

INSTITUTO TECNOLÓGICO SUPERIOR DE CALKINÍ

**“DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA DE EVALUACIÓN Y DESARROLLO DE
COMPETENCIAS EMPRENDEDORAS: UN ANÁLISIS DE SU IMPACTO EN EL DESEMPEÑO
INDIVIDUAL Y GRUPAL”**

TESIS PROFESIONAL

PARA OBTENER EL TÍTULO DE:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

QUE PRESENTA:

FERNANDO YAHIR PUC EUAN

DIRECTOR Y CO-DIRECTOR:

DRA. MARLENE MENDEZ MORENO

DR. ANGEL FRANCISCO CAN CABRERA

CALKINÍ, CAMPECHE, MÉXICO

MARZO, 2025

INSTITUTO TECNOLÓGICO SUPERIOR DE CALKINÍ

**“DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA DE EVALUACIÓN Y DESARROLLO DE
COMPETENCIAS EMPRENDEDORAS: UN ANÁLISIS DE SU IMPACTO EN EL DESEMPEÑO
INDIVIDUAL Y GRUPAL”**

TESIS PROFESIONAL

PARA OBTENER EL TÍTULO DE:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

QUE PRESENTA:

FERNANDO YAHIR PUC EUAN

DIRECTOR Y CO-DIRECTOR:

DRA. MARLENE MENDEZ MORENO

DR. ANGEL FRANCISCO CAN CABRERA

CALKINÍ, CAMPECHE, MÉXICO

MARZO, 2025



Educación
Secretaría de Educación Pública



Instituto Tecnológico Superior de Calkiní
Subdirección de Planeación y Desarrollo

Instituto Tecnológico Superior de Calkiní
Departamento de Administración y
Apoyo al Estudiante

Calkiní, Campeche 07/03/2025
P/03/25/660

INGENIERÍA EN SISTEMAS COMPUTACIONALES

CONSTANCIA DE NO INCONVENIENCIA

MODALIDAD: TESIS

C. FERNANDO YAHIR PUC EUAN 7567
PASANTE DE LA CARRERA DE
INGENIERÍA EN SISTEMAS COMPUTACIONALES.
PRESENTE

Con relación a lo establecido en el Reglamento de Estudios de Licenciatura, capítulo VII, fracción I, vigente en este Instituto, y una vez emitido el fallo favorable por el director DRA. MARLENE MÉNDEZ MORENO, y el co-director DR. ANGEL FRANCISCO CAN CABRERA, respectivamente, este Instituto consta de no tener inconveniencia para la realización del acto protocolario de titulación integral de su trabajo profesional titulado: **"DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA DE EVALUACIÓN Y DESARROLLO DE COMPETENCIAS EMPRENDEDORAS: UN ANÁLISIS DE SU IMPACTO EN EL DESEMPEÑO INDIVIDUAL Y GRUPAL."**

ATENTAMENTE

Excelencia en Educación Tecnológica®
Sabiduría, Fortaleza de Nuestra Cultura®

Dr. Jorge Alfonso Hau Puc.
Departamento de Administración y
Apoyo al Estudiante.

C.c.p. Archivo



INSTITUTO TECNOLÓGICO SUPERIOR
DE CALKINÍ

"DEPARTAMENTO DE ADMINISTRACIÓN
ESCOLAR Y APOYO A
ESTUDIANTES"

CLAVE: 04MSU0013P
CALKINÍ, CAMPECHE, MÉXICO.



2025
Año de
La Mujer
Indígena

Av. Ah Canul S/N por Carretera Federal, Calkiní, Campeche,
C.P. 24900 Tel. 996-8134870,
e-mail: se_dcalkini@tecnm.mx | itescam.edu.mx



DEDICATORIA

Este logro se lo dedico a mis abuelos Addy Collí y Manuel Collí las cuales son el motivo por el cual me impulsa a seguir adelante con mis proyectos.

A mi padre Manuel Fernando Puc Velázquez es muy importante para mí, ha sido mi guía y apoyo a lo largo de mi vida y es uno de los pilares que me impulsan a seguir cumpliendo con mis metas.

A mis abuelos Rosario Velázquez y Roque Puc por su apoyo y comprensión con mis proyectos y metas a lo largo de mi carrera.

A mi mamá Rosario que le tengo cariño y respeto por todo lo que ha hecho por mí.

A mis hermanos que han compartido buenos momentos conmigo y que vean que puedo cumplir con mis metas para que igual vean que si pueden cumplir los suyos.

"A quienes creen en mí incluso cuando yo mismo dudo. Su apoyo hace posible todo lo que soy."

(Yahir Puc)

AGRADECIMIENTOS

Primero quiero agradecer a mi padre Manuel y mi abuela Addy ellos dos han sido muy importantes en el transcurso de mis estudios me han apoyado en lo que necesite. Siempre han dado todo de sí para cumplir con mis metas. Les agradezco demasiado por todo lo que soy ahora y también por seguir con su apoyo incondicional, su cariño y sobre todo por creer en mí.

A mi familia que me han apoyado y motivado a seguir adelante, compartiendo buenos momentos con una sana convivencia.

A mis amigos, los REALES del Discord, Chucho, Axel, Jordi, Raul, Martín, Juan Pablo, Magdiel y David que a pesar de que me incluyeron a mediados de la carrera cuando no tenía la suficiente confianza para formar parte de ellos terminamos siendo amigos cercanos, viajando a lugares, hecho fiestas, jugado videojuegos donde me he divertido demasiado. Los quiero muchos amigos gracias por todo.

Al Strappy, Karla por ser mis mejores amigos que conozco desde hace tiempo, por su apoyo y consejos que me brindan a lo largo del tiempo gracias, los quiero.

A mis docentes, Dr. José Luis Lira Turriza, Dr. José Manuel Lira Turriza y Dra. Yaqueline Pech Uh por haber formado parte de mi carrera universitaria el cual me brindaron el conocimiento que hoy poseo y por las enseñanzas y consejos que me ayudaron a seguir adelante.

A mis asesores Dr. Marlene Méndez Moreno, Dr. Gonzalo Miguel Quetz Aguirre, MCM. Ángel Francisco Can Cabrera, Dr. Mario Ben-Hur Chuc Armendariz por su enseñanza y apoyo a lo largo de mi formación académica, por brindarme las observaciones y consejos en este proceso tan importante para la carrera.

CONTENIDO

DEDICATORIA.....	4
AGRADECIMIENTOS	5
ÍNDICE DE ILUSTRACIONES.....	9
ÍNDICE DE TABLAS.....	13
RESUMEN.....	14
Capítulo 1.....	15
INTRODUCCIÓN.....	15
1.2 ANTECEDENTES.	17
1.3 JUSTIFICACIÓN.	19
1.4 ALCANCES Y LIMITACIONES.....	21
1.5 OBJETIVOS GENERALES Y ESPECÍFICOS.....	22
1) Objetivo general.....	22
2) Objetivos específicos.....	22
Capítulo 2.....	24
MARCO TEÓRICO	24
2.1 METODOLOGÍA SCRUM.....	24
2.2 EMPRENDIMIENTO.	25
2.3 COMPETENCIAS EMPRENDEDORAS.	25
2.4 CARACTERÍSTICAS EMPRENDEDORAS PERSONALES (CEPS).....	25
2.5 EVALUACIÓN DE COMPETENCIAS EMPRENDEDORAS.....	27
2.6 MÉTODOS DE EVALUACIÓN DE COMPETENCIAS EMPRENDEDORAS.....	27
2.6 IMPACTO DE LA EVALUACIÓN EN EL DESARROLLO DE COMPETENCIAS:..	27
2.7 ARQUITECTURA DEL SISTEMA: FRONTEND Y BACKEND	28
2.8 SERVICIOS EN LARAVEL.....	29
2.9 GESTIÓN COLABORATIVA CON GITHUB.....	30
2.10 ESCALA LIKERT	30
Capítulo 3.....	31
METODOLOGÍA	31
3.1 MATERIALES.....	31
3.1.1 Backend	31

3.1.2 Frontend	33
3.1.3 Herramientas de desarrollo.	34
3.4 IMPLEMENTACIÓN DE LA METODOLOGÍA DEL PROYECTO	35
3.5 REQUERIMIENTOS	37
3.5.1 Requerimientos Funcionales	37
3.5.2 REQUERIMIENTOS NO FUNCIONALES.....	40
3.5 CASOS DE USO.....	41
3.5.1 Diagrama caso de uso para iniciar sesión.	41
3.5.2 Diagrama caso de uso para el estudiante.	42
3.5.3 Diagrama caso de uso para el maestro.	43
3.5.4 diagrama caso de uso para el administrador	44
3.5.5 Modelo relacional	45
3.6.1 Módulo de Registro.....	46
3.6.2. Módulo de Inicio de sesión	48
3.6.3 Módulo de bienvenida	49
3.6.4 Módulo de perfil de usuario.....	55
3.6.5 Módulo de análisis y reportes	76
3.6.7 Módulo edición de la página principal	77
3.6.8 Modulo Gestión de cuestionarios	78
3.7.1 Registro de alumno	82
3.7.2 Página principal	84
3.7.3 Login de usuarios.....	89
3.7.4 Dashboard del Alumno.....	90
3.7.5 Grupos	92
3.7.6 Cuestionarios para el alumno	93
3.7.7 Resultados del alumno	95

3.7.8 Dashboard para maestros	97
3.7.9 Lista de grupos	98
3.7.10 Vista de cuestionarios aplicados	102
3.7.11 Administración de alumnos del maestro	107
3.7.12 Dashboard del administrador.	108
3.7.13 Apartado Países.....	109
3.7.14 Apartado de instituciones.	111
3.7.15 Administración de los usuarios por institución.....	112
3.7.16 Administración de usuarios del sistema	117
3.7.17 Administración de cuestionarios.....	119
3.7.18 Ver detalles del cuestionario.....	120
3.7.19 Crear una nueva versión del cuestionario.....	122
3.7.20 Instrucciones del cuestionario	123
3.8 PRUEBAS DEL SISTEMA.....	129
CAPITULO 4. RESULTADOS Y DISCUSION.....	132
Capítulo 5. CONCLUSIONES Y RECOMENDACIONES.	149
REFERENCIAS BIBLIOGRÁFICAS	151
ANEXO	153

ÍNDICE DE ILUSTRACIONES

Ilustración 1 .Diagrama de uso para inicio sesión.	41
Ilustración 2. Diagrama de uso para el estudiante	42
Ilustración 3. Diagrama de uso para el maestro.....	43
Ilustración 4. Diagrama de uso para el administrador	44
Ilustración 5. Modelo Relacional	45
Ilustración 6. Diseño del registro del alumno	47
Ilustración 7. Diseño del login de usuarios.....	48
Ilustración 8. Diseño de la página principal	49
Ilustración 9. Diseño carrusel de imágenes	50
Ilustración 10. Diseño objetivo de la página	50
Ilustración 11. Diseño Aclaración	50
Ilustración 12. Diseño pie de pagina	51
Ilustración 13. Diseño ¿quiénes somos? (Historia)	51
Ilustración 14. Diseño ¿quiénes somos? (Misión y Visión)	52
Ilustración 15. Diseño te presento al equipo.....	52
Ilustración 16. Diseño te presento al equipo (Instituciones)	53
Ilustración 17. Diseño localización de las instituciones	53
Ilustración 18. Diseño localización de las instituciones	53
Ilustración 19. Diseño localización de las instituciones	54
Ilustración 20. Diseño cuestionario	54
Ilustración 21. Diseño Dashboard del alumno.....	55
Ilustración 22. Diseño búsqueda de grupo.....	56
Ilustración 23. Diseño visualización de grupos	56
Ilustración 24. Diseño de visualización de cuestionarios por grupo	57
Ilustración 25. Diseño visualización de información del perfil.....	58
Ilustración 26. Diseño visualización cuestionarios resueltos.	59
Ilustración 27. Diseño visualización resultados del alumno	60
Ilustración 28. Diseño visualización del Dashboard	61
Ilustración 29. Diseño de la vista agregar grupo	62
Ilustración 30. Diseño visualización de grupos	63
Ilustración 31. Diseño visualización de cuestionarios.....	63
Ilustración 32. Diseño visualización de alumnos del grupo	64
Ilustración 33. Diseño vista de agregar alumno al grupo	64
Ilustración 34. Diseño visualización de información del perfil.....	65
Ilustración 35. Diseño vista de gestión de alumnos	66
Ilustración 36. Diseño vista de resultados por grupo	67
Ilustración 37. Diseño Dashboard del Administrador	68
Ilustración 38. Diseño lista de instituciones	69
Ilustración 39. Diseño administración de usuarios por institución.....	70
Ilustración 40. Diseño lista de docentes por institución	70
Ilustración 41. Diseño vista registro de maestro	71

Ilustración 42. Diseño lista de grupos del docente	72
Ilustración 43- Diseño administración de usuarios del sistema.....	72
Ilustración 44. Diseño lista de docentes del sistema	73
Ilustración 45. Diseño de la vista de resultados por grupo	74
Ilustración 46. Diseño lista de alumnos del sistema.....	75
Ilustración 47. Diseño de la vista de resultados del alumno.....	76
Ilustración 48. Diseño de edición del perfil.....	77
Ilustración 49. Diseño de la edición del cuestionario	78
Ilustración 50. Diseño de cuestionarios aplicados.....	79
Ilustración 51. Diseño de instrucciones del cuestionario	80
Ilustración 52. Diseño del cuestionario	81
Ilustración 53. Vista del registro del alumno.....	82
Ilustración 54. Vista del registro del alumno.....	82
Ilustración 55. Datos ingresados.....	83
Ilustración 56. Datos ingresados.....	83
Ilustración 57. Vista del registro exitoso.	84
Ilustración 58. Página principal	84
Ilustración 59. Carrusel de imágenes.....	85
Ilustración 60. Objetivo de la página.....	85
Ilustración 61. Aclaración.....	86
Ilustración 62. Pie de página.....	86
Ilustración 63. Vista ¿quiénes somos?.....	87
Ilustración 64. Localización de las instituciones	87
Ilustración 65. Localización de las instituciones	88
Ilustración 66. Localización de las instituciones	88
Ilustración 67. Vista login de usuarios	89
Ilustración 68. Login de usuarios	89
Ilustración 69. Dashboard del alumno.....	90
Ilustración 70. Búsqueda de grupo	90
Ilustración 71. Información del grupo	91
Ilustración 72. Ingreso al grupo	91
Ilustración 73. Visualización de grupos	92
Ilustración 74. Visualización de cuestionarios (Si no hay disponibles)	92
Ilustración 75. Visualización de cuestionarios (Con cuestionarios resueltos).....	93
Ilustración 76. Visualización de cuestionarios (Con cuestionarios pendientes).....	93
Ilustración 77. Visualización del cuestionario.....	94
Ilustración 78. Visualización del cuestionario.....	94
Ilustración 79. Visualización de información (Resultados de autoevaluación).....	95
Ilustración 80.1 Visualización de resultados (Resultados de autoevaluación).....	95
Ilustración 81. Visualización de resultados (Gráfico de autoevaluación)	96
Ilustración 82. Visualización del Dashboard.....	97
Ilustración 83. Vista agregar grupo	97
Ilustración 84. Creación de nuevo grupo.....	98

Ilustración 85. Visualización de grupos	98
Ilustración 86. Visualización de cuestionarios (no disponibles)	99
Ilustración 87. Visualización de cuestionarios (Disponibles)	99
Ilustración 88. Visualización de alumnos del grupo.....	100
Ilustración 89. Visualización de cuestionarios	100
Ilustración 90. Alumno agregado al grupo	101
Ilustración 91. Lista de alumnos actualizada.....	101
Ilustración 92. Lista de cuestionarios aplicados	102
Ilustración 93. Vista publicar cuestionario existente a un grupo.....	103
Ilustración 94. Vista publicar cuestionario (Cuestionario seleccionado)	103
Ilustración 95. Vista publicar cuestionario (Cuestionario seleccionado)	104
Ilustración 96. Vista publicar cuestionario (Cuestionario seleccionado)	104
Ilustración 97. Mensaje cuestionario publicado.	105
Ilustración 98. Lista de cuestionario actualizada.....	105
Ilustración 99. Información del cuestionario aplicado en el grupo	106
Ilustración 100.1 Información del cuestionario aplicado en el grupo	106
Ilustración 101. Administración de alumnos del maestro	107
Ilustración 102. visualización de cuestionarios resueltos del alumno	107
Ilustración 103.1. visualización de resultados macro del	108
Ilustración 104. Dashboard del Administrador	108
Ilustración 105. Sección de países.....	109
Ilustración 106. Agregar país.....	110
Ilustración 107. Instituciones por país.....	110
Ilustración 108. Lista de instituciones	111
Ilustración 109. Instituciones filtradas.....	111
Ilustración 110. Lista de docentes por institución	112
Ilustración 111. Registro de maestro	113
Ilustración 112. Docente creado correctamente	114
Ilustración 113. Lista de docentes actualizada	114
Ilustración 114. Lista de grupos del docente	115
Ilustración 115. Lista de integrantes del grupo.....	115
Ilustración 116. Lista de alumnos de la institución	116
Ilustración 117. Administración de usuarios del sistema	117
Ilustración 118. Lista de docentes del sistema	117
Ilustración 119. Lista de alumnos del sistema.....	118
Ilustración 120. Lista de cuestionarios	119
Ilustración 121. Detalles del cuestionario.....	120
Ilustración 122. Detalles del cuestionario.....	120
Ilustración 123. Detalles del cuestionario (última página).....	121
Ilustración 124. Crear nueva versión del cuestionario	122
Ilustración 125. Crear nueva versión del cuestionario (última pregunta).....	122
Ilustración 126. Vista de instrucciones del cuestionario	123
Ilustración 127. Cuestionarios resueltos del alumno.....	124

Ilustración 128. Análisis de resultados por categoría.	124
Ilustración 129. Análisis de resultados por categoría parte 2.	125
Ilustración 130. Análisis de resultados por expresión.	125
Ilustración 131. Gráfica de resultados por expresión.	126
Ilustración 132. Resultados por pregunta – Expresión.	126
Ilustración 133. Información del alumno.	127
Ilustración 134. Resultados de autoevaluación.	127
Ilustración 135. Gráfico de autoevaluación.	127
Ilustración 136. Inicio de sesión del alumno.	132
Ilustración 137. Inicio de sesión del alumno.	132
Ilustración 138. Buscar grupo por código de acceso.	133
Ilustración 139. Información del grupo.	133
Ilustración 140. Ingreso exitoso al grupo.	133
Ilustración 141. Dashboard del estudiante.	134
Ilustración 142. Cuestionarios pendientes del estudiante.	134
Ilustración 143. Información del cuestionario.	135
Ilustración 144. Instrucciones del cuestionario.	135
Ilustración 145. Título y descripción del cuestionario.	135
Ilustración 146. Últimas preguntas y respuestas del cuestionario.	136
Ilustración 147. Información del alumno y cuestionario.	136
Ilustración 148. Tabla de resultados por puntaje y variable.	137
Ilustración 149. Gráfico araña de autoevaluación.	137
Ilustración 150. Dashboard del docente.	138
Ilustración 151. Sección de grupos del docente.	138
Ilustración 152. Visualización de cuestionarios aplicados del grupo.	139
Ilustración 153. Tabla de métricas del cuestionario por grupo.	139
Ilustración 154. Tabla de métricas del cuestionario por grupo.	140
Ilustración 155. Tabla de integrantes del grupo.	140
Ilustración 156. Cuestionarios resueltos del alumno.	141
Ilustración 157. Destalles del cuestionario y puntajes por categorías (Macro) parte 1.	141
Ilustración 158. Detalles del cuestionario y puntajes por categorías (Macro) parte 2.	142
Ilustración 159. Resultados “Micro” de la categoría ATRIBUTO EMPRENDEDOR.	143
Ilustración 160. Gráfica de resultados “Micro” de la categoría ATRIBUTO EMPRENDEDOR.	143
Ilustración 161. Resultados Micro y Gráfica de la categoría ACTITUD EMPRENDEDORA.	144
Ilustración 162. Resultados Micro y Gráfica de la categoría CAPACIDAD EMPRENDEDORA.	145
Ilustración 163. Resultados Micro y Gráfica de la categoría CAPACIDAD INTERPERSONAL.	145

Ilustración 164. Resultados Micro y Grafica de la categoría FACTOR DE CORRECCIÓN.	146
--	-----

ÍNDICE DE TABLAS

Tabla 1. Equipo del Proyecto.	35
Tabla 2. Tipos de usuarios.	36
Tabla 3. Requerimientos funcionales.	40
Tabla 4. Requerimientos no funcionales	40
Tabla 5. Pruebas realizadas de los módulos de bienvenida e inicio de sesión.	129
Tabla 6. Pruebas realizadas de los módulos de perfil de usuario, gestión de grupos y cuestionarios.	130
Tabla 7. Pruebas realizadas de los módulos de resultados y retroalimentación y exportación de resultados.	131

RESUMEN

Este estudio se centra en la implementación y análisis de una plataforma web para la evaluación de competencias emprendedoras en estudiantes universitarios, con el propósito de validar su impacto en la autoevaluación y en la optimización del proceso de medición de habilidades emprendedoras. A través de un enfoque cuantitativo, se analizaron los resultados obtenidos tras la aplicación del sistema, evaluando su eficacia en la recopilación, estructuración y visualización de datos de manera automatizada.

La plataforma permite a los estudiantes autoevaluar sus competencias de forma intuitiva mediante cuestionarios estructurados en escala Likert, mientras que los docentes pueden acceder a reportes detallados con métricas clave, eliminando la necesidad de cálculos manuales y facilitando la interpretación de los resultados. A nivel técnico, el sistema fue desarrollado utilizando Angular para el frontend y Laravel para el backend, con MySQL para la gestión de bases de datos y JWT para garantizar la seguridad en la autenticación de usuarios.

La metodología ágil Scrum se empleó para la implementación del sistema, permitiendo un desarrollo iterativo que facilitó ajustes y mejoras basadas en las pruebas realizadas. Los resultados de la evaluación indican que la plataforma logra una optimización significativa en la administración de evaluaciones, reduciendo tiempos de procesamiento de datos y proporcionando una estructura visual clara para la toma de decisiones.

A partir del análisis de los datos recopilados, se concluye que la plataforma no solo mejora la experiencia de autoevaluación para los estudiantes, sino que también representa una herramienta eficaz para la gestión académica de competencias emprendedoras. Como trabajo futuro, se recomienda la incorporación de módulos de retroalimentación personalizada y análisis predictivo, con el fin de potenciar su utilidad en entornos educativos y fortalecer el desarrollo de habilidades clave en los alumnos.

Capítulo 1

INTRODUCCIÓN.

El desarrollo de competencias emprendedoras en estudiantes universitarios se ha convertido en un factor importante dentro de la educación superior, ya que estas habilidades permiten a los jóvenes afrontar los retos que ofrece el campo laboral. Por ello, la evaluación de dichas competencias emprendedoras se considera un desafío para las instituciones educativas, ya que los métodos tradicionales utilizados, como cuestionarios escritos o encuestas manuales, no permiten un análisis inmediato ni una personalización del aprendizaje basada en las necesidades individuales de cada estudiante por lo que lleva tiempo al momento de evaluar dichas competencias. A falta de herramientas especializadas que faciliten la evaluación y el seguimiento continuo de estas competencias limita el desarrollo de estrategias efectivas para fortalecerlas de manera progresiva.

Ante esta problemática, el presente estudio propone el diseño e implementación de una plataforma web para la evaluación y desarrollo de competencias emprendedoras, con la finalidad de ofrecer un sistema que facilite la medición de habilidades clave y proporcione una retroalimentación personalizada a los estudiantes. A través de la utilización de cuestionarios estructurados basados en la escala Likert, por ello la plataforma permitirá a los usuarios conocer su nivel de competencias y recibir recomendaciones adaptadas a sus resultados, optimizando así el proceso de aprendizaje y fortalecimiento de habilidades emprendedoras.

En este contexto, la investigación se centra en responder la siguiente pregunta: ¿Cómo influye la utilización de una plataforma de evaluación de competencias emprendedoras en el desarrollo de habilidades clave a nivel individual en estudiantes universitarios? Para dar respuesta a la pregunta, se plantea evaluar la efectividad de la plataforma en la personalización de los cuestionarios, analizando en qué medida esta herramienta permite una evaluación eficaz de los estudiantes dando a conocer los resultados de los cuestionarios, así como la comparación de estos.

Para el desarrollo del sistema se utilizó la metodología ágil Scrum, la cual nos permitió estructurar el proceso de las tareas organizadas para asegurar una implementación eficiente y flexible. La arquitectura del sistema se compone por el frontend desarrollado en Angular y el backend realizado en Laravel, con una base de datos realizado en MYSQL y un sistema de autenticación mediante JWT para garantizar la seguridad de los usuarios y el sistema. Las tecnologías mencionadas fueron seleccionadas por su capacidad para desarrollar aplicaciones web dinámicas y escalables, asegurando una experiencia de usuario óptima con el sistema.

La finalidad de esta investigación es buscar contribuir al desarrollo de herramientas digitales que faciliten la evaluación de competencias emprendedoras en el ámbito universitario, promoviendo un sistema la cual beneficie a los docentes para la aplicación de dichas evaluaciones, así como la retroalimentación de los estudiantes. Asimismo, se espera

que la plataforma propuesta sirva como referencia para futuras iniciativas en la integración de tecnología educativa aplicada al emprendimiento, siendo una herramienta que beneficie tanto a docentes y alumnos para la evaluación de dichas competencias.

1.2 ANTECEDENTES.

Para ser un buen emprendedor, es crucial contar con ciertas competencias emprendedoras. La competencia de un emprendedor es la capacidad, habilidad, pericia o destreza que debe tener un emprendedor para poder cumplir objetivos. En otras palabras, las competencias emprendedoras están relacionadas con capacidades, destrezas, habilidades y aptitudes que posee un emprendedor, o puede adquirir, a través de procesos formativos y de capacitación.

Según (Tortajada, 2020) una competencia emprendedora es la forma en que una persona utiliza todos sus recursos personales para resolver una tarea en contexto, sujeto a diferentes componentes, como son las habilidades, actitudes, conocimientos y experiencias.

Por ende, existen diferentes herramientas para evaluar dichas competencias, una herramienta más usada es la escala Likert. La evaluación de competencias se ha vuelto cada vez más importantes en los programas de educación superior, para la evaluación de las competencias emprendedoras en el ámbito universitario el cuestionario es un instrumento para emplear.

(Bacigalupo et al. 2016) mencionan que el uso de las escalas Likert son una herramienta común para la autoevaluación de competencias, especialmente cuando se utilizan en modelos como el *EntreComp Framework*, que ha sido ampliamente usado en la educación emprendedora.

Definimos el cuestionario basándonos en Tapia Moreno (2010) como el conjunto de ítems diferentes que pueden ser planteados de forma interrogativa, enunciativa, afirmativa o negativa con varias alternativas, con un formato determinado, un orden de preguntas y un contenido concreto sobre el tema que queremos investigar. Por ello el objetivo del cuestionario será traducir las variables de preguntas concretas que nos proporcionan la información viable de ser cuantificada. (Aparicio, Palacios, Martínez, Verduzco y Retana, 2008)

Los sistemas de evaluación que se implementan en la educación superior son elaborados por profesores, (PhD, 2023) menciona que, para lograr construir las condiciones ideales para impulsar el desarrollo emprendedor, los institutos deben desarrollar una visión sistemática. Por lo que nos da a entender que el desarrollo de ciertas áreas concretas permita la funcionalidad y coordinación para potenciar al emprendedor, dando a conocer sus competencias y proporcionando ciertas recomendaciones para pulir las competencias las cuales el universitario ha salido bajo y pueda obtener una mejora de este.

(Lizbeth Alicia González Tamayo, 2019) menciona que, la evaluación de las competencias emprendedoras a través de instrumentos estandarizados permite a las instituciones educativas identificar perfiles emprendedores y proponer estrategias de mejora personalizadas para los estudiantes.

Por ello el proyecto que permite la evaluación de dichas competencias se propone para que las instituciones educativas identifiquen y mejoren los perfiles emprendedores de

los universitarios. A través de cuestionarios y empleando un método de evaluación con la escala Likert proporcione los puntajes obtenidos de cada una de las CEPS y recomendaciones a emplear para mejorarlas.

1.3 JUSTIFICACIÓN.

El desarrollo de competencias emprendedoras es una característica clave en el ámbito educativo, ya que estas habilidades permiten a los estudiantes afrontar los desafíos que proporciona el campo laboral. Sin embargo, la evaluación de estas competencias sigue siendo un reto para las instituciones educativas, debido a la falta de herramientas especializadas que permitan medir y analizar con precisión el progreso individual de cada estudiante. Los métodos tradicionales, como cuestionarios escritos y encuestas manuales, presentan limitaciones al no ofrecer un análisis inmediato ni una personalización del aprendizaje, lo que dificulta la implementación de estrategias efectivas para su fortalecimiento.

Este proyecto, orientado a la creación de un sistema web para la evaluación de competencias emprendedoras, tiene como objetivo principal ofrecer una solución efectiva que permita a los estudiantes adquirir un mejor conocimiento de sus capacidades en este ámbito. A través de este sistema, los estudiantes podrán identificar sus fortalezas y áreas de oportunidad en cuanto a sus habilidades emprendedoras, lo que les permitirá enfocarse en aquellas competencias que necesitan mejorar para desarrollar un perfil emprendedor sólido. De esta manera, se busca fomentar en los estudiantes una mentalidad emprendedora, preparándolos no solo para enfrentar los retos actuales, sino también para adaptarse a las demandas futuras de su determinado sector.

Además de beneficiar directamente a los estudiantes, este proyecto contribuirá al fortalecimiento de los programas académicos que ofrecen las instituciones. La herramienta propuesta no solo permitirá una evaluación individual de las competencias emprendedoras, sino que también proporcionará a los docentes y a las instituciones educativas información valiosa sobre el nivel de desarrollo de estas competencias en sus estudiantes. Este análisis facilitará a los maestros y directivos una visión más clara y objetiva del desempeño de los alumnos, permitiéndoles ajustar sus programas de apoyo y enseñanza de manera más efectiva y acorde a las necesidades reales del alumnado.

Es importante resaltar que actualmente existe una falta de herramientas específicas que permitan realizar una evaluación clara y objetiva de las competencias emprendedoras. Este vacío dificulta la tarea tanto para estudiantes como para docentes, quienes carecen de indicadores precisos que les ayuden a medir el progreso en el desarrollo de estas habilidades. Por lo tanto, el desarrollo de este sistema web servirá para cubrir esta carencia, al brindar una solución innovadora y accesible que facilite la identificación y mejora de las competencias emprendedoras de los alumnos.

La relevancia de este proyecto radica en su capacidad para adaptar las oportunidades de aprendizaje según el progreso individual de cada estudiante, permitiendo que cada usuario reciba una evaluación personalizada y recomendaciones específicas para mejorar sus competencias emprendedoras. Esto no solo beneficiará su formación profesional y su inserción en el mercado laboral, sino que también fomentará la creación de nuevos proyectos y emprendimientos, impulsando el desarrollo económico y la cultura emprendedora dentro del entorno universitario.

De este modo, se espera que los alumnos no solo estén mejor preparados para enfrentar los desafíos del mercado laboral, sino que también sean capaces de crear y gestionar sus propios proyectos empresariales, generando un impacto positivo en el desarrollo del ecosistema emprendedor.

Además, este proyecto será de gran utilidad para los docentes y directivos de las instituciones educativas, ya que les proporcionará una herramienta eficaz para evaluar el nivel de competencias emprendedoras de sus alumnos y así poder tomar decisiones informadas sobre cómo mejorar sus programas académicos. Con esta información, los docentes podrán ajustar sus metodologías y enfoques, y las instituciones podrán diseñar estrategias más adecuadas para apoyar a los estudiantes en el desarrollo de un perfil emprendedor. Asimismo, permitirá a las universidades establecer métricas claras de progreso en el ámbito del emprendimiento, lo que a su vez contribuirá a mejorar la calidad de la enseñanza y el aprendizaje en distintas áreas.

1.4 ALCANCES Y LIMITACIONES.

Alcance del proyecto.

El proyecto de investigación busca evaluar la influencia de una plataforma web para la medición de las competencias emprendedoras en estudiantes universitarios. Mediante la implementación y análisis del sistema, se busca evaluar su impacto en el desempeño individual de los estudiantes universitarios. La investigación busca determinar cómo la personalización de las evaluaciones y del contenido educativo influye en el desarrollo de habilidades clave en el ámbito del emprendimiento. Por lo tanto, los aspectos que se incluyen dentro de los alcances del estudio son:

- Desarrollo e implementación de una plataforma web diseñada para evaluar competencias emprendedoras en estudiantes universitarios.
- Edición de los cuestionarios con base al contexto de la institución y programa educativo referentes a las carreras de los universitarios.
- Implementación de un sistema de autenticación y registro de usuarios.
- Evaluación de la usabilidad y funcionalidad del sistema.
- Pruebas de implementación con grupos de estudiantes universitarios, analizando su experiencia en la plataforma y la manera en que esta influye en su desempeño individual.
- Generación de los informes de los usuarios en formato PDF.
- Realización de pruebas iniciales y corrección de errores.
- Capacitación para los usuarios finales del sistema.
- Documentación completa del sistema.
- Despliegue final del sistema en un servidor.
- Provisión de recomendaciones personalizadas para mejorar las competencias emprendedoras de los usuarios.

Limitaciones.

- La plataforma solo será desarrollada como un sitio web.
- Los usuarios son registrados con la matrícula proporcionada por las instituciones.
- Se necesitará el acceso a internet para utilizar el sistema, lo que puede ser un desafío en áreas con limitaciones de conectividad.
- El sistema se enfocará exclusivamente en estudiantes universitarios de distintas carreras.
- No incluirá el seguimiento o evaluación de emprendimientos una vez iniciados.
- No abordará el desarrollo de habilidades emprendedoras en sectores no educativos.
- El proyecto se limitará al contexto educativo universitario, sin incluir otros niveles académicos.

1.5 OBJETIVOS GENERALES Y ESPECÍFICOS.

1) Objetivo general.

Evaluar la influencia de una plataforma web en la medición y desarrollo de competencias emprendedoras en estudiantes universitarios, analizando el impacto en su desempeño individual y grupal, haciendo énfasis en la personalización de las evaluaciones y del contenido educativo.

2) Objetivos específicos.

- Implementar una interfaz que permita la gestión y organización de diferentes grupos de estudiantes para facilitar el monitoreo y evaluación continua.
- Implementar un sistema de cuestionarios en línea que evalúe las competencias emprendedoras de los estudiantes utilizando la escala Likert.
- Analizar el módulo de visualización de datos que permita presentar los resultados individuales y grupales de manera clara y comprensible para los usuarios.
- Implementar un sistema de almacenamiento seguro de los resultados de las evaluaciones que garantice la confidencialidad de la información personal y académica.
- Integrar herramientas de análisis estadístico que permitan al administrador generar conclusiones sobre el desarrollo emprendedor de los estudiantes a partir de los datos recopilados.
- Asegurar la protección de los datos mediante medidas de seguridad que garanticen su manejo adecuado y cumplan con normativas de privacidad.

1.6 Hipótesis

El uso de una plataforma web para la evaluación de competencias emprendedoras permitirá a los estudiantes universitarios realizar autoevaluaciones de manera rápida y sencilla, eliminando la necesidad de procesos manuales y proporcionando resultados inmediatos y precisos. Esto facilitará la identificación de sus fortalezas y áreas de mejora en el desarrollo de habilidades emprendedoras.

Capítulo 2

MARCO TEÓRICO

En este capítulo abordaremos los conceptos teóricos que son fundamentales para sustentar el análisis e implementación de una plataforma de evaluación y desarrollo de competencias emprendedoras

2.1 METODOLOGÍA SCRUM

Scrum es una metodología ágil diseñada para gestionar proyectos complejos de manera iterativa e incremental, enfocándose en la colaboración, la flexibilidad y la entrega continua de valor.

Principales Elementos de Scrum:

- Roles en Scrum:
 - Scrum Máster: Coordina al equipo, elimina impedimentos y fomenta la adopción de Scrum.
 - Product Owner: Gestiona el backlog del producto, priorizando las tareas para maximizar el valor entregado.
 - Equipo de Desarrollo: Conforman un grupo multifuncional encargado de diseñar, desarrollar y entregar el producto.
- Eventos de Scrum:
 - Sprint: Periodos definidos en los que se desarrolla un conjunto específico de funcionalidades.
 - Daily Scrum: Reuniones diarias breves para sincronizar el progreso y planificar el día.
 - Sprint Review: Presentación de los avances al final del sprint, recibiendo retroalimentación.
 - Sprint Retrospective: Análisis del proceso de trabajo, identificando mejoras para futuros Sprints.
- Artefactos de Scrum:
 - Product Backlog: Lista priorizada de requerimientos.
 - Sprint Backlog: Tareas seleccionadas para completarse durante un sprint.
 - Incremento: Producto funcional entregable al final de cada sprint.

Scrum permite estructurar y priorizar las tareas, garantizando entregas constantes y la flexibilidad para adaptarse a cambios en los requerimientos.

2.2 EMPRENDIMIENTO.

El emprendimiento se define como la capacidad de una persona para gestionar y organizar diversos factores de producción, innovar, asumir riesgos y enfrentar situaciones imprevistas. Según Fayyaz et al. (2009), el emprendimiento implica la habilidad de coordinar recursos, introducir innovaciones y tomar decisiones bajo incertidumbre para alcanzar objetivos específicos.

2.3 COMPETENCIAS EMPRENDEDORAS.

Las competencias emprendedoras son habilidades que permiten a los estudiantes actuar sobre ideas y oportunidades para generar resultados de valor.

(Martínez, et al. 2009) Menciona que las competencias emprendedoras buscan la autonomía de la persona y se orientan hacia la autorrealización de un proyecto vital centrado en la producción de bienes y servicios que satisfagan las necesidades de la comunidad. De igual manera, el mismo autor menciona que la cultura emprendedora es una alternativa de desarrollo y progreso que no solo refiere en el ámbito laboral, sino también en el plano social y medio ambiental. La necesidad de fomentar la cultura emprendedora tomando en cuenta las bases de las competencias emprendedoras, motiva por razones de orden personal, social y económico.

(Martínez, 2009) Menciona que las competencias emprendedoras buscan la autonomía de la persona y se orientan hacia la autorrealización del sujeto, como una vía para la realización de un proyecto vital centrado en la producción de bienes y servicios que satisfagan las necesidades de la comunidad.

2.4 CARACTERÍSTICAS EMPRENDEDORAS PERSONALES (CEPS).

Las características emprendedoras son aquellas cualidades y comportamientos personales que son claves necesarias para que un emprendedor sea exitoso. Estas características son importantes para el éxito en el emprendimiento ya que permite a los individuos enfrentar desafíos y aprovechar oportunidades en el ámbito empresarial.

Dichas características se dividen de la siguiente manera:

- **Atributo emprendedor:**
 - **Buscar y aprovechar las oportunidades:** Los emprendedores exitosos destacan por su capacidad para identificar necesidades insatisfechas en el mercado y desarrollar soluciones innovadoras, lo que les permite crear propuestas de valor únicas y competitivas (Silva, 2020).
 - **Fijar objetivos:** La capacidad de establecer metas claras y alcanzables orienta los esfuerzos y recursos de manera efectiva. Esto permite a los emprendedores definir objetivos específicos que guían el crecimiento de sus negocios (Espinoza Valencia, Real Pérez, & González González, 2021).

- **Planificar y hacer seguimiento sistemático:** La planificación estratégica y el monitoreo constante del progreso permiten a los emprendedores adaptarse a cambios y optimizar sus operaciones, asegurando la sostenibilidad a largo plazo (Barba Bayas, 2013).
- **Actitud emprendedora:**
 - **Ser persistentes:** La persistencia es clave para enfrentar y superar obstáculos, manteniendo el enfoque en los objetivos a pesar de las adversidades (Blanco, 2021).
 - **Cumplimiento de compromisos:** La capacidad de honrar acuerdos con clientes, proveedores y socios es esencial para construir una reputación sólida y relaciones de confianza en el ámbito empresarial (Cavero, 2023).
 - **Exigir eficiencia y calidad:** La búsqueda constante de mejorar la calidad de los productos o servicios y optimizar procesos es fundamental para satisfacer las expectativas del mercado y mantener la competitividad (Pedraza, 2023).
- **Capacidad emprendedora:**
 - **Correr riesgos moderados:** Implica evaluar cuidadosamente las posibles consecuencias antes de tomar decisiones que impliquen riesgos, asumiendo aquellos que ofrecen una relación favorable entre riesgo y beneficio (Vargas-Hernández & Mercado-Campuzano, 2020).
 - **Tener autoconfianza:** La seguridad en las propias habilidades y juicios permite a los emprendedores enfrentar desafíos y tomar decisiones con determinación, incluso en situaciones de incertidumbre (Bernal, 2024).
- **Capacidad interpersonal:**
 - **Buscar y conseguir información:** La proactividad para obtener información relevante ayuda en la toma de decisiones informadas. Los emprendedores deben estar al tanto de las últimas tendencias y necesidades del mercado (Cavero, 2023).
 - **Persuasión y creación de redes de apoyo:** La capacidad para influir en otras personas y establecer una red sólida de contactos brinda apoyo en diversos aspectos del emprendimiento, facilitando negociaciones y relaciones beneficiosas con clientes, proveedores e inversores (Bernal, 2024).

2.5 EVALUACIÓN DE COMPETENCIAS EMPRENDEDORAS.

Según (Quezada et al. 2021). Para la evaluación de competencias emprendedoras es importante identificar y potenciar las habilidades que son necesarias en los estudiantes universitarios que buscan desarrollar iniciativas empresariales exitosas. Estas competencias actúan en conjunto de conocimientos, habilidades y actitudes que permiten al emprendedor enfrentar desafíos y aprovechar oportunidades en el entorno empresarial.

2.6 MÉTODOS DE EVALUACIÓN DE COMPETENCIAS EMPRENDEDORAS.

Existen distintos métodos para la medición de las competencias emprendedoras algunas más destacables son:

- **Cuestionarios Estructurados:** Instrumentos diseñados para medir competencias específicas. Por ejemplo, el estudio de Quezada et al. (2021) desarrolló un instrumento basado en un modelo de tres categorías de competencias genéricas: personal, interpersonal y funcional. Este instrumento consta de 14 reactivos con gradientes de desempeño que los participantes responden según su experiencia (Quezada et al., 2021).
- **Entrevistas y Observación:** Métodos cualitativos que analizan el desempeño y las actitudes de los emprendedores en situaciones reales o simuladas. Estas técnicas permiten una comprensión profunda del comportamiento emprendedor y de cómo se manifiestan las competencias en la práctica (Quezada et al., 2021).
- **Autoevaluaciones:** Herramientas donde los propios emprendedores valoran sus competencias a través de escalas de medición. Este enfoque promueve la auto reflexión y el reconocimiento de áreas de mejora personal (Loor Ramos & Larreal Bracho, 2023).

2.6 IMPACTO DE LA EVALUACIÓN EN EL DESARROLLO DE COMPETENCIAS:

La evaluación de las competencias emprendedoras beneficia en diversos aspectos como la identificación de Fortalezas y Áreas de Mejora ya que permite a los emprendedores reconocer sus habilidades actuales y las áreas que requieren desarrollo adicional (Quezada et al., 2021).

De igual manera ayuda a las instituciones de educación superior a facilitar la creación de planes de formación adaptados a las necesidades específicas de los emprendedores, optimizando el proceso de aprendizaje.

Otro beneficio que ofrece es la retroalimentación a los individuos ya que al obtener los resultados través de la evaluación contribuye a aumentar la confianza de los emprendedores en sus capacidades para iniciar y gestionar proyectos empresariales.

2.7 ARQUITECTURA DEL SISTEMA: FRONTEND Y BACKEND

El sistema se divide en dos componentes principales: frontend y backend, integrados mediante servicios API para garantizar la interacción fluida y eficiente entre las capas.

2.7.1 Frontend

El frontend es la interfaz visual y funcional con la que interactúan los usuarios. En este proyecto, Angular fue seleccionado como framework principal debido a su capacidad para desarrollar aplicaciones de una sola página (SPA).

- Funciones del Frontend:
 - Proporcionar interfaces visuales intuitivas y responsivas.
 - Sincronizar datos con el backend mediante APIs REST.
 - Gestionar la interacción del usuario con la lógica del sistema.
- Angular como Framework del Frontend: Angular, desarrollado por Google, se destaca por su arquitectura modular y herramientas avanzadas. Sus principales características incluyen:
 - Componentes Reutilizables: Dividen la aplicación en partes independientes y reutilizables.
 - Data Binding Bidireccional: Automatiza la sincronización entre el modelo de datos y las vistas.
 - Enrutamiento: Permite la navegación entre diferentes vistas de la aplicación sin recargar la página.
 - Inyección de Dependencias: Facilita la integración de servicios externos.

2.7.2 Backend

El backend gestiona la lógica del negocio, los datos y la seguridad del sistema. Laravel fue utilizado para estructurar esta capa debido a su enfoque en la simplicidad y su robusta arquitectura MVC (Modelo-Vista-Controlador).

- Funciones del Backend:
 - Procesar solicitudes del frontend.
 - Implementar reglas de negocio y lógica compleja.
 - Garantizar la integridad y seguridad de los datos.
- Laravel como Framework del Backend: Laravel ofrece herramientas y características avanzadas que simplifican el desarrollo del backend:
 - ORM Eloquent: Facilita el manejo de bases de datos relacionales.
 - Middleware: Garantiza la autenticación y control de acceso.
 - Servicios API: Gestiona la comunicación entre frontend y backend de forma eficiente.

2.8 SERVICIOS EN LARAVEL

Los servicios en Laravel encapsulan funcionalidades específicas, promoviendo un diseño modular y organizado. Estos permiten separar responsabilidades dentro del backend y reutilizar lógica común en diferentes partes del sistema.

Ventajas de los Servicios en Laravel:

- Reutilización: Un servicio puede ser utilizado en múltiples componentes del sistema.
- Pruebas Unitarias: Facilitan la realización de pruebas individuales en la lógica encapsulada.
- Mantenimiento: Al segmentar responsabilidades, el código es más fácil de entender y mantener.

2.9 GESTIÓN COLABORATIVA CON GITHUB

GitHub es una plataforma de control de versiones basada en Git que facilita el trabajo colaborativo y la gestión de proyectos de desarrollo.

Características de GitHub Utilizadas:

- Control de Versiones: Registro detallado de cambios realizados en el código.
- Trabajo en Equipo: Uso de ramas para separar desarrollos individuales y facilitar la integración continua.
- Gestión de Issues: Seguimiento de tareas, errores y nuevas funcionalidades.

2.10 ESCALA LIKERT

La escala Likert es una herramienta comúnmente utilizada en encuestas para medir las actitudes o percepciones de los usuarios hacia un tema o producto. En este proyecto, la escala Likert se implementó en el diseño de cuestionarios interactivos.

Características de la Escala Likert:

- Formato: Consta de una serie de afirmaciones a las que los usuarios responden seleccionando un nivel de acuerdo o desacuerdo en una escala (por ejemplo, de 1 a 5).
- Medición: Permite cuantificar actitudes cualitativas, proporcionando datos numéricos para su análisis.
- Flexibilidad: Puede adaptarse a diferentes contextos y propósitos de investigación.

Capítulo 3

METODOLOGÍA

En este apartado se describirán los materiales y métodos usados en el transcurso del proyecto, por lo que a continuación proporcionaremos cada herramienta tecnológica y técnicas existentes que fueron escogidas para el desarrollo, así como las técnicas aplicadas durante las fases de implementación. El enfoque principal fue garantizar un flujo de trabajo ágil y eficiente, permitiendo que cada una de las etapas del proyecto pueda replicarse en otros contextos con las mismas condiciones.

3.1 MATERIALES.

3.1.1 Backend

Laravel



Ilustración 1. Logotipo Laravel.

Laravel es un framework PHP que sigue el patrón MVC (Modelo-Vista-Controlador) para el desarrollo de aplicaciones web. En este proyecto, Laravel fue empleado para gestionar el backend, brindando una plataforma robusta para la administración de usuarios, grupos y cuestionarios.

Las características clave de Laravel en este proyecto fueron:

- Rutas y controladores: Se configuraron rutas que conectan el frontend con las Apis del sistema.
- Migraciones y Eloquent ORM: Se gestionó la estructura de la base de datos mediante migraciones y se utilizaron relaciones en las tablas a través del ORM Eloquent.
- Middleware de autenticación: Laravel facilitó la implementación de la autenticación de usuarios mediante su sistema de middleware.

PHP

PHP fue el lenguaje utilizado en el backend, especialmente con Laravel. Su capacidad para gestionar la lógica del servidor e interactuar con bases de datos lo convierte en una opción ideal para desarrollar aplicaciones web robustas.

Las características principales de PHP en este proyecto fueron:

Desarrollo basado en servidores: PHP gestiona las solicitudes del frontend, interactúa con MySQL y devuelve los datos necesarios.

- Integración con Laravel: Al ser el lenguaje sobre el cual se basa Laravel, PHP permitió estructurar la aplicación bajo el patrón MVC.
- Facilidad y flexibilidad: PHP ofrece una curva de aprendizaje accesible y permite implementar rápidamente soluciones en el backend.

MySQL

MySQL fue la base de datos utilizada para almacenar la información del sistema. Se seleccionó por su capacidad para gestionar grandes volúmenes de datos y su integración con Laravel.

Algunos de los aspectos más relevantes de MySQL en este proyecto son:

- Gestión de tablas: Se crearon tablas para manejar la información de usuarios, grupos y cuestionarios.
- Relaciones entre tablas: La base de datos fue diseñada utilizando claves foráneas para garantizar la integridad referencial entre los módulos del sistema.

JSON Web Tokens (JWT)

JWT fue implementado para la autenticación de usuarios en el sistema. Este estándar asegura la transmisión segura de información entre cliente y servidor a través de tokens firmados digitalmente.

Las ventajas más notables de JWT en el proyecto incluyen:

- Autenticación basada en tokens: Los tokens generados identifican al usuario en cada solicitud al servidor, evitando la necesidad de gestionar sesiones.
- Seguridad: Los tokens incluyen firmas digitales que aseguran la integridad y autenticidad de la información transmitida.

3.1.2 Frontend

Angular



Ilustración 2. Logotipo Angular.

Angular es un framework de código abierto mantenido por Google que facilita el desarrollo de aplicaciones web dinámicas de una sola página (SPA). En este proyecto, Angular se utilizó para crear el frontend del sistema, lo que permitió diseñar interfaces interactivas y responsivas.

Algunas de las ventajas más destacadas de Angular son:

- Desarrollo estructurado: Gracias a TypeScript y a su enfoque modular, Angular permite un mantenimiento más sencillo del código.
- Data binding: Se implementó el enlace bidireccional de datos para sincronizar el modelo de datos con la vista de manera automática.
- Componentes reutilizables: La aplicación fue organizada en componentes, lo que favoreció la reutilización del código y una mejor estructuración de esta.

JavaScript

JavaScript es un lenguaje de programación interpretado y orientado a objetos, ampliamente utilizado para el desarrollo web. En este proyecto, se utilizó en el frontend, principalmente en las interacciones y lógica de los componentes de Angular.

Las características principales de JavaScript que se aprovecharon incluyen:

- Manipulación del DOM: Se utilizó para actualizar la interfaz de usuario de manera dinámica, sin necesidad de recargar la página.
- Funciones asíncronas: Se emplearon promesas y `async/await` para gestionar de manera eficiente las solicitudes a la API.

TypeScript

TypeScript es un lenguaje desarrollado por Microsoft que extiende JavaScript añadiendo tipado estático opcional. En este proyecto, TypeScript fue clave en el desarrollo del frontend con Angular, facilitando la creación de componentes robustos y escalables.

Entre sus características más destacadas en el proyecto se encuentran:

- Tipado estático: Definir tipos para variables, funciones y objetos ayuda a prevenir errores durante el desarrollo y a mantener el código más predecible.
- Compatibilidad con JavaScript: Al compilarse en JavaScript puro, TypeScript es compatible con cualquier entorno que soporte JavaScript.
- Herramientas modernas: TypeScript ofrece herramientas avanzadas de depuración y autocompletado, lo que aceleró el desarrollo en Angular.

Librería de Mat Icons

Se implementó esta librería para integrar iconos modernos y visualmente atractivos en el frontend, mejorando la experiencia y la interfaz de usuario.

3.1.3 Herramientas de desarrollo.

Visual Studio Code

IDE principal utilizado para el desarrollo, facilitando la edición de código y el uso de extensiones que optimizan el flujo de trabajo.

SourceTree

Herramienta de interfaz gráfica para Git, utilizada para facilitar la gestión de control de versiones y el trabajo colaborativo a través de GitHub.

GitHub

GitHub fue utilizado como plataforma de control de versiones para gestionar el código del proyecto de manera colaborativa. El uso de GitHub permitió:

- Control de versiones: Se mantuvo un historial detallado de los cambios en el código a lo largo del desarrollo.
- Colaboración: La gestión de ramas facilitó el trabajo colaborativo y la integración continua entre el frontend y el backend.

XAMPP

Utilizado para gestionar el entorno de desarrollo local, ejecutando servicios de Apache y MySQL para probar y desarrollar el backend.

3.4 IMPLEMENTACIÓN DE LA METODOLOGÍA DEL PROYECTO

Existen diversas metodologías para el desarrollo de este tipo de proyecto, pero se usó la metodología Scrum, ya que nos permite gestionar el proceso de desarrollo de manera ágil y flexible, asegurando la entrega continua de valor en cada iteración. Por lo que favorece al equipo en la entrega de tareas a realizar en cada momento específico siendo una forma más eficiente en el desarrollo del proyecto.

Por ello se realizaron reuniones diarias de seguimiento, conocidas como "daylis", donde cada miembro del equipo compartía el progreso realizado, los obstáculos encontrados y las tareas a realizar. También algunas reuniones se realizaban por determinado tiempo para la verificación de la funcionalidad del proyecto las cuales se conocen como "demo", donde el cliente verifica el avance funcional del proyecto y determina si lo realizado cumple con las historias de usuario que se realizaron en el "sprint" que fue otorgado a cada miembro del equipo y también da algunas recomendaciones para mejorar el sistema o cambie de opinión en un apartado en específico.

A continuación, se presenta el equipo del proyecto:

Rol	Descripción	Miembro
Product Owner	Coordinador del proyecto	Dr. Irene Alvarado Van der Laat
Scrum Máster	Responsable del proyecto	DR. Gonzalo Miguel Quetz Aguirre
Development Team	Desarrollador del proyecto	Br. Fernando Yahir Puc Euan Br. Fernanda Yazbeth Chable Cocon.

Tabla 1. Equipo del Proyecto.

Una vez presentado el equipo se realiza el análisis de requerimientos, donde se realizaron varias sesiones con el cliente del proyecto, los cuales sirvió para comprender las necesidades que desea cubrir con respecto a los usuarios finales, por ende, nos permitió establecer los requerimientos específicos para el sistema.

Las funcionalidades que requiere el sistema son los siguientes:

Gestión de Usuarios:

- Registro de usuarios con información básica.
- Autenticación y recuperación de contraseña.

Gestión de Cuestionarios:

- Asignación y configuración de cuestionarios por parte de los maestros.
- Respuestas y retroalimentación inmediata para los estudiantes.

Gestión de Grupos:

- Creación y gestión de grupos de estudiantes por parte de los maestros.
- Asignación de estudiantes a grupos específicos.
- Creación del código de acceso a los grupos por parte de los maestros.

Análisis y Reportes:

- Generación de informes detallados sobre los resultados de los cuestionarios.
- Visualización de datos a través de gráficos y estadísticas.

TIPO DE USUARIO	DESCRIPCIÓN
Estudiante	Es el usuario que ingresa al sistema para realizar los cuestionarios, que pertenece a grupos y visualiza sus resultados.
Maestro	Es el encargado de asignar cuestionarios, visualizar los resultados de sus alumnos, crear grupos, administrar a los alumnos y generar informes tanto grupales como individuales.
Administrador	Se encarga de gestionar usuarios, así como instituciones y carreras, categorías de preguntas, análisis de datos, editar y publicar los cuestionarios, editar la página principal de la plataforma, generar informes en base a los resultados de los alumnos tanto grupales como individuales.

Tabla 2. Tipos de usuarios.

3.5 REQUERIMIENTOS

En este apartado se detallaron los requerimientos del sistema propuesto, organizado por módulos y submódulos que se dividen en dos categorías las cuales son los requerimientos funcionales y no funcionales.

3.5.1 Requerimientos Funcionales

Los requerimientos funcionales son aquellos satisfacen las necesidades y expectativas del usuario. A continuación, se presentan los requerimientos funcionales organizados por módulos y submódulos:

ID	REQUERIMIENTOS FUNCIONALES
Módulo de Bienvenida	
R01	La página de inicio debe contener un menú donde se podrá acceder a las secciones Home, Quiénes somos, Contacto, Cuestionario, Registro e Inicio de sesión.
R02	El botón "Home" debe mostrar información relacionada con el cuestionario que se va a realizar, explicando los detalles del proceso, así como el objetivo del mismo.
R03	El botón "Quiénes somos" muestra la misión, visión y una breve historia del proyecto.
R04	El botón "Contacto" muestra el nombre de la institución, dirección, número de teléfono, sitio web, y un mapa con las universidades que participan en el proyecto.
R05	El botón "Cuestionario" muestra un mensaje de advertencia, indicando que el cuestionario solo se puede acceder después de iniciar sesión.
R06	El botón "Registro" dirige al usuario al formulario para dar paso al registro de este a la plataforma.
R07	El botón "Inicio de sesión" debe permitir a los usuarios registrados ingresar usando su correo y contraseña.
Módulo de Inicio de Sesión	
R08	Poder acceder con correo electrónico, matrícula y contraseña.
R09	Registro de nuevo usuario proporcionando información básica (Nombre, Apellidos, Edad, Genero, Institución, Matrícula, Tipo de usuario, carrera, País, correo electrónico, contraseña).
R10	Recuperación de contraseña mediante un enlace enviado por correo electrónico.
Módulo de Perfil de Usuario	
Submódulo Perfil de Estudiante:	
R11	Los estudiantes podrán registrarse y acceder a la plataforma utilizando su correo o matrícula y contraseña.
R12	Visualización de cuestionarios asignados por los profesores.
R13	Responder cuestionarios desde cualquier dispositivo con acceso a internet.
R14	Recepción de retroalimentación personalizada sobre sus competencias emprendedoras.

R15	Visualización del progreso individual en el transcurso de la carrera con base en los resultados de los cuestionarios.
R09	Consultar y editar sus datos del perfil de usuario.
Submódulo Perfil de Maestro	
R16	Asignación de cuestionarios a grupos específicos de alumnos, con filtros por carrera, grado o sección.
R17	Establecimiento de fechas de inicio y fin para los cuestionarios.
R18	Visualización de las respuestas de los alumnos, tanto individualmente como de forma grupal.
R19	Generación de informes detallados de resultados, incluyendo cualidades y retroalimentación por sección.
R20	Exportación de informes en formato PDF o CSV.
R21	Generación de códigos de acceso únicos para cada grupo.
R22	Consultar y editar sus datos del perfil de usuario.
Submódulo Perfil Administrador	
R23	Los administradores podrán crear, editar y eliminar usuarios de la plataforma (alumnos, maestros y administradores).
R24	Gestión de categorías y variables de preguntas.
R25	Visualización de los resultados de todos los cuestionarios de la plataforma.
R26	Generación de informes detallados, con gráficos y tablas, sobre cualidades individuales y grupales por carrera, grado y sección.
R27	Exportación de informes en formato PDF o CSV.
R28	Consulta de los datos de perfil de todos los usuarios.
R29	Seguimiento del uso de la plataforma.
R30	Provisión de retroalimentación personalizada a los alumnos según sus respuestas.
R31	Almacenamiento de respuestas y retroalimentación para seguimiento del progreso individual y grupal en el transcurso de la carrera.
Módulo de Gestión de Cuestionarios	
Submódulo Edición de Cuestionarios	
R32	Edición de preguntas y variables dentro de los cuestionarios, con la posibilidad de definir influencias positivas o negativas en los resultados por parte de los administradores.
Submódulo Configuración del Cuestionario	
R33	Configuración del cuestionario para un grupo específico y establecimiento de fecha límite para completarlo.
Submódulo Asignación de Cuestionarios:	
R34	Asignación de cuestionarios a estudiantes individuales o grupos completos por los profesores.
Submódulo Código de Acceso	
R35	Generación de códigos de acceso a los estudiantes para completar el cuestionario, garantizando que solo los estudiantes asignados puedan acceder.
Submódulo Respuestas de los Estudiantes	
R36	Los estudiantes podrán responder los cuestionarios en línea.

Submódulo Retroalimentación Inmediata	
R37	Retroalimentación inmediata a los estudiantes tras enviar el cuestionario, incluyendo cualidades positivas y negativas y recomendaciones para mejorar.
Submódulo Eliminación de Cuestionarios Completados	
R38	Los cuestionarios completados desaparecerán de la vista del estudiante.
Submódulo Cálculo de Resultados	
R39	Las preguntas tendrán una fórmula que se calcula a partir de la escala proporcionada por el administrador.
Submódulo Preguntas de Corrección	
R40	Las preguntas de corrección contarán con una fórmula independiente para aumentar la precisión del cuestionario.
Submódulo Aleatorización de Preguntas	
R41	Las preguntas podrán aleatorizarse y cambiar su sentido (positivo o negativo) en cada cuestionario.
Módulo de Gestión de Grupos	
Submódulo Creación de Grupos	
R42	Creación y gestión de grupos de estudiantes según carrera y grado por parte de los maestros.
Submódulo Asignación a Grupos	
R43	Asignación de estudiantes a grupos específicos mediante un código de acceso proporcionado por el maestro.
Módulo de Dashboard	
Submódulo Dashboard para Estudiantes	
R44	Visualización de todos los cuestionarios completados por el estudiante y su progreso académico.
Submódulo Dashboard para Maestros	
R45	Listado de cuestionarios y grupos gestionados por el maestro, con visualización de resultados individuales y grupales.
Submódulo Dashboard para Administrativos	
R46	Acceso a resultados y estadísticas de todos los cuestionarios aplicados, así como detalles de usuarios y administración.
Módulo de Análisis y Reportes	
Submódulo Gráficas de Resultados	
R47	Visualización gráfica de los resultados de los cuestionarios.
Submódulo Análisis de Datos	
R48	Análisis de datos obtenidos de los alumnos en base a sus características empresariales y capacidades.
Submódulo Generación de Informes	
R49	Generación automática de informes detallados sobre los resultados de los cuestionarios.
Submódulo Exportación a XML	
R50	Exportación de la lista de resultados y alumnos en formato XML.
Submódulo Exportación a PDF	

R51	Exportación de gráficas de resultados de forma individual o grupal en formato PDF.
-----	--

Tabla 3.Requerimientos funcionales.

3.5.2 REQUERIMIENTOS NO FUNCIONALES

Los requerimientos no funcionales son aquellos que nos garantizan que el sistema funcione de manera eficiente, segura y con una buena experiencia de usuario. A continuación, se presenta de igual manera los requerimientos no funcionales:

ID	Requerimientos No Funcionales
RN01	La aplicación debe ser compatible con navegadores modernos y estar actualizada con las últimas versiones de Angular y Laravel.
RN02	La aplicación debe ser completamente responsiva, funcionando de manera adecuada tanto en dispositivos móviles como en escritorios.
RN03	Los mensajes de error presentados a los usuarios deben ser claros, sencillos y fáciles de entender.
RN04	Se debe ofrecer un diseño de interfaz amigable, accesible y fácil de usar para todos los usuarios, independientemente de su nivel técnico.
RN05	El rendimiento de la aplicación debe ser óptimo, con tiempos de carga reducidos y una experiencia de uso fluida para todos los usuarios.
RN06	La seguridad de los datos será una prioridad, con medidas como el cifrado de contraseñas y protección contra ataques comunes (por ejemplo, inyecciones SQL, CSRF, XSS).
RN07	MySQL será utilizado como el sistema de gestión de bases de datos para almacenar la información de manera eficiente y segura.
RN08	La infraestructura de servidor debe ser adecuada para garantizar la alta disponibilidad del sistema y su escalabilidad ante un aumento de usuarios o datos.
RN09	La metodología de desarrollo de software será Scrum, lo que permitirá adaptabilidad y mejora continua durante el proceso de desarrollo.
RN10	El lenguaje de programación a utilizar será PHP para el backend con Laravel, y HTML5, TypeScript para el frontend con Angular.
RN11	La aplicación debe ser fácil de mantener y actualizar, asegurando que el código esté bien estructurado y documentado para futuras modificaciones.
RN12	Se requiere una conexión a Internet para acceder a la plataforma y realizar todas las funciones necesarias dentro del sistema.

Tabla 4. Requerimientos no funcionales

3.5 CASOS DE USO.

3.5.1 Diagrama caso de uso para iniciar sesión.

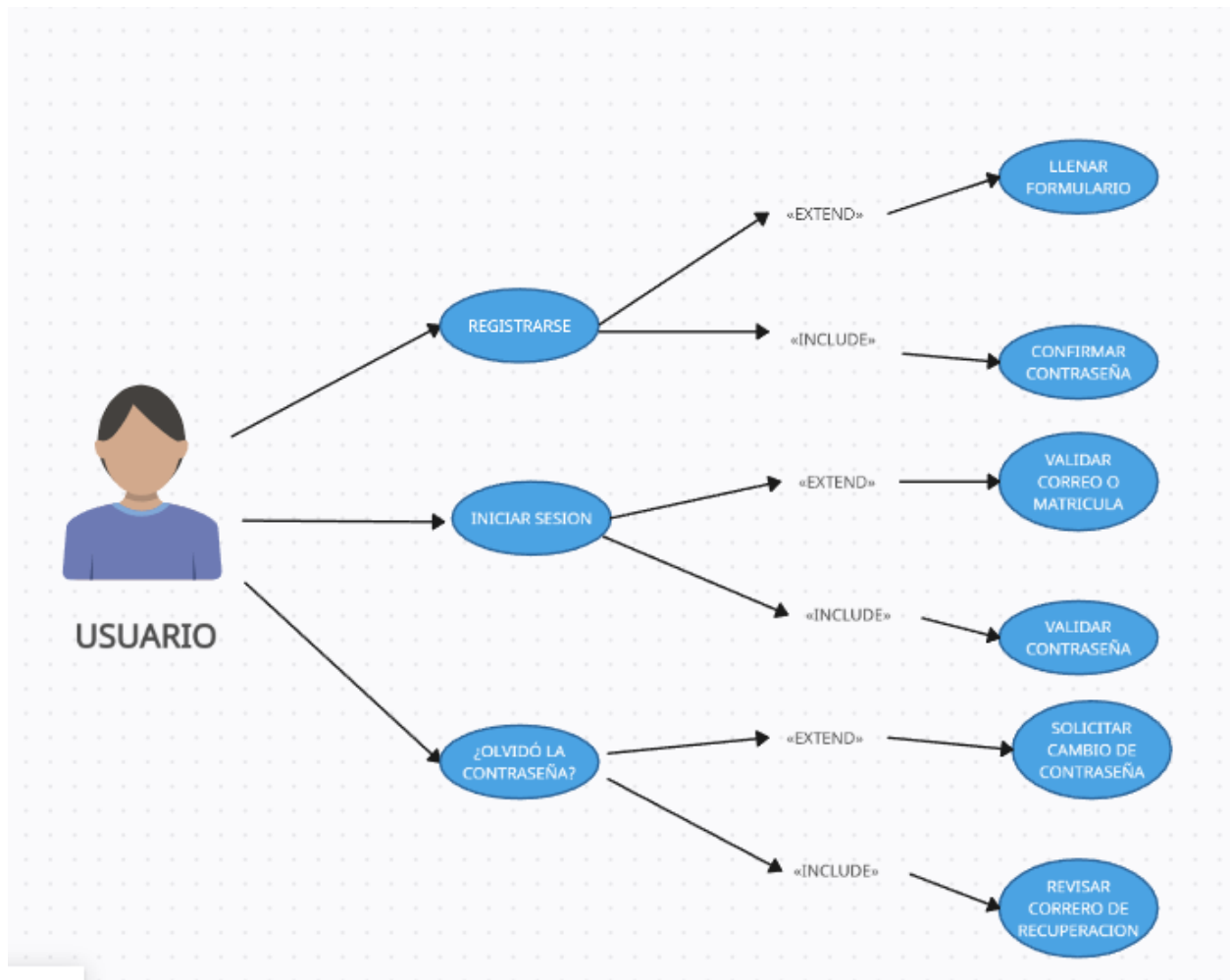


Ilustración 1 .Diagrama de uso para inicio sesión.

3.5.2 Diagrama caso de uso para el estudiante.

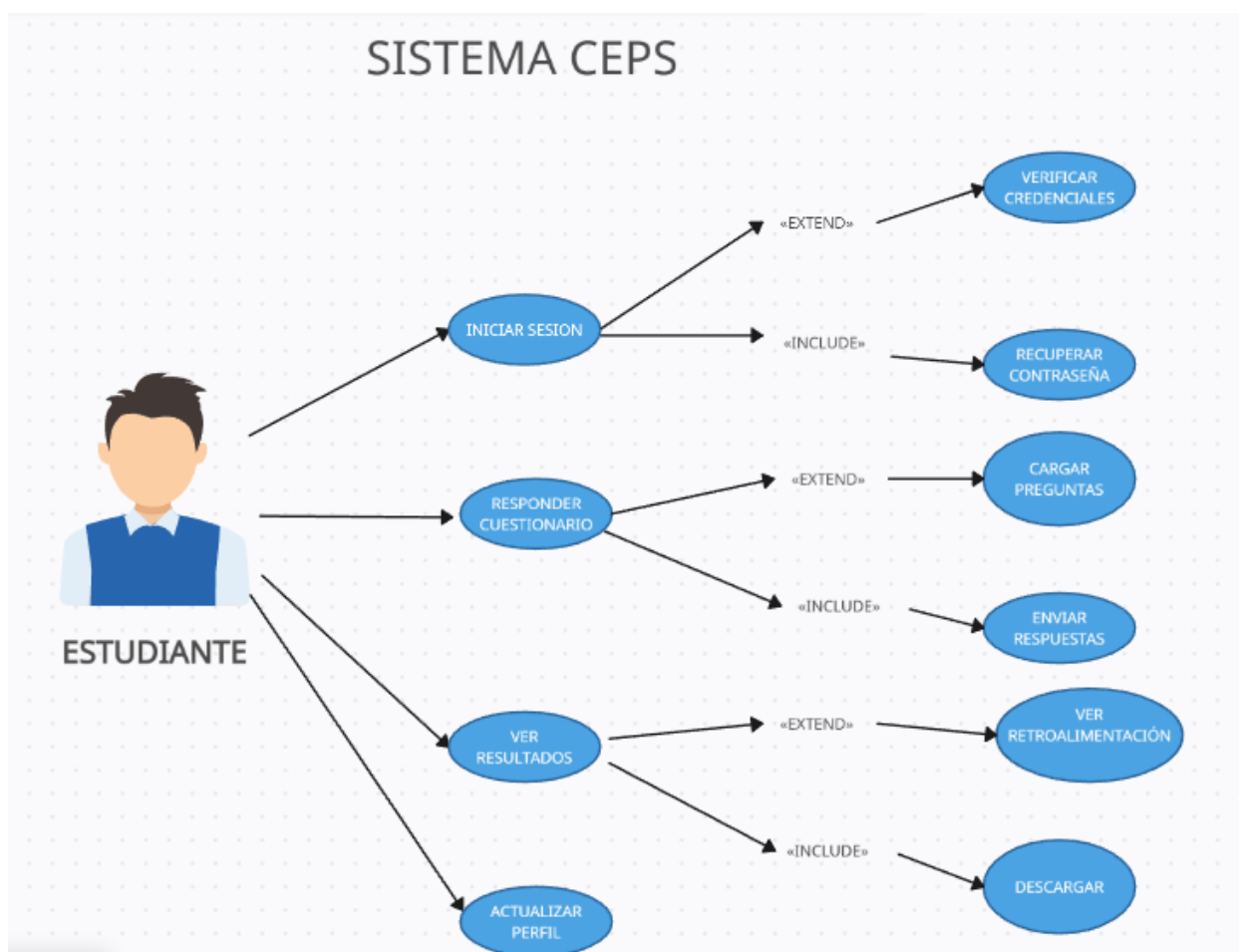


Ilustración 2. Diagrama de uso para el estudiante

3.5.3 Diagrama caso de uso para el maestro.

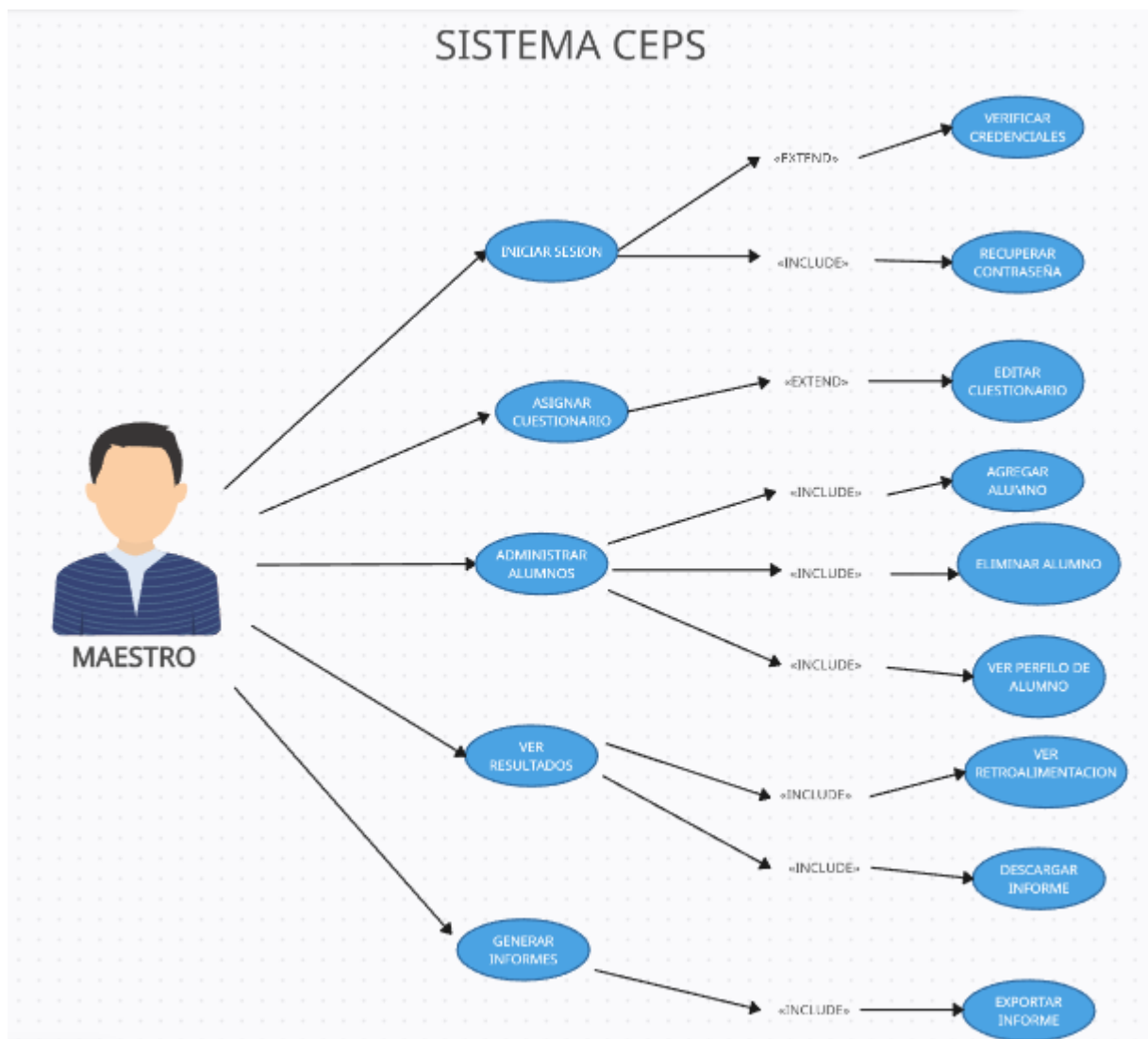


Ilustración 3. Diagrama de uso para el maestro

3.5.4 diagrama caso de uso para el administrador

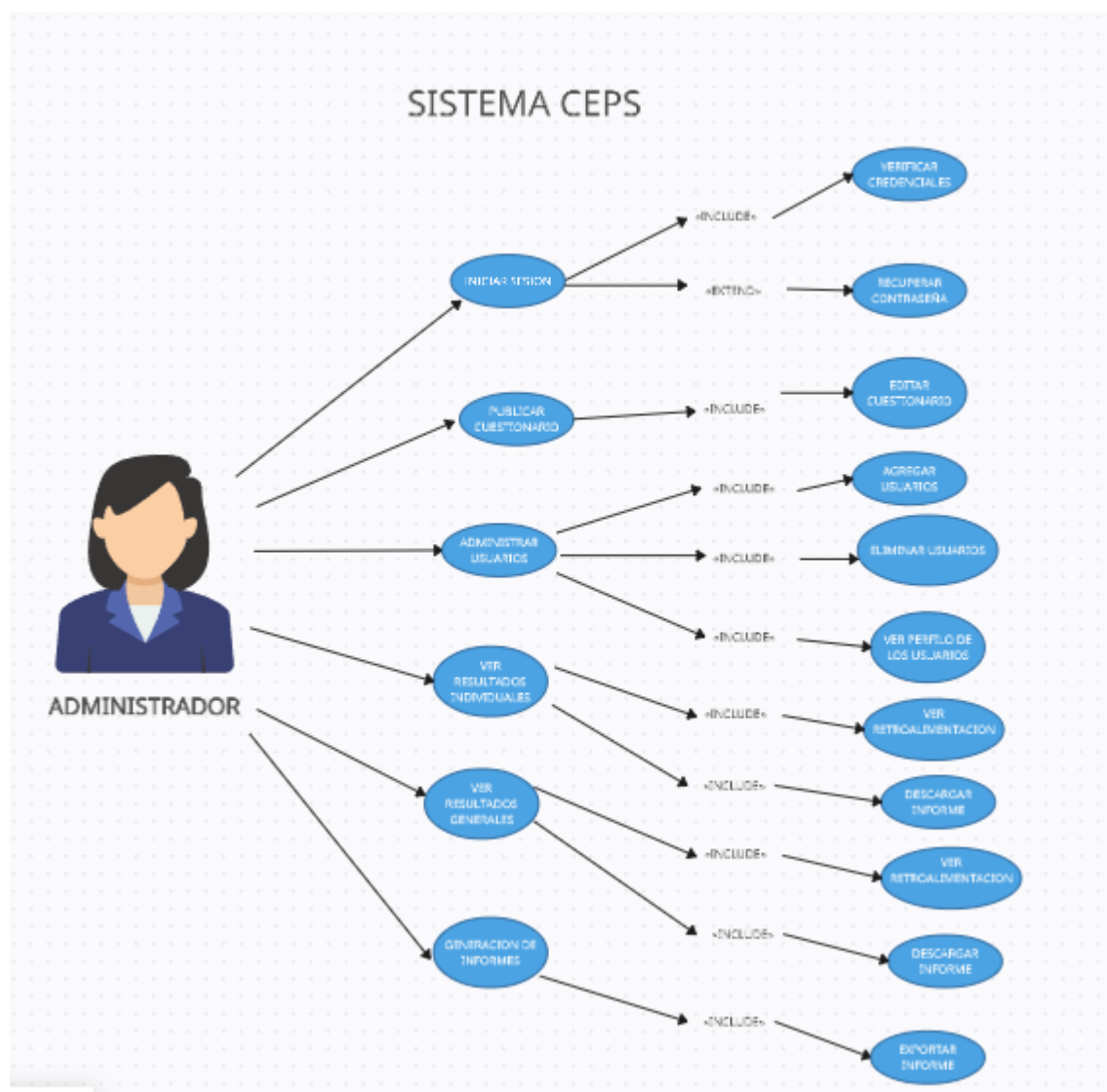


Ilustración 4. Diagrama de uso para el administrador

3.5.5 Modelo relacional

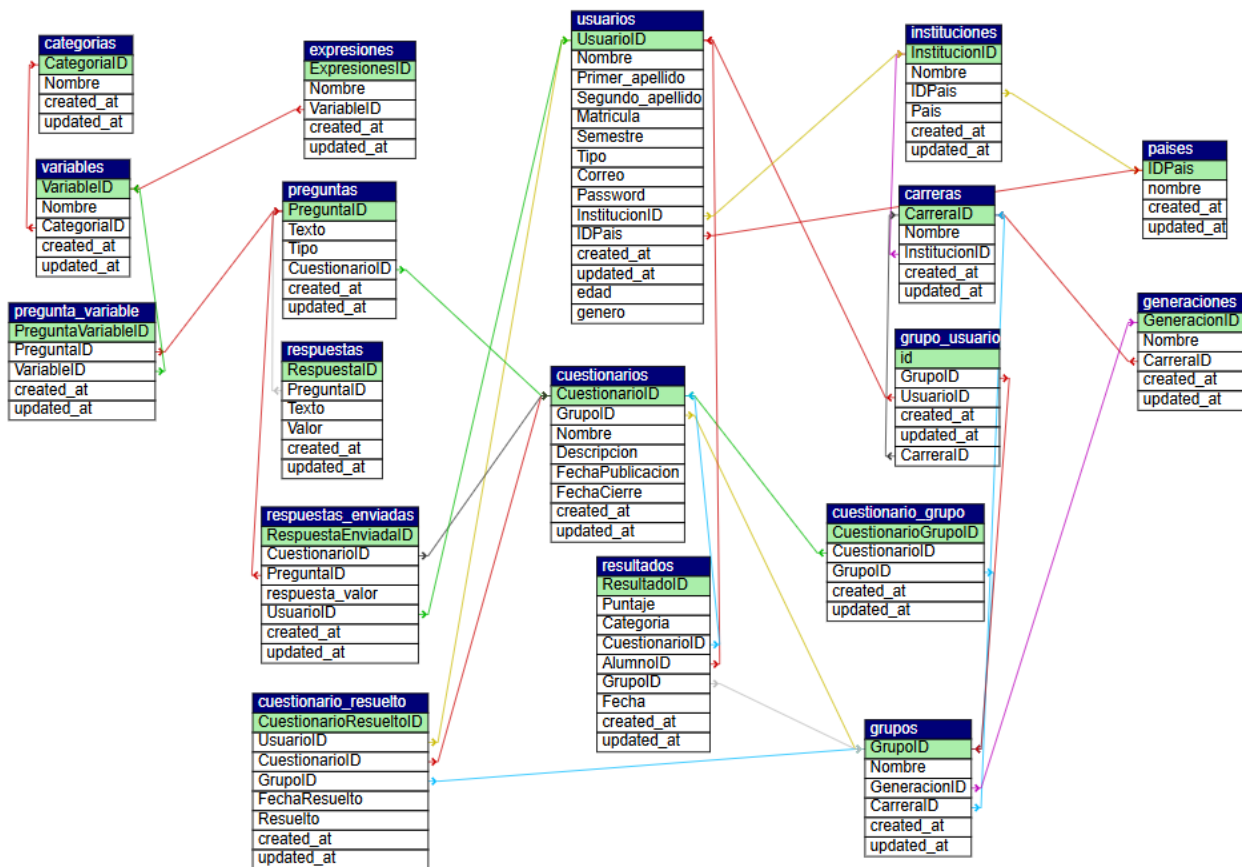


Ilustración 5. Modelo Relacional

3.6 DISEÑO PRELIMINAR DEL SISTEMA

3.6.1 Módulo de Registro

Este módulo proporciona a los usuarios registrarse a la plataforma. Al acceder al formulario de registro, el usuario verá los siguientes campos obligatorios:

- **Nombre:** Primer nombre del usuario.
- **Primer Apellido:** Primer apellido del usuario.
- **Segundo Apellido:** Segundo apellido del usuario.
- **Matrícula:** Número de identificación institucional para usuarios vinculados a universidades.
- **Semestre:** Semestre académico actual del usuario.
- **Institución:** Campo desplegable para seleccionar la institución a la que el usuario pertenece, basado en las instituciones participantes del proyecto.
- **País:** Selección del país de origen o residencia del usuario para una referencia geográfica.
- **Carrera:** Campo desplegable para seleccionar la carrera o programa académico del usuario.
- **Edad:** Campo numérico para ingresar la edad del usuario.
- **Género:** Selección de género con opciones estándar (e.g., femenino, masculino, otro).
- **Correo Electrónico:** Dirección de correo electrónico que servirá como nombre de usuario para el inicio de sesión y la recuperación de cuenta.
- **Contraseña:** Campo protegido para ingresar la contraseña, la cual debe cumplir con los requisitos de seguridad mínimos (e.g., longitud mínima, combinación de caracteres).
- **Confirmar Contraseña:** Campo adicional para volver a ingresar la contraseña, asegurando que coincide con la original.

TEAM ALPHA

INICIO

SOBRE NOSOTROS

CONTACTO

CUESTIONARIO CEP'S

Sign up

Login

Regístrate

Nombre:

Apellido Paterno:

Apellido Materno:

Institución:

Matrícula:

País:

Correo Electronico:

Contraseña:

Confirmar contraseña:

Regresar

Registrarse

Ilustración 6. Diseño del registro del alumno

3.6.2. Módulo de Inicio de sesión

Este módulo proporciona acceso seguro a la plataforma mediante un sistema de autenticación. Sus funciones incluyen:

- Acceso para usuarios registrados utilizando correo electrónico, matrícula y contraseña.
- **Registro de nuevos usuarios:** A través de un formulario que solicita datos básicos como nombre, apellidos, edad, género, institución, matrícula, tipo de usuario, carrera, país, correo electrónico y contraseña.
- **Recuperación de contraseña:** Ofrece la opción de restablecer la contraseña mediante un enlace enviado al correo electrónico registrado.
- **Redireccionamiento de Dashboard:** Redirige al usuario a su respectivo Dashboard dependiendo del tipo de usuario: Alumno, Maestro y Administrador.



Ilustración 7. Diseño del login de usuarios

3.6.3 Módulo de bienvenida

El módulo de bienvenida es la primera interacción del usuario con la plataforma. Incluye un diseño visualmente atractivo que refleja la identidad del sistema, con colores distintivos, el logo de la aplicación y un menú principal accesible desde la página de inicio. Este menú contiene las siguientes Submódulos:

- Home: Presenta una introducción al cuestionario, explicando su propósito, los detalles del proceso, y su importancia para los usuarios.
- Quiénes somos: Ofrece una descripción detallada de la misión, visión y breve historia del proyecto, destacando la colaboración con las instituciones participantes.
- Contacto: Proporciona información relevante sobre la institución, como dirección, número de teléfono, sitio web, y un mapa interactivo que muestra las universidades asociadas.
- Cuestionario: Muestra un mensaje de advertencia que informa al usuario que debe registrarse e iniciar sesión para acceder al cuestionario.
- Registro: Redirige al formulario de creación de cuenta, donde los nuevos usuarios pueden registrarse proporcionando su información básica.
- Inicio de sesión: Permite a los usuarios registrados acceder a la plataforma ingresando su correo electrónico o matrícula junto con su contraseña.



Ilustración 8. Diseño de la página principal



Ilustración 9. Diseño carrusel de imágenes



Ilustración 10. Diseño objetivo de la página



Ilustración 11. Diseño Aclaración

En la parte final del inicio contiene un pie de página el cual contiene información sobre el proyecto, los créditos, las instituciones que participan en el proyecto, las redes sociales y contacto. Las instituciones participantes y redes sociales ya contienen el direccionamiento a sus páginas oficiales.

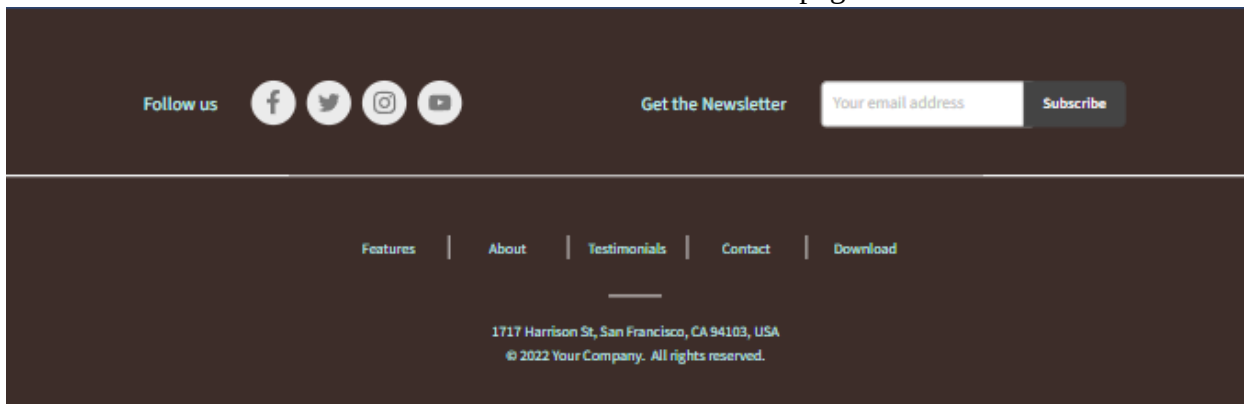


Ilustración 12. Diseño pie de pagina

Submódulo Quienes Somos



Ilustración 13. Diseño ¿quiénes somos? (Historia)



Ilustración 14. Diseño ¿quiénes somos? (Misión y Visión)



Ilustración 15. Diseño te presento al equipo

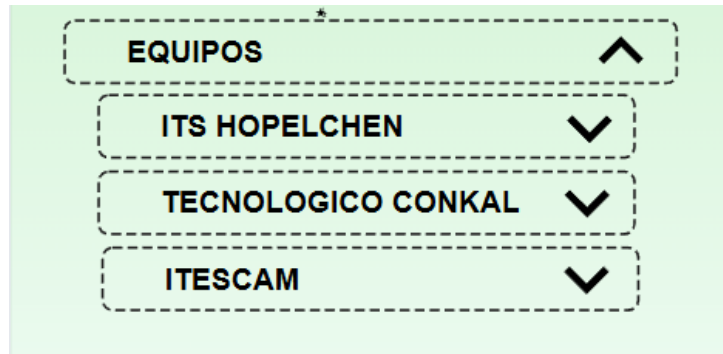


Ilustración 16. Diseño de presentación al equipo (Instituciones)

Submódulo Contacto



Ilustración 17. Diseño de localización de las instituciones



Ilustración 18. Diseño de localización de las instituciones



Ilustración 19. Diseño localización de las instituciones

Submódulo Cuestionario

Este apartado no permitirá al usuario ingresar al cuestionario hasta que se haya registrado e iniciado sesión en la plataforma.



Ilustración 20. Diseño cuestionario

3.6.4 Módulo de perfil de usuario

Este módulo permite la gestión personalizada de perfiles según el tipo de usuario: estudiante, maestro o administrador.

Submódulo Perfil de Estudiante

Este submódulo permite al estudiante administrar sus datos y acceder a grupos y cuestionarios asignados por los maestros. Las funcionalidades que incluye son:

- Registro y acceso mediante correo electrónico o matrícula y contraseña.
- Visualización de cuestionarios asignados.
- Respuesta a cuestionarios desde cualquier dispositivo con acceso a internet.
- Recepción de retroalimentación personalizada sobre competencias emprendedoras.
- Monitoreo del progreso académico basado en los resultados de los cuestionarios.
- Consulta y edición de datos del perfil de usuario.

Al darle iniciar sesión, redirige al Dashboard del alumno, el cual mostrará el nombre y correo del alumno, así como el día, el menú y los apartados a los que pueda ingresar.

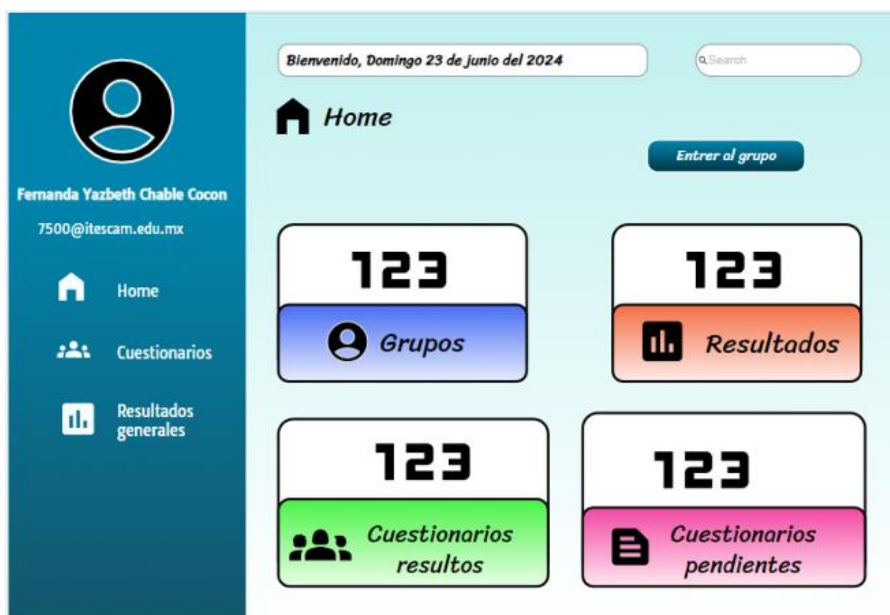


Ilustración 21. Diseño Dashboard del alumno

Al darle clic al botón entrar al grupo, se desplegará una ventana para buscar el grupo mediante el código que proporcionará el maestro. Al dar clic en buscar deberá aparecer la información del grupo, donde el alumno al verificar la información se una al grupo o al cancelar cierre la ventana.



Ilustración 22. Diseño búsqueda de grupo

Grupos

En el apartado de grupos, el alumno podrá visualizar sus respectivos grupos o bien, podrá unirse a otro.



Ilustración 23. Diseño visualización de grupos

Al entrar al grupo el alumno visualizará los cuestionarios resueltos.



Ilustración 24. Diseño de visualización de cuestionarios por grupo

Visualización del perfil del alumno

El alumno deberá tener un apartado donde visualice los datos de su perfil, así como tener una opción para editarlo.

The image shows a user profile interface. On the left is a dark blue sidebar with a white user icon, the name 'Fernanda Yazbeth Chable Cocon', the email '7500@itescam.edu.mx', and three navigation links: 'Home' (house icon), 'Grupos' (people icon), and 'Cuestionarios resueltos' (bar chart icon). The main area has a light blue header with the title 'PERFIL' and an 'Editar perfil' button. Below this is a white profile card containing the same user icon and name. To the right of the card is a list of fields with their corresponding values in input boxes: 'Correo Electronico' (7500@itescam.edu.mx), 'Matricula' (7500), 'Contraseña' (masked with asterisks), 'Carrera' (Ingeniería en sistemas computacion), 'Institucion' (Instituto Tecnológico Superior de Ca), 'Semestre' (Octavo), and 'Grupo' ('A').

Correo Electronico	7500@itescam.edu.mx
Matricula	7500
Contraseña	*****
Carrera	Ingeniería en sistemas computacion
Institucion	Instituto Tecnológico Superior de Ca
Semestre	Octavo
Grupo	"A"

Ilustración 25. Diseño visualización de información del perfil

Cuestionarios resueltos del alumno

En este apartado el alumno podrá visualizar todos sus cuestionarios resueltos.

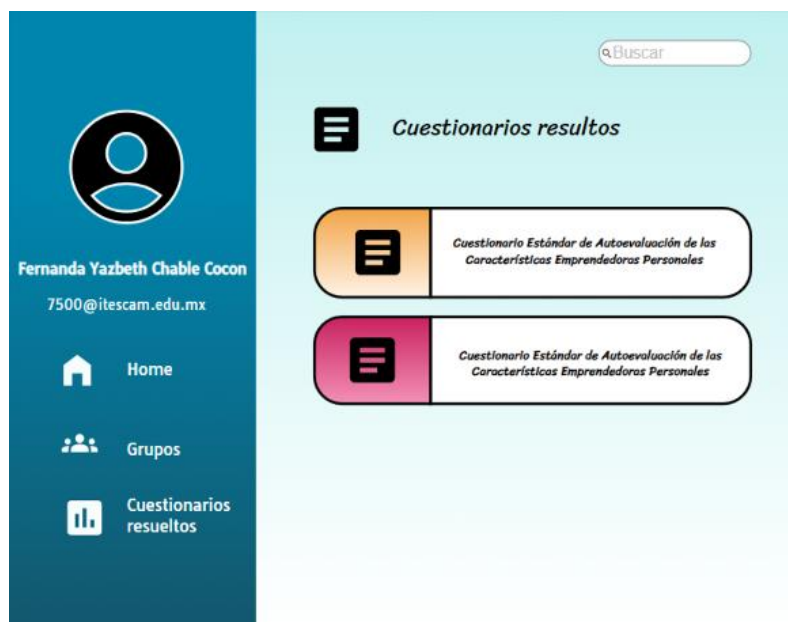


Ilustración 26. Diseño visualización cuestionarios resueltos.

Resultados del cuestionario

En este apartado se deberá ver los resultados del cuestionario, mostrando los resultados en una gráfica y una retroalimentación.

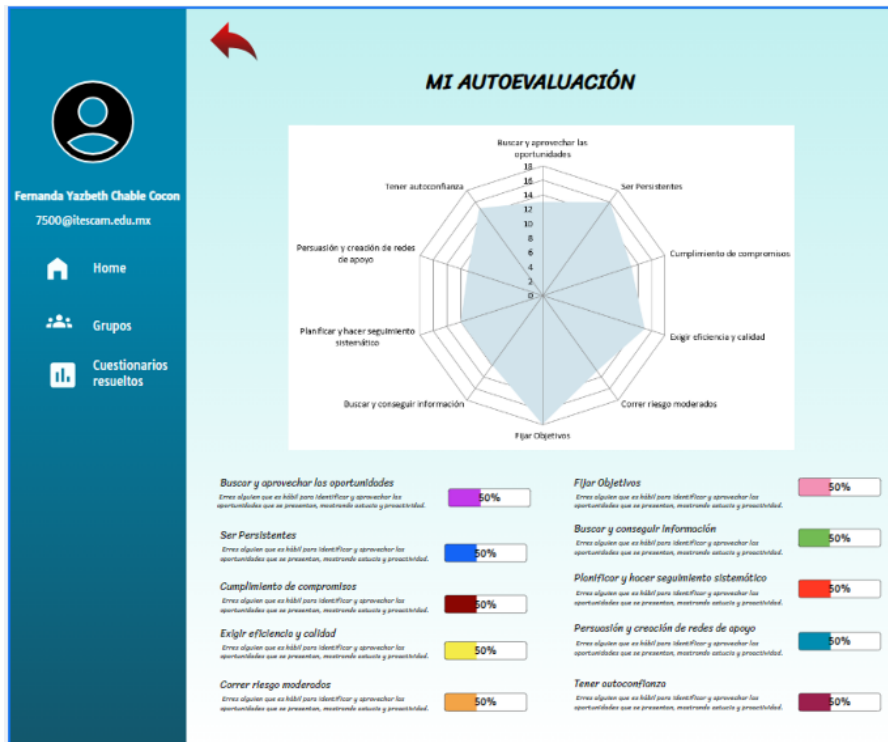


Ilustración 27. Diseño visualización resultados del alumno

Submódulo perfil de Maestro

Este submódulo permite a los maestros gestionar los cuestionarios asignados a los alumnos y permite acceder a los resultados individuales y grupales, de igual manera incluye las siguientes funcionalidades: Asignación de cuestionarios a grupos específicos utilizando filtros por carrera, grado o sección.

- Establecimiento de fechas de inicio y fin para los cuestionarios.
- Visualización de respuestas individuales y grupales de los estudiantes.
- Generación de informes detallados en formato PDF o CSV.
- Gestión de códigos de acceso únicos para grupos.
- Consulta y edición de los datos del perfil de usuario.

El maestro al iniciar sesión será dirigido a su Dashboard, el cual contiene su información del perfil y las acciones que podrá realizar.

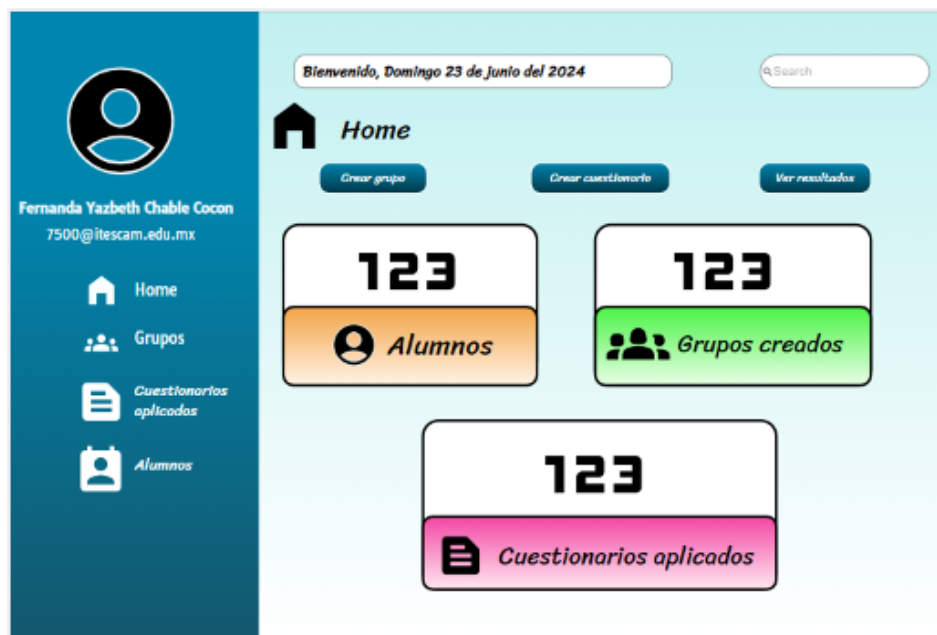


Ilustración 28. Diseño visualización del Dashboard

El maestro podrá crear nuevos grupos al darle clic al botón “Crear grupo”, el cual desplegará un formulario para ingresar los datos del grupo.

The image shows a web application interface for creating a group. On the left is a dark blue sidebar with a user profile and navigation menu. The main area on the right is light blue and contains a form titled 'Crear grupo' with several input fields and two buttons at the bottom.

Sidebar (Left):

- Profile icon: A circle with a stylized person.
- Username: **Fernanda Yazbeth Chable Cocon**
- Email: **7500@itescam.edu.mx**
- Menu items:
 - Home (house icon)
 - Grupos (group of people icon)
 - Cuestionarios aplicados (document icon)
 - Alumnos (calendar icon)

Main Area (Right):

Crear grupo

Form fields:

- Nombre de la Institución:
- Carrera:
- Semestre:
- Grupo:
- Ciclo escolar:

Buttons:

- Cancelar (dark blue button)
- Guardar (dark blue button)

Ilustración 29. Diseño de la vista agregar grupo

Una vez creado el grupo aparecerá una ventana flotante que informa que el grupo se ha creado correctamente y nos proporciona el código de acceso al grupo, al darle en aceptar procederá a cerrar la ventana y redirigir al usuario a la lista de grupos.

Lista de grupos

En el apartado de grupos se visualizarán todos los grupos existentes a las que pertenece el maestro, podrá crear un grupo o ingresar a la información del grupo.

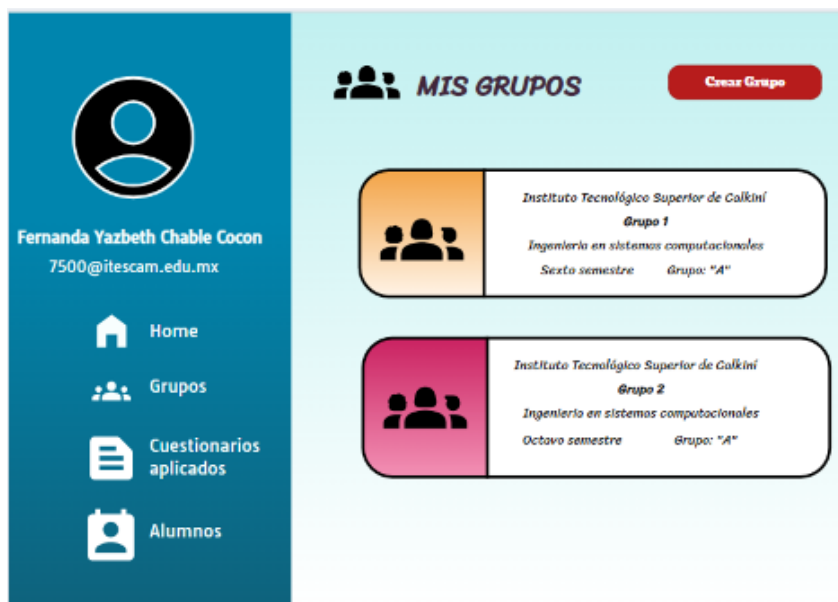


Ilustración 30. Diseño visualización de grupos

Al darle clic a un grupo se podrá visualizar los datos del grupo, los cuestionarios aplicados y de igual manera ver los integrantes del grupo.



Ilustración 31. Diseño visualización de cuestionarios

Al darle clic al botón de ver integrantes, deberá mostrar la lista de alumnos que pertenecen al grupo con sus respectivas acciones ya sea eliminar el alumno del grupo o ver sus resultados.



Ilustración 32. Diseño visualización de alumnos del grupo

Al darle clic al botón de buscar y añadir alumno, redirigirá a una ventana donde buscaremos al alumno, por su nombre, matricula o correo. Al darle clic a buscar mostrara los datos del alumno y la opción de añadirlo al grupo.



Ilustración 33. Diseño vista de agregar alumno al grupo

Visualización del perfil del maestro

El administrador deberá tener un apartado donde visualice los datos de su perfil, así como tener una opción para editarlo.

PERFIL [Editar perfil](#)

Fernanda Yazbeth Chable Cocon
7500@itescam.edu.mx

Home
Grupos
Cuestionarios resueltos

Correo Electronico: 7500@itescam.edu.mx
Matricula: 7500
Contraseña: *****
Carrera: Ingenieria en sistemas computacion
Institucion: Instituto Tecnológico Superior de Ca
Semestre: Octavo
Grupo: "A"

Ilustración 34. Diseño visualización de información del perfil

Gestión de Alumnos

En este apartado el maestro deberá visualizar todos los alumnos que forman parte de algún grupo la cual le pertenece al maestro.

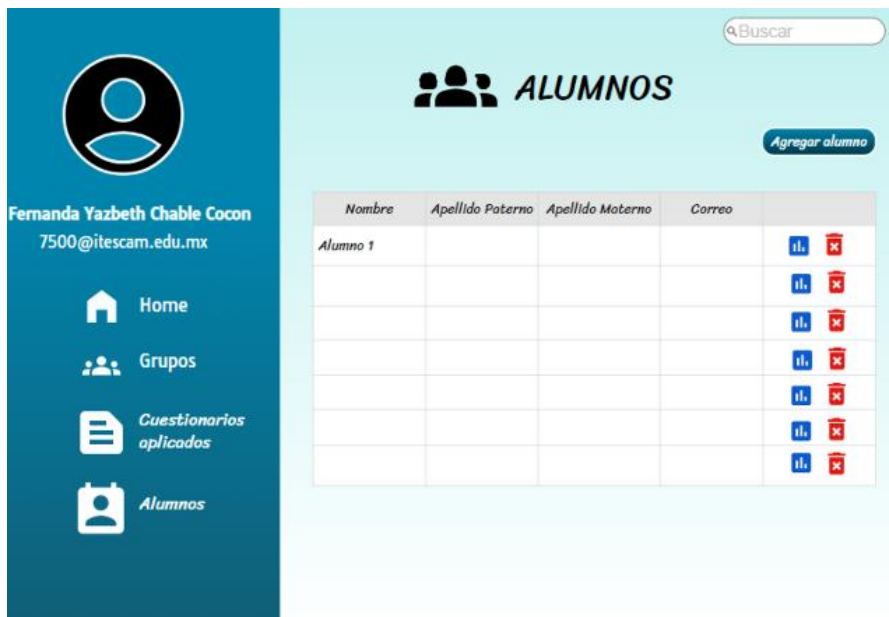


Ilustración 35. Diseño vista de gestión de alumnos

Visualización de resultados por grupo

En este apartado el maestro deberá visualizar los resultados generales del grupo, visualizando información del grupo, la gráfica de los resultados y el porcentaje de resultados que se obtuvieron.

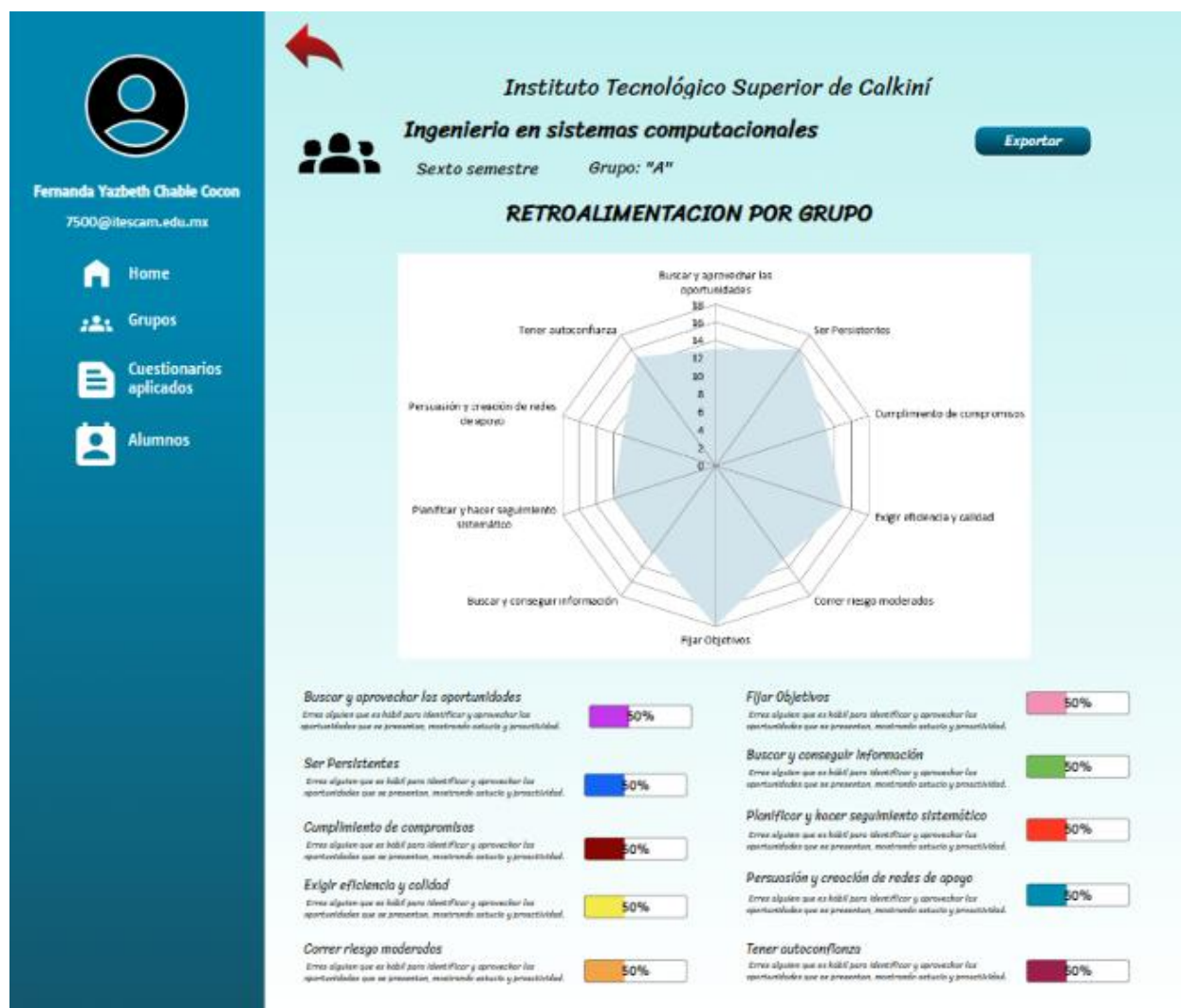


Ilustración 36. Diseño vista de resultados por grupo

Submódulo del administrador

Este submódulo permite a los administrativos el acceso completo para gestionar los usuarios y datos de la plataforma. Sus funcionalidades son:

- Gestión completa de usuarios (creación, edición y eliminación).
- Configuración y administración de categorías y variables de preguntas.
- Visualización de resultados globales de los cuestionarios.
- Generación y exportación de informes detallados en diversos formatos.
- Seguimiento del uso de la plataforma.
- Provisión de retroalimentación personalizada y almacenamiento de datos para el progreso a lo largo del tiempo.

Al iniciar sesión como administrador se redirigirá a su respectivo Dashboard, el cual contendrá su menú con las funciones que le corresponde, se podrá visualizar la cantidad de alumnos y maestros existentes, así como informes, instituciones y cuestionarios aplicados.

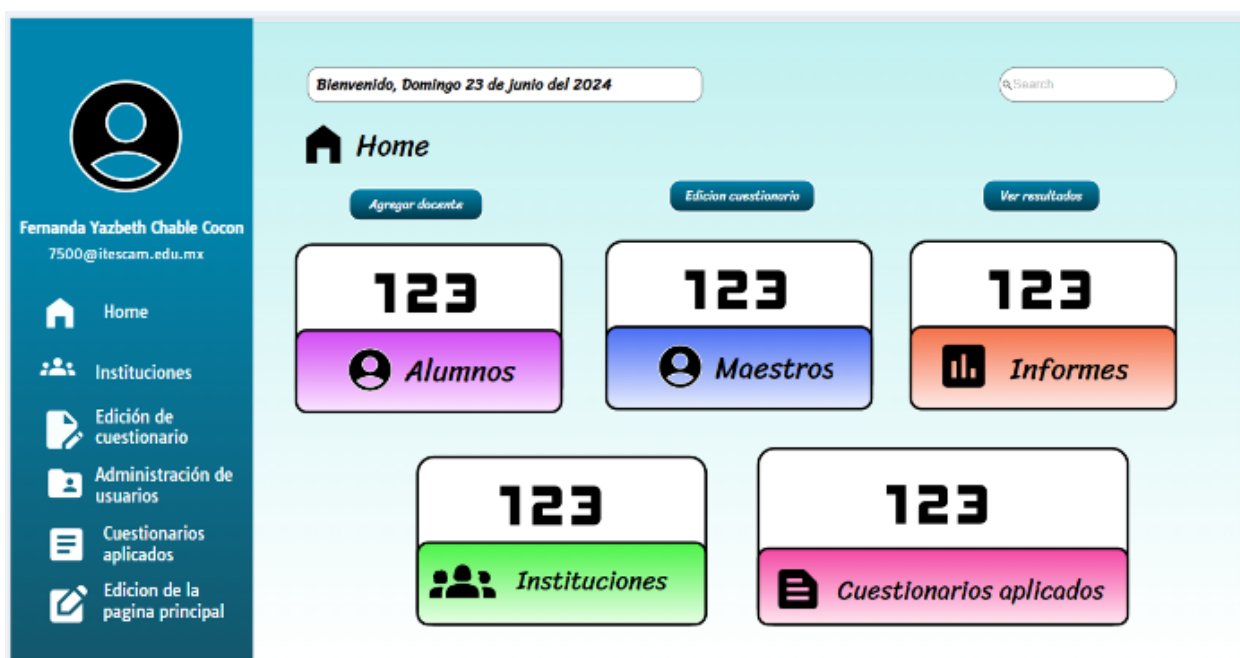


Ilustración 37. Diseño Dashboard del Administrador

Apartado de instituciones

Al ingresar al apartado de instituciones se visualizará todas las instituciones existentes por defecto, de igual manera se tiene un filtro para visualizar las instituciones que pertenezcan a un país en específico.

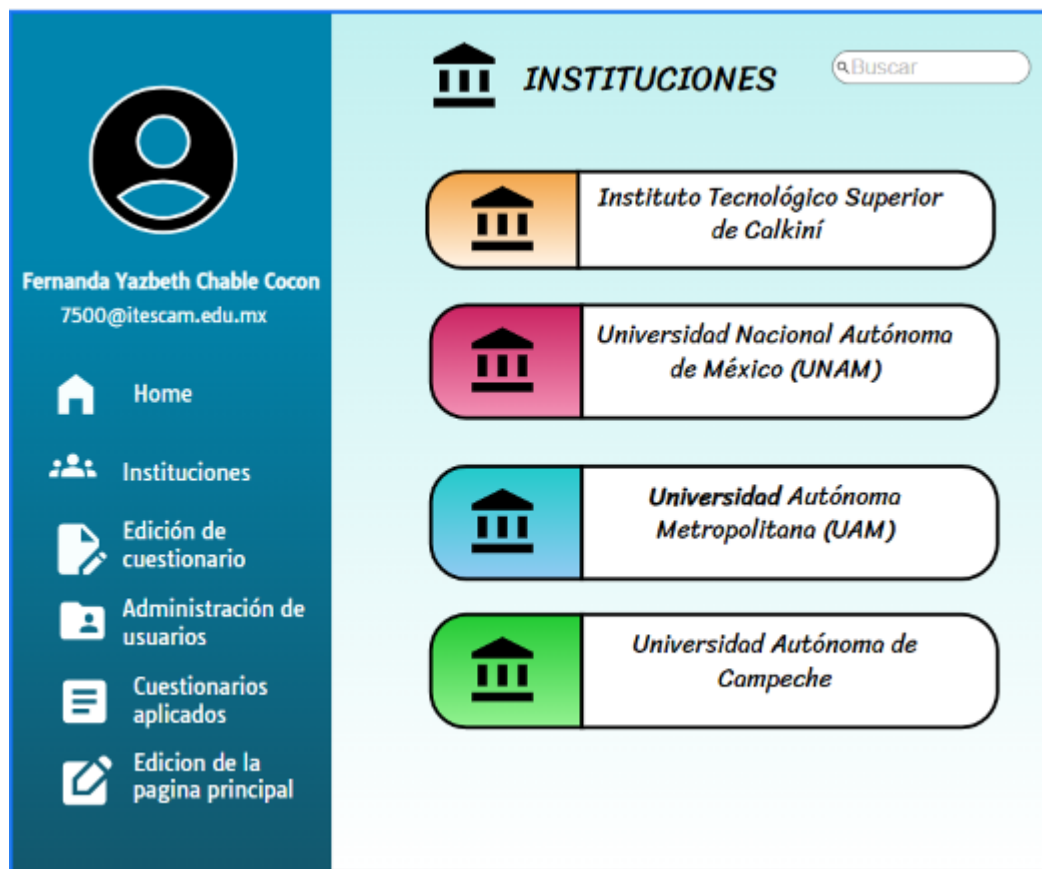


Ilustración 38. Diseño lista de instituciones

Administración de los usuarios

Al ingresar a la institución, redirigirá al administrador a la administración de usuarios de esa institución.



Ilustración 39. Diseño administración de usuarios por institución

Al ingresar a maestros se visualizará la lista de maestros que pertenecen a la institución con la opción de agregar a un nuevo maestro, así como visualizar los grupos que pertenecen al maestro o eliminarlas de la institución.



Ilustración 40. Diseño lista de docentes por institución

Al darle clic al botón “agregar docente” redirigirá al administrador al formulario para añadir un nuevo docente con el país e institución preestablecido.

Fernanda Yazbeth Chable Cocon
7500@itescam.edu.mx

- Home
- Instituciones
- Edición de cuestionario
- Administración de usuarios
- Cuestionarios aplicados
- Edición de la pagina principal

Registro docente

Nombres:

Apellido Paterno: Apellido Materno:

Institución:

Matrícula:

País:

Correo Electronico:

Contraseña:

Confirmar contraseña:

[Regresar](#) [Agregar docente](#)

Ilustración 41. Diseño vista registro de maestro

Luego redirigirá a la lista de maestros. Ahora en la lista de maestros al darle clic en el icono se podrán observar los alumnos que posee el maestro seleccionado



Ilustración 42. Diseño lista de grupos del docente

Ahora al darle clic en el menú de administración de usuarios en general, se podrá visualizar el menú para visualizar la lista de todos los usuarios del sistema.



Ilustración 43- Diseño administración de usuarios del sistema

Al ingresar a los maestros, se podrá visualizar la lista de todos los maestros.

The screenshot displays a web application interface for managing teachers. On the left is a dark blue sidebar with a user profile for 'Fernanda Yazbeth Chable Cocon' and a list of navigation options: Home, Instituciones, Edición de cuestionario, Administración de usuarios, Cuestionarios aplicados, and Edición de la pagina principal. The main content area has a light blue header with a search bar, a title 'ADMINISTRACION DE MAESTROS EN LA PLATAFORMA', and a 'DOCENTES' section with an 'Agregar docente' button. Below this is a table with columns for Nombre, Apellido Paterno, Apellido Materno, Carrera, Institución, and Correo. The table contains one row labeled 'Docente 1' and several empty rows. Each row has three action icons: a blue group of people, a green plus sign, and a red minus sign.

Nombre	Apellido Paterno	Apellido Materno	Carrera	Institución	Correo	
Docente 1						  
						  
						  
						  
						  
						  
						  

Ilustración 44. Diseño lista de docentes del sistema

El administrador podrá visualizar los resultados del grupo del maestro.

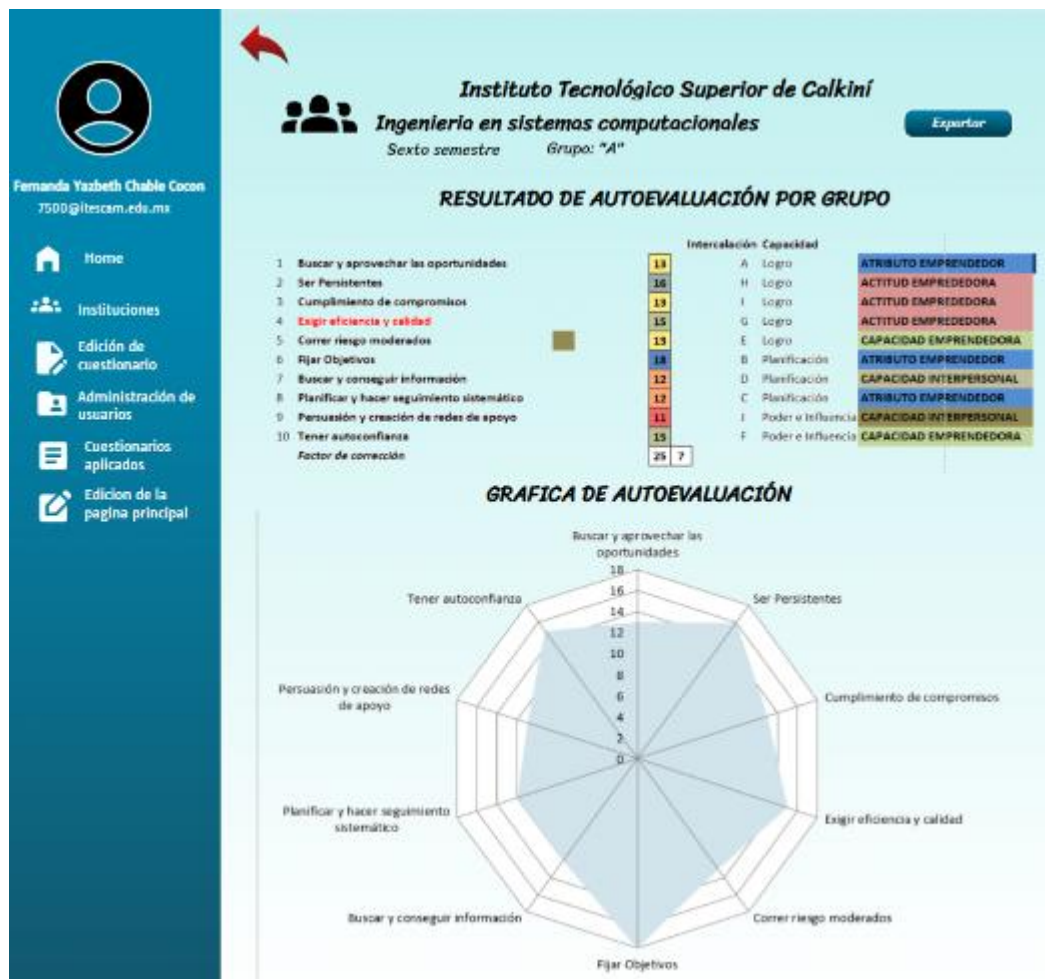


Ilustración 45. Diseño de la vista de resultados por grupo

De la misma manera con los alumnos se podrá visualizar la lista de todos los alumnos.

ADMINISTRACION DE ALUMNOS EN LA PLATAFORMA

ALUMNOS

Nombre	Apellido Paterno	Apellido Materno	Carrera	Institución	Correo	Semestre	
Alumno 1							 
							 
							 
							 
							 
							 
							 

Ilustración 46. Diseño lista de alumnos del sistema

3.6.5 Módulo de análisis y reportes

Este módulo tiene la finalidad de proporcionar análisis detallados sobre los resultados de los cuestionarios los cuales incluye los siguientes submódulos:

- Gráficas de Resultados: Visualización de resultados en formatos gráficos claros y comparativos.
- Análisis de Datos: Evaluación detallada basada en las características y competencias empresariales de los usuarios.
- Generación de Informes: Informes automatizados de cuestionarios con detalles clave.
- Exportación a XML y PDF: Generación de archivos exportables para facilitar la distribución y almacenamiento de datos.

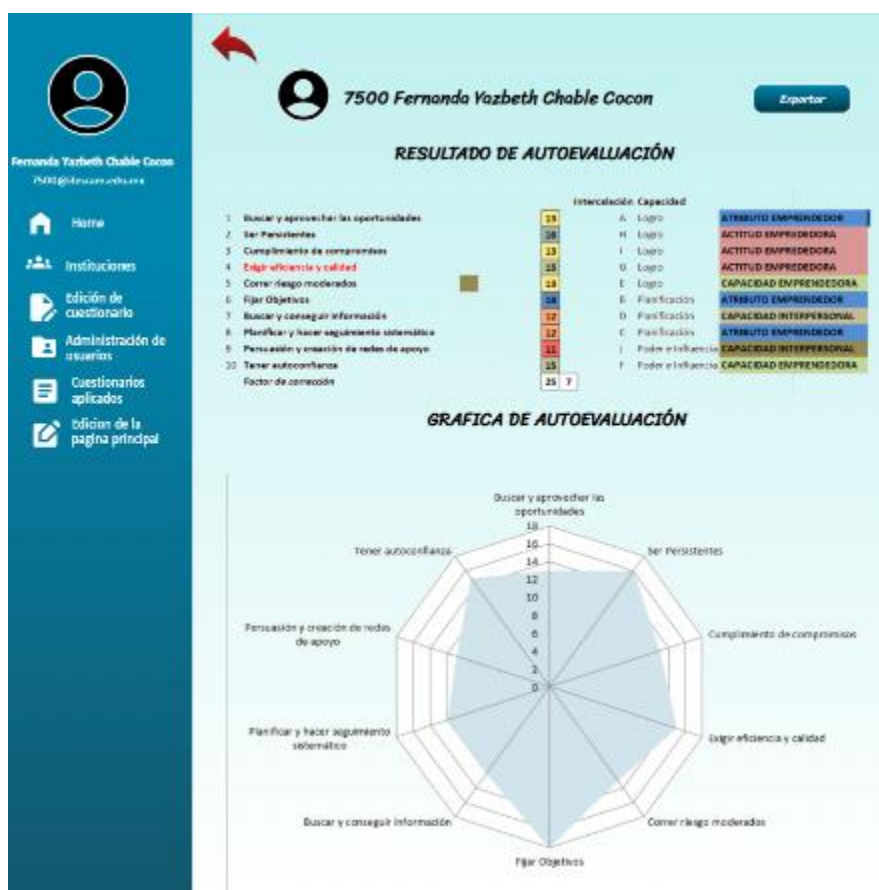


Ilustración 47. Diseño de la vista de resultados del alumno

3.6.7 Módulo edición de la página principal

Diseño del apartado de edición de la página principal donde se podrán editar ciertos componentes del inicio y que solamente el administrador podrá realizarlo.



Ilustración 48. Diseño de edición del perfil

3.6.8 Modulo Gestión de cuestionarios

Este módulo permite a los administradores y maestros gestionar todos los aspectos relacionados con los cuestionarios asignados a los estudiantes, los submódulos que incluye son:

- **Edición de Cuestionarios:** Modificación de preguntas y variables, con influencia definida por los administradores.
- **Configuración del Cuestionario:** Permite asignar cuestionarios a grupos y establecer fechas límite.
- **Asignación de Cuestionarios:** Dirigida a estudiantes individuales o grupos completos.
- **Código de Acceso:** Generación de códigos únicos para garantizar accesibilidad controlada.
- **Respuestas de los Estudiantes:** Recopila las respuestas enviadas en línea.
- **Retroalimentación Inmediata:** Brinda retroalimentación automática con recomendaciones personalizadas.
- **Eliminación de Cuestionarios Completados:** Oculta cuestionarios finalizados del perfil del estudiante.
- **Cálculo de Resultados:** Implementación de fórmulas para obtener puntajes basados en una escala predefinida.
- **Preguntas de Corrección:** Aumenta la precisión mediante una fórmula especial.
- **Aleatorización de Preguntas:** Genera variabilidad en las preguntas para cada usuario.

En este apartado el administrador podrá editar las preguntas, la variable y signo, posteriormente podrá guardar el cuestionario en una nueva versión.

The screenshot displays a web interface titled "Edición del cuestionario" with a pencil icon. It features a table with three columns: "Cuestionario", "Variable", and "Signo". The table contains seven rows, each with a text input field labeled "Pregunta 1", a dropdown menu labeled "Select", and another dropdown menu labeled "Select". At the bottom of the interface, there are four buttons: "Anterior", "Siguiente", "Cancelar", and "Guardar".

Cuestionario	Variable	Signo
Pregunta 1	Select	Select
Pregunta 1	Select	Select
Pregunta 1	Select	Select
Pregunta 1	Select	Select
Pregunta 1	Select	Select
Pregunta 1	Select	Select
Pregunta 1	Select	Select

Anterior Siguiente Cancelar Guardar

Ilustración 49. Diseño de la edición del cuestionario

Diseño de cuestionarios aplicados

En este apartado se tiene contemplado visualizar los cuestionarios aplicados por todos los maestros.



Ilustración 50. Diseño de cuestionarios aplicados

Instrucciones del cuestionario

Diseño de las instrucciones del cuestionario para los alumnos.

Cuestionario Estándar de Autoevaluación de las Características Emprendedoras Personales (CEPs)²

INSTRUCCIONES

a) El cuestionario consta de 55 declaraciones breves. Lea cuidadosamente cada una y decida qué tan acertadamente le describe. Sea honesto consigo mismo. Recuerde que nadie tiene puntaje perfecto. Este es un autoanálisis, por lo tanto el único que juzgará su resultado es usted mismo

b) Seleccione el número que corresponde para designar la media en que cada declaración describe su comportamiento:

1= Nunca	2= Raras veces	3 = Algunas veces	4= Usualmente	5= Siempre
----------	----------------	-------------------	---------------	------------

c) Algunas declaraciones pueden ser similares, pero ninguna es igual, por lo tanto deben ser respondidas todas

d) Si tiene duda sobre alguna, pase a la siguiente, y al finalizar vuelva sobre las que dejó sin contestar

Cancelar

Empezar

Ilustración 51. Diseño de instrucciones del cuestionario

Diseño del cuestionario

Cuestionario Estándar de Autoevaluación de las Características Emprendedoras Personales (CEPs)²

Lea cuidadosamente cada una y decida qué tan acertadamente le describe. Sea honesto consigo mismo.

1) *Me esmero en buscar cosas que necesitan hacerse.*
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

2) *Cuando me enfrente a un problema difícil invierto gran cantidad de tiempo en encontrar una solución.*
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

3) *Termino mi trabajo a tiempo.*
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

4) *Me molesta cuando las cosas no se hacen debidamente.*
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

5) *Prefiero situaciones en las que puedo controlar la mayor cantidad de la información al máximo.*
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

6) *Me gusta pensar sobre el futuro.*
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

7) *Cuando comienzo una tarea o un proyecto nuevo, reúno toda la información posible antes de darle curso.*
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

8) *Planifico un proyecto grande dividiéndolo en tareas de menor tamaño*
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

9) *Logro que otros apoyen mis recomendaciones.*
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

10) *Me siento confiado en que puede tener éxito en cualquier actividad que me propongo ejecutar.*
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

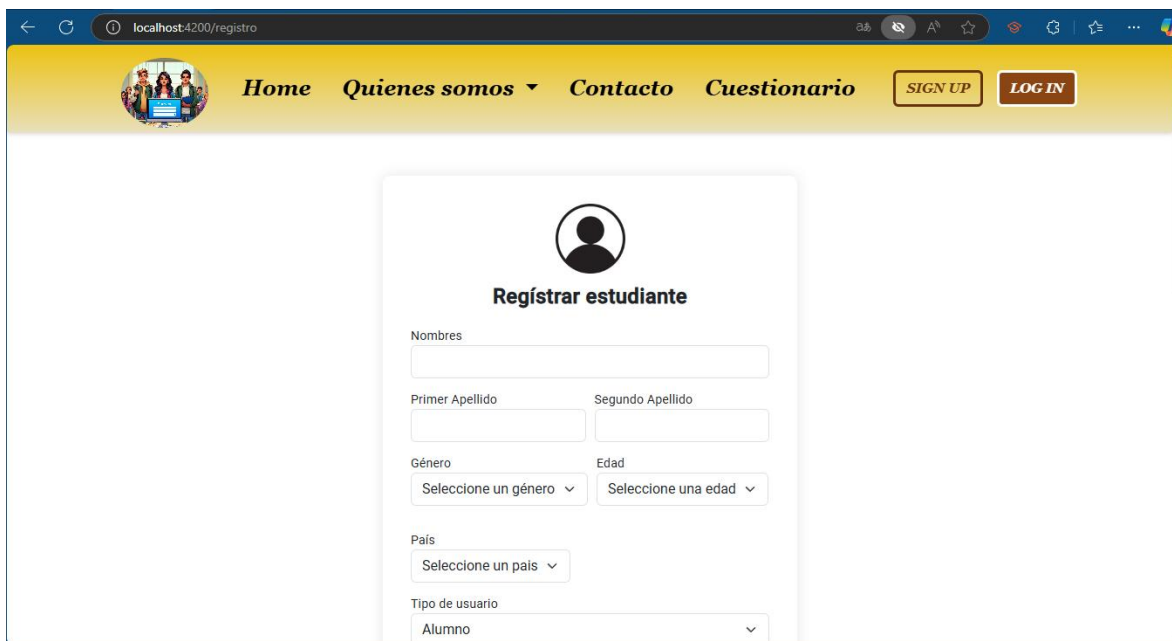
[Anterior](#) [Siguiente](#)

Ilustración 52. Diseño del cuestionario

3.7 Implementación del sistema.

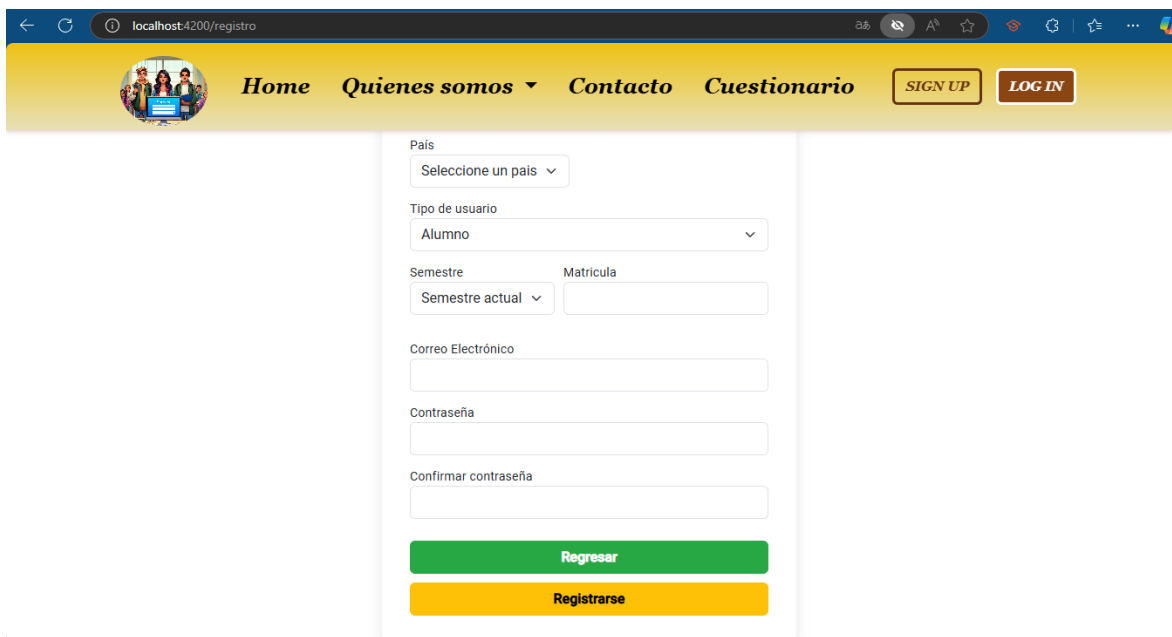
3.7.1 Registro de alumno

La primera vista del sistema es el registro del estudiante, por ende, el usuario alumno deberá registrarse a la plataforma.



The screenshot shows a web browser window with the URL 'localhost:4200/registro'. The header is yellow and contains a logo on the left, navigation links 'Home', 'Quienes somos', 'Contacto', and 'Cuestionario', and two buttons 'SIGN UP' and 'LOG IN' on the right. The main content area features a white card titled 'Registrar estudiante' with a user icon. The form includes the following fields: 'Nombres' (text input), 'Primer Apellido' and 'Segundo Apellido' (text inputs), 'Género' (dropdown menu with 'Seleccione un género'), 'Edad' (dropdown menu with 'Seleccione una edad'), 'País' (dropdown menu with 'Seleccione un país'), and 'Tipo de usuario' (dropdown menu with 'Alumno').

Ilustración 53. Vista del registro del alumno



This screenshot shows the same registration form as the previous one, but with additional fields visible below the 'Tipo de usuario' dropdown. These fields include: 'País' (dropdown menu with 'Seleccione un país'), 'Semestre' (dropdown menu with 'Semestre actual'), 'Matricula' (text input), 'Correo Electrónico' (text input), 'Contraseña' (text input), and 'Confirmar contraseña' (text input). At the bottom of the form are two buttons: a green 'Regresar' button and a yellow 'Registrarse' button.

Ilustración 54. Vista del registro del alumno

Datos ingresados del estudiante

The screenshot shows a web browser window with the address bar displaying 'localhost:4200/registro'. The page has a yellow header with a logo on the left and navigation links: 'Home', 'Quienes somos', 'Contacto', and 'Cuestionario'. There are also 'SIGN UP' and 'LOG IN' buttons. The main content area features a white card titled 'Registrar estudiante' with a user icon. The form includes the following fields: 'Nombres' (Antonio Israel), 'Primer Apellido' (Dominguez), 'Segundo Apellido' (Xiu), 'Género' (Masculino), 'Edad' (25), 'País' (Mexico), 'Institución' (Itescam), 'Tipo de usuario' (Alumno), 'Semestre', and 'Matricula'.

Registrar estudiante

Nombres
Antonio Israel

Primer Apellido
Dominguez

Segundo Apellido
Xiu

Género
Masculino

Edad
25

País
Mexico

Institución
Itescam

Tipo de usuario
Alumno

Semestre

Matricula

Ilustración 55. Datos ingresados

The screenshot shows the same web browser window, but the form is now filled out with additional information. The 'Tipo de usuario' is 'Alumno', 'Semestre' is '2°', and 'Matricula' is '7895'. The 'Carrera' is 'Ingeniería en Sistemas Computacionales', 'Correo Electrónico' is 'AntonioXiu@gmail.com', 'Contraseña' is '*****', and 'Confirmar contraseña' is '*****'. At the bottom of the form are two buttons: 'Regresar' (green) and 'Registrarse' (yellow).

Tipo de usuario
Alumno

Semestre
2°

Matricula
7895

Carrera
Ingeniería en Sistemas Computacionales

Correo Electrónico
AntonioXiu@gmail.com

Contraseña

Confirmar contraseña

Regresar

Registrarse

Ilustración 56. Datos ingresados

Al darle clic a registrarse aparecerá una ventana con el mensaje de que el estudiante ha sido creado correctamente, al aceptar el botón “aceptar” cerrará el dialogo.

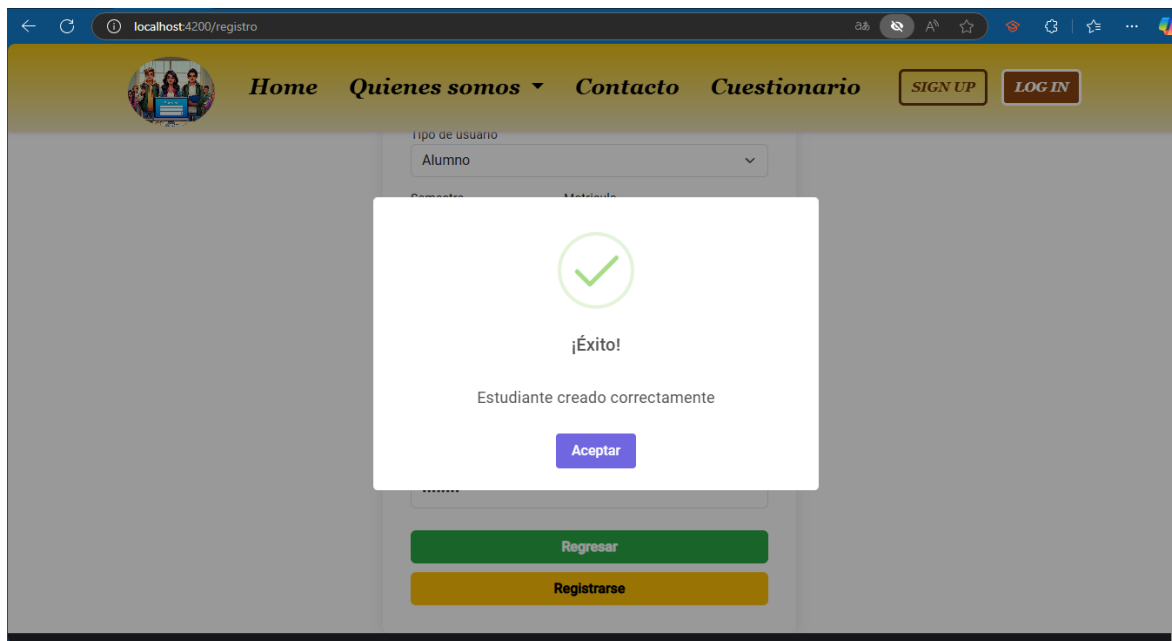


Ilustración 57. Vista del registro exitoso.

3.7.2 Página principal

Luego me retornará a la página de inicio, la página de inicio contiene información sobre la encuesta, un carrusel de imágenes, el objetivo del cuestionario como se ve a continuación:



Ilustración 58. Página principal



Ilustración 59. Carrusel de imágenes



Ilustración 60. Objetivo de la página



Ilustración 61. Aclaración

En la parte final del inicio contiene un pie de página el cual contiene información sobre el proyecto, los créditos, las instituciones que participan en el proyecto, las redes sociales y contacto. Las instituciones participantes y redes sociales ya contienen el direccionamiento a sus páginas oficiales.

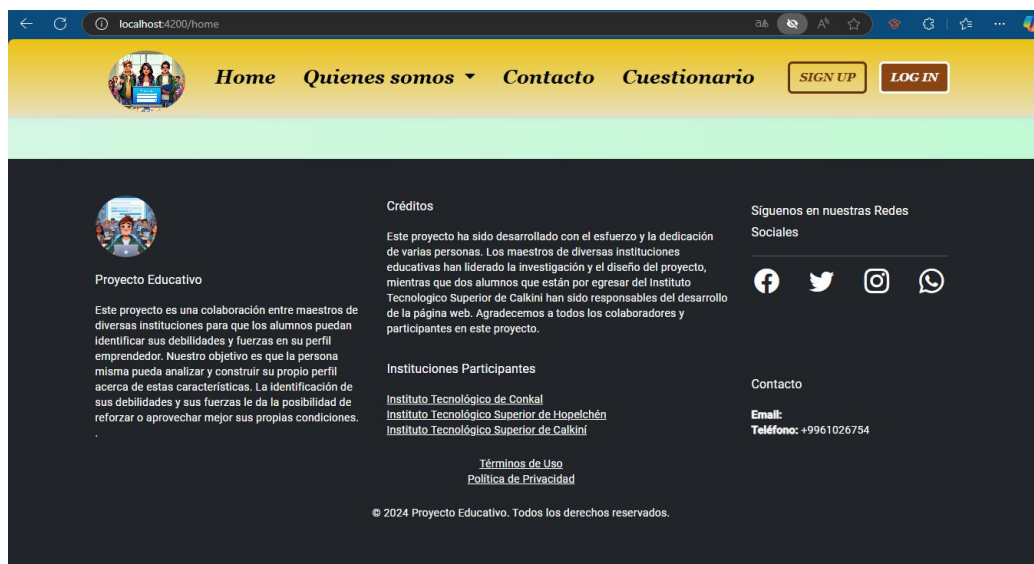


Ilustración 62. Pie de página

Quienes somos

En el encabezado nos dirigimos a quienes somos donde nos mostrará información del equipo que compone este proyecto, aun no se ingresa esa información hasta que el cliente proporcione los datos del equipo por ende aún está en proceso agregar la información.

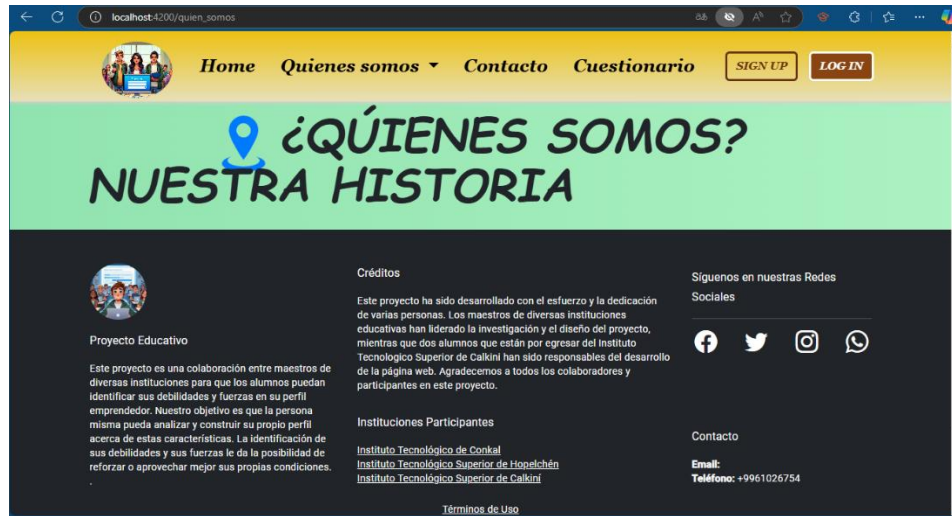


Ilustración 63. Vista ¿quiénes somos?

Contacto

Ahora en el apartado de contacto, nos mostrará el directorio y localización de las instituciones participantes del proyecto.



Ilustración 64. Localización de las instituciones



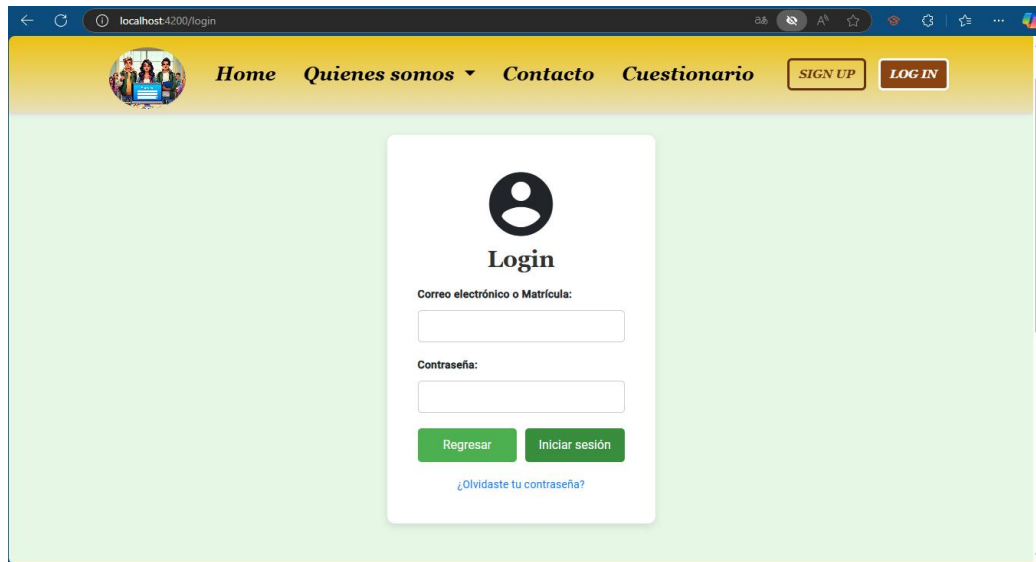
Ilustración 65. Localización de las instituciones



Ilustración 66. Localización de las instituciones

3.7.3 Login de usuarios

En el apartado de cuestionario no se podrá ingresar hasta que el usuario inicie sesión y pertenezca a un grupo donde el maestro le haya asignado un cuestionario de igual manera.



localhost:4200/login

Home Quienes somos Contacto Cuestionario SIGN UP LOG IN

Login

Correo electrónico o Matrícula:

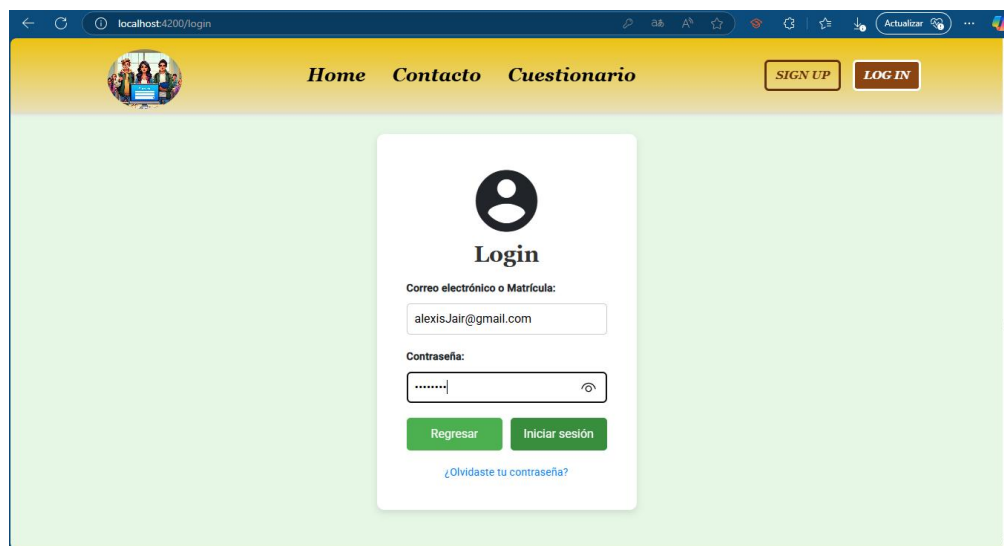
Contraseña:

Regresar Iniciar sesión

[¿Olvidaste tu contraseña?](#)

Ilustración 67. Vista login de usuarios

Ahora en el login, el usuario puede ingresar a su cuenta ya sea con correo o matricula, usaremos la matricula del usuario que se creó anteriormente para verificar el funcionamiento.



localhost:4200/login

Home Contacto Cuestionario SIGN UP LOG IN Actualizar

Login

Correo electrónico o Matrícula:

alexisJair@gmail.com

Contraseña:

.....

Regresar Iniciar sesión

[¿Olvidaste tu contraseña?](#)

Ilustración 68. Login de usuarios

3.7.4 Dashboard del Alumno

Al darle iniciar sesión, nos dirigirá al Dashboard del alumno, el cual nos muestra el nombre y correo del alumno, así como el día, el menú y los apartados a los que pueda ingresar.

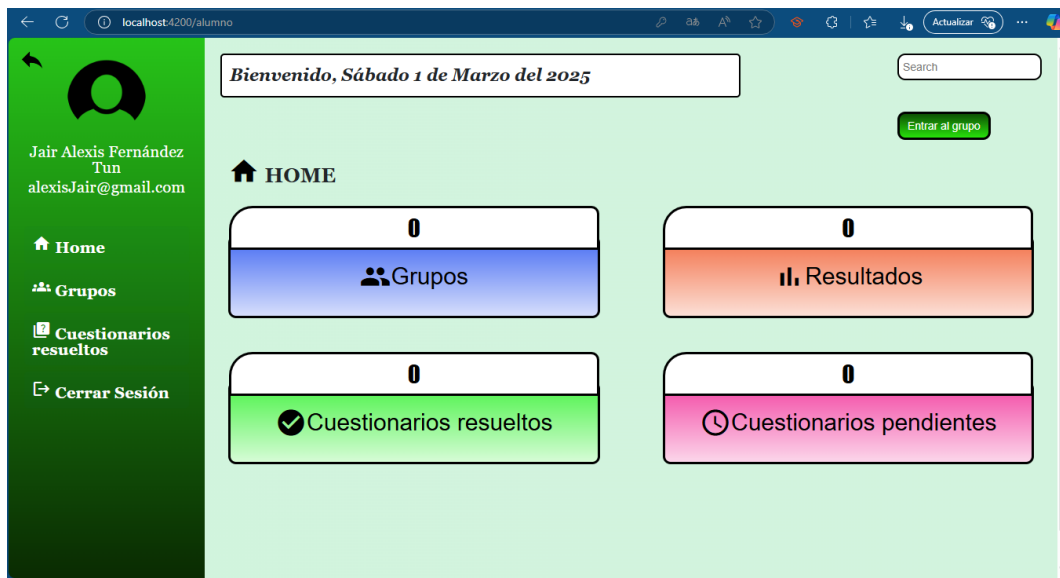


Ilustración 69. Dashboard del alumno

Al darle clic al botón entrar al grupo, se desplegará una ventana para buscar el grupo mediante el código que proporcionará el maestro.

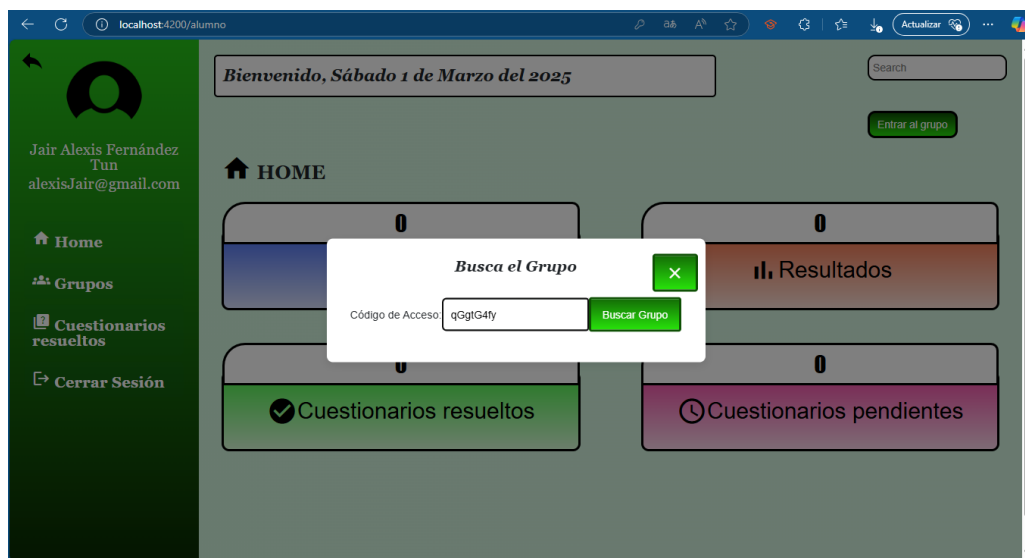


Ilustración 70. Búsqueda de grupo

Al dar clic en buscar nos aparecerá información del grupo donde el alumno al verificar la información se una al grupo o al cancelar cierre la ventana.

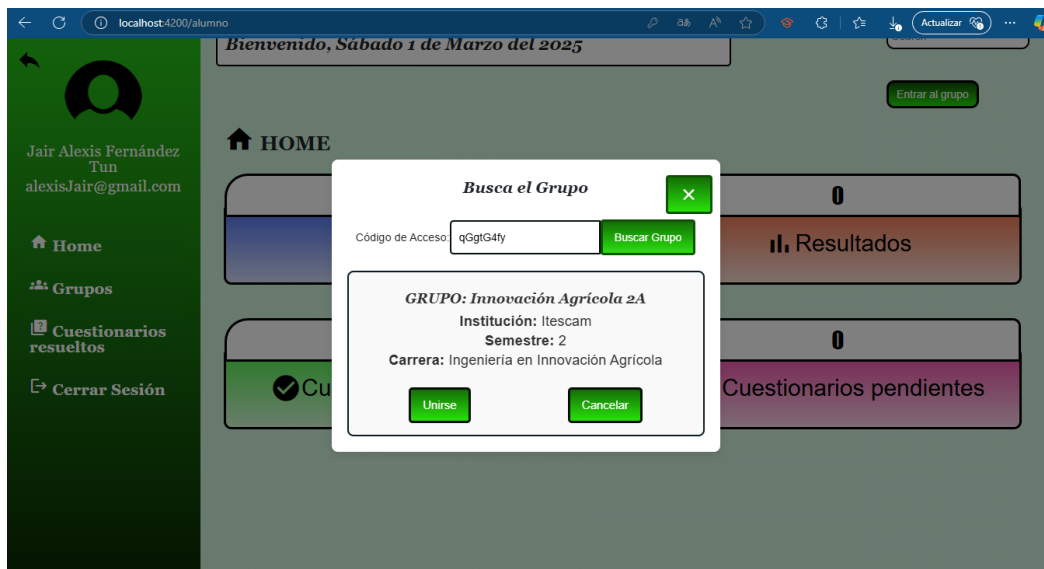


Ilustración 71. Información del grupo



Ilustración 72. Ingreso al grupo

3.7.5 Grupos

En el apartado de grupos, el alumno podrá visualizar sus respectivos grupos o bien, podrá unirse a otro.

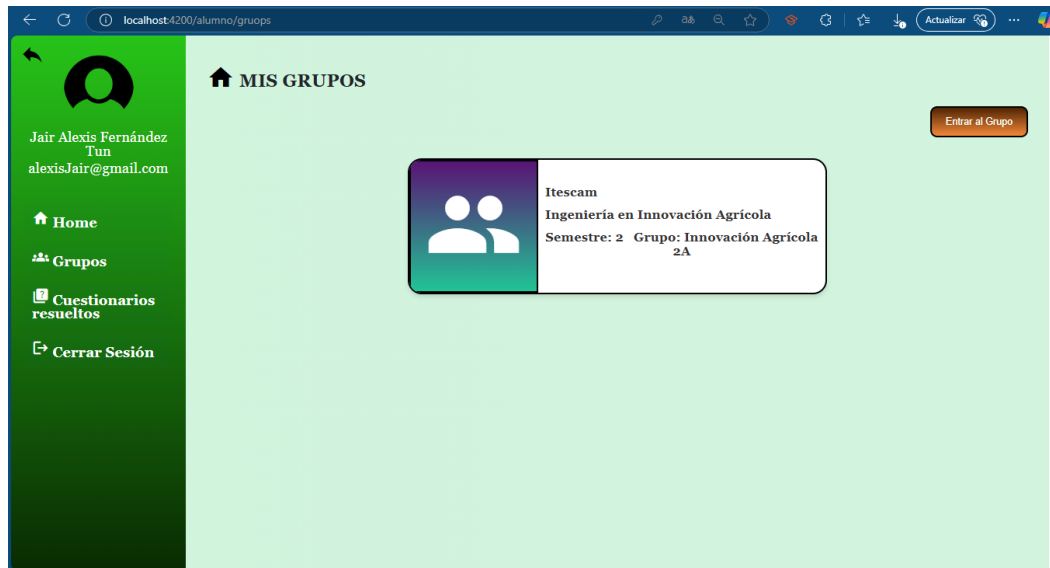


Ilustración 73. Visualización de grupos

Al entrar al grupo el alumno visualizará los cuestionarios resueltos.

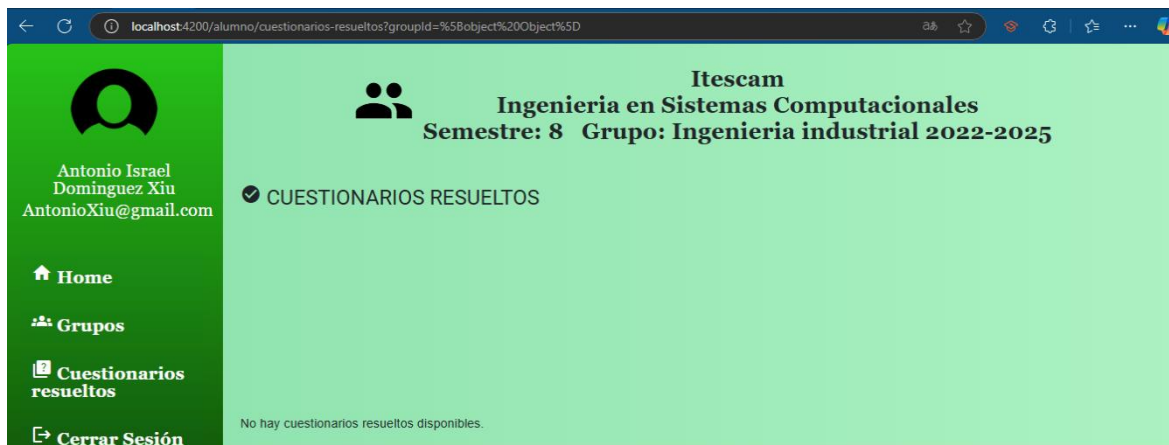


Ilustración 74. Visualización de cuestionarios (Si no hay disponibles)

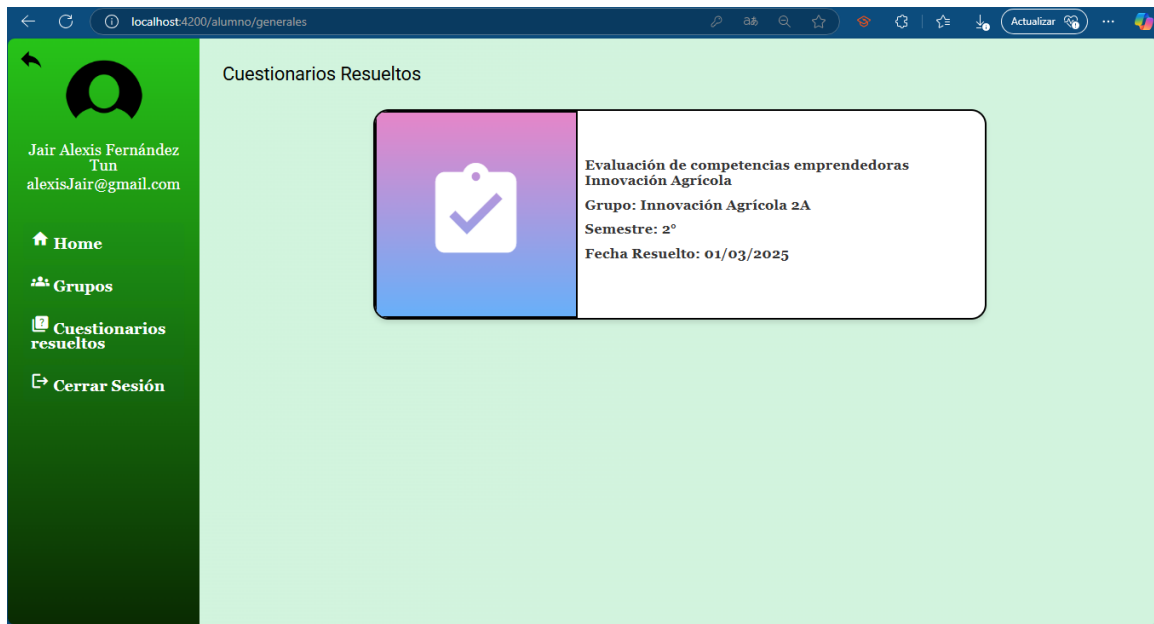


Ilustración 75. Visualización de cuestionarios (Con cuestionarios resueltos)

3.7.6 Cuestionarios para el alumno

El alumno para poder visualizar los cuestionarios pendientes del alumno, se podrá visualizar el nombre del cuestionario, su descripción, la fecha de publicación y cierre.

Posteriormente le redirigirá al alumno al cuestionario para resolver.



Ilustración 76. Visualización de cuestionarios (Con cuestionarios pendientes)

Versión para el ITESCAM



Esta versión es para la institución del ITESCAM con cambios en algunas palabras para el entendimiento en la región

Me esmero en buscar cosas que necesitan hacerse.

- ☐ Nunca
- ☐ Raras veces
- ☐ Algunas veces
- ☐ Usualmente
- ☐ Siempre

Cuando me enfrento a un problema difícil invierto gran cantidad de tiempo en encontrar una solución.

- ☐ Nunca
- ☐ Raras veces
- ☐ Algunas veces
- ☐ Usualmente
- ☐ Siempre

Termino mi trabajo a tiempo.

- ☐ Nunca
- ☐ Raras veces

Ilustración 77. Visualización del cuestionario

Termino mi trabajo a tiempo.

- ☐ Nunca
- ☐ Raras veces
- ☐ Algunas veces
- ☐ Usualmente
- ☐ Siempre

Me molesta cuando las cosas no se hace debidamente.

- ☐ Nunca
- ☐ Raras veces
- ☐ Algunas veces
- ☐ Usualmente
- ☐ Siempre

Prefiero situaciones en las que puedo controlar la mayor cantidad de la información al máximo.

- ☐ Nunca
- ☐ Raras veces
- ☐ Algunas veces
- ☐ Usualmente
- ☐ Siempre

Anterior

Siguiente

Ilustración 78. Visualización del cuestionario

3.7.7 Resultados del alumno

Al darle clic al cuestionario resuelto el alumno podrá visualizar los resultados y una retroalimentación. De igual forma la opción de exportar los resultados en un PDF.

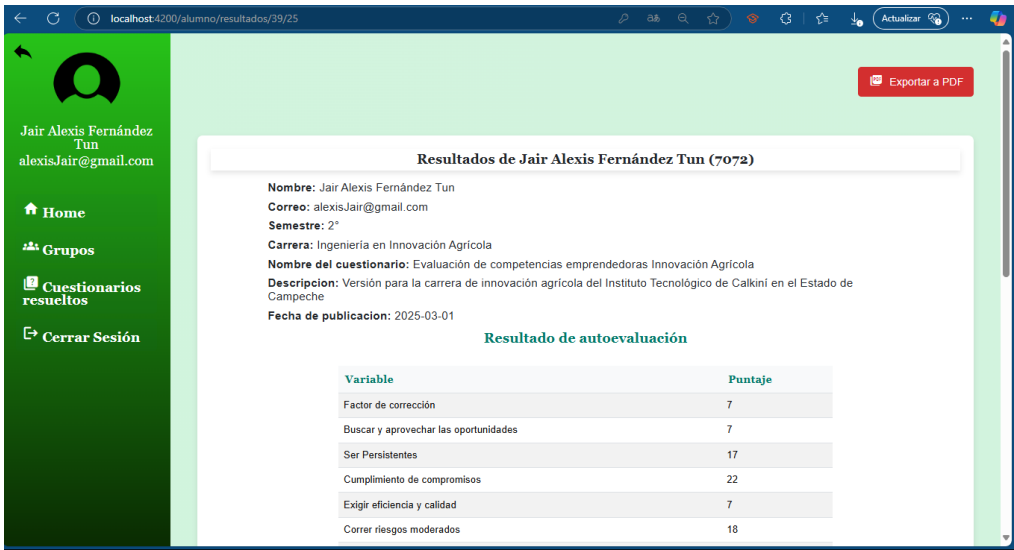


Ilustración 79. Visualización de información (Resultados de autoevaluación)

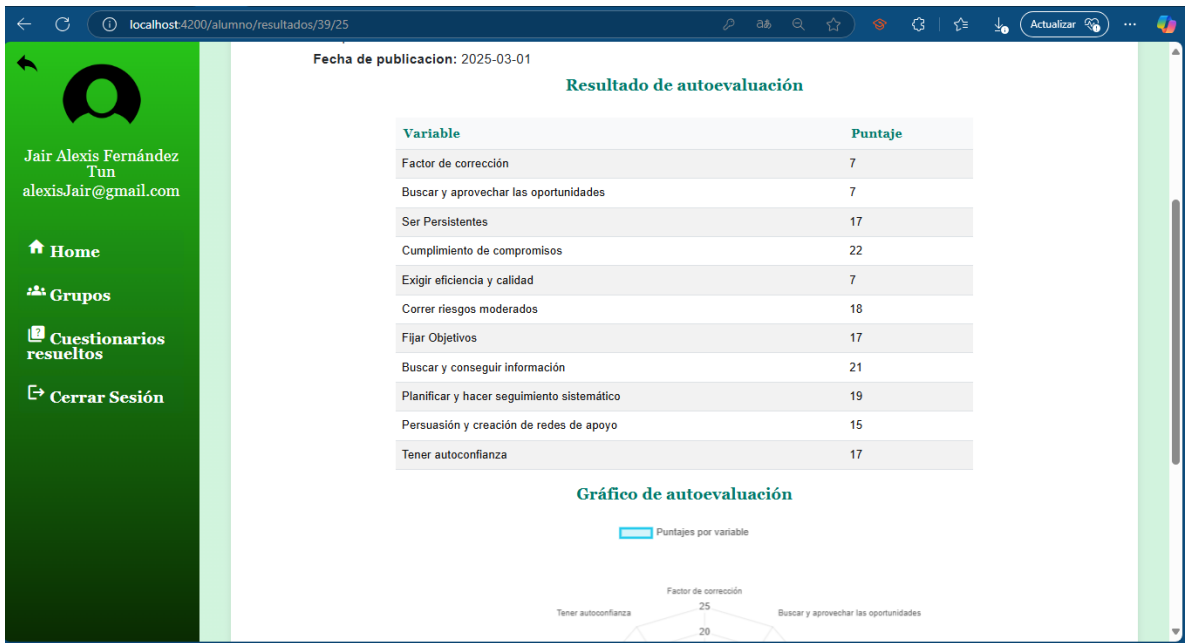


Ilustración 80.1 Visualización de resultados (Resultados de autoevaluación)

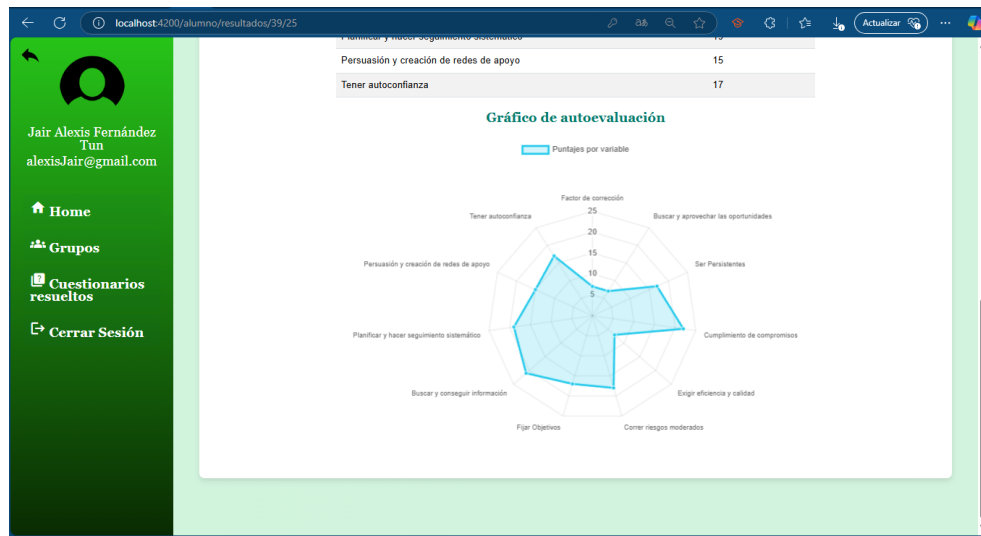


Ilustración 81. Visualización de resultados (Gráfico de autoevaluación)

Si el alumno desea cerrar la sesión se le redirigirá a la página principal y deberá ingresar nuevamente.

3.7.8 Dashboard para maestros

El maestro al iniciar sesión será dirigido a su Dashboard, el cual contiene su información del perfil y las acciones que podrá realizar.

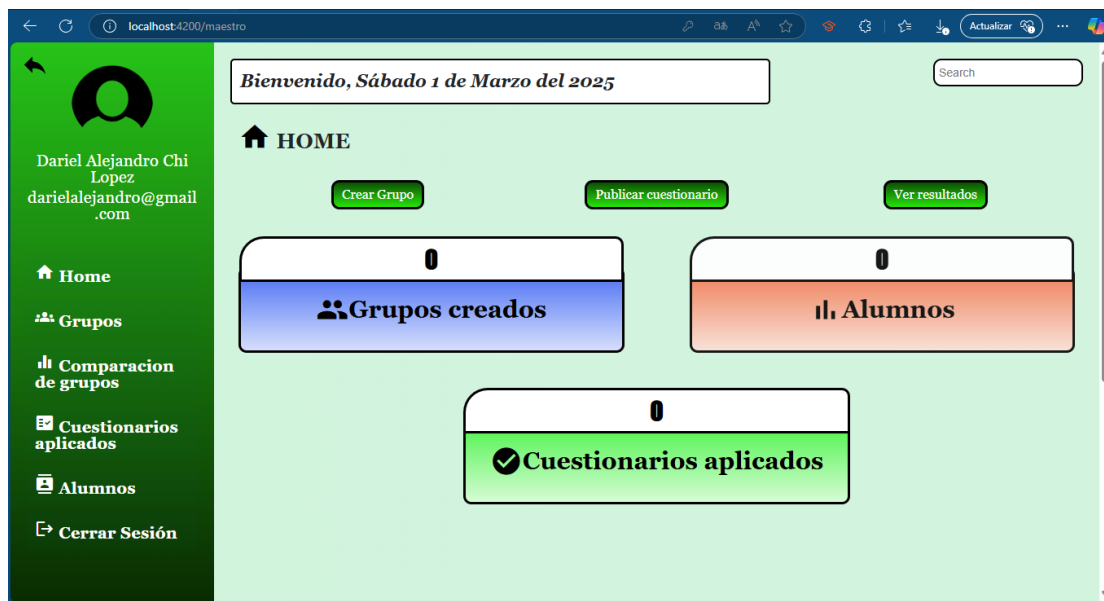


Ilustración 82. Visualización del Dashboard

El maestro podrá crear nuevos grupos al darle clic al botón “Crear grupo”, el cual desplegará un formulario para ingresar los datos del grupo.

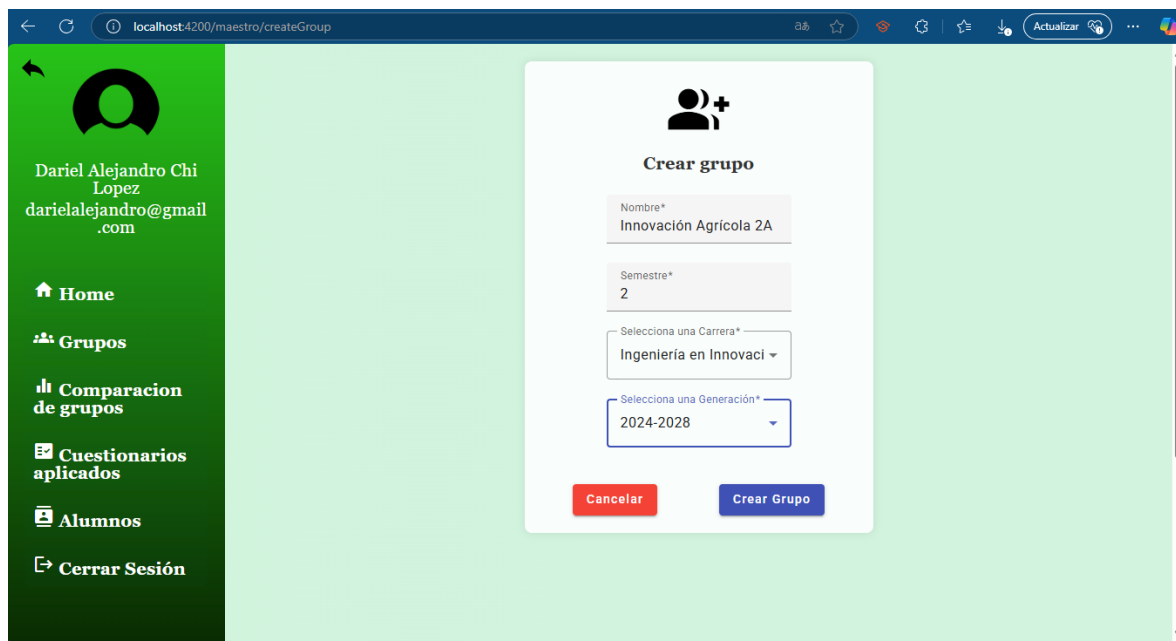


Ilustración 83. Vista agregar grupo

Una vez creado el grupo aparecerá una ventana flotante que informa que el grupo se ha creado correctamente y nos proporciona el código de acceso al grupo, al darle en aceptar procederá a cerrar la ventana y redirigir al usuario a la lista de grupos.

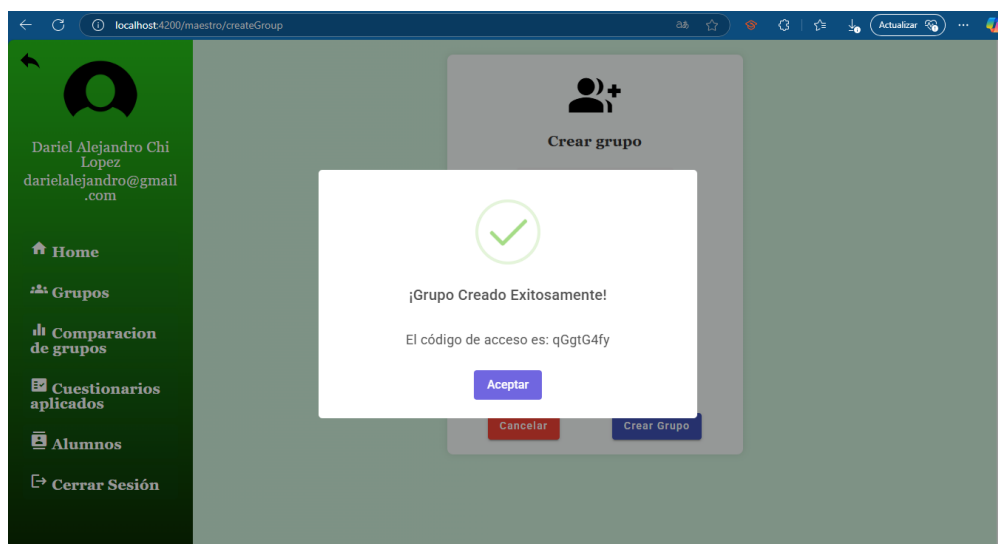


Ilustración 84. Creación de nuevo grupo

3.7.9 Lista de grupos

En el apartado de grupos se visualizarán todos los grupos existentes a las que pertenece el maestro, podrá crear un grupo o ingresar a la información del grupo.



Ilustración 85. Visualización de grupos

Al darle clic a un grupo se podrá visualizar los datos del grupo, los cuestionarios aplicados y de igual manera ver los integrantes del grupo.

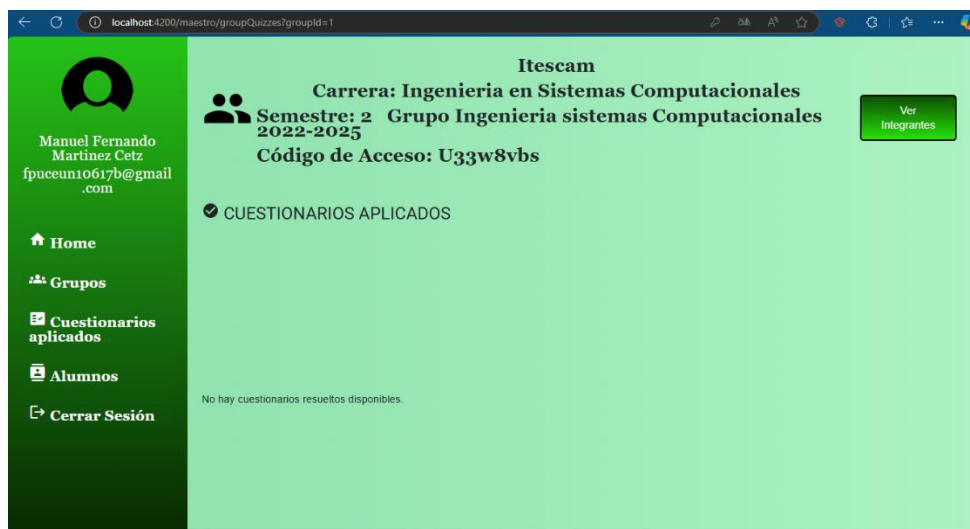


Ilustración 86. Visualización de cuestionarios (no disponibles)

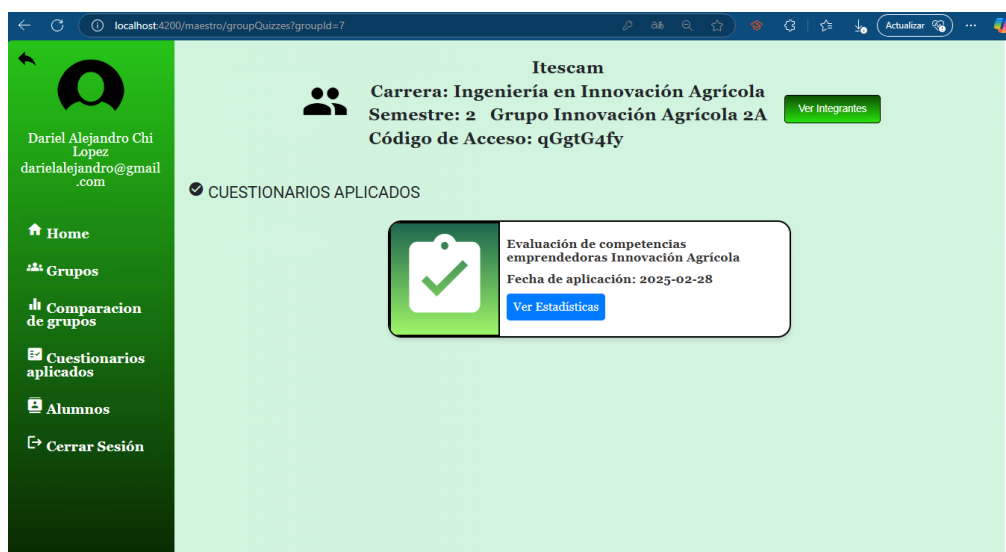
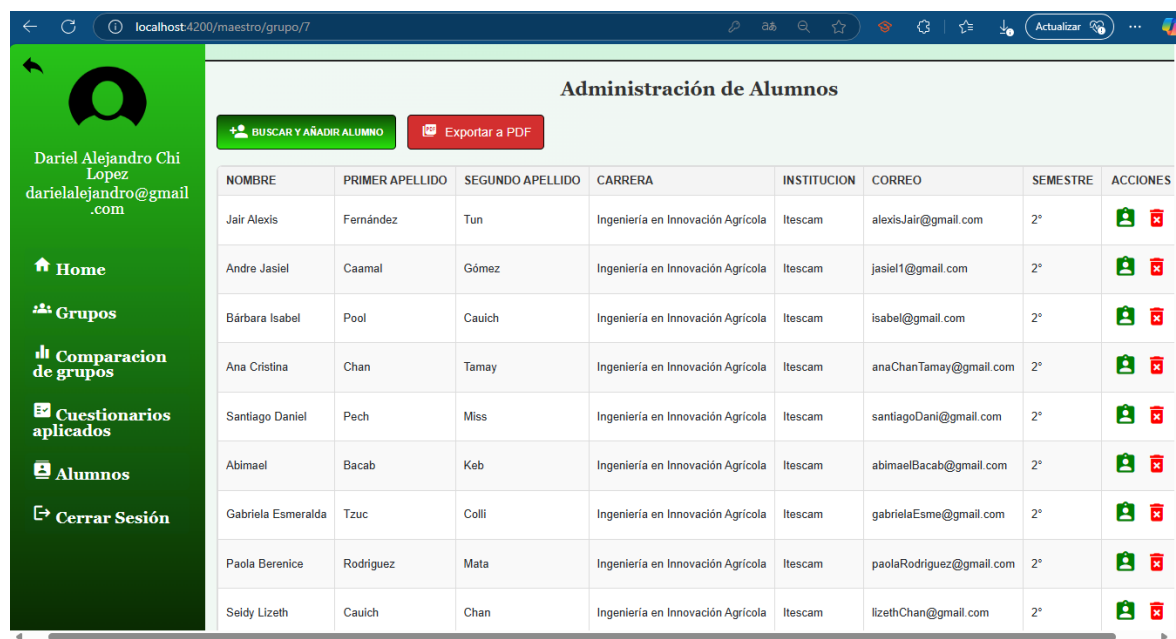


Ilustración 87. Visualización de cuestionarios (Disponibles)

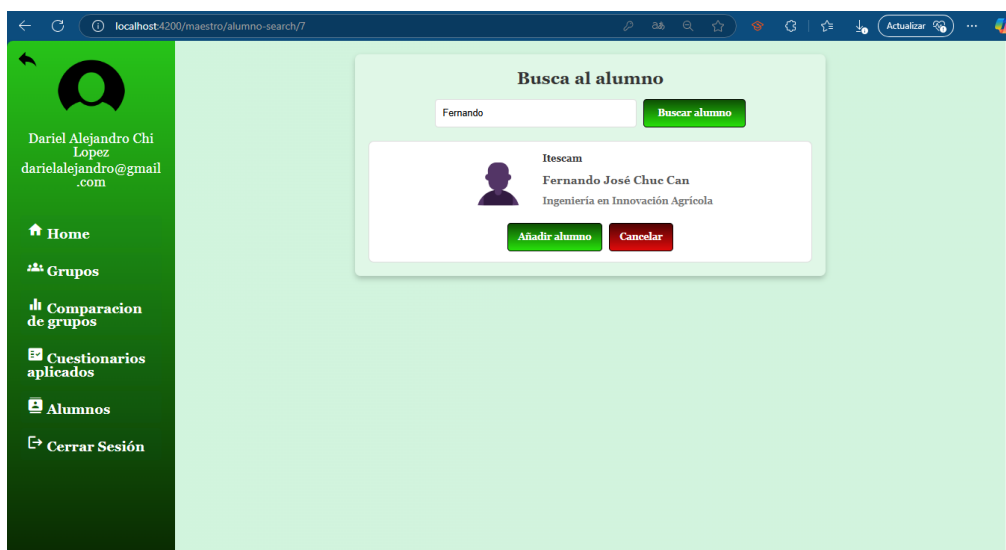
Al darle clic al botón de ver integrantes, nos mostrará la lista de alumnos que pertenecen al grupo con sus respectivas acciones ya sea eliminar el alumno del grupo o ver sus resultados. De igual forma el docente podrá exportar la lista de alumnos en formato PDF.



NOMBRE	PRIMER APELLIDO	SEGUNDO APELLIDO	CARRERA	INSTITUCION	CORREO	SEMESTRE	ACCIONES
Jair Alexis	Fernández	Tun	Ingeniería en Innovación Agrícola	Itescam	alexisJair@gmail.com	2º	
Andre Jasiel	Caamal	Gómez	Ingeniería en Innovación Agrícola	Itescam	jasiel1@gmail.com	2º	
Bárbara Isabel	Pool	Cauich	Ingeniería en Innovación Agrícola	Itescam	isabel@gmail.com	2º	
Ana Cristina	Chan	Tamay	Ingeniería en Innovación Agrícola	Itescam	anaChanTamay@gmail.com	2º	
Santiago Daniel	Pech	Miss	Ingeniería en Innovación Agrícola	Itescam	santiagoDani@gmail.com	2º	
Abimael	Bacab	Keb	Ingeniería en Innovación Agrícola	Itescam	abimaelBacab@gmail.com	2º	
Gabriela Esmeralda	Tzuc	Colli	Ingeniería en Innovación Agrícola	Itescam	gabrielaEsme@gmail.com	2º	
Paola Berenice	Rodriguez	Mata	Ingeniería en Innovación Agrícola	Itescam	paolaRodriguez@gmail.com	2º	
Seidy Lizeth	Cauich	Chan	Ingeniería en Innovación Agrícola	Itescam	lizethChan@gmail.com	2º	

Ilustración 88. Visualización de alumnos del grupo

Al darle clic al botón de buscar y añadir alumno, nos redirigirá a una ventana donde buscaremos al alumno, por su nombre, matricula o correo. Al darle clic a buscar nos mostrara los datos del alumno y la opción de añadirlo al grupo.



Busca al alumno

Fernando

Itescam

Fernando José Chuc Can

Ingeniería en Innovación Agrícola

Ilustración 89. Visualización de cuestionarios

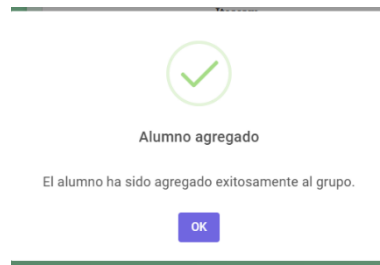


Ilustración 90. Alumno agregado al grupo

Redirigirá al maestro a la lista de alumnos actualizada con el nuevo usuario.

NOMBRE	PRIMER APELLIDO	SEGUNDO APELLIDO	CARRERA	INSTITUCION	CORREO	SEMESTRE	ACCION
Jair Alexis	Fernández	Tun	Ingeniería en Innovación Agrícola	Itescam	alexisJair@gmail.com	2º	
Andre Jasiel	Caamal	Gómez	Ingeniería en Innovación Agrícola	Itescam	jasiel1@gmail.com	2º	
Barbara Isabel	Pool	Cauich	Ingeniería en Innovación Agrícola	Itescam	isabel@gmail.com	2º	
Ana Cristina	Chan	Tamay	Ingeniería en Innovación Agrícola	Itescam	anaChanTamay@gmail.com	2º	
Santiago Daniel	Pech	Miss	Ingeniería en Innovación Agrícola	Itescam	santiagoDani@gmail.com	2º	
Abimael	Bacab	Keb	Ingeniería en Innovación Agrícola	Itescam	abimaelBacab@gmail.com	2º	
Gabriela Esmeralda	Tzuc	Colli	Ingeniería en Innovación Agrícola	Itescam	gabrielaEsme@gmail.com	2º	
Paola Berenice	Rodriguez	Mata	Ingeniería en Innovación Agrícola	Itescam	paolaRodriguez@gmail.com	2º	
Seidy Lizeth	Cauich	Chan	Ingeniería en Innovación Agrícola	Itescam	lizethChan@gmail.com	2º	

Ilustración 91. Lista de alumnos actualizada

3.7.10 Vista de cuestionarios aplicados

Al ingresar a la sección de cuestionarios aplicados se podrán observar todos los cuestionarios que han sido publicados por el docente. La información del cuestionario se presenta en tarjetas donde muestra el nombre del cuestionario, carrera, grupo, fecha de publicación y fecha de cierre.

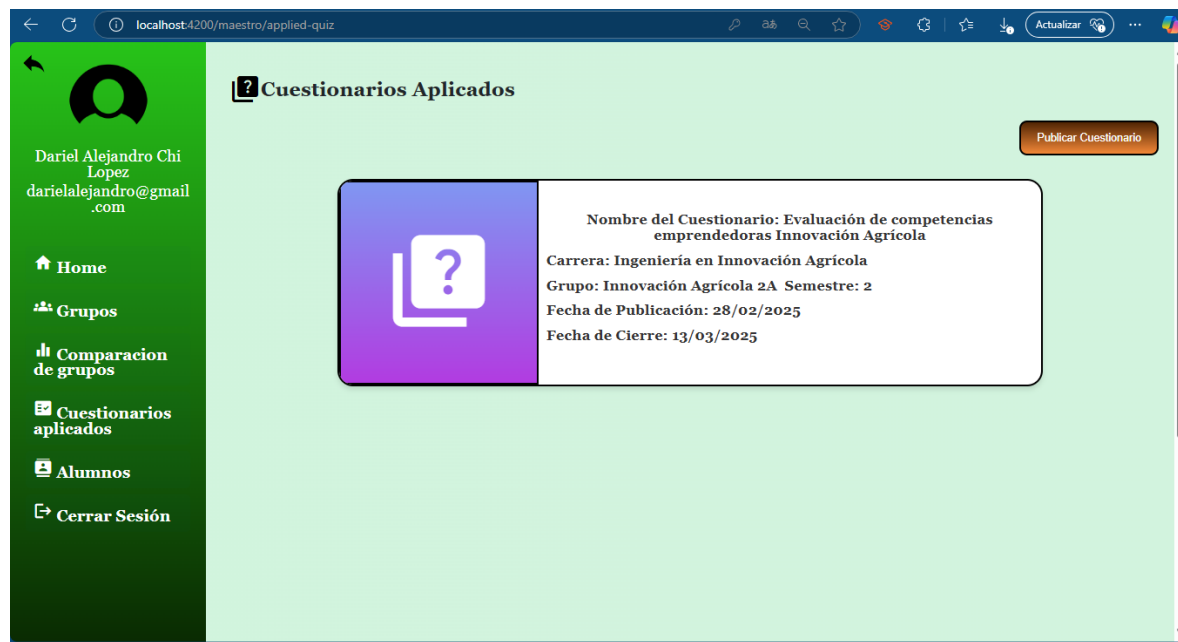


Ilustración 92. Lista de cuestionarios aplicados

También el maestro tiene la opción de publicar un cuestionario ya existente ya sea en el mismo grupo o en uno diferente, al darle clic al botón de “publicar cuestionario” se mostrará una vista el cual el maestro podrá seleccionar la versión del cuestionario, el grupo al cual se aplicará, la fecha de publicación y la fecha de cierre.

PUBLICAR CUESTIONARIO EXISTENTE A UN GRUPO

Seleccionar Cuestionario

Grupo

Fecha de Publicación

Fecha de Cierre

Anterior Siguiente

Publicar Cuestionario Cancelar

Manuel Fernando Martínez Cetz
fpuceun10617b@gmail.com

Home Grupos Cuestionarios aplicados Alumnos Cerrar Sesión

Ilustración 93. Vista publicar cuestionario existente a un grupo

El maestro al seleccionar el cuestionario se mostrarán todas las preguntas y opciones del cuestionario. Cabe mencionar que el cuestionario tiene un total de 55 preguntas por ende se podrán visualizar 10 preguntas por página.

PUBLICAR CUESTIONARIO EXISTENTE A UN GRUPO

Seleccionar Cuestionario

Grupo

Fecha de Publicación

Fecha de Cierre

Preguntas del Cuestionario

1. Me esmero en buscar cosas que necesitan hacerse.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

2. Cuando me enfrento a un problema difícil invierto gran cantidad de tiempo en encontrar una solución.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

3. Terminé mi trabajo a tiempo.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

4. Me molesta cuando las cosas no se hace debidamente.

Actualizar ...

Dariel Alejandro Chi Lopez
darielalejandrolopez@gmail.com

Home Grupos Comparacion de grupos Cuestionarios aplicados Alumnos Cerrar Sesión

Ilustración 94. Vista publicar cuestionario (Cuestionario seleccionado)

localhost:4200/maestro/post-quiz

Preguntas del Cuestionario

1. Me esmero en buscar cosas que necesitan hacerse.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
2. Cuando me enfrento a un problema difícil invierto gran cantidad de tiempo en encontrar una solución.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
3. Terminó mi trabajo a tiempo.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
4. Me molesta cuando las cosas no se hacen debidamente.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
5. Prefiero situaciones en las que puedo controlar la mayor cantidad de la información al máximo.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
6. Me gusta pensar sobre el futuro.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
7. Cuando comienza una tarea o un proyecto nuevo, reúno toda la información posible antes de darle curso.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
8. Planifico un proyecto grande dividiéndolo en tareas de menor tamaño.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
9. Logro que otros apoyen mis recomendaciones.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
10. Me siento confiado en que puede tener éxito en cualquier actividad que me propongo ejecutar.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

Anterior Publicar Cuestionario Siguiente Cancelar

Ilustración 95. Vista publicar cuestionario (Cuestionario seleccionado)

localhost:4200/maestro/post-quiz

Fecha de Publicación
26/02/2025

Fecha de Cierre
13/03/2025

Preguntas del Cuestionario

51. Me valgo de varias fuentes de información al buscar ayuda para llevar a cabo tareas o proyectos.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
52. Si no resulta una determinada estrategia para hacer frente a un problema, busco otro.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
53. Puedo lograr que personas con firmes convicciones y opiniones cambien de modo de pensar.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
54. Me mantengo firme en mis decisiones, aún cuando otras personas me contradigan enérgicamente.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre
55. Cuando no se algo no reparo en admitirlo.
☐ Nunca ☐ Raras veces ☐ Algunas veces ☐ Usualmente ☐ Siempre

Anterior Publicar Cuestionario Siguiente Cancelar

Ilustración 96. Vista publicar cuestionario (Cuestionario seleccionado)

Luego de publicar el cuestionario podrá aparecerá un mensaje el cual indicará que el cuestionario ha sido publicado correctamente.

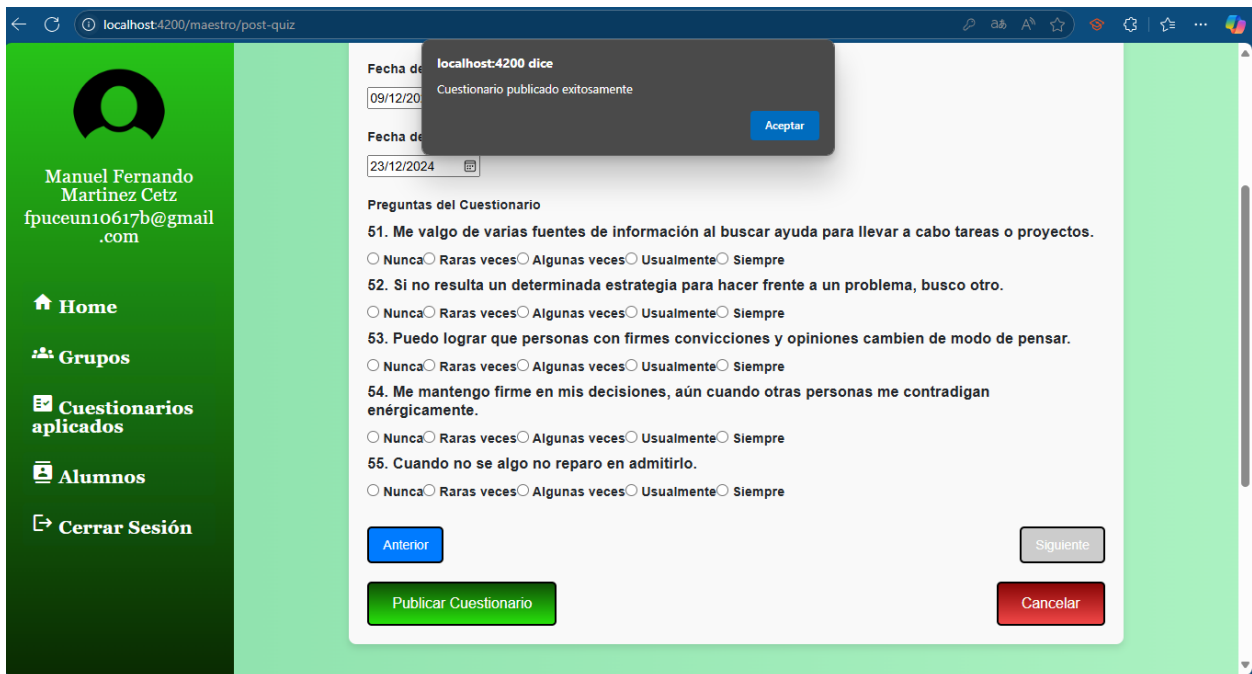


Ilustración 97. Mensaje cuestionario publicado.

Luego al darle aceptar al mensaje redirigirá al maestro a la lista de cuestionarios.



Ilustración 98. Lista de cuestionario actualizada

Al darle clic a un cuestionario aplicado se mostrará una vista con la información del grupo a la cual se aplicó el cuestionario, así como los integrantes del grupo y los cuestionarios aplicados en ese grupo en particular. De igual manera con la opción de exportar los datos del cuestionario.



Dariel Alejandro Chi Lopez
darielalejandro@gmail.com

Home

Grupos

Comparacion de grupos

Cuestionarios aplicados

Alumnos

Cerrar Sesión

Exportar a PDF

Carrera: Ingeniería en Innovación Agrícola

Grupo: Innovación Agrícola 2A Semestre: 2

Código de acceso: qGgtG4fy

Integrantes del Grupo

Nombre	Primer Apellido	Segundo Apellido	Carrera
Jair Alexis	Fernández	Tun	Ingeniería en Innovación Agrícola
Andre Jasiel	Caamal	Gómez	Ingeniería en Innovación Agrícola
Bárbara Isabel	Pool	Cauich	Ingeniería en Innovación Agrícola
Ana Cristina	Chan	Tamay	Ingeniería en Innovación Agrícola
Santiago Daniel	Pech	Miss	Ingeniería en Innovación Agrícola
Abimael	Bacab	Keb	Ingeniería en Innovación Agrícola
Gabriela Esmeralda	Tzuc	Colli	Ingeniería en Innovación Agrícola
Paola Berenice	Rodríguez	Mata	Ingeniería en Innovación Agrícola
Seidy Lizeth	Cauich	Chan	Ingeniería en Innovación Agrícola
Cesar Osvaldo	Pech	Bonilla	Ingeniería en Innovación Agrícola

Ilustración 99. Información del cuestionario aplicado en el grupo



Dariel Alejandro Chi Lopez
darielalejandro@gmail.com

Home

Grupos

Comparacion de grupos

Cuestionarios aplicados

Alumnos

Cerrar Sesión

Abimael	Bacab	Keb	Ingeniería en Innovación Agrícola
Gabriela Esmeralda	Tzuc	Colli	Ingeniería en Innovación Agrícola
Paola Berenice	Rodríguez	Mata	Ingeniería en Innovación Agrícola
Seidy Lizeth	Cauich	Chan	Ingeniería en Innovación Agrícola
Cesar Osvaldo	Pech	Bonilla	Ingeniería en Innovación Agrícola
Ismael Alexander	Maas	Martínez	Ingeniería en Innovación Agrícola
Daniel David	Uc	Tun	Ingeniería en Innovación Agrícola
Fernando José	Chuc	Can	Ingeniería en Innovación Agrícola
Jesús David	Suárez	Kantun	Ingeniería en Innovación Agrícola

Cuestionarios Aplicados

Nombre: Evaluación de competencias emprendedoras Innovación Agrícola

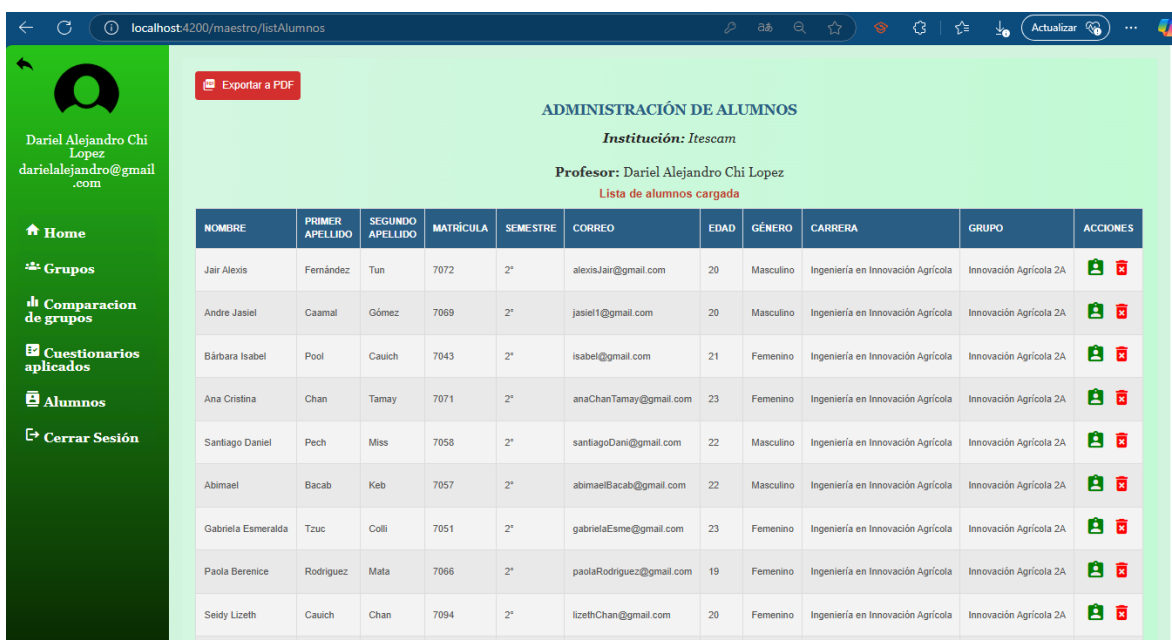
Detalles de Cuestionario: Versión para la carrera de innovación agrícola del Instituto Tecnológico de Calkiní en el Estado de Campeche

Mostrar Detalles

Ilustración 100.1 Información del cuestionario aplicado en el grupo

3.7.11 Administración de alumnos del maestro

En este apartado se podrá visualizar el nombre de la institución a la que pertenece el maestro, de igual manera su nombre y la lista de alumnos con los siguientes datos: Nombre, Apellidos, Edad, Genero, Carrera y Grupo. Con 2 acciones, visualizar los resultados del alumno y eliminarlo del grupo.



NOMBRE	PRIMER APELLIDO	SEGUNDO APELLIDO	MATRÍCULA	SEMESTRE	CORREO	EDAD	GÉNERO	CARRERA	GRUPO	ACCIONES
Jair Alexis	Fernández	Tun	7072	2*	alexisJair@gmail.com	20	Masculino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Andre Jasiel	Caamal	Gómez	7069	2*	jasiel1@gmail.com	20	Masculino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Bárbara Isabel	Pool	Caulch	7043	2*	isabel@gmail.com	21	Femenino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Ana Cristina	Chan	Tamay	7071	2*	anaChanTamay@gmail.com	23	Femenino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Santiago Daniel	Pech	Miss	7058	2*	santiagoDani@gmail.com	22	Masculino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Abimael	Bacab	Keb	7057	2*	abimaelBacab@gmail.com	22	Masculino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Gabriela Esmeralda	Tzuc	Colli	7051	2*	gabrielaEsme@gmail.com	23	Femenino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Paola Berenice	Rodriguez	Mata	7066	2*	paolaRodriguez@gmail.com	19	Femenino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Seldy Lizeth	Caulch	Chan	7094	2*	lizethChan@gmail.com	20	Femenino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 

Ilustración 101. Administración de alumnos del maestro



NOMBRE	PRIMER APELLIDO	SEGUNDO APELLIDO	MATRÍCULA	SEMESTRE	CORREO	EDAD	GÉNERO	CARRERA	GRUPO	ACCIONES
Jair Alexis	Fernández	Tun	7072	2*	alexisJair@gmail.com	20	Masculino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Andre Jasiel	Caamal	Gómez	7069	2*	jasiel1@gmail.com	20	Masculino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Bárbara Isabel	Pool	Caulch	7043	2*	isabel@gmail.com	21	Femenino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Ana Cristina	Chan	Tamay	7071	2*	anaChanTamay@gmail.com	23	Femenino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Santiago Daniel	Pech	Miss	7058	2*	santiagoDani@gmail.com	22	Masculino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Abimael	Bacab	Keb	7057	2*	abimaelBacab@gmail.com	22	Masculino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Gabriela Esmeralda	Tzuc	Colli	7051	2*	gabrielaEsme@gmail.com	23	Femenino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Paola Berenice	Rodriguez	Mata	7066	2*	paolaRodriguez@gmail.com	19	Femenino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 
Seldy Lizeth	Caulch	Chan	7094	2*	lizethChan@gmail.com	20	Femenino	Ingeniería en Innovación Agrícola	Innovación Agrícola 2A	 

Ilustración 102. visualización de cuestionarios resueltos del alumno

De igual forma se puede visualizar los resultados del cuestionario con los porcentajes obtenidos por categorías al igual que exportar la información en formato PDF.

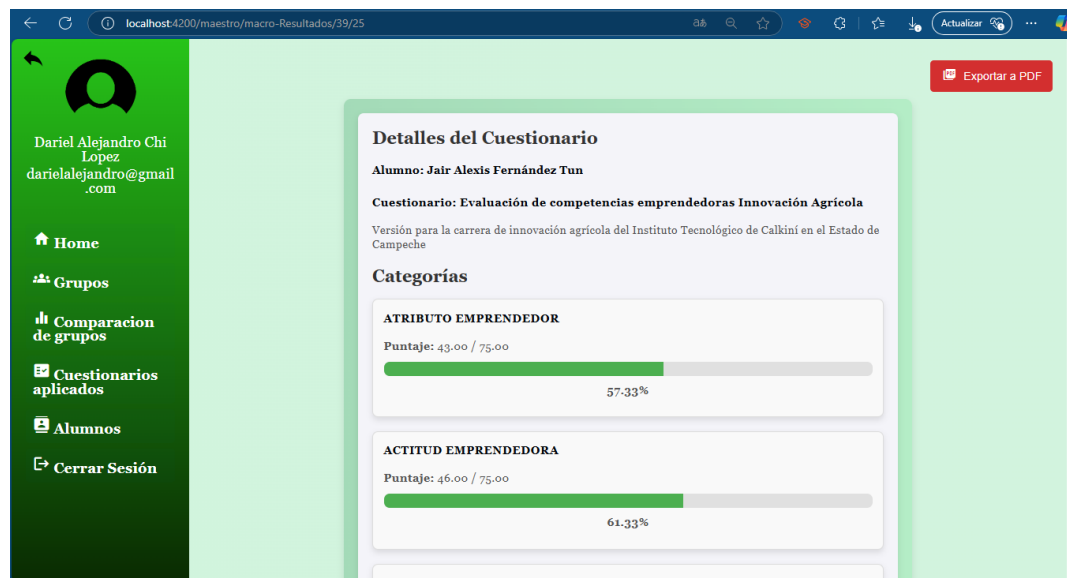


Ilustración 103.1. visualización de resultados macro del

3.7.12 Dashboard del administrador.

Al iniciar sesión como administrador se redirigirá a su respectivo Dashboard, el cual contendrá su menú con las funciones que le corresponde, se podrá visualizar la cantidad de alumnos y maestros existentes, así como informes, instituciones y cuestionarios aplicados.

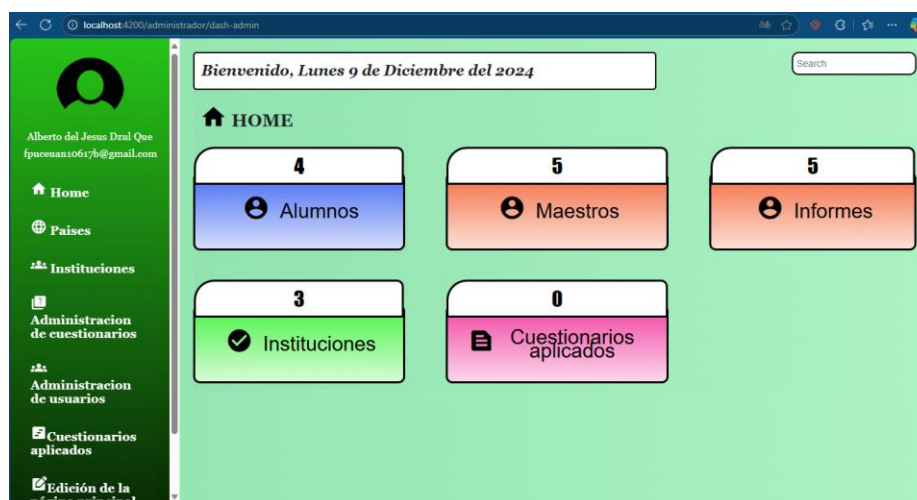


Ilustración 104. Dashboard del Administrador

3.7.13 Apartado Países

Al darle clic en la sección países, el administrador podrá visualizar todos los países del sistema, con un filtro para buscar el país en específico.

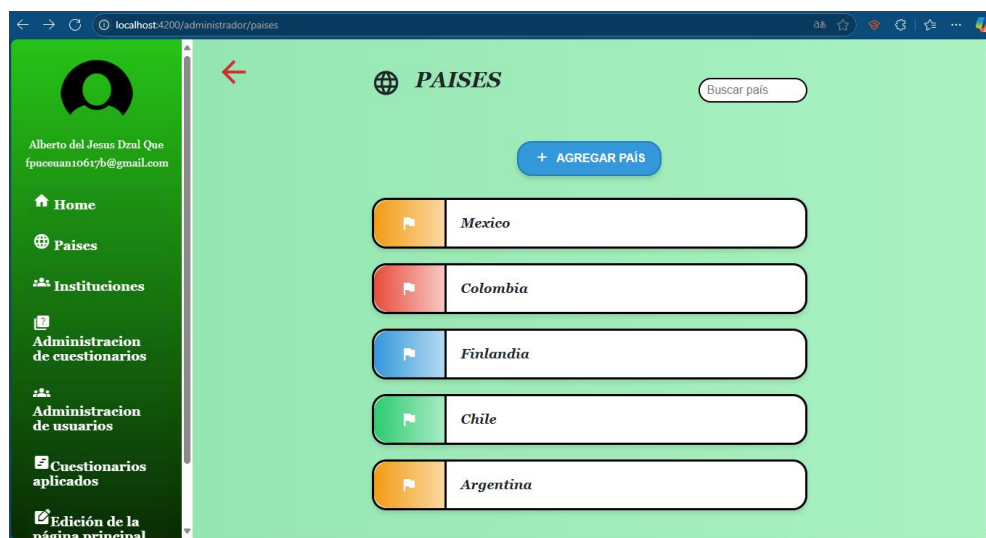


Ilustración 105. Sección de países

Al darle clic en el botón “Agregar País”, se desplegará una ventana para crear un nuevo país al sistema, al darle en agregar guardará el país al sistema.

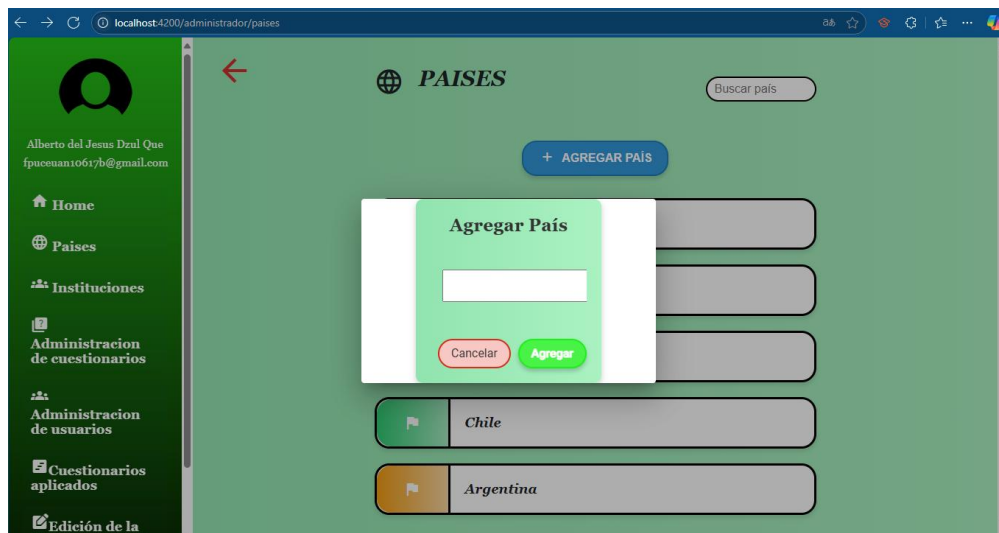


Ilustración 106. Agregar país

De igual manera al darle clic a un país en específico mostrará las instituciones que pertenecen a ese país.



Ilustración 107. Instituciones por país

3.7.14 Apartado de instituciones.

Al ingresar al apartado de instituciones se visualizará todas las instituciones existentes por defecto, de igual manera se agregó un filtro para visualizar las instituciones que pertenezcan a un país en específico.

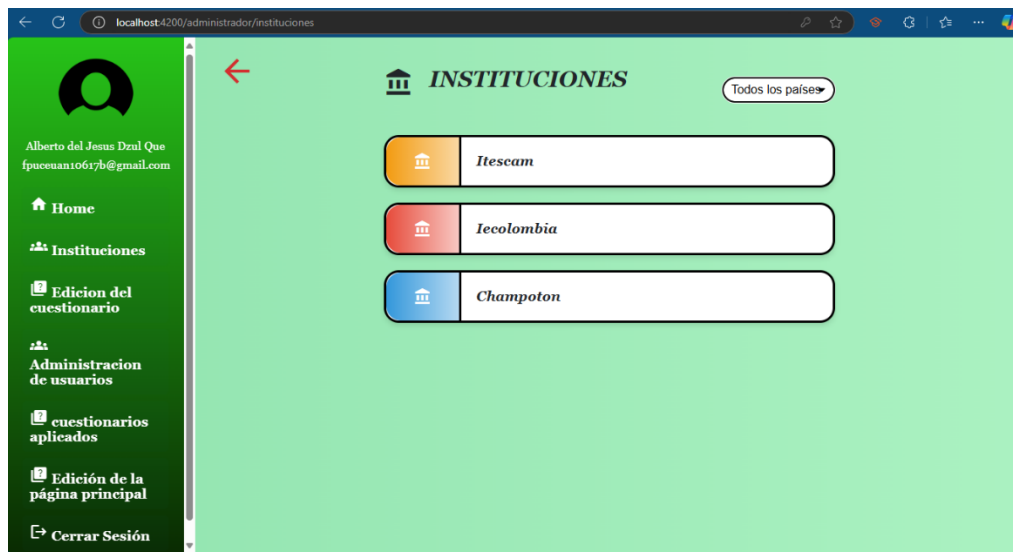


Ilustración 108. Lista de instituciones



Ilustración 109. Instituciones filtradas

3.7.15 Administración de los usuarios por institución

Al ingresar a la institución, redirigirá al administrador a la administración de usuarios de esa institución.



Ilustración 3.54 Administración de usuarios por institución

Al ingresar a maestros se visualizará la lista de maestros que pertenecen a la institución con la opción de agregar a un nuevo maestro, así como visualizar los grupos que pertenecen al maestro o eliminarlas de la institución.

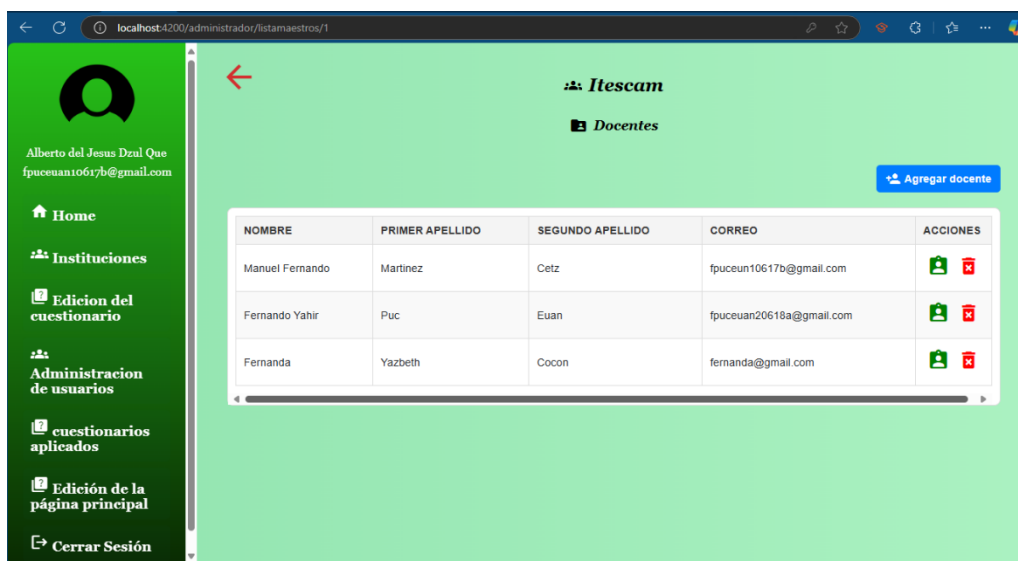


Ilustración 110. Lista de docentes por institución

Al darle clic al botón “agregar docente” redirigirá al administrador al formulario para añadir un nuevo docente con el país e institución preestablecido.

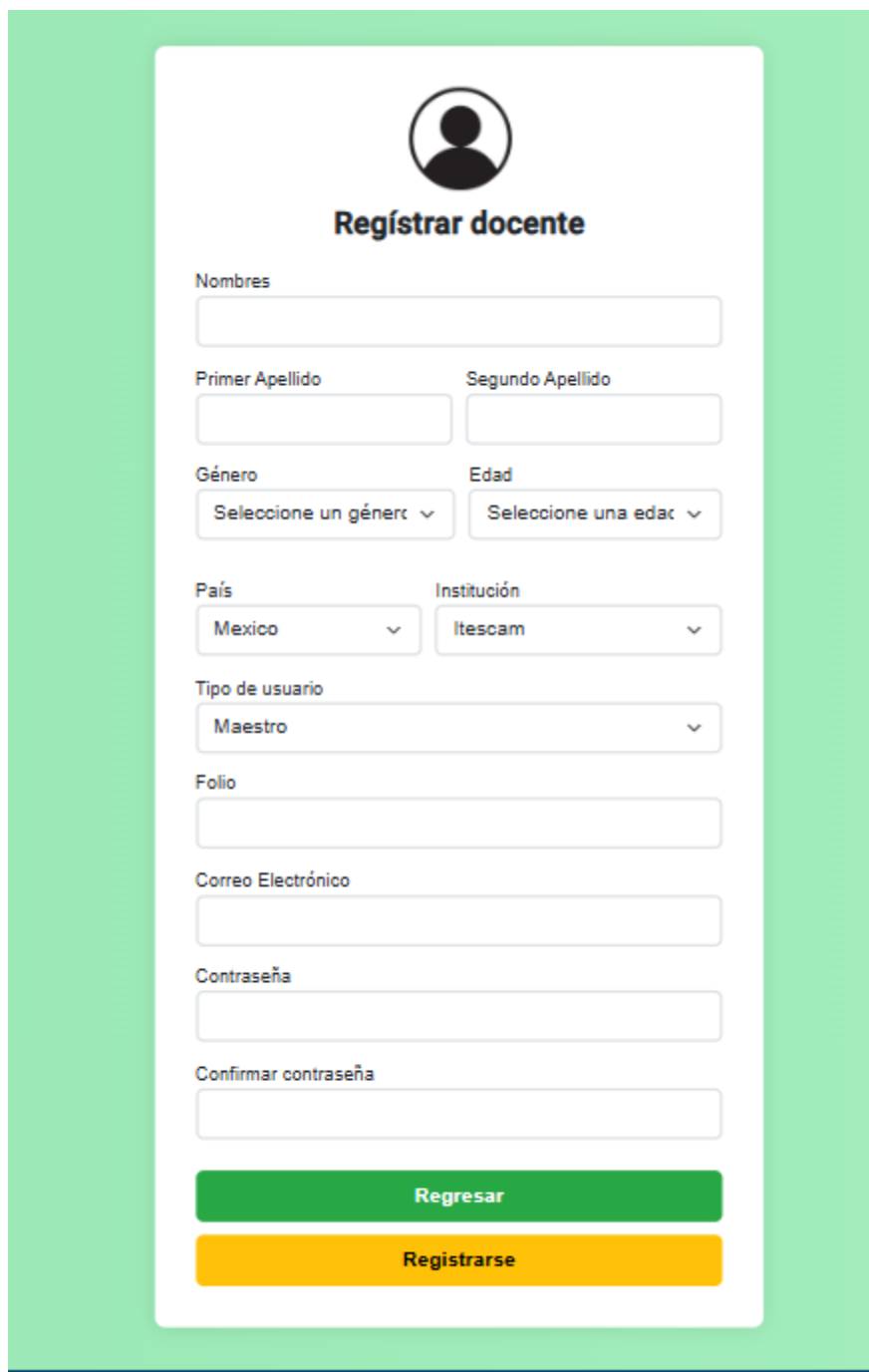


Ilustración 111. Registro de maestro

Formulario de registro de docente:

- Regístrar docente**
- Nombres**:
- Primer Apellido**:
- Segundo Apellido**:
- Género**:
- Edad**:
- País**:
- Institución**:
- Tipo de usuario**:
- Folio**:
- Correo Electrónico**:
- Contraseña**:
- Confirmar contraseña**:
- Regresar** (botón verde)
- Registrarse** (botón amarillo)

Ilustración 111. Registro de maestro

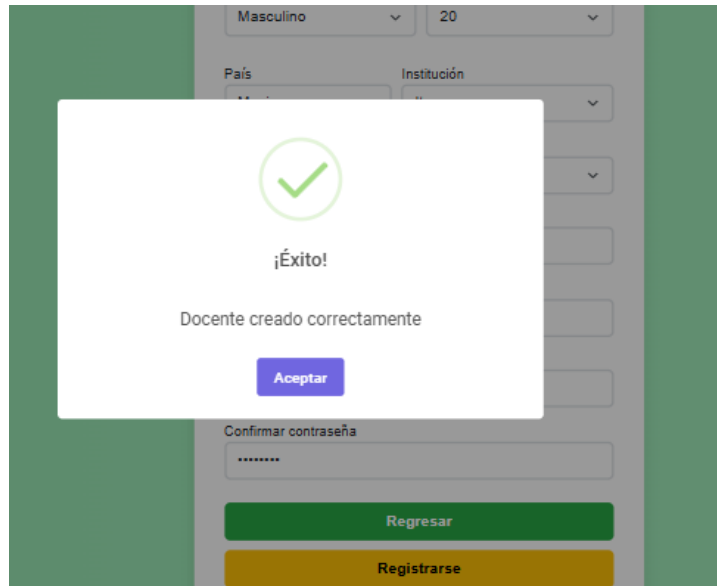


Ilustración 112. Docente creado correctamente

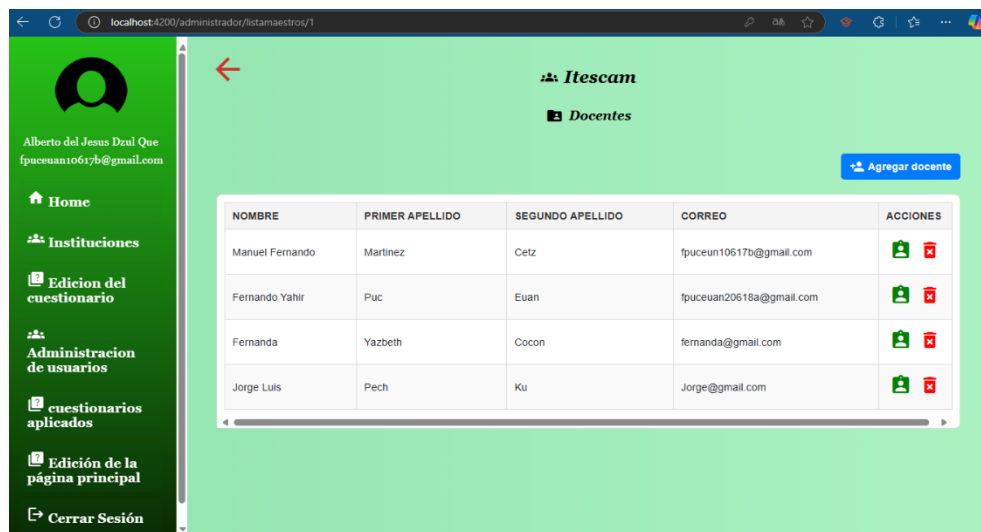


Ilustración 113. Lista de docentes actualizada

Luego redirigirá a la lista de maestros, posteriormente en la lista de maestros al darle clic en el icono se podrán observar los grupos que posee el maestro seleccionado

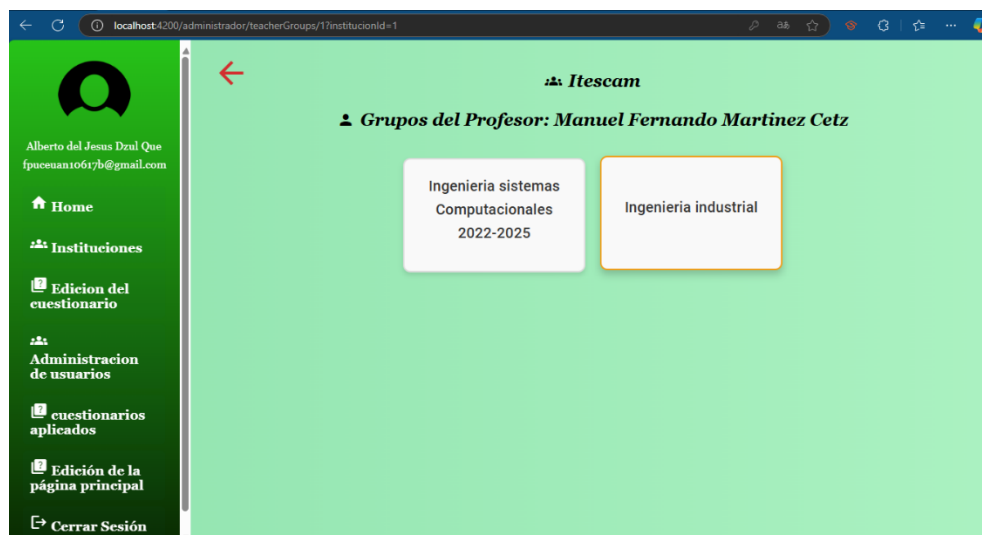


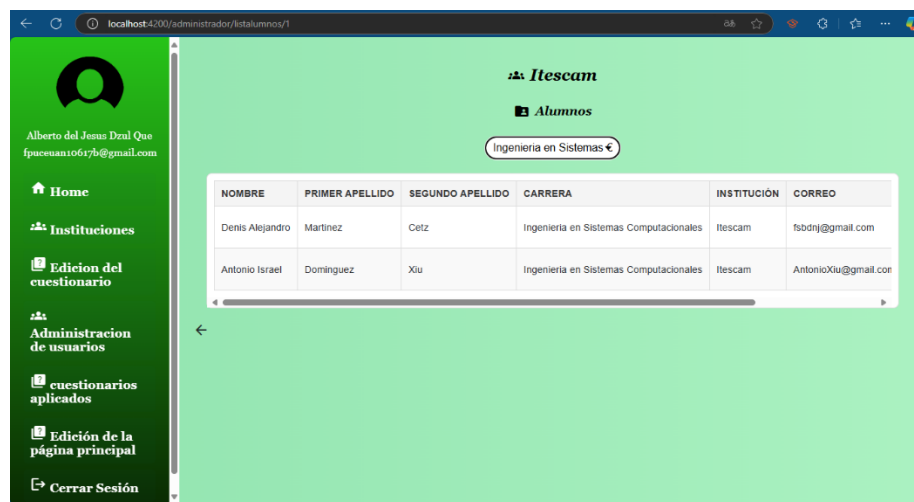
Ilustración 114. Lista de grupos del docente

Luego de ingresar al grupo se podrán observar los integrantes, junto con la información del grupo.



Ilustración 115. Lista de integrantes del grupo

Ahora regresando al apartado de administración de usuarios por institución, el maestro podrá visualizar la lista de todos los alumnos o filtrada por carrera.



The screenshot shows a web application interface. On the left is a green sidebar with a user profile at the top (Alberto del Jesus Dzul Quo, fpuceuan10617b@gmail.com) and a list of navigation links: Home, Instituciones, Edición del cuestionario, Administración de usuarios, Cuestionarios aplicados, Edición de la página principal, and Cerrar Sesión. The main content area has a light green background. At the top, it displays the 'Itescam' logo and the title 'Alumnos'. Below this is a dropdown menu currently set to 'Ingeniería en Sistemas'. A table lists the students, with columns for NOMBRE, PRIMER APELLIDO, SEGUNDO APELLIDO, CARRERA, INSTITUCIÓN, and CORREO. Two students are listed: Denis Alejandro Martinez Celz and Antonio Israel Dominguez Xiu, both from Itescam and studying Ingeniería en Sistemas Computacionales.

NOMBRE	PRIMER APELLIDO	SEGUNDO APELLIDO	CARRERA	INSTITUCIÓN	CORREO
Denis Alejandro	Martinez	Celz	Ingeniería en Sistemas Computacionales	Itescam	fsbdnj@gmail.com
Antonio Israel	Dominguez	Xiu	Ingeniería en Sistemas Computacionales	Itescam	AntonioXiu@gmail.com

Ilustración 116. Lista de alumnos de la institución

3.7.16 Administración de usuarios del sistema

Ahora al darle clic en el menú de administración de usuarios en general, se podrá visualizar el menú para visualizar la lista de todos los usuarios del sistema.

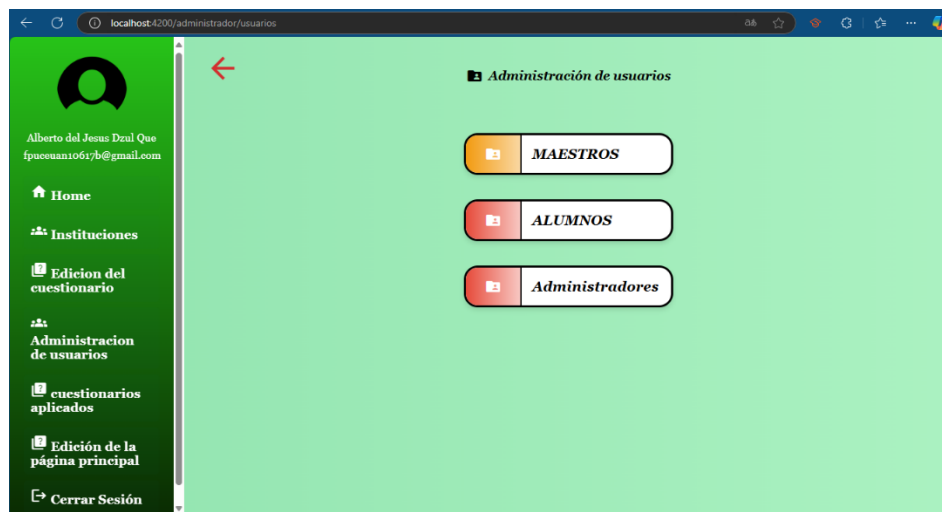


Ilustración 117. Administración de usuarios del sistema

Al ingresar a los maestros, se podrá visualizar la lista de todos los maestros, también filtrarlos por institución y país.

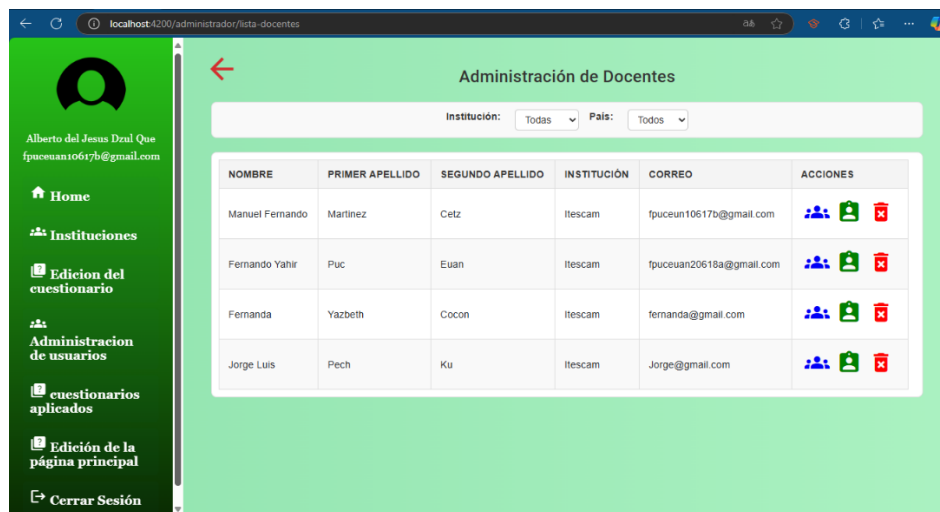


Ilustración 118. Lista de docentes del sistema

De la misma manera con los alumnos se podrá visualizar la lista de todos los alumnos con filtros por carrera, institución y país.

NOMBRE	PRIMER APELLIDO	SEGUNDO APELLIDO	CARRERA	INSTITUCIÓN	PAÍS	CORREO
Mauricio	Colli	Uru	Industrial	Itescam	Mexico	fernandoj
Denis Alejandro	Martinez	Cetz	Ingeniería en Sistemas Computacionales	Itescam	Mexico	fsbnj@g
Maria Isabel	Rodriguez	Pech	Ingeniería en Mecatronica	Itecolombia	Colombia	maria@gr
Antonio Israel	Dominguez	Xiu	Ingeniería en Sistemas Computacionales	Itescam	Mexico	AntonioXi

Ilustración 119. Lista de alumnos del sistema

3.7.17 Administración de cuestionarios

En la sección administración de cuestionarios el administrador al ingresar podrá visualizar la lista de las versiones del cuestionario que existen en el sistema, en este caso solamente el administrador podrá visualizarlo y podrá crear distintas versiones.

En la lista de cuestionarios se podrán observar todas las versiones existentes del cuestionario, los cuales esta ordenado en una lista lo cual incluye: Nombre, descripción del cuestionario, las fechas de publicación por grupo y un botón “Ver detalles” para visualizar la versión del cuestionario.



Ilustración 120. Lista de cuestionarios

3.7.18 Ver detalles del cuestionario

En este apartado al dar clic en el botón “ver detalles” el administrador podrá ver la información del cuestionario, se visualiza el nombre, la descripción, las fechas de publicación por grupo. De igual manera todas las preguntas con su signo y variables los cuales se visualizarán 10 preguntas por página incluyendo un botón para regresar a la lista de cuestionarios.

The screenshot shows a web application interface. On the left is a green sidebar with a user profile for 'Alberto del Jesus Dzul Que' and a list of menu items: Home, Países, Instituciones, Administración de cuestionarios, Administración de usuarios, cuestionarios aplicados, and Edición de la página principal. The main content area has a title 'Evaluación de competencias emprendedoras Innovación Agrícola'. Below the title is a description: 'Versión para la carrera de innovación agrícola del Instituto Tecnológico de Calkiní en el Estado de Campeche'. It also shows 'Fechas por Grupo:' with 'Grupo: Innovación Agrícola 2A' and 'Publicación: 2/28/25'. The 'Preguntas' section lists three items, each with a 'Signo: Positiva' and a 'Variables' field containing a text input. The first question is '1. Me esmero en buscar cosas que necesitan hacerse.' with the variable 'Iniciativa'. The second is '2. Cuando me enfrento a un problema difícil invierto gran cantidad de tiempo en encontrar una solución.' with the variable 'Persistente'. The third is '3. Termino mi trabajo a tiempo.'

Ilustración 121. Detalles del cuestionario

This screenshot shows the continuation of the questionnaire details. It displays questions 9 and 10. Question 9 is '9. Logro que otros apoyen mis recomendaciones.' with the variable 'Persuacion'. Question 10 is '10. Me siento confiado en que puede tener éxito en cualquier actividad que me propongo ejecutar.' with the variable 'Autoconfianza'. At the bottom of the main content area, there are four buttons: 'Anterior' (disabled), 'Página 1 de 6', 'Siguiente' (disabled), and 'Regresar a la lista' (active).

Ilustración 122. Detalles del cuestionario

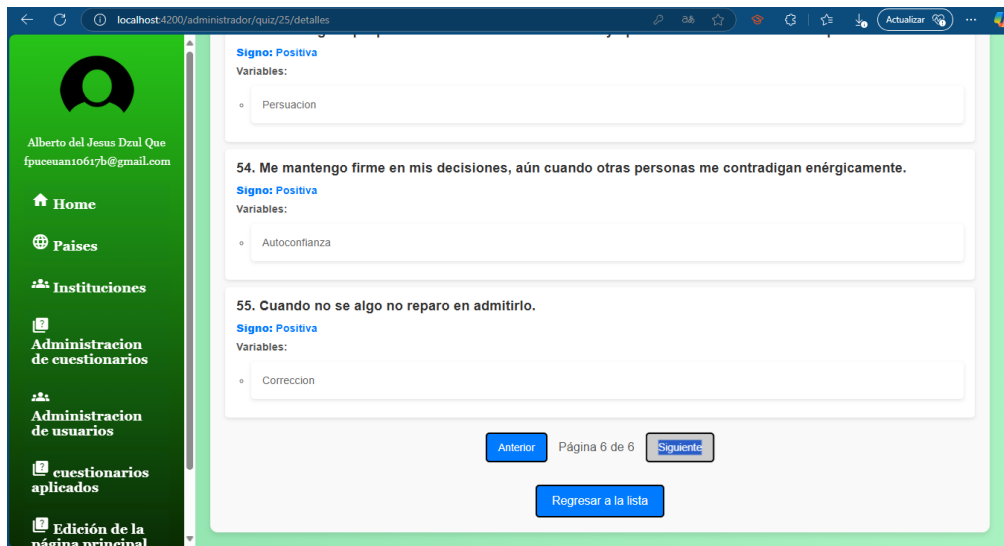


Ilustración 123. Detalles del cuestionario (última página)

3.7.19 Crear una nueva versión del cuestionario.

En este apartado el administrador puede crear una nueva versión del cuestionario, asignando el nombre de la versión y su descripción. por ende, al ingresar a esta sección se muestran las preguntas del cuestionario original ya que se crea una nueva versión de este. De igual manera se solicitó que solamente se pueda modificar las preguntas y el signo de la pregunta y las variables se mantienen intactas para respetar el formato del cuestionario.

localhost:4200/administrador/create-quiz

Alberto del Jesus Dzul Que
fpuceuan10617b@gmail.com

Home
Países
Instituciones
Administración de cuestionarios
Administración de usuarios
Cuestionarios aplicados
Edición de la página principal
Cerrar Sesión

CREAR UN NUEVO CUESTIONARIO

Nombre
Evaluación de competencias emprendedoras Innovación Agrícola

Descripción
Versión para la carrera de innovación agrícola del Instituto Tecnológico de Calkiní en el Estado de Campeche

PREGUNTAS DEL CUESTIONARIO

Pregunta	Variables	Signo
Pregunta 1 Me esmero en buscar cosas que neces	Iniciativa	Positiva
Pregunta 2 Cuando me enfrento a un problema difi	Persistente	Positiva
Pregunta 3 Termino mi trabajo a tiempo.	Cumplimiento	Positiva
Pregunta 4 Me molesta cuando las cosas no se ha	Eficiencia	Negativa

Ilustración 124. Crear nueva versión del cuestionario

localhost:4200/administrador/create-quiz

Alberto del Jesus Dzul Que
fpuceuan10617b@gmail.com

Home
Países
Instituciones
Administración de cuestionarios
Administración de usuarios
Cuestionarios aplicados
Edición de la página principal
Cerrar Sesión

Pregunta 51 Me valgo de varias fuentes de informac	Buscar	Positiva
Pregunta 52 Si no resulta un determinada estrategi	Planificación	Positiva
Pregunta 53 Puedo lograr que personas con firmes	Persuasion	Positiva
Pregunta 54 Me mantengo firme en mis decisiones,	Autoconfianza	Positiva
Pregunta 55 Cuando no se algo no reparo en admiti	Correccion	Positiva

Anterior Siguiente

Guardar Cuestionario Cancelar

Ilustración 125. Crear nueva versión del cuestionario (última pregunta)

3.7.20 Instrucciones del cuestionario

En este apartado se muestran las instrucciones del cuestionario que observará el alumno.

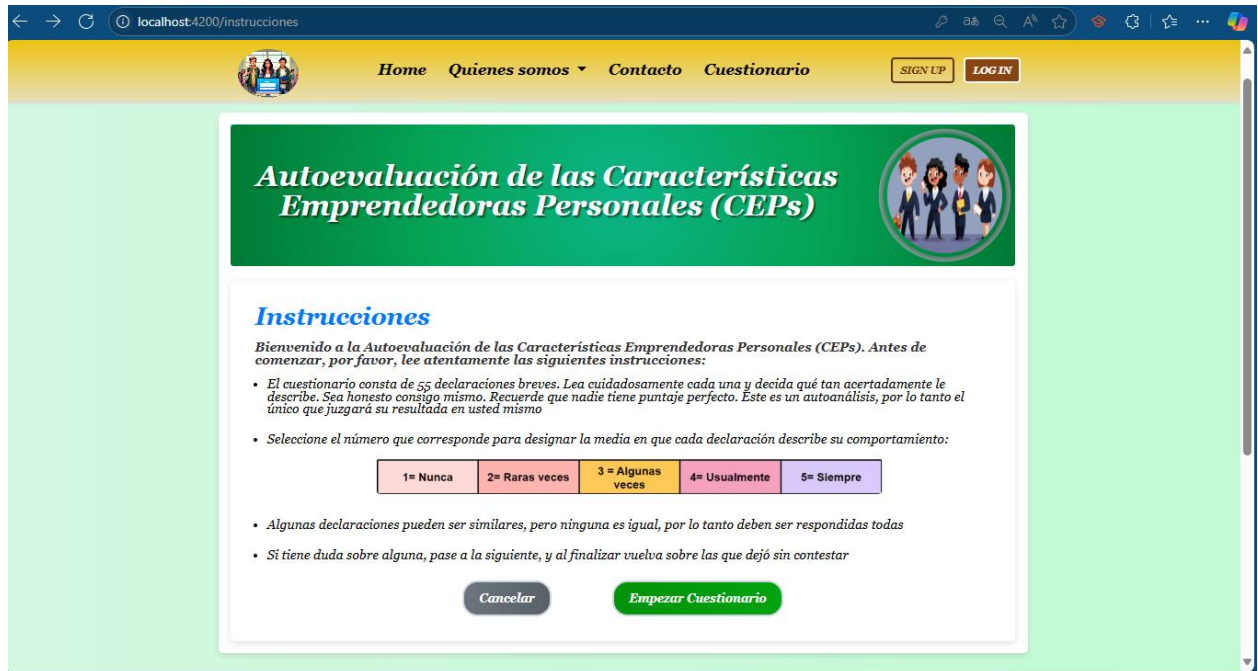


Ilustración 126. Vista de instrucciones del cuestionario

3.7.21 Generación de informes de los resultados.

El sistema incluye una funcionalidad específica para la generación de informes, diseñada para facilitar el acceso y análisis de los resultados obtenidos en las evaluaciones. Esta herramienta está disponible para los administradores, quienes pueden generar reportes detallados con información clave sobre el desempeño de los estudiantes.

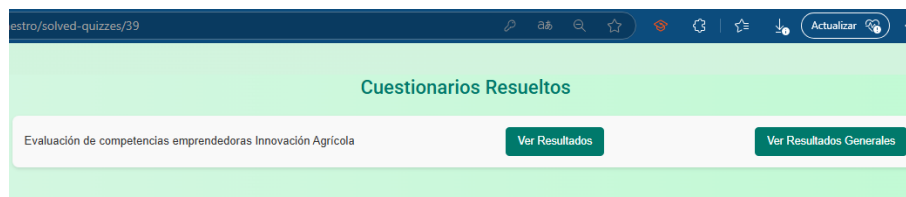


Ilustración 127. Cuestionarios resueltos del alumno.

Por lo que el administrador puede ingresar a los resultados de los alumnos del cuestionario el cual seleccione, al ingresar podrá observar la información del alumno y el cuestionario. Posteriormente visualizará el porcentaje obtenido de los alumnos por categorías (Macro).

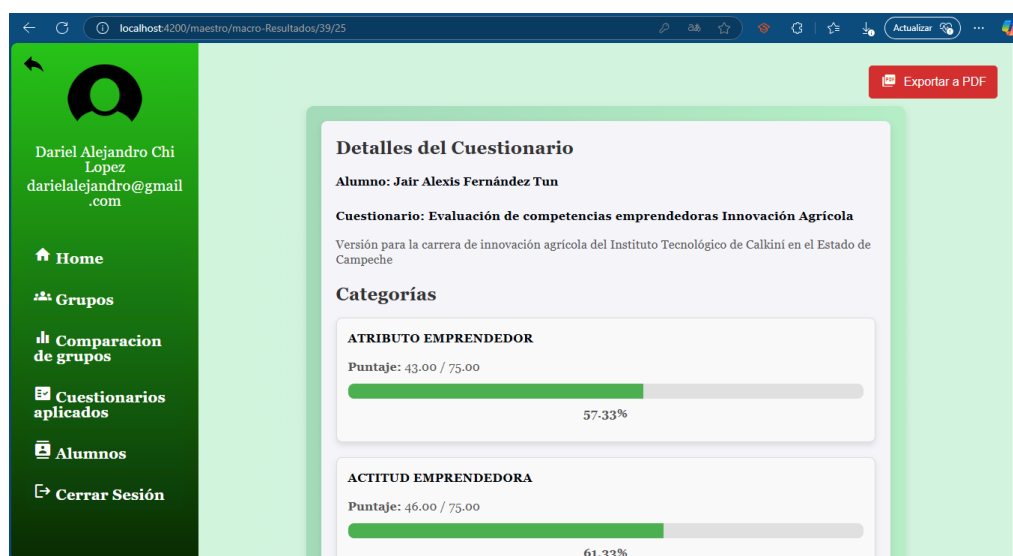


Ilustración 128. Análisis de resultados por categoría.

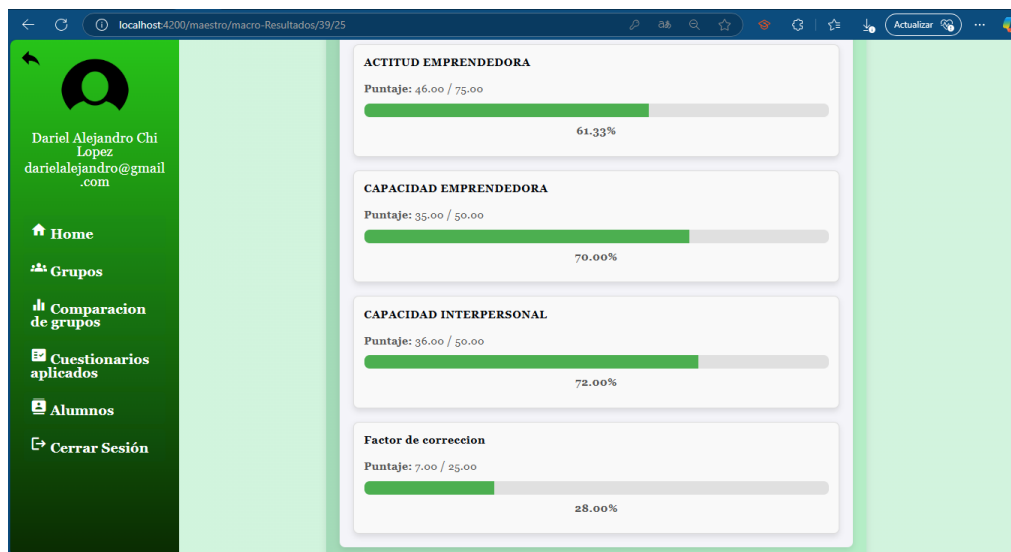


Ilustración 129. Análisis de resultados por categoría parte 2.

El administrador al ingresar al puntaje de la categoría podrá observar los resultados por expresión (Micro). En este de igual forma se observan los detalles del cuestionario y los puntajes obtenidos por expresión.

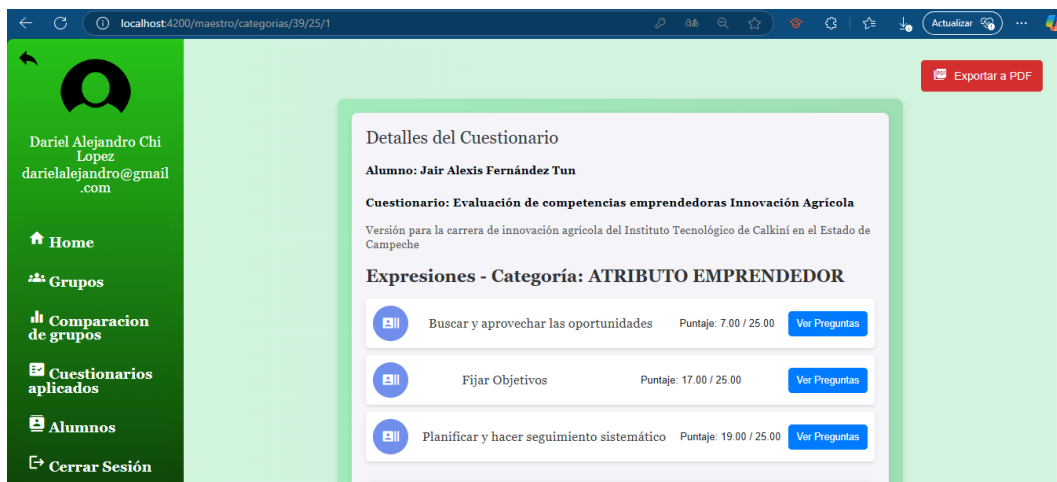


Ilustración 130. Análisis de resultados por expresión.

De igual forma de podrá observar una gráfica de los puntajes obtenidos que pertenecen a la categoría.

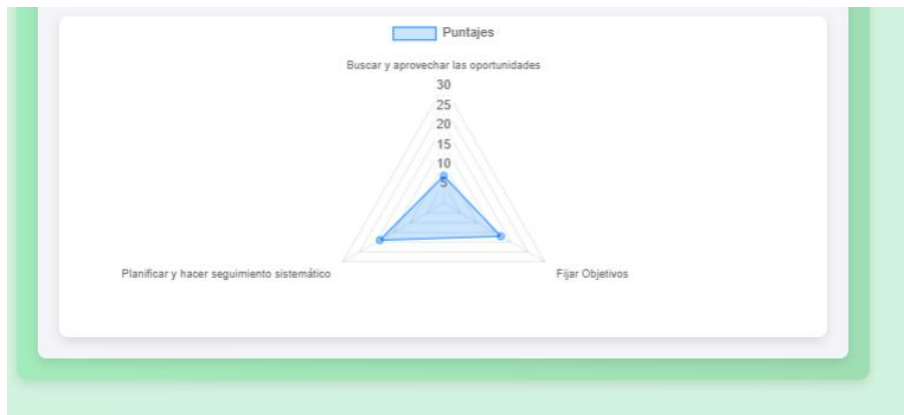


Ilustración 131. Gráfica de resultados por expresión.

El administrador podrá realizar un análisis más preciso al ingresar con el botón “Ver Preguntas” para la visualización a las preguntas que corresponden a la expresión seleccionada. En ella puede observar la información del alumno y cuestionario, así como una tabla con las preguntas, las respuestas del alumno y el puntaje máximo. Finalmente cuenta con la opción de exportar la información en formato PDF.

Detalles del Cuestionario

Alumno: **Jair Alexis Fernández Tun**

Cuestionario: **Evaluación de competencias emprendedoras Innovación Agrícola**

Versión para la carrera de innovación agrícola del Instituto Tecnológico de Calkini en el Estado de Campeche

Preguntas - Expresión: Buscar y aprovechar las oportunidades

Pregunta	Respuesta del alumno	Puntaje máximo
② Me esmero en buscar cosas que necesitan hacerse.	3	5
② Hago lo que se necesita hacer sin que otros tengan que pedirme que lo haga.	5	5
② Me gustan los desafíos y nuevas oportunidades.	5	5
⑦ Prefiero desempeñar tareas que domino a la perfección y en las que me siento seguro.	5	5
⑦ Me aventura a hacer cosas nuevas y diferentes de lo que he realizado en el pasado.	3	5

Ilustración 132. Resultados por pregunta – Expresión.

El administrador de igual forma puede obtener el informe general del alumno como se visualiza en la siguiente imagen con la opción de exportarlo en formato PDF.



Ilustración 133. Información del alumno.

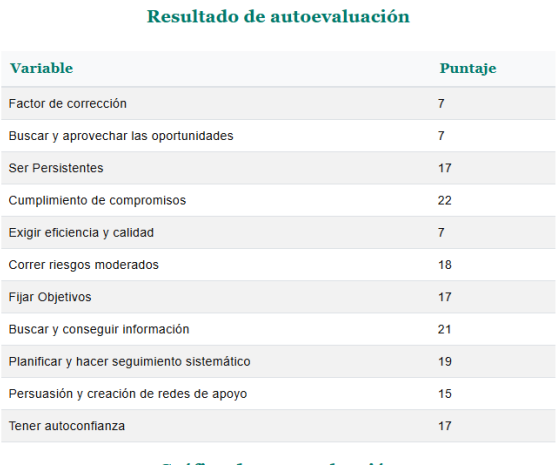


Ilustración 134. Resultados de autoevaluación.



Ilustración 135. Gráfico de autoevaluación.

Es importante destacar que el módulo de retroalimentación sigue en desarrollo, por lo que se continuará trabajando en la optimización de esta funcionalidad. La meta es proporcionar un sistema más robusto que no solo presente los resultados de manera clara, sino que también ofrezca recomendaciones automáticas y personalizadas a los estudiantes, basadas en su desempeño en las evaluaciones para los usuarios.

3.8 PRUEBAS DEL SISTEMA.

Se realizaron pruebas del sistema para garantizar la correcta funcionalidad y usabilidad, por ende, se llevaron a cabo diversas pruebas enfocadas en el módulo de bienvenida, módulo de inicio de sesión y el módulo del alumno, ya que este fue el principal usuario objetivo en esta etapa del desarrollo. Dado que las interfaces de maestro y administrador aún se encuentran en proceso de implementación, se decidió excluirlas de las pruebas en esta fase, priorizando aquellas funcionalidades que ya están completamente desarrolladas y operativas.

Módulo	Submódulo	Prueba	Descripción	Resultado	Observaciones
Módulo de bienvenida	Home	Visualización del Home	Verificación de las secciones que componen el Home se visualicen correctamente.	Correcto	Todas las secciones se visualizan correctamente.
	Menú de navegación	Acceso a las secciones del menú.	Validación de los botones de navegación del menú.	Correcto	Todos los botones funcionan correctamente.
	Sección del cuestionario	Acceso al apartado del cuestionario.	Verificación de acceso a la sección del cuestionario para la visualización del mensaje de advertencia del cuestionario.	Correcto	Sin observaciones.
Módulo de inicio de sesión	Registro	Registro de usuario.	Validación de datos ingresados y restricciones.	Funcional	Mejorar la validación de contraseñas y correos.
	Inicio de sesión	Inicio de sesión del usuario.	Comprobación del acceso con las credenciales correctas Matricula o Correo y Contraseña.	Correcto	Funciona y redirige al usuario a su determinado Dashboard.
	Recuperación de contraseña	Recuperación de contraseña del usuario.	Verificación del proceso de restablecimiento de contraseña.	No disponible	Función aun en desarrollo, se integrará en futuras actualizaciones.

Tabla 5. Pruebas realizadas de los módulos de bienvenida e inicio de sesión.

Módulo	Submódulo	Prueba	Descripción	Resultado	Observaciones
Módulo Perfil de Usuario	Perfil de estudiante	Visualización del Dashboard.	Visualizar el Dashboard del alumno con las secciones que le corresponden y el funcionamiento de este.	Correcto	Sin observaciones.
		Edición del perfil.	Prueba de modificación de los datos del usuario.	Funcional	Funciona correctamente a excepción de la foto del perfil del usuario.
		Visualización del historial de cuestionarios.	Comprobación del acceso a cuestionarios que han sido completados.	Correcto	Sin observaciones.
Módulo gestión de grupos	Ingreso a un grupo	Verificar el ingreso a un grupo.	Verificar el ingreso del código correcto, y la confirmación del ingreso al grupo.	Correcto	Sin observaciones.
	Visualización de grupos	Verificación de la lista de grupos.	Verificación de los grupos a los que pertenece el usuario.	Funcional	Funciona correctamente.
Módulo de cuestionarios	Acceso a cuestionarios	Acceso a los cuestionarios.	Verificar el acceso a los cuestionarios disponibles que haya asignado el docente.	Funcional.	Sin observaciones.
	Resolución de cuestionarios	Realización de los cuestionarios.	Verificar el almacenamiento de las respuestas en la base de datos.	Correcto	Sin observaciones.
	Restricción de cuestionarios	Restricción del acceso a cuestionarios que hayan sido cerrados o completados.	Verificar que el usuario no pueda acceder a los cuestionarios que hayan sido cerrados o completados.	Correcto	Sin observaciones.

Tabla 6. Pruebas realizadas de los módulos de perfil de usuario, gestión de grupos y cuestionarios.

Módulo	Submódulo	Prueba	Descripción	Resultado	Observaciones
Módulo de resultados y retroalimentación	Visualización de resultados	Visualizar los resultados de la autoevaluación del estudiante.	Generación correcta del informe con el puntaje obtenido y el gráfico de resultados.	Funcional.	Funciona correctamente.
	Retroalimentación	Retroalimentación personalizada.	Evaluación de la claridad de las recomendaciones.	No disponible	Se sigue trabajando en esta área, se agregará en próximas versiones.
Módulo de exportación de resultados	Generación de informes	Generación de informes individuales de los resultados obtenidos del estudiante.	Intento de exportación de resultados en formato PDF.	Funcional	Funciona correctamente.

Tabla 7. Pruebas realizadas de los módulos de resultados y retroalimentación y exportación de resultados.

CAPITULO 4. RESULTADOS Y DISCUSION.

El propósito de esta sección es analizar los resultados obtenidos por un estudiante de segundo semestre de la carrera de Innovación Agrícola del Instituto Tecnológico de Calkiní en el Estado de Campeche en la evaluación de sus competencias emprendedoras dentro de la plataforma web desarrollada. Además, se evaluará si la plataforma cumple con los criterios propuestos en la hipótesis, permitiendo una autoevaluación eficiente y proporcionando resultados esperados.

El estudiante se registró de manera exitosa a la plataforma el cual posteriormente inició sesión a la plataforma con sus credenciales.

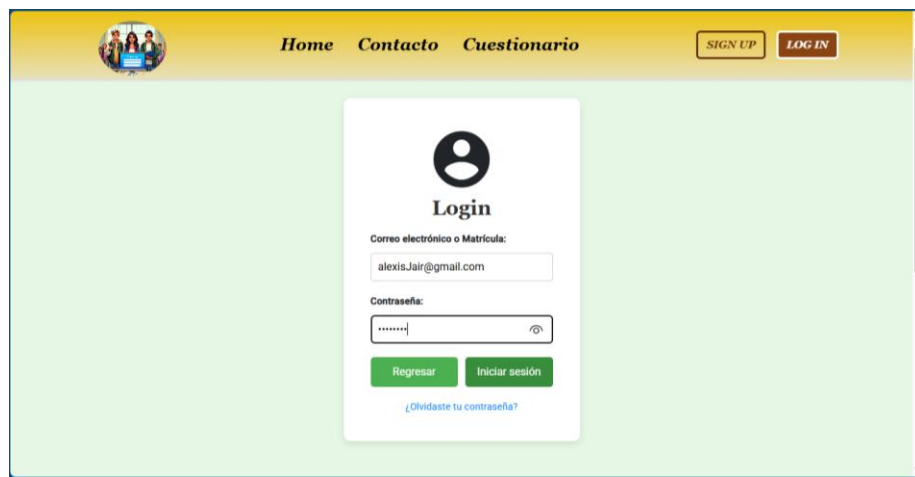


Ilustración 136. Inicio de sesión del alumno.

Seguidamente el sistema lo redirige al Dashboard del estudiante donde se observan las secciones en las cuales puede ingresar. Se observan que los contadores de las secciones grupos, resultados, cuestionarios resueltos y cuestionarios pendientes están en cero ya que es un usuario nuevo en el sistema.

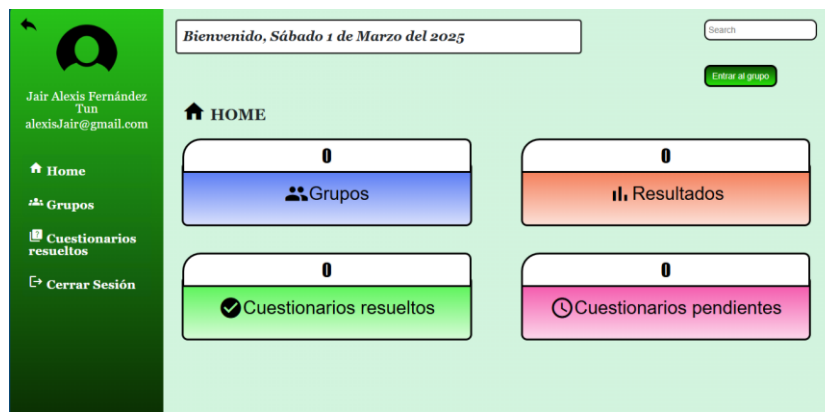


Ilustración 137. Inicio de sesión del alumno.

El docente previamente proporcionó un código de acceso por el cual podrá ingresar al grupo donde se encuentra el cuestionario publicado por el mismo. El estudiante ingresó a la opción de buscar grupo y posteriormente introdujo el código proporcionado previamente, al darle clic en buscar grupo aparecieron los datos del grupo y se unió al grupo.

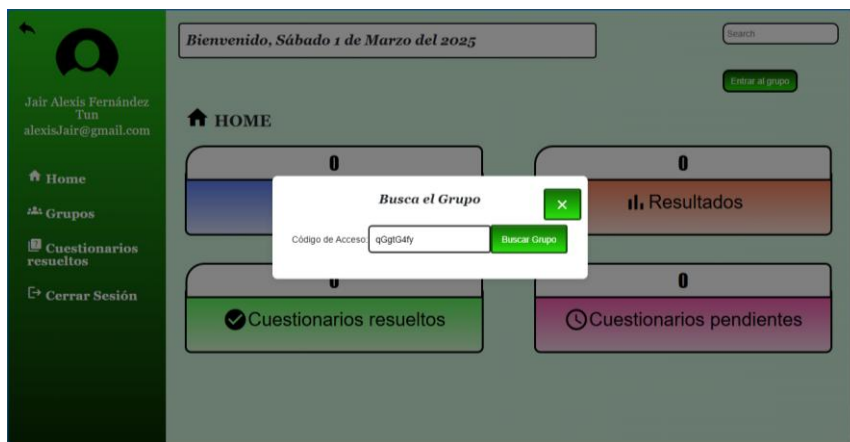


Ilustración 138. Buscar grupo por código de acceso.

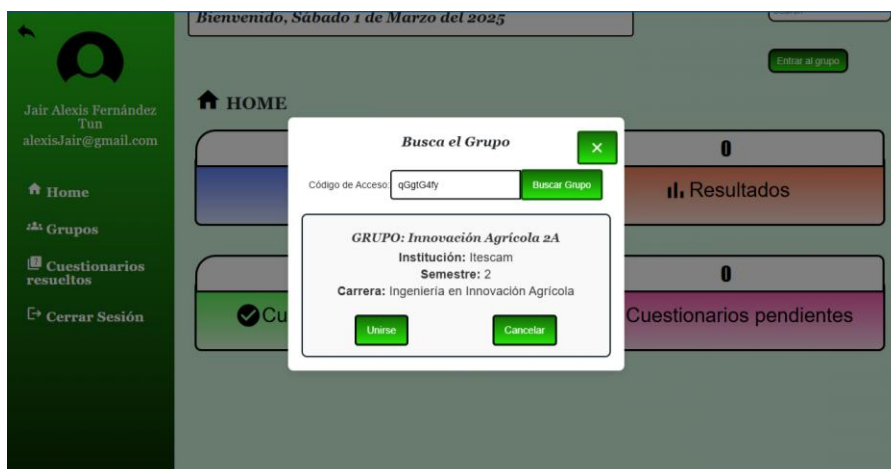


Ilustración 139. Información del grupo.

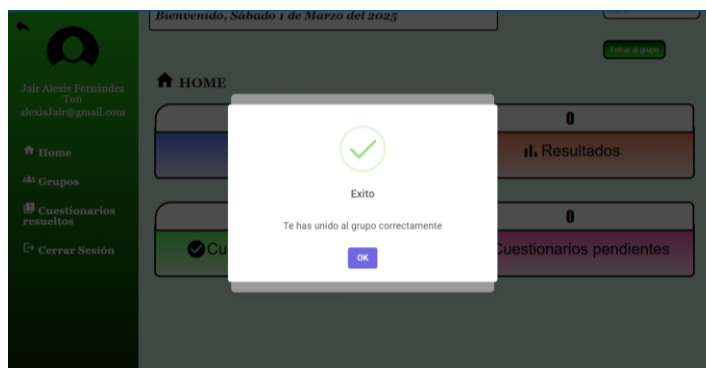


Ilustración 140. Ingreso exitoso al grupo.

El alumno una vez ingresado al grupo se redirige al Dashboard donde puede visualizar que ya está en un grupo y tiene un cuestionario pendiente por resolver.

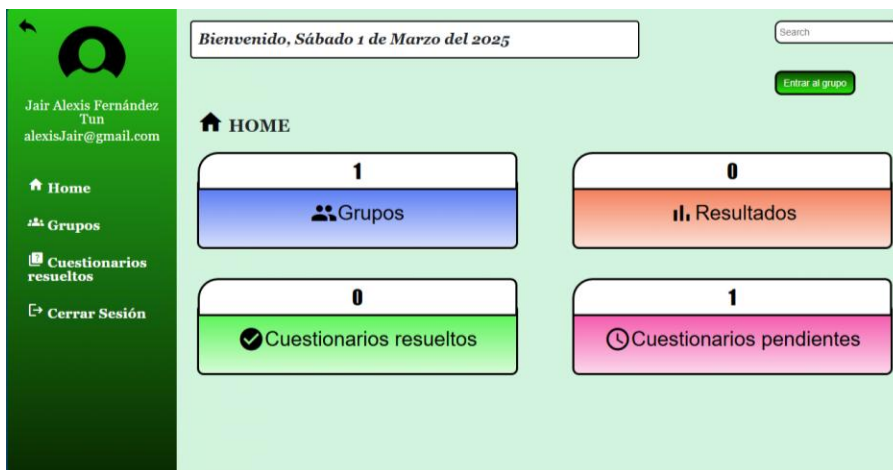


Ilustración 141. Dashboard del estudiante.

El alumno procede a dirigirse a los cuestionarios pendientes en donde le aparecerá solamente los cuestionarios disponibles asignados por el profesor. Dichos cuestionarios serán presentados en formas de tarjetas con información importante. El alumno procede a ingresar al cuestionario.

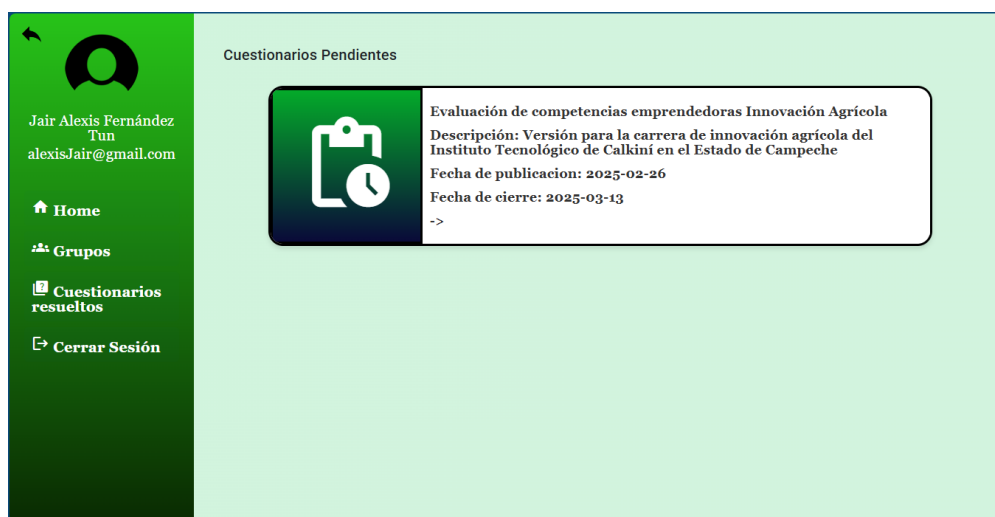


Ilustración 142. Cuestionarios pendientes del estudiante.

Una vez ingresado el cuestionario el alumno observará la información del cuestionario y comienza con el proceso de la autoevaluación.

The screenshot shows a web interface for a questionnaire. On the left is a green sidebar with a user profile (Jair Alexis Fernández Tun, alexisJair@gmail.com) and navigation links: Home, Grupos, Cuestionarios resueltos, and Cerrar Sesión. The main content area has a green header with the title 'Autoevaluación de las Características Emprendedoras Personales (CEPs)' and an illustration of three business people. Below the header, the section 'Instrucciones' (Instructions) is displayed. It includes a welcome message and two bullet points: 'El cuestionario consta de 55 declaraciones breves. Lea cuidadosamente cada una y decida qué tan acertadamente le describe. Sea honesto consigo mismo. Recuerde que nadie tiene puntaje perfecto. Este es un autoanálisis, por lo tanto el único que juzgará su resultado es usted mismo' and 'Seleccione el número que corresponde para designar la media en que cada declaración describe su comportamiento:'. Below the text is a horizontal row of five colored boxes representing a Likert scale: 1= Nunca (pink), 2= Raras veces (light orange), 3= Algunas veces (yellow), 4= Usualmente (light blue), and 5= Siempre (purple).

Ilustración 143. Información del cuestionario.

This screenshot is similar to the previous one but includes additional instructions and buttons. The 'Instrucciones' section now includes a third bullet point: 'Algunas declaraciones pueden ser similares, pero ninguna es igual, por lo tanto deben ser respondidas todas' and a fourth: 'Si tiene duda sobre alguna, pase a la siguiente, y al finalizar vuelva sobre las que dejó sin contestar'. At the bottom of the instruction area are two buttons: 'Cancelar' (Cancel) and 'Empezar Cuestionario' (Start Questionnaire).

Ilustración 144. Instrucciones del cuestionario.

The screenshot shows the evaluation section of a questionnaire titled 'Evaluación de competencias emprendedoras Innovación Agrícola'. The sidebar is identical to the previous screenshots. The main content area has a green header with the title and an illustration of three business people. Below the header, it specifies 'Versión para la carrera de innovación agrícola del Instituto Tecnológico de Calkiní en el Estado de Campeche'. The first evaluation item is 'Me valgo de varias fuentes de información al buscar ayuda para llevar a cabo tareas o proyectos.' with radio button options: Nunca, Raras veces, Algunas veces, Usualmente, and Siempre (selected). The second item is 'Si no resulta una determinada estrategia para hacer frente a un problema, busco otro.' with radio button options: Nunca and Raras veces.

Ilustración 145. Título y descripción del cuestionario.

☐ Raras veces
☐ Algunas veces
☒ Usualmente
☐ Siempre

Me mantengo firme en mis decisiones, aún cuando otras personas me contradigan energicamente.

☐ Nunca
☐ Raras veces
☐ Algunas veces
☒ Usualmente
☐ Siempre

Cuando no se algo no reparo en admitirlo.

☐ Nunca
☐ Raras veces
☐ Algunas veces
☐ Usualmente
☒ Siempre

Anterior Enviar

Ilustración 146. Ultimas preguntas y respuestas del cuestionario.

Una vez completada la evaluación el sistema generó automáticamente los resultados del estudiante lo mostró en una pantalla organizando la información del estudiante y el cuestionario, también organizó los resultados de la autoevaluación en una tabla donde muestra las variables y puntajes por cada variable del cuestionario, finalmente generó la gráfica para presentar los puntajes obtenidos para la facilitar la interpretación.

Exportar a PDF

Resultados de Jair Alexis Fernández Tun (7072)

Nombre: Jair Alexis Fernández Tun
Correo: alexisJair@gmail.com
Semestre: 2º
Carrera: Ingeniería en Innovación Agrícola
Nombre del cuestionario: Evaluación de competencias emprendedoras Innovación Agrícola
Descripción: Versión para la carrera de innovación agrícola del Instituto Tecnológico de Calkiní en el Estado de Campeche
Fecha de publicación: 2025-03-01

Resultado de autoevaluación

Variable	Puntaje
Factor de corrección	7
Buscar y aprovechar las oportunidades	7
Ser Persistentes	17
Cumplimiento de compromisos	22
Resistencia y calidad	7

Ilustración 147. Información del alumno y cuestionario.

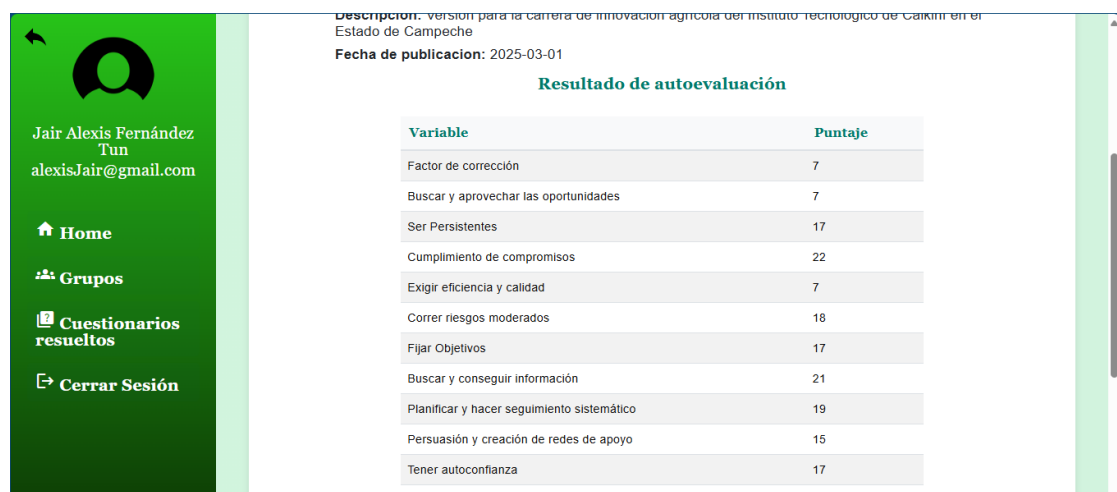


Ilustración 148. Tabla de resultados por puntaje y variable.



Ilustración 149. Gráfico araña de autoevaluación.

Con base a los resultados obtenidos y observaciones realizadas en el proceso realizado se puede afirmar que el estudiante pudo completar el cuestionario sin dificultades técnicas, de igual forma el estudiante aseguró que la interfaz del sistema es intuitiva por lo que no tuvo un problema al ingresar a cada sección y presentar el cuestionario. De igual forma la generación de resultados permitió al estudiante obtener un diagnóstico rápido eliminando la necesidad de cálculos manuales por parte del docente.

Para continuar con el análisis de resultados el sistema proporciona al docente la herramienta para visualizar los resultados de una forma más amplia, proporcionando los resultados macro y micro donde el podrá realizar un análisis más detallado.

El docente al iniciar sesión de igual forma que el estudiante, lo redirigirá a su respectivo Dashboard donde se mostrará las secciones las cual puede ingresar y los contadores de grupos creados, la cantidad de alumnos que tiene a su disposición y los cuestionarios aplicados. Previamente el docente había creado el grupo y proporcionado el código de acceso a los estudiantes, posteriormente publicó el cuestionario al grupo que le pertenece a la ingeniería en innovación agrícola 2A dichos procesos se describen en el subtema “3.7 Implementación del sistema”.

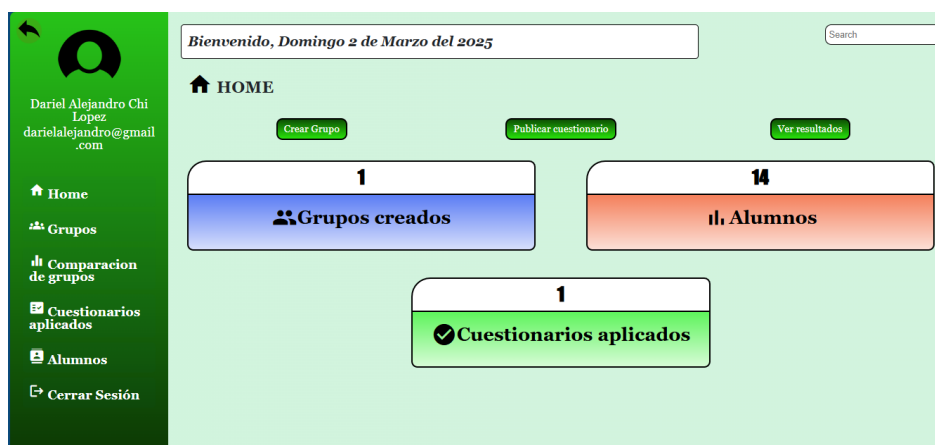


Ilustración 150. Dashboard del docente.

Ahora bien, para visualizar los resultados del cuestionario aplicado, el docente se dirige a la sección de “Grupos creados” o darle clic en el botón “Ver resultados”. En este caso el docente se dirige a la sección de grupos creados donde se observan la lista de grupos que ha creado el docente, la información de estos se muestra en forma de tarjeta con la información relevante del grupo.



Ilustración 151. Sección de grupos del docente.

Al ingresar al grupo el docente puede visualizar su información, sus cuestionarios aplicados y tiene las opciones de visualizar los integrantes del grupo o las estadísticas grupales que se obtuvieron en el cuestionario.

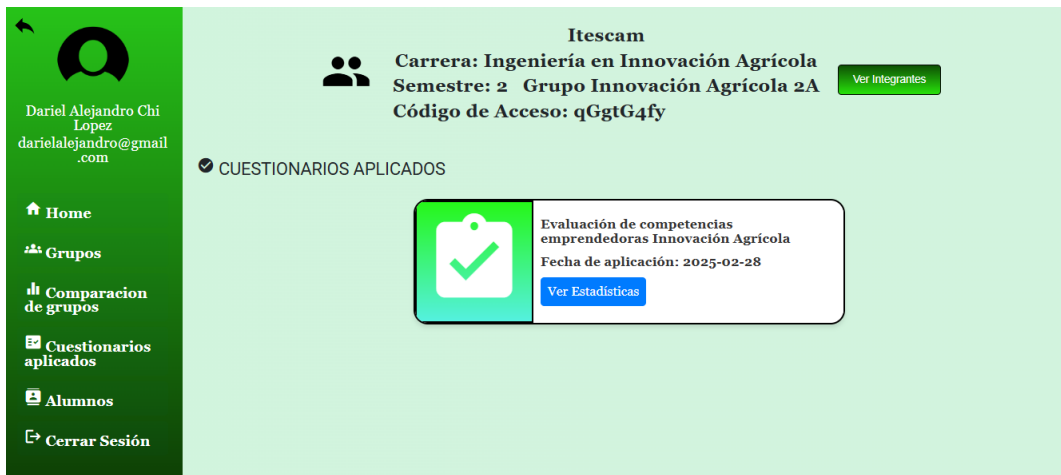


Ilustración 152. Visualización de cuestionarios aplicados del grupo.

Para el análisis de las estadísticas grupales el sistema cuenta con una sección diseñada específicamente para apoyar a docentes y administradores en la evaluación de competencias emprendedoras. Esta funcionalidad permite consultar de manera estructurada los resultados obtenidos en los cuestionarios aplicados a los estudiantes.

Al acceder a esta sección, se muestra una tabla dinámica que presenta los resultados agregados del grupo seleccionado. En ella se incluyen métricas clave para cada una de las categorías y variables evaluadas en el cuestionario, tales como el promedio, desviación estándar, valores mínimo y máximo, mediana y cuartiles. Gracias a estos datos, los docentes y administradores pueden obtener una visión general del desempeño del grupo en cada una de las dimensiones analizadas.

</

Ilustración 153. Tabla de métricas del cuestionario por grupo.

De igual forma el sistema cuenta con una herramienta de visualización gráfica interactiva. Esta función permite a los usuarios seleccionar una categoría específica y elegir la métrica de interés para su análisis. La representación en gráficos de barras facilita la interpretación de los datos, al mostrar de forma clara la distribución de los puntajes en cada variable dentro de la categoría seleccionada.

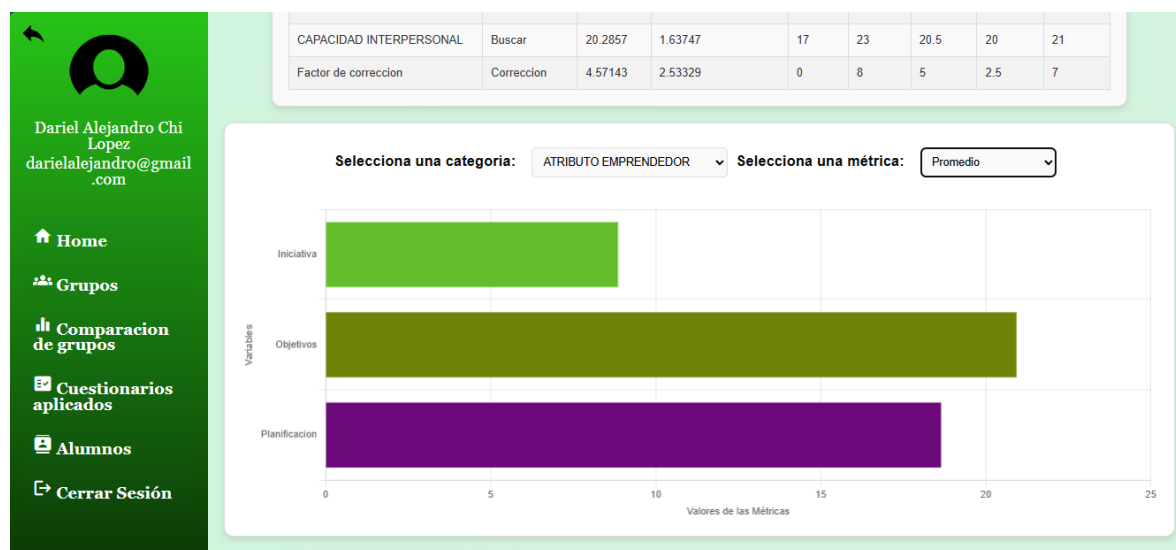


Ilustración 154. Tabla de métricas del cuestionario por grupo.

El docente se dirigirá al botón “Ver integrantes” donde nos redirigirá a la tabla con todos los estudiantes que pertenecen al grupo, mostrando su información relevante. La tabla cuenta con una columna llamada “Acciones” la cual el docente podrá visualizar los resultados del estudiante o puede eliminar al estudiante del grupo en caso de requerirlo. De igual forma el docente con ayuda de los botones que se encuentran arriba de la tabla puede tanto buscar un alumno y añadirlo al grupo o exportar la lista de integrantes del grupo.

Administración de Alumnos							
Exportar a PDF							
NOMBRE	PRIMER APELLIDO	SEGUNDO APELLIDO	CARRERA	INSTITUCION	CORREO	SEMESTRE	ACCIONES
Jair Alexis	Fernández	Tun	Ingeniería en Innovación Agrícola	Itescam	alexisJair@gmail.com	2º	
Andre Jasiel	Caamal	Gómez	Ingeniería en Innovación Agrícola	Itescam	jasiel1@gmail.com	2º	
Bárbara Isabel	Pool	Cauich	Ingeniería en Innovación Agrícola	Itescam	isabel@gmail.com	2º	
Ana Cristina	Chan	Tamay	Ingeniería en Innovación Agrícola	Itescam	anaChanTamay@gmail.com	2º	
Santiago Daniel	Pech	Miss	Ingeniería en Innovación Agrícola	Itescam	santiagoDani@gmail.com	2º	
Abimael	Bacab	Keb	Ingeniería en Innovación Agrícola	Itescam	abimaelBacab@gmail.com	2º	
Gabriela Esmeralda	Tzuc	Colli	Ingeniería en Innovación Agrícola	Itescam	gabrielaEsme@gmail.com	2º	
Paola Berenice	Rodriguez	Mata	Ingeniería en Innovación Agrícola	Itescam	paolaRodriguez@gmail.com	2º	
Seidy Lizeth	Cauich	Chan	Ingeniería en Innovación Agrícola	Itescam	lizethChan@gmail.com	2º	

Ilustración 155. Tabla de integrantes del grupo.

Ahora bien, para continuar con el análisis de resultados, el docente deberá ingresar al resultado del alumno que eligió sujeto de prueba. Al dirigirse a los resultados del estudiante mediante el botón de resultados de la columna “Acciones” redirigirá a la vista de cuestionarios resueltos del estudiante donde podrá visualizar los resultados de esa encuesta de forma más precisa o ver los resultados de la encuesta de manera general.

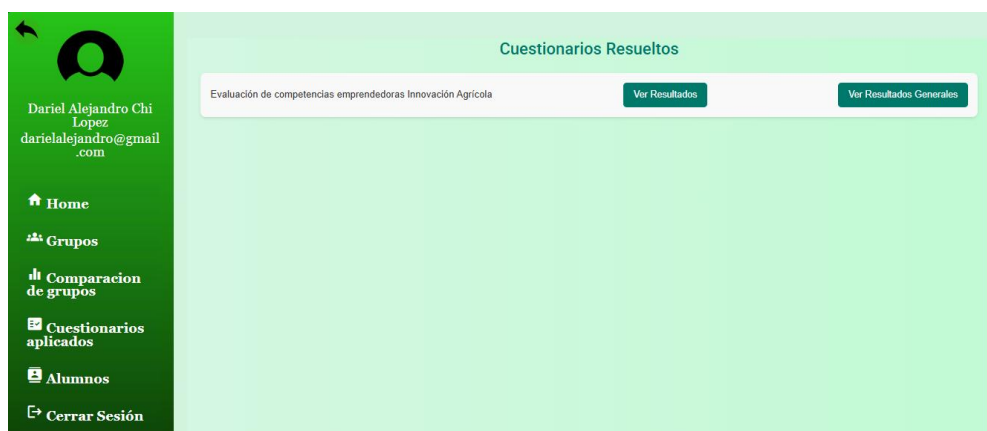


Ilustración 156. Cuestionarios resueltos del alumno.

Análisis Macro

El docente al ingresar a los resultados del alumno detallados del estudiante visualizará los detalles del cuestionario que en este caso son presentados por categorías que conocemos como “Macro”, se puede visualizar el puntaje obtenido de cada categoría por lo que se realizará un análisis más detallado.

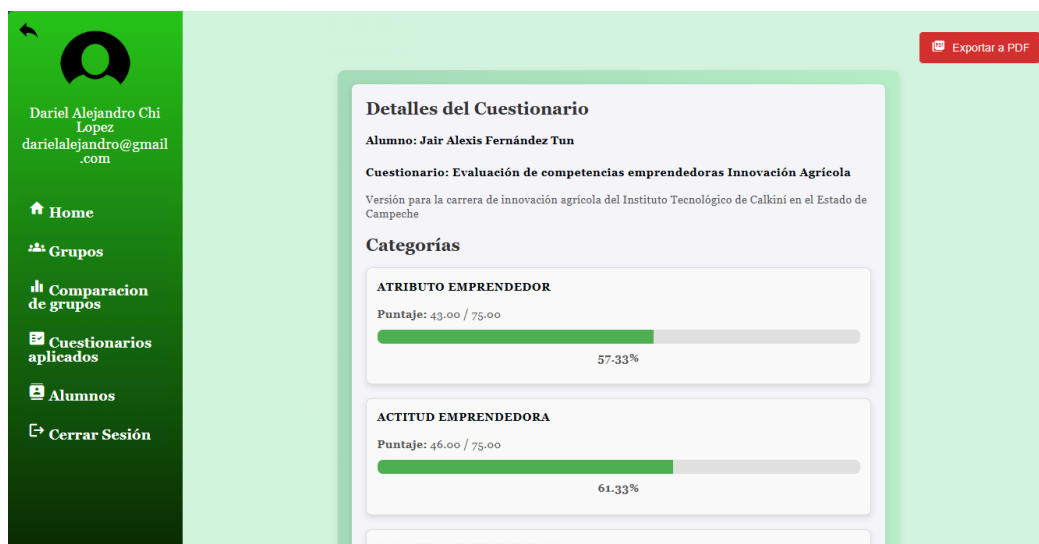


Ilustración 157. Destalles del cuestionario y puntajes por categorías (Macro) parte 1.

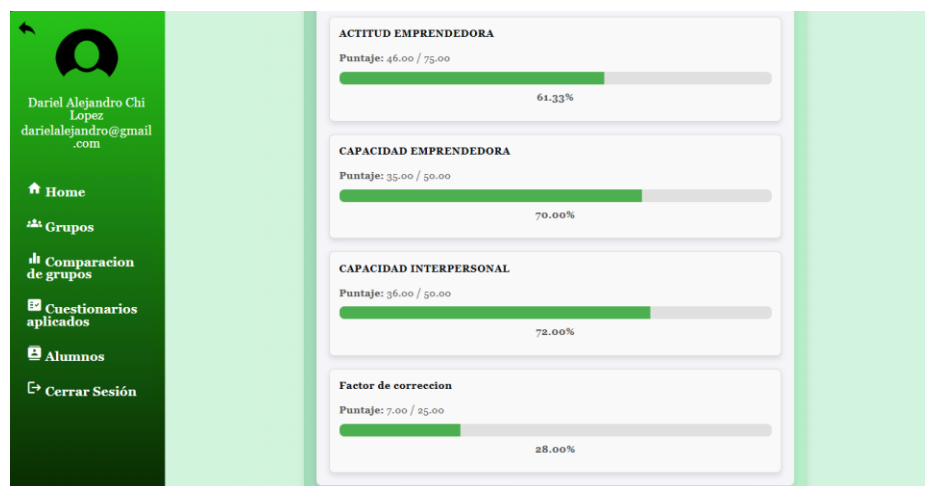


Ilustración 158. Detalles del cuestionario y puntajes por categorías (Macro) parte 2.

El sistema proporciona al docente una visión general del desempeño del estudiante en distintas categorías de competencias emprendedoras, lo que permite evaluar su perfil de manera estructurada. En este nivel de análisis, se presentan los resultados agrupados por atributo emprendedor, actitud emprendedora, capacidad emprendedora, capacidad interpersonal y factor de corrección.

Los resultados globales muestran que el estudiante tiene un perfil equilibrado en competencias emprendedoras, con fortalezas en la capacidad emprendedora e interpersonal, mientras que su atributo emprendedor y actitud emprendedora pueden desarrollarse más.

Estos valores reflejan el nivel de desarrollo del estudiante de sus características emprendedoras (CEPS). Por lo que a partir de estos datos el docente ya puede realizar un diagnóstico inicial sobre sus fortalezas y áreas de mejora del estudiante donde podrá identificar los patrones generales en su desempeño.

Análisis Micro

En esta sección el sistema proporcionará las herramientas para un análisis más detallado “Micro” de cada categoría desglosando los puntajes que obtuvo el estudiante en variables específicas para comprender con mayor precisión las competencias del estudiante.

Categoría: ATRIBUTO EMPRENDEDOR

El sistema permite analizar con detalle cada una de las variables de la categoría atributo emprendedor, desglosando los puntajes obtenidos por el estudiante en cada expresión perteneciente a la categoría.



Ilustración 159. Resultados “Micro” de la categoría ATRIBUTO EMPRENDEDOR.



Ilustración 160. Grafica de resultados “Micro” de la categoría ATRIBUTO EMPRENDEDOR.

El sistema proporciona una representación estructurada de los datos, permitiendo observar el nivel de desempeño del estudiante en cada expresión evaluada. En esta categoría,

el estudiante presenta diferencias en sus puntajes, obteniendo valores más altos en planificación y seguimiento y puntajes más bajos en búsqueda y aprovechamiento de oportunidades.

Categoría: ACTITUD EMPRENDEDORA.

La plataforma permite un análisis más detallado. En esta categoría, el estudiante presenta puntajes altos en cumplimiento de compromisos, lo que sugiere una capacidad para asumir responsabilidades y cumplir con tareas asignadas. Por otro lado, presenta puntajes más bajos en eficiencia y calidad, lo que indica una diferencia en su desempeño dentro de esta categoría.



Ilustración 161. Resultados Micro y Grafica de la categoría ACTITUD EMPRENDEDORA.

Categoría: CAPACIDAD EMPRENDEDORA.

En esta categoría, el estudiante presenta puntajes equilibrados en correr riesgos moderados y autoconfianza, lo que sugiere una tendencia a asumir desafíos con precaución y cierta seguridad en sus habilidades. Sin embargo, se pueden notar diferencias entre estas subcompetencias dentro de la misma categoría.

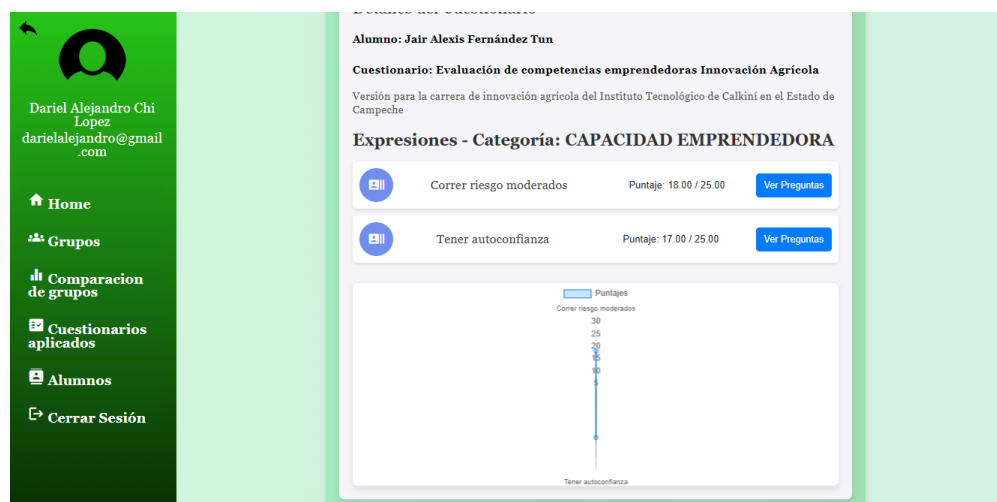


Ilustración 162. Resultados Micro y Grafica de la categoría CAPACIDAD EMPRENDEDORA.

Categoría: CAPACIDAD INTERPERSONAL.

En esta categoría, el estudiante presenta un puntaje alto en la búsqueda y consecución de información, lo que indica una capacidad favorable para investigar y obtener datos relevantes. Sin embargo, en persuasión y creación de redes de apoyo su puntaje es más bajo, lo que refleja una diferencia en su desempeño dentro de esta categoría.

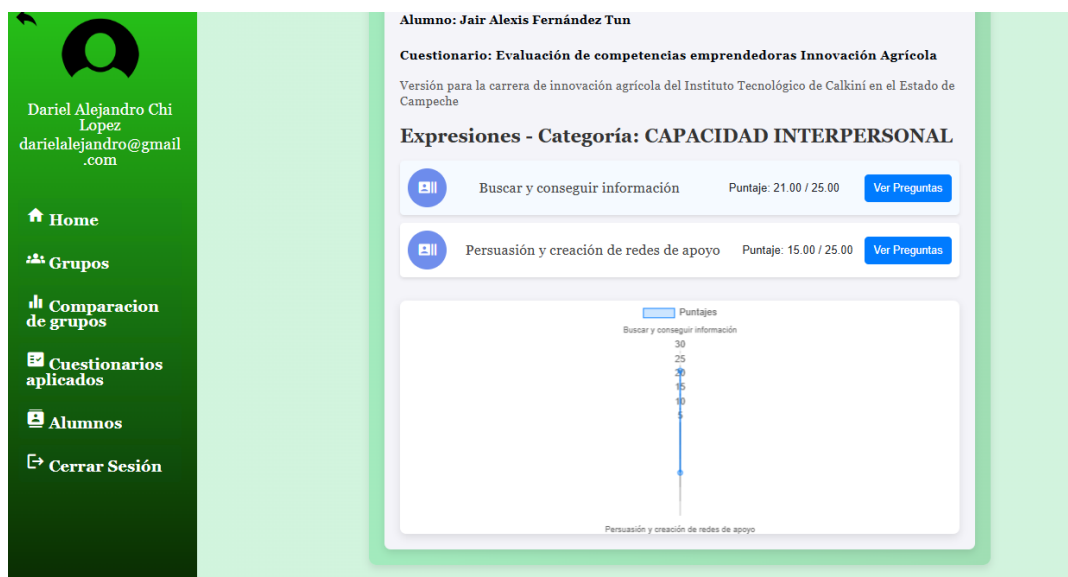


Ilustración 163. Resultados Micro y Grafica de la categoría CAPACIDAD INTERPERSONAL.

Categoría: FACTOR DE CORRECCIÓN.

El sistema proporciona una representación clara de este factor, permitiendo observar su influencia en la evaluación general. En este caso, el estudiante obtuvo un puntaje de 7 sobre 25, lo que sugiere que su nivel de sesgo es bajo o moderado. Este puntaje es utilizado como referencia para ajustar la interpretación de los resultados en otras categorías y asegurar que la evaluación sea más objetiva.

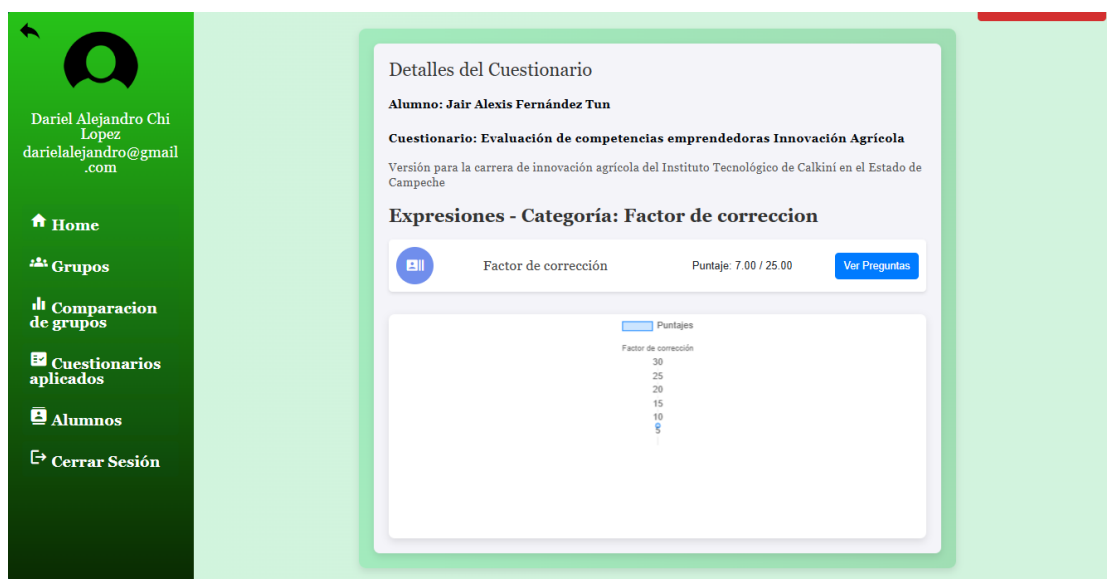


Ilustración 164. Resultados Micro y Grafica de la categoría FACTOR DE CORRECCIÓN.

El propósito de la investigación fue evaluar si la plataforma web puede facilitar la autoevaluación de competencias emprendedoras en estudiantes universitarios, optimizando el proceso y proporcionando resultados de manera eficiente. Para ello se planteó la siguiente pregunta:

¿Cómo influye la utilización de una plataforma de evaluación de competencias emprendedoras en el desarrollo de habilidades clave a nivel individual en estudiantes universitarios?

A partir de esta pregunta se planteó la siguiente hipótesis: El uso de una plataforma web para la evaluación de competencias emprendedoras permitirá a los estudiantes universitarios realizar autoevaluaciones de manera rápida y sencilla, eliminando la necesidad de procesos manuales y proporcionando resultados inmediatos y precisos.

Por ende, con ayuda de las pruebas y resultados obtenidos se demuestra que la plataforma cumple con la hipótesis ya que el sistema facilita el proceso de autoevaluación donde el estudiante pudo realizar la autoevaluación sin dificultad alguna y sin la necesidad de solicitar ayuda externa. También, el sistema genera los resultados de manera automática e inmediata eliminando la necesidad de cálculos manuales y entregó los resultados estructurada correctamente mediante gráficos y tablas que permitieron al alumno y docente interpretar los datos con facilidad, sin la necesidad de realizar un análisis complejo.

Otro punto más importante es que el sistema permitió al docente analizar los resultados macro y micro haciendo que no muestre los resultados de forma general, sino que también desglosando los puntajes por categorías y expresiones que pertenecen a dichas categorías.

Estos hallazgos indican que la implementación de herramientas digitales puede mejorar significativamente el proceso de evaluación, ofreciendo una solución más práctica y accesible tanto para estudiantes como para docentes.

La implementación de la plataforma web para la evaluación de competencias emprendedoras ha demostrado ser una herramienta efectiva que impacta de manera positiva en el desarrollo de habilidades clave a nivel individual en los estudiantes universitarios. A través de un proceso estructurado y automatizado, la plataforma permite que cada estudiante pueda obtener un diagnóstico detallado de su perfil emprendedor, identificando con claridad sus fortalezas y aquellas áreas en las que puede mejorar.

Uno de los aspectos más relevantes de esta plataforma es su capacidad para proporcionar una retroalimentación inmediata. Una vez que el estudiante completa la autoevaluación, el sistema genera los resultados de manera automática, sin necesidad de intervención manual por parte del docente. Esto no solo agiliza el proceso, sino que también permite que los estudiantes tengan acceso a información clara y organizada sobre sus competencias, lo que les facilita tomar decisiones informadas sobre su desarrollo profesional.

La accesibilidad y diseño amigable del sistema son importantes en su influencia sobre la evaluación de habilidades. Al eliminar la necesidad de procesos manuales y cálculos complejos, la plataforma ofrece una experiencia de usuario intuitiva, permitiendo que cualquier estudiante, independientemente de su nivel de familiaridad con herramientas digitales, pueda completar la autoevaluación sin dificultades técnicas. Además, al presentar los resultados en distintos niveles de análisis, tanto macro como micro, el sistema brinda una visión integral del desempeño del estudiante, permitiendo un análisis más profundo y estructurado de su progreso.

Otro aspecto por destacar es la optimización del tiempo y los recursos en el proceso de evaluación. En comparación con métodos tradicionales, la plataforma reduce significativamente el tiempo requerido para la recopilación, corrección y análisis de los resultados, permitiendo que los docentes puedan enfocarse en el diseño de estrategias que favorezcan el crecimiento de los estudiantes, en lugar de invertir esfuerzos en tareas operativas.

Con base en estos hallazgos, se concluye que la plataforma tiene un impacto significativo en la evaluación y desarrollo de competencias emprendedoras en estudiantes universitarios. Al brindar un diagnóstico preciso y accesible, permitir la obtención de resultados inmediatos y organizados, optimizar el proceso de evaluación y ofrecer herramientas para un análisis detallado, esta tecnología se posiciona como una solución eficaz para fortalecer la formación de los futuros emprendedores. Estos resultados sugieren

que la integración de herramientas digitales en la educación puede representar un cambio para el beneficio en la forma en que se evalúan y desarrollan las habilidades individuales, proporcionando información clara y estructurada que permite una mejor toma de decisiones en el ámbito académico y profesional.

Capítulo 5. CONCLUSIONES Y RECOMENDACIONES.

El presente estudio tuvo como propósito evaluar la funcionalidad y el impacto de una plataforma web en la autoevaluación de competencias emprendedoras en estudiantes universitarios, buscando determinar si esta herramienta facilita el proceso de evaluación y proporciona resultados de manera estructurada, inmediata y sin la necesidad de cálculos manuales.

A lo largo del desarrollo y pruebas del sistema, se logró comprobar que la plataforma permite que los estudiantes realicen su autoevaluación de manera sencilla e intuitiva, eliminando la necesidad de intermediarios o procesos tediosos que podrían ralentizar el análisis de sus competencias. Además, el sistema no solo automatiza la obtención de resultados, sino que organiza la información en un formato comprensible para el usuario, presentando tablas y gráficos que facilitan su interpretación sin requerir conocimientos especializados.

Otro aspecto clave es la optimización del tiempo y la carga de trabajo para los docentes. Gracias a la automatización, el sistema genera reportes inmediatos que eliminan la necesidad de evaluaciones manuales y proporcionan una visión general y detallada del perfil emprendedor del estudiante. La funcionalidad de desglosar los resultados en niveles macro y micro permite a los docentes y especialistas analizar con mayor precisión cada una de las categorías evaluadas, ofreciendo una herramienta que va más allá de una simple calificación.

A partir de los resultados obtenidos, se puede afirmar que la plataforma desarrollada cumple con los objetivos planteados y valida la hipótesis inicial, ya que proporciona un entorno digital donde la autoevaluación se lleva a cabo de manera rápida, sencilla y sin la necesidad de procesos manuales. La generación inmediata de resultados estructurados permite que tanto estudiantes como docentes cuenten con información útil para la toma de decisiones en el ámbito del desarrollo de competencias emprendedoras.

Asimismo, se identifica que la digitalización de este tipo de evaluaciones representa una alternativa viable y eficiente frente a los métodos tradicionales, ya que no solo optimiza el tiempo de aplicación y análisis, sino que también brinda accesibilidad a los resultados desde cualquier dispositivo, mejorando la experiencia del usuario y favoreciendo el seguimiento del progreso del estudiante en distintas etapas de su formación.

Finalmente, como línea de mejora y con el objetivo de incrementar la utilidad del sistema, se sugiere la implementación de un módulo de retroalimentación personalizada, que permita ofrecer recomendaciones basadas en los resultados obtenidos, ayudando al estudiante a fortalecer sus competencias en función de su desempeño. Esta integración permitiría que la plataforma no solo sirva como una herramienta de evaluación, sino también como un apoyo para el crecimiento y desarrollo de los futuros emprendedores.

Con base en estos hallazgos, se concluye que la plataforma desarrollada es una solución efectiva, accesible y estructurada para la evaluación de competencias emprendedoras, facilitando el diagnóstico y la toma de decisiones tanto para estudiantes como para docentes.

Recomendaciones

1. Implementar medidas adicionales de seguridad, como cifrado avanzado y protección contra vulnerabilidades, para garantizar la confidencialidad y la integridad de los datos de los usuarios.
2. Continuar mejorando la experiencia del usuario mediante pruebas con estudiantes y maestros, para identificar posibles áreas de mejora en las interfaces y funcionalidades.
3. Proveer sesiones de capacitación para los usuarios finales, orientadas a maximizar el aprovechamiento de las herramientas desarrolladas y garantizar el correcto uso del sistema.
4. Implementar una funcionalidad que, basada en los resultados obtenidos en los cuestionarios, brinde sugerencias adaptadas a cada estudiante sobre cómo mejorar sus competencias emprendedoras. Esto enriquecería el sistema al no solo evaluar, sino también guiar el aprendizaje.
5. Hay que asegurar que la plataforma cuente con una versión optimizada para dispositivos móviles, mejorando la accesibilidad para los usuarios que prefieran utilizarla en smartphones o tablets.
6. Capacitar a los usuarios para un uso correcto del sistema.

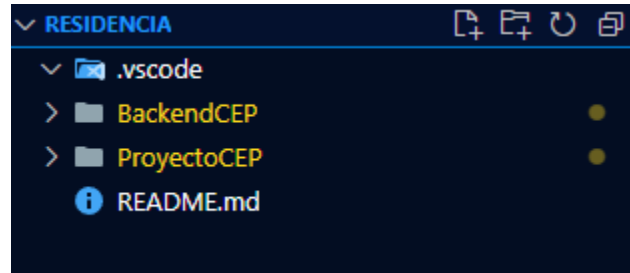
REFERENCIAS BIBLIOGRÁFICAS

- Aparicio, A., Palacios, W.D., Martínez, A.M., Verduzco C. Y Retana, E. (2008). El cuestionario. Apuntes del Departamento de Métodos de Investigación Avanzada. Recuperado de [http://www.uam.es/personal_pdi/stmaria/jmurillo/Met_Inves_Avan/Presentaciones/Cuestionario_\(trab\).pdf](http://www.uam.es/personal_pdi/stmaria/jmurillo/Met_Inves_Avan/Presentaciones/Cuestionario_(trab).pdf)
- Bacigalupo, M., Kampylis, P., Punie, Y., & Van den Brande, G. (2016). EntreComp: The Entrepreneurship Competence Framework. European Commission.
- Bacigalupo, M., Kampylis, P., Punie, Y., & Van den Brande, G. (2016). EntreComp: The Entrepreneurship Competence Framework. Luxembourg: Publication Office of the European Union. Disponible en: <https://publications.jrc.ec.europa.eu/repository/handle/JRC101581>
- Barba Bayas, D. (2013). Características Emprendedoras Personales (CEP) de hombres y mujeres gerentes de las Cooperativas de Ahorro y Crédito de Riobamba: investigación 2010-2011. Revista Ciencia UNEMI, 6(9), 62-68. Recuperado de <https://www.redalyc.org/pdf/5826/582663861008.pdf>
- Bernal, R. (2024). La autoconfianza como factor determinante en el éxito emprendedor. Revista de Psicología Empresarial, 9(2), 56-70.
- Bernal, R. (2024). Persuasión y networking en el ámbito emprendedor. Revista Internacional de Negocios y Emprendimiento, 11(3), 88-101.
- Blanco, E. (2021). Características emprendedoras en universitarios próximos a egresar: un estudio descriptivo. Revista de Investigación en Ciencias Sociales y Humanidades, 5(2), 45-60. Recuperado de https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2007-18332015000200003
- Cavero, O. (2023). Cumplimiento de compromisos en el emprendimiento: clave para el éxito empresarial. Revista de Estudios Empresariales, 12(3), 78-90.
- Chávez Moreno, E. A. (2020). Análisis comparativo de competencias emprendedoras entre estudiantes de la UABC. RIDE. Revista Iberoamericana para la Investigación y el Desarrollo Educativo, 10(20).
- Espinoza Valencia, F. J., Real Pérez, I., & González González, M. (2021). Características emprendedoras de los propietarios de las PyMES en Sonora. Revista de Investigación Académica Sin Frontera, 34(1), 1-15. Recuperado de <https://revistainvestigacionacademicasinfrontera.unison.mx/index.php/RDIASF/article/view/322/335>

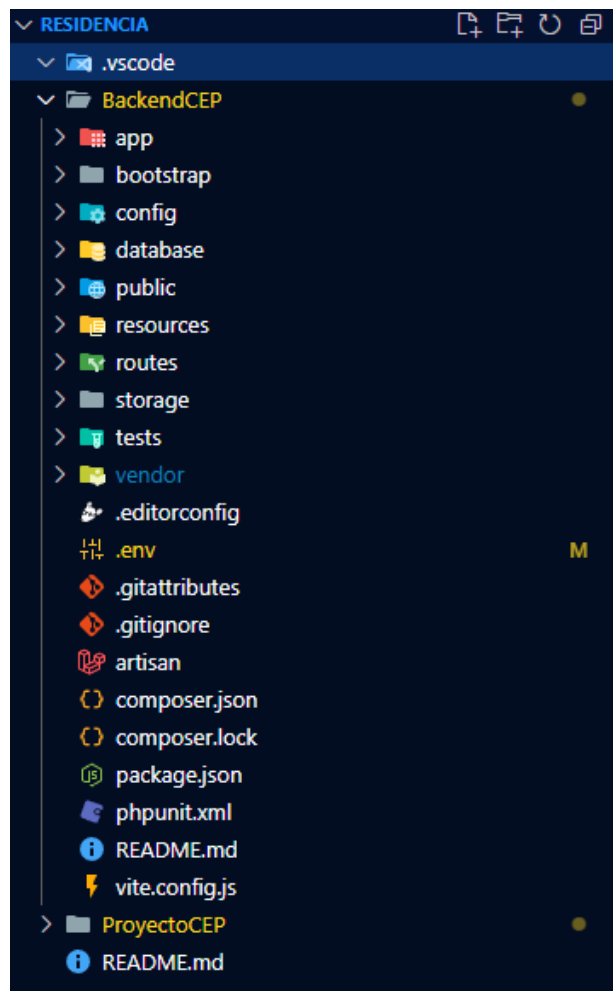
- Fayyaz, A., Mian, S. A., & Khan, J. (2009). State of entrepreneurship in developing countries: The case of Pakistan. *International Journal of Entrepreneurship and Small Business*, 7(4), 491-523.
- Martínez, F. M. y Carmona, G. (2009). Aproximación al concepto de "competencias emprendedoras": valor social e implicaciones educativas. *REICE: Revista Iberoamericana sobre Calidad, Eficacia y Cambio en Educación*, <https://www.redalyc.org/articulo.oa?id=55114063007>
- Olmos, R. E., Sánchez, R. F. G., & Vera, E. A. (2012). Desarrollo de competencias emprendedoras: Un análisis explicativo con estudiantes universitarios. *Cuadernos de estudios empresariales*, 22, 29
- PhD, I. A. (3 de 1 de 2023). Guía para la implementación de un Programa sistémico Emprendedor para el Instituto Tecnológico de Calkiní, Campeche. *Emprendedurismo y Economía ambiental*.
- Quezada, M. E., Vega-Valero, C. Z., & Nava-Quiroz, C. (2021). Evaluación de las competencias genéricas del comportamiento emprendedor. *Retos*, 11(22), 301-314. Recuperado de: <https://retos.ups.edu.ec/index.php/retos/article/view/22.2021.07>
- Tortajada, E. G. (2020). Diseño y validación inicial de un instrumento de medición de la competencia emprendedora sobre su tratamiento y comunicación en las aulas universitarias. *Comunicación y Hombre*, (16), 193-224
- Tapia Moreno, Francisco Javier (2010). Cómo elaborar un cuestionario. Notas de estadística aplicada a la administración, contaduría e informática administrativa, Departamento de Matemáticas, Universidad de Sonora. Recuperado de [http://www.mat.uson.mx/~ftapia/Lecturas%20Adicionales%20\(C%C3%B3mo%20dise%C3%B1ar%20una%20encuesta\)/C%C3%B3moElaborarUnCuestionario.pdf](http://www.mat.uson.mx/~ftapia/Lecturas%20Adicionales%20(C%C3%B3mo%20dise%C3%B1ar%20una%20encuesta)/C%C3%B3moElaborarUnCuestionario.pdf)

ANEXO

En los siguientes anexos (1 - 8) se presenta la arquitectura empleada para el proyecto, con sus requerimientos.



Anexo 1. Estructura del proyecto



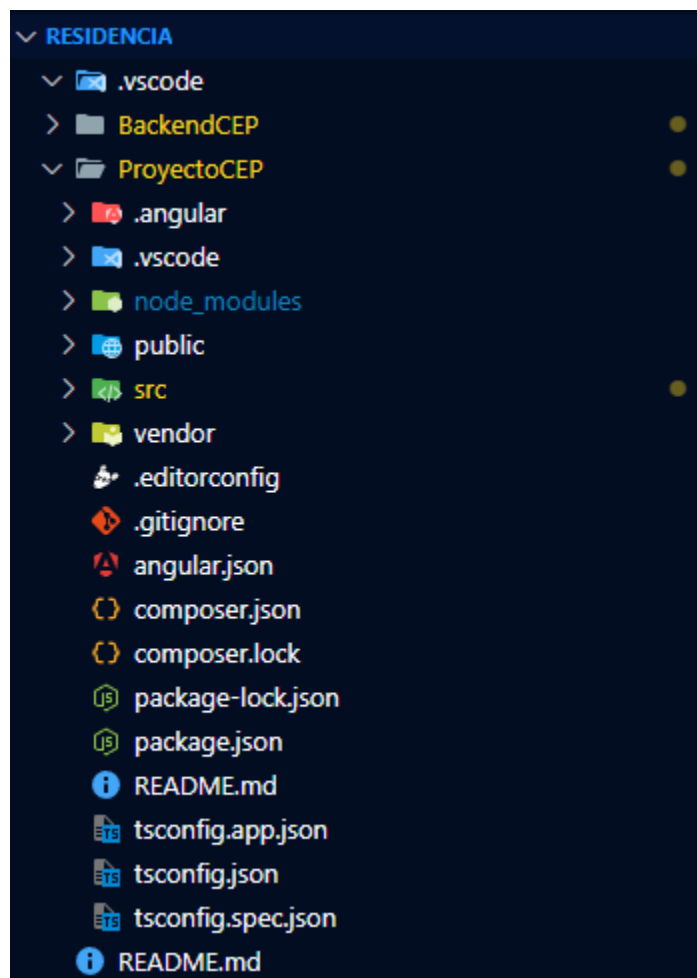
Anexo 2. Estructura del proyecto (Backend)

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:jXue+yQoM3JLfDNW8Qh705e8E3KNbCEKdFI0i1K+Ec4=
APP_DEBUG=true
APP_TIMEZONE=UTC
APP_URL=http://localhost
```

Anexo 3. Conexión con la base de datos (Backend)

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=proyectcep
DB_USERNAME=root
DB_PASSWORD=
```

Anexo 4. Conexión con la base de datos (Backend)



Anexo 5. Estructura del proyecto (Frontend)



Anexo 7. Estructura del proyecto (Frontend)

```
"dependencies": {
  "@angular/animations": "^18.0.0",
  "@angular/cdk": "^18.2.5",
  "@angular/common": "^18.0.0",
  "@angular/compiler": "^18.0.0",
  "@angular/core": "^18.0.0",
  "@angular/forms": "^18.0.0",
  "@angular/material": "^18.2.5",
  "@angular/platform-browser": "^18.0.0",
  "@angular/platform-browser-dynamic": "^18.0.0",
  "@angular/platform-server": "^18.0.0",
  "@angular/router": "^18.0.0",
  "@angular/ssr": "^18.0.3",
  "@fortawesome/fontawesome-free": "^6.6.0",
  "@popperjs/core": "^2.11.8",
  "bootstrap": "^5.3.3",
  "bootstrap-icons": "^1.11.3",
  "chart.js": "^4.4.4",
  "express": "^4.18.2",
  "material-design-icons-iconfont": "^6.7.0",
  "rxjs": "~7.8.0",
  "sweetalert": "^2.1.2",
  "sweetalert2": "^11.14.0",
  "tslib": "^2.3.0",
  "zone.js": "~0.14.3"
},
```

Anexo 8. Librerías usadas del proyecto (Frontend)

Frontend

Ahora en los siguientes anexos se presentan las funciones más importantes realizadas para las vistas del proyecto.

Para el registro del alumno se creó un formulario donde se añadirán los datos del estudiante, en ella se adecua el tipo de usuario del alumno, así como los validadores necesarios para que se respeten la estructura del registro.

```
this.formGroup = new FormGroup({
  nombre: new FormControl('', [nameValidator, makeRequired, lengthValidator({ min: 1, max: 30 })]),
  papellido: new FormControl('', [nameValidator, makeRequired, lengthValidator({ min: 1, max: 15 })]),
  sapellido: new FormControl('', [nameValidator, makeRequired, lengthValidator({ min: 1, max: 15 })]),
  matricula: new FormControl('', [makeRequired, lengthValidator({ min: 1, max: 15 })]),
  semestre: new FormControl('', [makeRequired]),
  tipo: new FormControl('Alumno', []),
  correo: new FormControl('', [emailValidator(), makeRequired, lengthValidator({ min: 1, max: 30 })]),
  contraseña: new FormControl('', [makeRequired, lengthValidator({ min: 8, max: 15 })]),
  confirmcontraseña: new FormControl('', [makeRequired, lengthValidator({ min: 8, max: 15 })]),
  edad: new FormControl('', [makeRequired]),
  genero: new FormControl('', [makeRequired]),
  institucion: new FormControl('', [makeRequired]),
  pais: new FormControl('', [makeRequired]),
  carrera: new FormControl('', [makeRequired])
}, { validators: confirmPassword });
```

Anexo 9. Formulario del alumno

De igual manera en el formulario se agregaron 3 subscripciones a un observable las cuales tienen la finalidad de obtener los países e instituciones del sistema y una función el cual mostrará una lista de edades para que el usuario seleccione la edad en un rango de 17 a 99 años.

```
this.catalogs.getCountries().subscribe((response: ResponseGet<Countries[]>) => {
  this.Countries = response.data;
});

this.formGroup.get('pais')?.valueChanges.subscribe(selectedCountryId => {
  this.onCountryChange(selectedCountryId);
});

this.formGroup.get('institucion')?.valueChanges.subscribe(selectedInstitutionId => {
  this.onInstitutionChange(selectedInstitutionId);
});

this.ages = Array.from({ length: 99 - 17 + 1 }, (_, i) => i + 17);
}
```

Anexo 10. Subscripciones para países e instituciones y rango de edad

El componente tiene 2 funciones importantes, la primera función “onCountryChange” tiene la finalidad de obtener las instituciones que pertenecen a un país, es decir, filtra las instituciones de un país. La segunda función “onInstitutionChange” tiene la misma funcionalidad, pero en este caso Filtra las carreras que pertenecen a la institución seleccionada.

```
Complexity is 3 Everything is cool!
onCountryChange(IDPais: string) {
  if (IDPais) {
    this.catalogs.getInstitutionsId(IDPais).subscribe((response: ResponseGet<Institutions[]>) => {
      this.FilteredInstitutions = response.data;
    });
  } else {
    this.FilteredInstitutions = [];
  }
}
Complexity is 3 Everything is cool!
onInstitutionChange(InstitucionID: string) {
  if (InstitucionID) {
    this.catalogs.getDegreesId(InstitucionID).subscribe((response: ResponseGet<Degrees[]>) => {
      this.FilteredDegrees = response.data;
    });
  } else {
    this.FilteredDegrees = [];
  }
}
```

Anexo 11. Funciones para obtener instituciones y carreras filtradas

La función “onSubmit” sirve para mandar los campos que han sido llenados por el usuario y a partir del servicio registerStudent almacena los datos del nuevo usuario y despliega un mensaje de que el usuario se ha creado correctamente o muestra un error si el usuario no se pudo crear.

```
Complexity is 7 It's time to do something...
93 onSubmit() {
94   if (this.formGroup.invalid) {
95     this.formGroup.markAllAsTouched();
96   } else {
97     const createStudent: createStudent = {
98       Nombre: this.formGroup.value.nombre,
99       Primer_apellido: this.formGroup.value.papellido,
100      Segundo_apellido: this.formGroup.value.sapellido,
101      Matricula: this.formGroup.value.matricula,
102      Semestre: this.formGroup.value.semestre,
103      Tipo: this.formGroup.value.tipo,
104      Correo: this.formGroup.value.correo,
105      Password: this.formGroup.value.contraseña,
106      edad: this.formGroup.value.edad,
107      genero: this.formGroup.value.genero,
108      InstitucionID: this.formGroup.value.institucion,
109      IDPais: this.formGroup.value.pais,
110      CarreraID: this.formGroup.value.carrera
111    };
112    this.userService.registerStudent(createStudent).subscribe(
113      (response) => {
114        Swal.fire({
115          title: '¡Éxito!',
116          text: 'Estudiante creado correctamente',
117          icon: 'success',
118          confirmButtonText: 'Aceptar'
119        });
120      },
121      (error) => {
122        if (error.status === 422 && error.error.message) {
123          ServerErrorValidator.setServerErrors(error.error.message, this.formGroup);
124        } else {
125          Swal.fire({
126            title: 'Error',
127            text: 'Ocurrió un error al crear el estudiante.',
128            icon: 'error',
129            confirmButtonText: 'Aceptar'
130          }).then(() => {
131            this.goBack();
132          });
133        }
134      }
135    );
136  }
137}
```

Anexo 12. Función onSubmit

```

1 <div class="container">
2   
3   <h2 class="form-title">Registrar estudiante</h2>
4
5   <form [formGroup]="formGroup" (ngSubmit)="onSubmit()">
6
7     <div class="form-group">
8       <text-input label="Nombres" [controller]="formGroup.controls['nombre']" type="text"></text-input>
9     </div>
10
11    <div class="form-group form-group-inline">
12      <text-input label="Primer Apellido" [controller]="formGroup.controls['papellido']" type="text"></text-input>
13      <text-input label="Segundo Apellido" [controller]="formGroup.controls['sapellido']" type="text"></text-input>
14    </div>
15
16    <div class="form-group form-group-inline">
17      <div class="form-group">
18        <label for="genero">Género</label>
19        <select class="form-select" id="genero" formControlName="genero">
20          <option value="">Seleccione un género</option>
21          <option value="Masculino">Masculino</option>
22          <option value="Femenino">Femenino</option>
23          <option value="Otro">Otro</option>
24        </select>
25        <div *ngIf="formGroup.get('genero')?.invalid && formGroup.get('genero')?.touched">
26          <span class="error-message">Selecciona un género válido</span>
27        </div>
28      </div>
29
30      <div class="form-group">
31        <label for="edad">Edad</label>
32        <select class="form-select" id="edad" formControlName="edad">
33          <option value="">Seleccione una edad</option>
34          <option *ngFor="let age of ages" [value]="age">{{ age }}</option>
35        </select>
36        <div *ngIf="formGroup.controls['edad'].invalid && formGroup.controls['edad'].touched">
37          <span class="error-message">Selecciona una edad válida</span>
38        </div>
39      </div>
40    </div>
41

```

Anexo 13. Código para la vista del registro 1

```

42     <div class="form-group-inline">
43         <div class="form-group">
44             <label for="pais">País</label>
45             <select class="form-select" id="pais" formControlName="pais">
46                 <option value="">Selecione un pais</option>
47                 <option *ngFor="let IDPais of Countries" [value]="IDPais.IDPais">
48                     {{ IDPais.Nombre }}
49                 </option>
50             </select>
51             <div *ngIf="formGroup.get('pais')?.invalid && formGroup.get('pais')?.touched">
52                 <span class="error-message">Selecciona un país válido </span>
53             </div>
54         </div>
55
56         <div class="form-group" *ngIf="formGroup.get('pais')?.value">
57             <label for="institucion">Institución</label>
58             <select class="form-select" id="institucion" formControlName="institucion">
59                 <option value="">Selecione una institución</option>
60                 <option *ngFor="let institucion of FilteredInstitutions" [value]="institucion.InstitucionID">
61                     {{ institucion.Nombre }}
62                 </option>
63             </select>
64             <div *ngIf="formGroup.get('institucion')?.invalid && formGroup.get('institucion')?.touched">
65                 <span class="error-message">Selecciona una institución válida</span>
66             </div>
67         </div>
68     </div>
69
70     <div class="form-group">
71         <div class="form-group">
72             <label for="tipo">Tipo de usuario</label>
73             <select class="form-select" id="tipo" formControlName="tipo" [disabled]="true">
74                 <option value="Alumno" selected>Alumno</option>
75             </select>
76         </div>
77     </div>

```

Anexo 14. Código para la vista del registro 2

```

79     <div class="form-group-inline">
80         <div class="form-group">
81             <div class="form-group">
82                 <label for="tipo">Semestre</label>
83                 <select class="form-select" id="semestre" formControlName="semestre">
84                     <option value="">Semestre actual</option>
85                     <option value="2º" selected>2º</option>
86                     <option value="6º" selected>6º</option>
87                     <option value="8º" selected>8º</option>
88                 </select>
89                 <div *ngIf="formGroup.get('semestre')?.invalid && formGroup.get('semestre')?.touched">
90                     <span class="error-message">Seleccione un semestre</span>
91                 </div>
92             </div>
93         </div>
94     </div>
95     <text-input label="Matricula" [controller]="formGroup.controls['matricula']" type="text"></text-input>
96     <div *ngIf="formGroup.get('matricula')?.errors?.['serverError']">
97         <span class="error-message">{{ formGroup.get('matricula')?.errors?.['serverError'] }}</span>
98     </div>
99 </div>
100
101 <div class="form-group" *ngIf="formGroup.get('institucion')?.value">
102     <label for="carrera">Carrera</label>
103     <select class="form-select" id="carrera" formControlName="carrera">
104         <option value="">Seleccione una carrera</option>
105         <option *ngFor="let carrera of FilteredDegrees" [value]="carrera.CarreraID">
106             {{ carrera.Nombre }}
107         </option>
108     </select>
109     <div *ngIf="formGroup.get('carrera')?.invalid && formGroup.get('carrera')?.touched">
110         <span class="error-message">Seleccione una carrera válida</span>
111     </div>
112 </div>

```

Anexo 15. Código para la vista del registro 3

```

116 <div class="form-group">
117     <text-input label="Correo Electrónico" [controller]="formGroup.controls['correo']" type="text"></text-input>
118     <div *ngIf="formGroup.get('correo')?.errors?.['serverError']">
119         <span class="error-message">{{ formGroup.get('correo')?.errors?.['serverError'] }}</span>
120     </div>
121 </div>
122
123 <div class="form-group">
124     <text-input label="Contraseña" [controller]="formGroup.controls['contraseña']" type="password"></text-input>
125 </div>
126
127 <div class="form-group">
128     <text-input label="Confirmar contraseña" [controller]="formGroup.controls['confirmcontraseña']" type="password"></text-input>
129     <div *ngIf="formGroup.errors?.['passwordMismatch'] && formGroup.get('confirmcontraseña')?.touched">
130         <span class="error-message">{{ formGroup.errors?.['passwordMismatch'].message }}</span>
131     </div>
132 </div>
133
134 <button type="button" class="btn btn-back" [routerLink]="['/home']">Regresar</button>
135 <button type="submit" class="btn btn-register">Registrarse</button>
136
137 </form>
138 </div>

```

Anexo 16. Código para la vista del registro 4

En la vista de la página principal está compuesta por 6 contenedores el cual contiene: header, componente principal, carrusel de imágenes, objetivo del cuestionario, aclaración y footer.

```

14'      <!-- Opciones del menú -->
15      <div class="collapse navbar-collapse nav-underline" id="navbarNav">
16        <ul class="navbar-nav mx-auto">
17          <hr class="text-black-50">
18          <li class="nav-item">
19            <a class="nav-link text-black" [routerLink]="['/home']">Home</a>
20          </li>
21          <li class="nav-item dropdown">
22            <a class="nav-link text-black dropdown-toggle" [routerLink]="['/quien_somos']"
23              id="navbarDarkDropdownMenuLink" role="button" data-bs-toggle="dropdown" aria-expanded="false">
24              Quienes somos
25            </a>
26          </li>
27          <li class="nav-item">
28            <a class="nav-link text-black" [routerLink]="['/locacion']">Contacto</a>
29          </li>
30          <li class="nav-item">
31            <a class="nav-link text-black" [routerLink]="['/instrucciones']">Cuestionario</a>
32          </li>
33        </ul>
34      </div>
35      <!-- Botones para el iniciar sesion y registrar -->
36      <hr class="text-white-50 bg-warning">
37      <div class="d-grid gap-2 d-md-flex justify-content-md-end">
38        <button type="button" class="btn btn-outline-signup me-md-2" [routerLink]="['/registro']">SIGN UP</button>
39        <button type="button" class="btn btn-login ml-2" [routerLink]="['/login']">LOG IN</button>
40      </div>
41    </div>
42  </nav>

```

Anexo 17. Opciones del menú del header

```

Go to component | Fernanda-Chabre, hace 3 meses | 1 autor (Fernanda-Chabre)
1  <body class="contenedorprincipal">
2    <!-- Contenedor principal con la imagen de fondo -->
3    <div class="container-fluid fondo-imagen d-flex align-items-center alto">
4      <div class="fondo-degradado ml-auto text-center p-4">
5        <h1 class="titulo">Autoevaluación de las Características Emprendedoras Personales</h1>
6        <button type="button" class="btn btn-primary btn-lg mt-4" [routerLink]="['/instrucciones']">¡CONTESTA AHORA!</button>
7      </div>
8    </div>
9    <div class="carrusel">
10     <app-carrusel></app-carrusel>
11   </div>
12
13   <div class="contenedor1">
14     <app-nota></app-nota>
15   </div>
16
17   <div class="contenedor2">
18     <app-nota2></app-nota2>
19   </div>
20 </body>

```

Anexo 18. Contenedor principal


```

Go to component | Fernanda-Chable, hace 4 meses | 1 author (Fernanda-Chable)
1 <div class="container-fluid fondo">
2   <div class="content">
3     <div class="text">
4       <h1>¿Cuál es el objetivo del cuestionario?</h1>
5       <p>Su objetivo principal es que la persona misma pueda analizar y construir
6         su propio perfil acerca de estas características.
7         La identificación de sus debilidades y sus fuerzas le da la posibilidad de
8         reforzar o aprovechar mejor sus propias condiciones.
9       </p>
10    </div>
11    <div class="images">
12      
13      
14    </div>
15  </div>
16 </div>

```

Anexo 19. Objetivo del cuestionario

```

<footer class="bg-dark text-white py-5 footer-with-top-margin">
  <div class="container">
    <div class="row">
      <!-- Logo y Proyecto -->
      <div class="col-md-4 mb-2 d-flex flex-column justify-content-between">
        <div>
          <a href="#" class="text-reset text-uppercase d-flex align-items-center logo-link mb-3">
            
            <span class="logo-text"></span>
          </a>
          <h3>Proyecto Educativo</h3>
          <p>Este proyecto es una colaboración entre maestros de
            diversas instituciones para que los alumnos puedan identificar sus debilidades
            y fuerzas en su perfil emprendedor. Nuestro objetivo es que la persona misma
            pueda analizar y construir su propio perfil acerca de estas características.
            La identificación de sus debilidades y sus fuerzas le
            da la posibilidad de reforzar o aprovechar
            mejor sus propias condiciones. </p>
        </div>
      </div>
      <!-- Contacto e Instituciones -->
      <div class="col-md-5 mb-2 d-flex flex-column justify-content-between">
        <div>
          <h3>Créditos</h3>
          <p>Este proyecto ha sido desarrollado con el esfuerzo
            y la dedicación de varias personas.
            Los maestros de diversas instituciones educativas
            han liderado la investigación y el diseño del proyecto,
            mientras que dos alumnos que están por egresar del Instituto Tecnológico Superior
            de Calkini han sido responsables del desarrollo de la página web.
            Agradecemos a todos los colaboradores y participantes en
            este proyecto.</p>
        </div>
        <div>
          <h3>Instituciones Participantes</h3>
          <ul class="list-unstyled">
            <li><a href="https://www.conkal.tecnm.mx/" class="text-reset">Instituto Tecnológico de Conkal</a></li>
            <li><a href="https://hopelchen.tecnm.mx/portal/" class="text-reset">Instituto Tecnológico Superior de Hopelchén</a></li>
            <li><a href="https://itescam.edu.mx/portal/" class="text-reset">Instituto Tecnológico Superior de Calkiní</a></li>
          </ul>
        </div>
      </div>
    </div>
  </div>

```

Anexo 20. Código del Footer 1

```

46      <!-- Redes sociales y Créditos -->
47      <div class="col-md-3 mb-2 d-flex flex-column justify-content-between">
48          <div class="social-icons mb-3">
49              <h3>Síguenos en nuestras Redes Sociales</h3>
50              <hr class="social-line">
51              <ul class="list-unstyled d-flex justify-content-between">
52                  <!-- Facebook -->
53                  <li><a href="https://www.facebook.com/?locale=es_LA" class="facebook">
54                      <i class="bi bi-facebook fs-2"></i>
55                  </a></li>
56                  <!-- Twitter -->
57                  <li><a href="https://twitter.com/?lang=es" class="x">
58                      <i class="bi bi-twitter fs-2"></i>
59                  </a></li>
60                  <!-- Instagram -->
61                  <li><a href="https://www.instagram.com" class="instagram">
62                      <i class="bi bi-instagram fs-2"></i>
63                  </a></li>
64                  <!-- WhatsApp -->
65                  <li><a href="https://www.whatsapp.com" class="whatsapp">
66                      <i class="bi bi-whatsapp fs-2"></i>
67                  </a></li>
68              </ul>
69          </div>
70          <div>
71              <h3>Contacto</h3>
72              <ul class="list-unstyled">
73                  <li><strong>Email:</strong> </li>
74                  <li><strong>Teléfono:</strong> +9961026754</li>
75              </ul>
76          </div>
77      </div>
78  </div>

```

Anexo 21. Código del Footer parte 2

```

79      <div class="row">
80          <div class="col text-center">
81              <ul class="list-unstyled d-inline-block">
82                  <li><a href="/terminos-de-uso" class="text-reset">Términos de Uso</a></li>
83                  <li><a href="/politica-de-privacidad" class="text-reset">Política de Privacidad</a></li>
84              </ul>
85              <p class="mt-2">&copy; 2024 Proyecto Educativo. Todos los derechos reservados.</p>
86          </div>
87      </div>
88  </div>
89  </footer>
90 </div>

```

Anexo 22. Código del Footer parte 3

Para el apartado de localización se contiene la información del tecnológico la estructura para una institución se repite de manera similar para la información de las otras dos instituciones.

```

2 <div class="title-container">
3   
4   <h1 class="titulo">DONDE ENCONTRARNOS</h1>
5 </div>
6 <br>
7 <!-- Contenedor del instituto de calkini-->
8 <div class="content-container">
9   <div class="title-container">
10    
11    <h1 class="centered-title">Instituto Tecnológico Superior de Calkiní en el estado de campeche</h1>
12  </div>
13  <br>
14  <div class="content-below">
15    <p class="text-left">
16      Dirección: Avenida AH-Canun, Sin Número, San Felipe, 24900 Calkiní, Camp.<br>
17      Teléfono: 996 813 4870
18    </p>
19    <a href="https://www.itescam.mx/" class="text-reset">Sitio web: https://www.itescam.mx</a>
20  </div>
21  <div class="map-right">
22    <iframe src="https://www.google.com/maps/embed?pb=!1m14!1m8!1m3!1d14254.95661178538!2d-90.0442127610134!3d20.345074120220303!3m2!1i1"
23      width="100%" height="100%" style="border:0;" allowfullscreen="" loading="lazy"></iframe>
24  </div>
25 </div>
26 </div>

```

Anexo 23. Contacto

```

<div class="title-container">
  
  <h1 class="titulo">¿QUIÉNES SOMOS?</h1>
</div>
<div class="container">
  <h1 class="titulo">NUESTRA HISTORIA</h1>
</div>

```

Anexo 24. Quienes somos

Para el login de usuarios se creó el formulario el cual admite correo o matricula y la contraseña. De igual manera una función la cual a partir del AuthService se obtiene el tipo de usuario y redirige al usuario a su respectivo Dashboard con un switch.

```

22 login() {
23   this.authService.login(this.FormGroup).subscribe(
24     Complexity is 10 it's time to do something...
25     (response) => {
26       console.log('Respuesta del API:', response);
27       localStorage.setItem('access_token', response.access_token);
28       localStorage.setItem('user_id', response.usuario.UsuarioID.toString());
29
30       // Obtenemos el tipo de usuario
31       const tipoUsuario = response.usuario.Tipo;
32
33       // Redirigimos al dashboard correspondiente según el tipo de usuario
34       switch (tipoUsuario) {
35         case 'Alumno':
36           this.router.navigate(['/alumno']);
37           break;
38         case 'Maestro':
39           this.router.navigate(['/maestro']);
40           break;
41         case 'Admin':
42           this.router.navigate(['/administrador']);
43           break;
44         default:
45           this.router.navigate(['/home']);
46           break;
47       }
48     },
49     (error) => {
50       console.error('Login failed', error);
51     }
52   );
53 }
54

```

Anexo 25. Función login

```
1 <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
2 <div class="login-container">
3   <div class="login-box">
4     <span class="material-icons">account_circle</span>
5     <h1>Login</h1>
6
7     <form (submit)="login()">
8       <label for="Correo">Correo electrónico o Matrícula:</label>
9       <input type="text" [(ngModel)]="FormGroup.Correo" name="Correo" required />
10
11       <label for="Password">Contraseña:</label>
12       <input type="password" [(ngModel)]="FormGroup.Password" name="Password" required />
13
14       <div class="button-group">
15         <button type="button" class="btn regresar-btn">Regresar</button>
16         <button type="submit" class="btn login-btn">Iniciar sesión</button>
17       </div>
18
19       <a href="#" class="forgot-password">¿Olvidaste tu contraseña?</a>
20     </form>
21   </div>
22 </div>
```

Anexo 26. Código de la vista del login

Para el Dashboard del estudiante tiene su propio menú el cual es donde podrá navegar el alumno. Donde al ingresar muestra el nombre, apellidos del alumno, así como su correo.

```

<div *ngIf="isLoggedIn" class="dashboard">
  <div class="sidebar">
    <div class="user-info">
      
      <p *ngIf="student">{{ student.Nombre }} {{ student.Primer_apellido }} {{ student.Segundo_apellido }}</p>
      <p *ngIf="student">{{ student.Correo }}</p>
    </div>
    <div class="menu">
      <a [routerLink]="['/alumno/dash-estu']">
        <mat-icon>home</mat-icon> Home
      </a>
      <a [routerLink]="['/alumno/gruops']">
        <mat-icon>groups</mat-icon> Grupos
      </a>
      <a [routerLink]="['/alumno/generales']">
        <mat-icon>quiz</mat-icon> Cuestionarios resueltos
      </a>
      <a (click)="logout()" *ngIf="isLoggedIn">
        <mat-icon>logout</mat-icon> Cerrar Sesión
      </a>
    </div>
  </div>

  <div class="main-content">
    <div class="content-container">
      <router-outlet></router-outlet>
    </div>
  </div>
</div>

```

Anexo 27. Código de la vista del menú

```

35   loadStudentData(): void {
36     const studentId = this.authService.getUserId();
37     console.log('ID del estudiante:', studentId);
38
39     if (studentId) {
40       this.userService.getStudent(studentId).subscribe(
41         response => {
42           console.log('Respuesta del API:', response);
43           if (response.data) {
44             this.student = response.data;
45             console.log('Datos del estudiante:', this.student);
46           } else {
47             console.error('No se encontraron datos del estudiante');
48           }
49         },
50         error => {
51           console.error('Error al cargar datos del estudiante', error);
52         }
53       );
54     } else {
55       console.error('No se encontró un ID de usuario autenticado');
56     }
57   }

```

Anexo 28. Función para cargar información del alumno del menú.

Para el Dashboard del alumno se agregó una bienvenida con la fecha del día el cual inició sesión, también incluye un menú el cual puede observar la cantidad de grupos, resultados, cuestionarios resueltos y cuestionarios pendientes.

```

1 <div class="main-content">
2 <header class="header">
3 <div class="header-box">
4 <h1>Bienvenido, {{ currentDate }}</h1>
5 </div>
6 <div class="search-and-button">
7 <input type="search" placeholder="Search">
8 <button class="entrar-grupo" (click)="openEntrarGrupo()">Entrar al grupo</button>
9 </div>
10 </header>
11
12 <section class="home">
13 <div class="home-box">
14 <h1><mat-icon>home</mat-icon> HOME</h1>
15 </div>
16 </section>
17
18 <div class="buttons">
19 <button class="button blue" [routerLink]="['/alumno/grupos']">
20 <mat-icon>group</mat-icon> Grupos <span class="counter">{{ totalGrupos }}</span>
21 </button>
22 <button class="button orange" [routerLink]="['/alumno/Resultados generales']">
23 <mat-icon>bar_chart</mat-icon> Resultados <span class="counter">{{ totalResultados }}</span>
24 </button>
25 <button class="button green" [routerLink]="['/alumno/generales']">
26 <mat-icon>check_circle</mat-icon> Cuestionarios resueltos <span class="counter">{{ totalCuestionariosResueltos }}</span>
27 </button>
28 <button class="button pink" [routerLink]="['/alumno/pendiente']">
29 <mat-icon>schedule</mat-icon> Cuestionarios pendientes <span class="counter">{{ totalCuestionariosPendientes }}</span>
30 </button>
31 </div>
32
33 </div>
34
35 <app-enter-group *ngIf="showEnterGroupPopup" (close)="closePopup()"></app-enter-group>

```

Anexo 29. Código vista del dashboard.

Se realizaron funciones para obtener la fecha de bienvenida y contadores del total de grupos, total de resultados, cuestionarios resueltos, cuestionarios pendientes.

Al ingresar al Dashboard el componente obtiene el id del usuario y la función para obtener el día, de igual manera inicializa las funciones para obtener los contadores del total de grupos, total de cuestionarios resueltos, total de cuestionarios pendientes y total de resultados.

```

39  ngOnInit(): void {
40      this.usuarioID = this.authService.getUserId();
41      this.currentDate = this.getFormattedDate();
42      this.getTotalGrupos();
43      this.getTotalResultados();
44      this.getTotalCuestionariosResueltos();
45      this.getTotalCuestionariosPendientes();
46  }
47
48  // Función para formatear la fecha
49  getFormattedDate(): string {
50      const today = new Date();
51      const daysOfWeek = ['Domingo', 'Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado'];
52      const monthsOfYear = ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto',
53          'Septiembre', 'Octubre', 'Noviembre', 'Diciembre'];
54
55      const dayOfWeek = daysOfWeek[today.getDay()];
56      const dayOfMonth = today.getDate();
57      const month = monthsOfYear[today.getMonth()];
58      const year = today.getFullYear();
59
60      return `${dayOfWeek} ${dayOfMonth} de ${month} del ${year}`;
61  }
62
63  getTotalGrupos(): void {
64      if (this.usuarioID !== null) {
65          this.alumnoGruposService.getAlumnoGrupos(this.usuarioID).subscribe(
66              (response) => {
67                  if (response.data) {
68                      this.totalGrupos = response.data.length;
69                  } else {
70                      this.totalGrupos = 0;
71                  }
72              },
73              (error) => {
74                  console.error('Error al obtener grupos:', error);
75              }
76          );
77      }
78  }

```

Anexo 30. Inicializador de funciones, función “formattedDate” y “getTotalGrupos”.

```

80  getTotalResultados(): void {
81      if (this.usuarioID !== null) {
82          this.solvedQuizzesService.getCuestionariosResueltosPorAlumno(this.usuarioID).subscribe(
83              Complexity is 3 Everything is cool!
84              (response) => {
85                  if (response.data && response.data.length) {
86                      this.totalResultados = response.data.length;
87                  } else {
88                      this.totalResultados = 0;
89                  }
90              },
91              (error) => {
92                  console.error('Error al obtener resultados:', error);
93              }
94          );
95      }
96      Complexity is 4 Everything is cool!
97      getTotalCuestionariosResueltos(): void {
98          if (this.usuarioID !== null) {
99              this.solvedQuizzesService.getCuestionariosResueltosPorAlumno(this.usuarioID).subscribe(
100                  (response) => {
101                      this.totalCuestionariosResueltos = response.data.length;
102                  },
103                  (error) => {
104                      console.error('Error al obtener cuestionarios resueltos:', error);
105                  }
106              );
107          }
108      }

```

Anexo 31. Funciones “getTotalResultados” y “getTotalCuestionariosResueltos”

```

108  getTotalCuestionariosPendientes(): void {
109      if (this.usuarioID !== null) {
110          this.alumnoGruposService.getAlumnoGrupos(this.usuarioID).subscribe(
111              Complexity is 6 It's time to do something...
112              (response) => {
113                  if (response.data && response.data.length > 0) {
114                      let totalPendientes = 0;
115                      Complexity is 3 Everything is cool!
116                      response.data.forEach(grupo => {
117                          const grupoId = grupo.GrupoID;
118                          console.log('Grupo ID:', grupoId);
119                          this.solvedQuizzesService.obtenerCuestionariosPendientes(this.usuarioID!, grupoId).subscribe(
120                              (cuestionariosResponse) => {
121                                  totalPendientes += cuestionariosResponse.data.length;
122                                  this.totalCuestionariosPendientes = totalPendientes;
123                              },
124                              (error) => {
125                                  console.error('Error al obtener cuestionarios pendientes para el grupo', grupoId, ':', error);
126                              }
127                          );
128                      });
129                  } else {
130                      this.totalCuestionariosPendientes = 0;
131                  }
132              },
133              (error) => {
134                  console.error('Error al obtener grupos:', error);
135                  this.totalCuestionariosPendientes = 0;
136              }
137          );
138      } else {
139          console.error('usuarioID es null');
140          this.totalCuestionariosPendientes = 0;
141      }

```

Anexo 32. Función “GetTotalCuestionariosPendientes”

Para ingresar a un grupo mediante una pop-up donde se ingresa el código de acceso, por lo que se creó una función “buscarGrupo” el cual a partir de buscar por código mediante el servicio “getGrupoByCodigo” y otra función “unirseGrupo” el cual permite al usuario ingresarlo al grupo relacionándolo con su id. También se agregó una función “cancelar” el cual como su nombre lo indica, cancela el proceso para ingresar a un grupo.

```
28 buscarGrupo() {
29     const codigoAcceso = this.form.get('codigo')?.value;
30     if (this.form.invalid) {
31         return;
32     }
33     this.loading = true;
34     const codigo = this.form.value.codigo_acceso;
35     this.grupoService.getGrupoByCodigo(codigo).subscribe(
36         (response) => {
37             this.grupoInfo = response.data;
38             this.loading = false;
39         },
40         (error) => {
41             this.error = error.error.message;
42             this.loading = false;
43             this.cancelar();
44         }
45     );
46 }
47
48 Complexity is 3 Everything is cool!
49 unirseGrupo(usuarioId: number) {
50     const codigo = this.form.value.codigo_acceso;
51     this.grupoService.joinGrupo(usuarioId, codigo).subscribe(
52         (response) => {
53             Swal.fire('Exito', 'Te has unido al grupo correctamente', 'success');
54         },
55         (error) => {
56             Swal.fire('Error', error.error.message, 'error');
57         }
58     );
59 }
60 cancelar() {
61     this.form.reset();
62     this.grupoInfo = null;
63     this.error = '';
64     this.close.emit();
65 }
```

Anexo 33. Funciones “buscarGrupo”, “unirseGrupo” y “cancelar”

```

<div class="overlay">
  <div class="pop-up">
    <button class="close-button" (click)="cancelar()"><mat-icon>close</mat-icon></button>
    <h2>Busca el Grupo</h2>

    <div class="input-group">
      <form [formGroup]="form" (ngSubmit)="buscarGrupo()">
        <label for="codigo_acceso">Código de Acceso:</label>
        <input id="codigo_acceso" formControlName="codigo_acceso" type="text" />
        <button type="submit" [disabled]="loading">Buscar Grupo</button>
      </form>
    </div>

    <div *ngIf="grupoInfo" class="grupo-container">
      <div class="grupo-info">
        <h3>GRUPO: {{ grupoInfo.Nombre }}</h3>
        <p><strong>Institución:</strong> {{ grupoInfo.carrera.institucion.Nombre}}</p>
        <p><strong>Semestre:</strong> {{ grupoInfo.semestre }}</p>
        <p><strong>Carrera:</strong> {{ grupoInfo.carrera.Nombre }}</p>
      </div>
      <div class="grupo-actions">
        <button (click)="unirseGrupo(1)">Unirse</button>
        <button (click)="cancelar()">Cancelar</button>
      </div>
      <div *ngIf="error">{{ error }}</div>
    </div>
  </div>
</div>

```

Anexo 34. Código de la vista de la ventana buscar grupo

Para la sección grupos se codificó la vista para mostrar la lista de grupos en las que está escrito el alumno estas tarjetas contiene información del grupo como el nombre de institución, carrera, semestre y nombre del grupo, también contiene un botón para ingresar a un grupo. La información del grupo se mostrará en una tarjeta con una función para visualizar los cuestionarios resueltos de ese grupo.

```

1 <section class="home">
2   <div class="home-box">
3     <h1><mat-icon>home</mat-icon> MIS GRUPOS</h1>
4   </div>
5 </section>
6
7 <div class="groups-header">
8   <button class="enter-group-btn" (click)="openEntrarGrupo()">Entrar al Grupo</button>
9 </div>
10
11 <div *ngIf="alumno">
12   <h2>{{ alumno.Nombre }} {{ alumno.Primer_apellido }} {{ alumno.Segundo_apellido }}</h2>
13 </div>
14
15 <div class="groups-container">
16   <div *ngFor="let group of groups; let i = index" (click)="enterGroup(group)" style="cursor: pointer;">
17     <div class="group-card">
18       <div class="icon-container" [ngStyle]="{'background': getRandomGradient()}">
19         <i class="material-icons">group</i>
20       </div>
21       <div class="info-container">
22         <p>{{ group.carrera?.institucion?.Nombre }}</p>
23         <p>{{ group.carrera.Nombre }}</p>
24         <p>Semestre: {{ group.semestre }} &nbsp; Grupo: {{ group.Nombre }}</p>
25       </div>
26     </div>
27   </div>
28 </div>
29
30 <div *ngIf="!groups || groups.length === 0">No hay grupos disponibles.</div>
31
32 <app-enter-group *ngIf="showEnterGroupPopup" (close)="closePopup()"></app-enter-group>
33
34

```

Anexo 35. Código de la vista de grupos.

Para el funcionamiento de esta vista se crearon ciertas funciones. Se añadió un inicializador para obtener el id del usuario para la autenticación. Se creó la función “getAlumnogrupos” el cual obtiene a partir del servicio “alumnoGruposService” los grupos a los que le pertenece el alumno. Otra función que se agregó fue la de “enterGroup” la cual redirige al alumno a la vista de los cuestionarios resueltos del grupo.

```
29 ngOnInit() {
30   const usuarioID = this.authService.getUserId();
31   if (usuarioID) {
32     this.getAlumnoGrupos(usuarioID);
33   } else {
34     console.error('No se encontró el usuario autenticado');
35   }
36   setTimeout(() => {
37     this.getRandomGradient();
38   });
39 }
40
41 getAlumnoGrupos(usuarioID: number) {
42   this.alumnoGruposService.getAlumnoGrupos(usuarioID).subscribe((response: any) => {
43     this.groups = response.data;
44     console.log('Grupos obtenidos:', this.groups);
45   });
46 }
47
48 enterGroup(group: { GrupoID: number }) {
49   console.log('Entering group:', group);
50   const groupId = group.GrupoID;
51   this.router.navigate(['/alumno/cuestionarios-resueltos'], { queryParams: { groupId } });
52 }
53
```

Anexo 36. Inicializador, Función “getAlumnos” y Función “enterGroup”.

Ahora para la vista de cuestionarios resueltos por grupo, se creó la vista el cual muestra la información del grupo tales como: la institución, carrera, semestre y grupo. También una lista de los cuestionarios resueltos por el alumno mostrando la información de la versión del cuestionario y la fecha la cual se resolvió. Si no existen los cuestionarios mostrará “No hay cuestionarios resueltos disponibles”.

```
2 <div *ngIf="grupo && grupo.Nombre && alumno" class="grupo-info">
3   <mat-icon class="grupo-icon">group</mat-icon>
4   <div class="grupo-text">
5     <p *ngIf="grupo.carrera?.Institucion?.Nombre">{{ grupo.carrera.Institucion.Nombre }}</p>
6     <p *ngIf="grupo.carrera?.Nombre">{{ grupo.carrera.Nombre }}</p>
7     <p>Semestre: {{ grupo.semestre }} &nbsp; Grupo: {{ grupo.Nombre }}</p>
8   </div>
9 </div>
10 <section class="home">
11   <div class="home-box">
12     <h1><mat-icon>check_circle</mat-icon> CUESTIONARIOS RESUELTOS</h1>
13   </div>
14 </section>
15
16 <div class="groups-container">
17   <div *ngFor="let quiz of solvedQuizzes">
18     <div class="quiz-card" (click)="goToQuizResults(quiz.CuestionarioID)">
19       <div class="icon-container" [ngStyle]="{'background': getRandomGradient()}">
20         <i class="material-icons">assignment_turned_in</i>
21       </div>
22       <div class="info-container">
23         <p>{{ quiz.Nombre }}</p>
24         <p>Fecha resuelto: {{ quiz.FechaResuelto | date:'dd/MM/yyyy' }}</p>
25       </div>
26     </div>
27   </div>
28
29   <div *ngIf="!solvedQuizzes.length">No hay cuestionarios resueltos disponibles.</div>
30
```

Anexo 37. Código de la vista de cuestionarios resueltos.

Para la funcionalidad del cuestionario para que pueda obtener los cuestionarios resueltos del alumno, se inicializa el id del usuario y el id del grupo para que a partir del servicio “alumnoGruposService” y “quizService” obtenga los cuestionarios que ha resuelto el alumno del grupo en específico.

```
ngOnInit() {
  const usuarioID = Number(localStorage.getItem('user_id'));
  const groupId = this.route.snapshot.queryParamMap.get('groupId');
  console.log('GroupId recibido:', groupId);

  if (!groupId) {
    console.error('No se encontró el groupId en los parámetros');
    return;
  }
  const groupIdNumber = parseInt(groupId, 10);
  if (isNaN(groupIdNumber)) {
    console.error('El groupId no es un número válido:', groupId);
    return;
  }
  console.log('groupId convertido:', groupIdNumber);
  this.alumnoGruposService.getAlumnoGrupos(usuarioID).subscribe(
    (response) => {
      console.log('Respuesta de grupos:', response);
      const grupos = response.data;
      if (grupos && grupos.length > 0) {
        this.grupo = grupos.find(grupo => grupo.GrupoID === groupIdNumber);
        console.log('Grupo encontrado:', this.grupo);
        if (this.grupo) {
          this.quizService.obtenerCuestionariosResueltos(usuarioID, this.grupo.GrupoID).subscribe(
            (quizzes) => {
              console.log('Cuestionarios resueltos:', quizzes);
              this.solvedQuizzes = quizzes.data;
            },
            (error) => {
              console.error('Error al obtener cuestionarios resueltos:', error);
            }
          );
        } else {
          console.warn('No se encontró el grupo con el groupId proporcionado');
        }
      } else {
        console.warn('No se encontraron grupos para este alumno');
      }
    }
  );
}
```

Anexo 38. Inicializador de cuestionarios resueltos.

Para los cuestionarios pendientes se creó la vista con el mismo formato de tarjetas, en este caso contiene el nombre del cuestionario, descripción, fecha de publicación y fecha de cierre.

```
<div class="cuestionarios-container">
  <h2>Cuestionarios Pendientes</h2>
  <div class="quiz-container">
    <div *ngFor="let cuestionario of cuestionariosPendientes" >
      <div class="quiz-card" (click)="irACuestionario(cuestionario.CuestionarioID, cuestionario.GrupoID)" >
        <div class="icon-container" [ngStyle]='{"background": getRandomGradient()}'>
          <i class="material-icons">pending_actions</i>
        </div>
        <div class="cuestionario-card">
          <p>{{ cuestionario.Nombre }}</p>
          <p>Descripción: {{ cuestionario.Descripcion }}</p>
          <p>Fecha de publicacion: {{cuestionario.FechaPublicacion}}</p>
          <p>Fecha de cierre: {{cuestionario.FechaCierre}}</p>
        </div>
      </div>
    </div>
  </div>
</div>
```

Anexo 39. Código de la vista del apartado cuestionarios pendientes.

Para la funcionalidad de este apartado, similar al componente anterior se inicializa el id del usuario para cargar los grupos del alumno para posteriormente cargar los cuestionarios que aún no han sido resueltos con las funciones “cargarGruposDe Alumno” y “cargarCuestionariosPendientes”. También se agregó una función para redirigir al cuestionario pendiente.

```
ngOnInit(): void {
  this.usuarioId = this.authService.getUserId();
  if (this.usuarioId !== null) {
    this.cargarGruposDelAlumno();
  } else {
    console.error('No se encontró el ID del usuario logueado.');
```

Complexity is 6 It's time to do something...

```
  }
  cargarGruposDelAlumno(): void {
    if (this.usuarioId !== null) {
      this.alumnoGruposService.getAlumnoGrupos(this.usuarioId).subscribe(
        response => {
          if (response && response.data) {
            this.gruposDelAlumno = response.data;
            this.cargarCuestionariosPendientes();
          } else {
            console.log('No se encontraron grupos para el alumno:', response);
          }
        },
        error => {
          console.error('Error al obtener grupos del alumno:', error);
        }
      );
    }
  }
}
```

Anexo 40. Inicializador y función “cargarGruposDelAlumno”.

```
Complexity is 6 It's time to do something...
cargarCuestionariosPendientes(): void {
  if (this.usuarioId !== null) {
    this.gruposDelAlumno.forEach(grupo => {
      this.solvedQuizzesService.obtenerCuestionariosPendientes(this.usuarioId!, grupo.GrupoID).subscribe(
        response => {
          if (response && response.data && response.data.length > 0) {
            this.cuestionariosPendientes = [...this.cuestionariosPendientes, ...response.data];
            console.log('Cuestionarios pendientes para el grupo ${grupo.GrupoID}:', response.data);
          } else {
            console.log('No se encontraron cuestionarios pendientes para el grupo ${grupo.GrupoID}.');
          }
        },
        error => {
          console.error('Error al obtener cuestionarios pendientes para el grupo ${grupo.GrupoID}:', error);
        }
      );
    });
  }
}
```

Anexo 41. Función “cargarCuestionariosPendientes”

```

    irACuestionario(cuestionarioID: number, grupoID: number) {
      console.log(`Navigating to quiz with ID: ${cuestionarioID} and group ID: ${grupoID}`);
      if (cuestionarioID && grupoID) {
        this.router.navigate([`/alumno/cuestionarios/${cuestionarioID}/preguntas`], {
          queryParams: { grupoID: grupoID }
        });
      } else {
        console.error('El ID del cuestionario o del grupo es indefinido o nulo.');
```

Anexo 42. Función “irACuestionario”.

Para visualizar el cuestionario del usuario se crearon una serie de funciones para darle una funcionalidad correcta al cuestionario. Primero en el inicializador se obtiene el id del usuario que ha sido autenticado y el id del cuestionario mediante el id del grupo. Luego la primera función “cargarCuestionario” a partir del “cuestionarioService” de la función “getPreguntasRespuestas” se muestran las preguntas y respuestas del cuestionario mostrando 5 preguntas por página.

```

30  < ngOnInit(): void {
31    this.usuarioID = this.authService.getUserId();
32  <  if (this.usuarioID === null) {
33    console.error('Usuario no logueado o ID de usuario no encontrado.');
```

Anexo 43. Inicializador del cuestionario y función “cargarCuestionario”.

De igual manera se creó la función para enviar las respuestas.

```
66   enviarRespuestas(event?: Event): void {
67     if (event) event.preventDefault();
68
69     const respuestas = Object.keys(this.respuestasSeleccionadas).map(preguntaId => ({
70       CuestionarioID: this.cuestionarioID,
71       PreguntaID: +preguntaId,
72       respuesta_valor: this.respuestasSeleccionadas[+preguntaId]
73     }));
74     if (this.usuarioID && this.cuestionarioID && this.grupoID) {
75       this.cuestionarioService.enviarRespuestas(this.usuarioID, respuestas).subscribe(
76         response => {
77           console.log('Respuestas guardadas:', response);
78           const data = {
79             UsuarioID: this.usuarioID,
80             CuestionarioID: this.cuestionarioID,
81             GrupoID: this.grupoID
82           };
83           this.solvedQuizzesService.marcarCuestionarioResuelto(data).subscribe(
84             res => {
85               console.log('Cuestionario marcado como resuelto:', res);
86               this.router.navigate(['/alumno/resultados/${this.usuarioID}/${this.cuestionarioID}']);
87             },
88             error => {
89               console.error('Error al marcar cuestionario como resuelto:', error);
90             }
91           );
92         },
93         error => {
94           console.error('Error al enviar respuestas:', error);
95         }
96       );
97     } else {
98       console.error('Faltan datos: UsuarioID, CuestionarioID o GrupoID.');
```

Anexo 44. Función “enviarRespuestas”.

```
102   paginaAnterior(): void {
103     if (this.paginaActual > 0) {
104       this.paginaActual--;
105       this.actualizarPreguntasPaginaActual();
106     }
107   }
108
109   paginaSiguiente(): void {
110     if (this.paginaActual < this.totalPaginas - 1) {
111       this.paginaActual++;
112       this.actualizarPreguntasPaginaActual();
113     }
114   }
115
116   isLastPage(): boolean {
117     return this.paginaActual === this.totalPaginas - 1;
118   }
119
120   actualizarPreguntasPaginaActual(): void {
121     this.preguntasPaginaActual = this.preguntas.slice(this.paginaActual * 5, (this.paginaActual + 1) * 5);
122   }
123 }
```

Anexo 45. Funciones para la navegación del cuestionario.

```
<div class="container my-5">
  <div class="text-center mb-4">
    <div class="title-bar d-flex align-items-center justify-content-between">
      <h1 class="display-4">{{ cuestionario.nombre }}</h1>
      
    </div>
    <br>
    <h2 class="lead">{{ cuestionario.descripcion }}</h2>
  </div>
  <form (ngSubmit)="enviarRespuestas($event)">
    <div *ngFor="let pregunta of preguntasPaginaActual" class="mb-4">
      <div class="card p-3 mb-3">
        <h3 class="card-title">{{ pregunta.Texto }}</h3>
        <div class="form-group">
          <div *ngFor="let respuesta of pregunta.respuestas">
            <div class="form-check">
```

Anexo 46. Código de la vista del cuestionario parte 1.

```
      <input
        type="radio"
        class="form-check-input"
        [name]='pregunta' + pregunta.PreguntaID
        [value]="respuesta.Valor"
        (change)="onAnswerChange(pregunta.PreguntaID, respuesta.Valor)"/>
      <label class="form-check-label">{{ respuesta.Texto }}</label>
    </div>
  </div>
</div>
</div>
</div>
</div>
</div>
<div class="d-flex justify-content-between">
  <button type="button" class="btn btn-secondary" (click)="paginaAnterior()" [disabled]="paginaActual === 0">Anterior</button>
  <button *ngIf="isLastPage();" type="button" (click)="enviarRespuestas()" class="btn btn-success">Enviar</button>
  <ng-template #siguienteButton>
    <button type="button" class="btn btn-primary" (click)="paginaSiguiente()" [disabled]="paginaActual >= totalPaginas - 1">Siguiente</button>
  </ng-template>
</div>
</form>
</div>
```

Anexo 47. Código de la vista del cuestionario parte 2.

Para visualizar los resultados del alumno se creó la vista de los resultados mostrando la información del alumno, tales como: Nombre, apellidos, correo, semestre y carrera. De igual manera contiene los puntajes obtenidos para cada variable y una gráfica para una mejor retroalimentación.

```
Go to component | You, hace 4 segundos | 2 authors (Fernanda-Chable and one other)
1 <div class="container my-5">
2   <div class="datos-student">
3     <div *ngIf="alumno">
4       <h2>Resultados de {{ alumno.Nombre }} {{ alumno.Primer_apellido }} {{ alumno.Segundo_apellido }} {{ alumno.Matricula }}</h2>
5       <p><strong>Nombre:</strong> {{ alumno.Nombre }} {{ alumno.Primer_apellido }} {{ alumno.Segundo_apellido }} </p>
6       <p><strong>Correo:</strong> {{ alumno.Correo }}</p>
7       <p><strong>Semestre:</strong> {{ alumno.Semestre }}</p>
8       <p><strong>Carrera:</strong> {{ alumno.CarreraID.Nombre }}</p>
9     </div>
10  </div>
```

Anexo 48. Código de la vista de resultados del cuestionario parte 1


```

<div class="text-center mb-4">
  <h2 class="display-5">Resultado de autoevaluación</h2>
</div>
<div class="row">
  <div class="col-md-8 offset-md-2">
    <div *ngIf="hasPuntajes">
      <div class="table-container">
        <table class="table table-striped table-sm">
          <thead>
            <tr>
              <th>Variable</th>
              <th>Puntaje</th>
            </tr>
          </thead>
          <tbody>
            <tr>
              <td>Factor de corrección</td>
              <td>{{ puntajes.puntaje_variable_0 }}</td>
            </tr>
            <tr>
              <td>Buscar y aprovechar las oportunidades</td>
              <td>{{ puntajes.puntaje_variable_1 }}</td>
            </tr>
            <tr>
              <td>Ser Persistentes</td>
              <td>{{ puntajes.puntaje_variable_2 }}</td>
            </tr>
            <tr>
              <td>Cumplimiento de compromisos</td>
              <td>{{ puntajes.puntaje_variable_3 }}</td>
            </tr>
            <tr>
              <td>Exigir eficiencia y calidad</td>
              <td>{{ puntajes.puntaje_variable_4 }}</td>
            </tr>
            <tr>
              <td>Correr riesgos moderados</td>
              <td>{{ puntajes.puntaje_variable_5 }}</td>
            </tr>
          </tbody>
        </table>
      </div>
    </div>
  </div>
</div>

```

Anexo 49. Código de la vista de resultados del cuestionario parte 2

```

<tr>
  <td>Fijar Objetivos</td>
  <td>{{ puntajes.puntaje_variable_6 }}</td>
</tr>
<tr>
  <td>Buscar y conseguir información</td>
  <td>{{ puntajes.puntaje_variable_7 }}</td>
</tr>
<tr>
  <td>Planificar y hacer seguimiento sistemático</td>
  <td>{{ puntajes.puntaje_variable_8 }}</td>
</tr>
<tr>
  <td>Persuasión y creación de redes de apoyo</td>
  <td>{{ puntajes.puntaje_variable_9 }}</td>
</tr>
<tr>
  <td>Tener autoconfianza</td>
  <td>{{ puntajes.puntaje_variable_10 }}</td>
</tr>
</tbody>
</table>
</div>
<!-- Gráfico de radar -->
<h2 class="display-5 text-center">Gráfico de autoevaluación</h2>
<div class="text-center">
  <canvas id="graficaAraña" width="800" height="0"></canvas>
</div>
</div>
<div *ngIf="!hasPuntajes">
  <p class="text-center">No se encontraron datos para el cuestionario seleccionado.</p>
</div>
</div>
</div>

```

Anexo 50. Código de la vista de resultados del cuestionario parte 3

Para darle funcionalidad a la vista de resultados del alumno se inicializan el id de usuario y el id de cuestionario, el cual llama el servicio “getStudent” para obtener la información del alumno mediante su id, para posteriormente almacenar la información en una variable data.

```

ngOnInit(): void {
  const usuarioID = this.route.snapshot.paramMap.get('usuarioID');
  const cuestionarioID = this.route.snapshot.paramMap.get('cuestionarioID');
  console.log('usuarioID:', usuarioID);
  console.log('cuestionarioID:', cuestionarioID);
  if (!usuarioID || !cuestionarioID) {
    console.error('ID de usuario o CuestionarioID no válidos:', { usuarioID, cuestionarioID });
    return;
  }
  const usuarioIDNumerico = +usuarioID;
  const cuestionarioIDNumerico = +cuestionarioID;
  // Llamar al servicio para obtener la información del alumno por ID
  this.userService.getStudent(usuarioIDNumerico).subscribe(
    (data) => {
      this.alumno = data.data; // Guardar la información del alumno en la variable
      console.log('Información del alumno:', this.alumno);
    },
    (error) => {
      console.error('Error al obtener los datos del alumno:', error);
    }
  );
}

```

Anexo 51. Inicializador de id de usuario y cuestionario.

De igual manera se obtiene los puntajes a partir del “servicioService” con la función “obtenerPuntajesPorCuestionario” mediante el id de usuario y el cuestionario, también se le agrego un mensaje de error en caso de no obtener puntajes validos o al no recibir los puntajes por problemas del servicio.

```
this.puntajeService.obtenerPuntajesPorCuestionario(usuarioIDNumerico, cuestionarioIDNumerico).subscribe(  
  Complexity is 15 You must be kidding  
  data => {  
    console.log('Datos obtenidos:', JSON.stringify(data, null, 2));  
    if (data && Object.keys(data).length > 0) {  
      this.puntajes = {  
        puntaje_variable_0: Number(data.puntaje_variable_0) || 0,  
        puntaje_variable_1: Number(data.puntajes_variables.puntaje_variable_1) || 0,  
        puntaje_variable_2: Number(data.puntajes_variables.puntaje_variable_2) || 0,  
        puntaje_variable_3: Number(data.puntajes_variables.puntaje_variable_3) || 0,  
        puntaje_variable_4: Number(data.puntajes_variables.puntaje_variable_4) || 0,  
        puntaje_variable_5: Number(data.puntajes_variables.puntaje_variable_5) || 0,  
        puntaje_variable_6: Number(data.puntajes_variables.puntaje_variable_6) || 0,  
        puntaje_variable_7: Number(data.puntajes_variables.puntaje_variable_7) || 0,  
        puntaje_variable_8: Number(data.puntajes_variables.puntaje_variable_8) || 0,  
        puntaje_variable_9: Number(data.puntajes_variables.puntaje_variable_9) || 0,  
        puntaje_variable_10: Number(data.puntajes_variables.puntaje_variable_10) || 0,  
      };  
      this.hasPuntajes = true;  
      setTimeout(() => this.createRadarChart(), 100);  
    } else {  
      console.warn('No se recibieron puntajes válidos', data);  
      this.hasPuntajes = false;  
    }  
  },  
  error => {  
    console.error('Error al obtener puntajes:', error);  
  }  
);  
}
```

Anexo 52. Obtener puntajes del cuestionario

Para crear la gráfica de telaraña se realizó la función “createRadarChart” el cual verifica que los puntajes obtenidos sean válidos para cada variable asignando el tipo de gráfico que se va a dibujar, configurando de igual manera el rango máximo y mínimo para mostrar los datos. Si en caso de que los datos no son válidos, la función muestra un mensaje de error en la consola.

```
createRadarChart(): void {
  if (this.puntajes && this.hasPuntajes) {
    const data = [
      this.puntajes.puntaje_variable_0,
      this.puntajes.puntaje_variable_1,
      this.puntajes.puntaje_variable_2,
      this.puntajes.puntaje_variable_3,
      this.puntajes.puntaje_variable_4,
      this.puntajes.puntaje_variable_5,
      this.puntajes.puntaje_variable_6,
      this.puntajes.puntaje_variable_7,
      this.puntajes.puntaje_variable_8,
      this.puntajes.puntaje_variable_9,
      this.puntajes.puntaje_variable_10
    ];
    const ctx = document.getElementById('graficaAraña') as HTMLCanvasElement;
    if (ctx) {
      if (this.chart) {
        this.chart.destroy();
      }
      this.chart = new Chart(ctx, {
        type: 'radar',
        data: {
          labels: [
            'Factor de corrección',
            'Buscar y aprovechar las oportunidades',
            'Ser Persistentes',
            'Cumplimiento de compromisos',
            'Exigir eficiencia y calidad',
            'Correr riesgos moderados',
            'Fijar Objetivos',
            'Buscar y conseguir información',
            'Planificar y hacer seguimiento sistemático',
            'Persuasión y creación de redes de apoyo',
            'Tener autoconfianza'
          ],
          datasets: data
        }
      });
    }
  }
}
```

Anexo 53. Función “createRadarChar” parte 1.

```

120     datasets: [{
121         label: 'Puntajes por variable',
122         data: data,
123         fill: true,
124         backgroundColor: 'rgba(34, 202, 236, 0.2)',
125         borderColor: 'rgb(34, 202, 236)',
126         pointBackgroundColor: 'rgb(34, 202, 236)',
127         pointBorderColor: '#fff',
128         pointHoverBackgroundColor: '#fff',
129         pointHoverBorderColor: 'rgb(34, 202, 236)'
130     }]
131     },
132     options: {
133         scales: {
134             r: {
135                 suggestedMin: 0,
136                 suggestedMax: 30,
137                 ticks: {
138                     stepSize: 5
139                 }
140             }
141         }
142     }
143 });
144 } else {
145     console.error('No se encontró el contexto del canvas para el gráfico.');
```

Anexo 53. Función “createRadarChar” parte 2.

Para el maestro se creó su propio menú este de igual manera que el Dashboard del alumno obtiene la información del usuario, pero en este caso cambia las secciones del menú. Ahora las secciones que contiene es el Home, grupos, cuestionarios aplicados y alumnos.

```
<div *ngIf="isLoggedIn" class="dashboard">
  <div class="sidebar">
    <div class="user-info">
      
      <p *ngIf="teacher">{{ teacher.Nombre }} {{ teacher.Primer_apellido }} {{ teacher.Segundo_apellido }}</p>
      <p *ngIf="teacher">{{ teacher.Correo }}</p>
    </div>
    <div class="menu">
      <a [routerLink]="['/maestro/dashTeacher']">
        <mat-icon>home</mat-icon> Home
      </a>
      <a [routerLink]="['/maestro/gruopsTeacher']">
        <mat-icon>groups</mat-icon> Grupos
      </a>
      <a [routerLink]="['/maestro/applied-quiz']">
        <mat-icon>fact_check</mat-icon> Cuestionarios aplicados
      </a>
      <a [routerLink]="['/maestro/listAlumnos']">
        <mat-icon>contacts_product</mat-icon> Alumnos
      </a>
      <a (click)="logout()" *ngIf="isLoggedIn">
        <mat-icon>logout</mat-icon> Cerrar Sesión
      </a>
    </div>
  </div>
  <div class="main-content">
    <div class="content-container">
      <router-outlet></router-outlet>
    </div>
  </div>
</div>
```

Anexo 54. Código de la vista del menú del maestro.

Como en el caso del alumno se usa la misma estructura para obtener la información del maestro, mediante el id de usuario y también incluye la función para cerrar la sesión.

```
ngOnInit(): void {
  this.isLoggedIn = this.authService.isLoggedIn();

  if (this.isLoggedIn) {
    this.loadTeacherData();
  } else {
    console.error('El usuario no está autenticado');
  }
}

Complexity is 5 Everything is cool!
loadTeacherData(): void {
  const teacherId = this.authService.getUserId();
  console.log('ID del maestro:', teacherId);
  if (teacherId) {
    this.userService.getTeacher(teacherId).subscribe(
      response => {
        console.log('Respuesta del API:', response);
        if (response.data) {
          this.teacher = response.data;
          console.log('Datos del maestro:', this.teacher);
        } else {
          console.error('No se encontraron datos del maestro');
        }
      },
      error => {
        console.error('Error al cargar datos del maestro', error);
      }
    );
  } else {
    console.error('No se encontró un ID de usuario autenticado');
  }
}

// Método para cerrar sesión
logout(): void {
  this.authService.logout();
  this.isLoggedIn = false;
  this.router.navigate(['/login']);
}
```

Anexo 55. Funciones para el menú del maestro.

El maestro podrá crear grupo con la siguiente función la cual al llenar el formulario con la función “onSubmit” el realiza una subscripción al servicio “createGroup” del “teacherGroupsService” generando un código de acceso aleatoriamente, si el proceso se ha realizado correctamente muestre un mensaje indicando que el grupo de ha creado exitosamente y en caso de que no, muestre el error.

```
Complexity is 7 It's time to do something...
onSubmit() {
  if (this.groupForm.valid) {
    const groupData = {
      ...this.groupForm.value,
      usuario_id: this.userid
    };
    this.teacherGroupsService.createGroup(groupData).subscribe({
      Complexity is 4 Everything is cool!
      next: (response) => {
        console.log('Grupo creado exitosamente:', response);
        const codigoAcceso = response.data?.codigo_acceso;
        console.log('Codigo de acceso', codigoAcceso);
        if (codigoAcceso) {
          Swal.fire({
            title: '¡Grupo Creado Exitosamente!',
            text: `El código de acceso es: ${codigoAcceso}`,
            icon: 'success',
            confirmButtonText: 'Aceptar'
          }).then((result) => {
            if (result.isConfirmed) {
              this.router.navigate(['/maestro/createGroup']);
            }
          });
        } else {
          this.snackBar.open('Grupo creado, pero no se obtuvo el código de acceso.', 'Cerrar', {
            duration: 5000,
          });
        }
        this.groupForm.reset();
      },
    });
  }
}
```

Anexo 56. Función almacenar datos del grupo parte 1.

```

    error: (error) => {
      console.error('Error al crear el grupo', error);
      this.snackBar.open('Error al crear el grupo. Inténtalo de nuevo.', 'Cerrar', {
        duration: 5000,
      });
    }
  });
} else {
  this.snackBar.open('Por favor, completa todos los campos requeridos.', 'Cerrar', {
    duration: 5000,
  });
}
}
}
```

Anexo 57. Función almacenar datos del grupo parte 1.

El maestro tiene la función de publicar cuestionarios a sus grupos por ello se creó el componente para dicha función. Para el funcionamiento del componente se inicializa el id del maestro para obtener sus grupos, de igual manera los cuestionarios existentes, posteriormente se cargan las preguntas del cuestionario.

```
34   ngOnInit() {
35     const usuarioID = this.authService.getUserId();
36     if (usuarioID !== null) {
37       this.obtenerCuestionarios();
38       this.obtenerGrupos(usuarioID);
39     } else {
40       console.error('No se encontró el ID de usuario logueado');
41     }
42   }
43 }
44 obtenerCuestionarios() {
45   this.cuestionarioService.getQuizVersions().subscribe((data: any) => {
46     console.log('Cuestionarios:', data.cuestionarios);
47     this.cuestionarios = Array.isArray(data.cuestionarios) ? data.cuestionarios : [];
48   });
49 }
50 obtenerGrupos(usuarioID: number) {
51   this.teacherGroupsService.getGruposDeMaestro(usuarioID).subscribe((data: any) => {
52     console.log('Grupos:', data);
53     this.grupos = Array.isArray(data) ? data : [];
54   });
55 }
56 cargarPreguntas() {
57   if (this.cuestionarioSeleccionado) {
58     this.cuestionarioService.getPreguntasRespuestas(this.cuestionarioSeleccionado).subscribe((data: any) => {
59       console.log('Preguntas:', data.preguntas);
60       this.preguntas = data.preguntas;
61     });
62   }
63 }
```

Anexo 58. Inicializador, funciones “obtenerCuestionarios”, “obtenerGrupos” y “cargarPreguntas”.

Para publicar el cuestionario se realizó la función “publicarCuestionario” en ella se realiza la condición si la fecha de publicación y fecha de cierre han sido establecidas, obtiene el id del grupo al cual se va a publicar y las fechas anteriormente para realizar la subscripción al “cuestionarioService” para la función “publishQuiz”, si el cuestionario se publica correctamente se mostrará un mensaje el cual indica que el cuestionario se publicó correctamente y si no muestra el mensaje error al publicar el cuestionario. También se puede cancelar el proceso.

```
publicarCuestionario() {  
  if (!this.fechaPublicacion || !this.fechaCierre) {  
    alert('Por favor, completa ambas fechas antes de publicar.');    return;  
  }  
  const payload = {  
    grupo_id: this.grupoSeleccionado,  
    fecha_publicacion: this.fechaPublicacion,  
    fecha_cierre: this.fechaCierre,  
  };  
  
  this.cuestionarioService.publishQuiz(this.cuestionarioSeleccionado, payload).subscribe({  
    next: () => {  
      alert('Cuestionario publicado exitosamente');  
      this.router.navigate(['/maestro/quiz']);  
    },  
    error: (err) => {  
      console.error('Error al publicar el cuestionario:', err);  
      alert('Hubo un error al publicar el cuestionario.');    },  
  });  
}  
  
cancelar() {  
  this.router.navigate(['/maestro/quiz']);  
}
```

Anexo 60. Función “publicarCuestionario”.

El maestro Puede visualizar la información del grupo con sus integrantes del grupo, de igual manera los cuestionarios que han sido aplicados en el siguiente componente.

```

Go to component | You, hace 48 segundos | 2 authors (Fernanda-Chable and one other)
1  <div *ngIf="grupo">
2    <section class="quiz-details">
3
4      <div class="quiz-header">
5        <h2><strong>Carrera:</strong> {{ grupo.carrera.Nombre }}</h2>
6        <h2><strong>Grupo:</strong> {{ grupo.Nombre }}&nbsp;<strong>Semestre:</strong> {{ grupo.semestre }}</h2>
7        <h2><strong>Código de acceso:</strong> {{ grupo.codigo_acceso }}</h2>
8      </div>
9
10     <div class="group-members">
11       <h3>Integrantes del Grupo</h3>
12       <table>
13         <thead>
14           <tr>
15             <th>Nombre</th>
16             <th>Primer Apellido</th>
17             <th>Segundo Apellido</th>
18             <th>Carrera</th>
19           </tr>
20         </thead>
21         <tbody>
22           <tr *ngFor="let integrante of integrantes">
23             <td>{{ integrante.Nombre }}</td>
24             <td>{{ integrante.Primer_apellido }}</td>
25             <td>{{ integrante.Segundo_apellido }}</td>
26             <td>{{ integrante.Carrera.Nombre }}</td>
27           </tr>
28         </tbody>
29       </table>
30     </div>
31
32     <div class="quiz-list">
33       <h3>Cuestionarios Aplicados</h3>
34       <ul>
35         <li *ngFor="let cuestionario of cuestionarios" (click)="selectCuestionario(cuestionario)">
36           <p><strong>Nombre:</strong> {{ cuestionario.Nombre }}</p>
37           <p><strong>Detalles de Cuestionario:</strong> {{ cuestionario.Descripcion }}</p>
38         </li>
39       </ul>
40     </div>
41

```

Anexo 61. Código la vista de información del cuestionario aplicado.

Se crearon varias funciones para la vista de información del cuestionario aplicado al grupo, al acceder a la vista se obtiene el id del grupo, de igual manera se inicializan las funciones para obtener detalles del grupo, obtener cuestionarios aplicados y obtener miembros del grupo.

```
36   ngOnInit(): void {
37     this.grupoId = +this.route.snapshot.paramMap.get('grupoId')!;
38     this.getGroupDetails();
39     this.getAppliedQuizzes();
40     this.getGroupMembers();
41   }
42
43   Complexity is 3 Everything is cool!
44   getGroupDetails() {
45     this.teacherGroupsService.getGroupById(this.grupoId).subscribe(
46       (response) => {
47         this.grupo = response.data;
48       },
49       (error) => {
50         console.error('Error al obtener los detalles del grupo:', error);
51       }
52     );
53   },
54   Complexity is 3 Everything is cool!
55   getAppliedQuizzes() {
56     this.teacherGroupsService.getAppliedQuizzesByGroup(this.grupoId).subscribe(
57       (response) => {
58         this.cuestionarios = response.data;
59       },
60       (error) => {
61         console.error('Error al obtener los cuestionarios aplicados:', error);
62       }
63     );
64   },
65   Complexity is 3 Everything is cool!
66   getGroupMembers() {
67     this.teacherGroupsService.getIntegrantes(this.grupoId).subscribe(
68       (response) => {
69         this.integrantes = response.data;
70       },
71       (error) => {
72         console.error('Error al obtener los integrantes del grupo:', error);
73       }
74     );
75   }
```

Anexo 62. Inicializador y funciones “getGroupDetails”, “getAppliedQuizzes” y “getGroupMembers”.

Para el Dashboard de administrador Funciona de la misma manera que el de maestro y alumno obteniendo la información del administrador, pero en este caso las secciones que posee son: Países, Instituciones, Administración de cuestionarios, Administración de usuarios, Cuestionarios aplicados y Edición de la página principal.

```
Go to component | You, el mes pasado | 1 author (You)
1 <div class="main-content">
2 <header class="header">
3 <div class="header-box">
4 <h1>Bienvenido, {{ currentDate }}</h1>
5 </div>
6 <div class="search-and-button">
7 <input type="search" placeholder="Search">
8 </div>
9 </header>
10
11
12 <section class="home">
13 <div class="home-box">
14 <h1><mat-icon>home</mat-icon> HOME</h1>
15 </div>
16 </section>
17
18 <div class="buttons">
19 <button class="button blue" [routerLink]="['/administrador/lista-estudiantes']">
20 <mat-icon>account_circle</mat-icon> Alumnos <span class="counter">{{totalAlumnos}}</span>
21 </button>
22 <button class="button orange" [routerLink]="['/administrador/lista-docentes']">
23 <mat-icon>account_circle</mat-icon> Maestros <span class="counter">{{ totalMaestros }}</span>
24 </button>
25 <button class="button orange" [routerLink]="['/administrador/lista-docentes']">
26 <mat-icon>account_circle</mat-icon> Informes <span class="counter">{{ totalMaestros }}</span>
27 </button>
28 <button class="button green" [routerLink]="['/administrador/instituciones']">
29 <mat-icon>check_circle</mat-icon> Instituciones <span class="counter">{{ totalInstituciones }}</span>
30 </button>
31 <button class="button pink" [routerLink]="['/alumno/pendiente']">
32 <mat-icon>text_snippet</mat-icon> Cuestionarios aplicados <span class="counter">{{ totalCuestionariosPendientes }}</span>
33 </button>
34 </div>
35
36 </div>
```

Anexo 63. Código de la vista del Dashboard del administrador.

```
Go to component | You, antayayr | 1 author (You)
1 <div *ngIf="isLoggedIn" class="dashboard">
2 <div class="sidebar">
3 <div class="user-info" (click)="navigateToProfile()">
4 
5 <p *ngIf="Administrator">{{ Administrator.Nombre }} {{ Administrator.Primer_apellido }} {{ Administrator.Segundo_apellido }}</p>
6 <p *ngIf="Administrator">{{ Administrator.Correo }}</p>
7 </div>
8 <div class="menu">
9 <a [routerLink]="['/administrador/dash-admin']"> You, el mes pasado • Frontend Creación del Dashboard de administrador...
10 <mat-icon>home</mat-icon> Home
11 </a>
12 <a [routerLink]="['/administrador/paises']">
13 <mat-icon>language</mat-icon> Países
14 </a>
15 <a [routerLink]="['/administrador/instituciones']">
16 <mat-icon>groups</mat-icon> Instituciones
17 </a>
18 <a [routerLink]="['/administrador/quiz']">
19 <mat-icon>quiz</mat-icon> Administracion de cuestionarios
20 </a>
21
22 <a [routerLink]="['/administrador/usuarios']">
23 <mat-icon>groups</mat-icon> Administracion de usuarios
24 </a>
25
26 <a [routerLink]="['']">
27 <mat-icon>view_timeline</mat-icon> Cuestionarios aplicados
28 </a>
29
30 <a [routerLink]="['']">
31 <mat-icon>edit_square</mat-icon> Edición de la página principal
32 </a>
33
34 <a (click)="logout()" *ngIf="isLoggedIn">
35 <mat-icon>logout</mat-icon> Cerrar Sesión
36 </a>
37 </div>
38 </div>
```

Anexo 64. Código de la vista del menú del administrador.

En el apartado de países se creó una función a partir del servicio “getCountries” de catalogs el cual se cargan todos los países registrados en el sistema, también se puede obtener un país con la función “filterCountries”.

```
Complexity is 4 Everything is cool!
30 loadCountries(): void {
31   this.catalogs.getCountries().subscribe({
32     next: (response) => {
33       this.countries = response.data || [];
34       this.filteredCountries = this.countries;
35     },
36     error: (error) => {
37       console.error('Error al cargar los países', error);
38     }
39   });
40 }
41
42 goToInstitutions(countryId: number): void {
43   this.router.navigate(['/administrador/institucionesf', countryId]);
44 }
Complexity is 3 Everything is cool!
45 filterCountries(): void {
46   // Filtrar la lista de países en base a la consulta de búsqueda
47   this.filteredCountries = this.countries.filter(country =>
48     this.searchQuery ? country.Nombre.toLowerCase().includes(this.searchQuery.toLowerCase()) : true
49   );
50 }
51
```

Anexo 65. Funciones “loadCountries” y “filterCountries”.

Para agregar un país se creó un formulario el cual contiene una suscripción al servicio “CreateCountries” agregando el id del país y su nombre. Si el país se agregó correctamente mostrará el mensaje del país agregado y si no mostrará el error obtenido. Se agregó una función para cancelar el proceso.

```
20 constructor(  
21   private fb: FormBuilder,  
22   private catalogs: CatalogsService,  
23   public dialogRef: MatDialogRef<AddCountryComponent>  
24 ) {  
25   this.countryForm = this.fb.group({  
26     Nombre: ['', Validators.required]  
27   });  
28 }  
29  
30 Complexity is 5 Everything is cool!  
31 onSubmit(): void {  
32   if (this.countryForm.valid) {  
33     const newCountry: CreateCountries = { IDPais: 0, Nombre: this.countryForm.value.Nombre };  
34     this.catalogs.createCountry(newCountry).subscribe({  
35       next: (response) => {  
36         console.log('País agregado:', response);  
37         this.dialogRef.close(true);  
38       },  
39       error: (error: any) => {  
40         console.error('Error al agregar país:', error);  
41       }  
42     });  
43   }  
44 }  
45 onCancel(): void {  
46   this.dialogRef.close(false);  
47 }  
48 }
```

Anexo 66. Código componente agregar país

De la misma forma para la lista de países se realizó el proceso para las instituciones. En este caso se añadió un filtro para obtener las instituciones filtradas por país.

```
26 ngOnInit(): void {  
27   this.loadInstitutions();  
28   this.loadCountries();  
29 }  
30 Complexity is 4 Everything is cool!  
31 loadInstitutions(): void {  
32   this.catalogs.getInstitutions().subscribe({  
33     next: (response) => {  
34       this.institutions = response.data || [];  
35       this.filteredInstitutions = this.institutions;  
36     },  
37     error: (error) => {  
38       console.error('Error al cargar las instituciones', error);  
39     }  
40   });  
41 }  
42 Complexity is 4 Everything is cool!  
43 loadCountries(): void {  
44   this.catalogs.getCountries().subscribe({  
45     next: (response) => {  
46       this.countries = response.data || [];  
47     },  
48     error: (error) => {  
49       console.error('Error al cargar los países', error);  
50     }  
51   });  
52 }
```

Anexo 67. Funcionalidad de la vista lista de instituciones.

```

51  filterInstitutions(): void {
52    if (this.selectedCountryId) {
53      // Si hay un país seleccionado, utilizar el servicio para obtener solo las instituciones de ese país
54      this.catalogs.getInstitutionsId(this.selectedCountryId).subscribe({
55        next: (response) => {
56          // Filtrar por nombre en base al query de búsqueda localmente
57          this.filteredInstitutions = response.data?.filter(institution =>
58            this.searchQuery ? institution.Nombre.toLowerCase().includes(this.searchQuery.toLowerCase()) : true
59          ) || [];
60        },
61        error: (error) => {
62          console.error('Error al cargar las instituciones filtradas', error);
63          this.filteredInstitutions = [];
64        }
65      });
66    } else {
67      // Si no hay un país seleccionado, utilizar el servicio getInstitutions() para obtener todas las instituciones
68      this.catalogs.getInstitutions().subscribe({
69        next: (response) => {
70          // Filtrar por nombre en base al query de búsqueda localmente
71          this.filteredInstitutions = response.data?.filter(institution =>
72            this.searchQuery ? institution.Nombre.toLowerCase().includes(this.searchQuery.toLowerCase()) : true
73          ) || [];
74        },
75        error: (error) => {
76          console.error('Error al cargar todas las instituciones', error);
77          this.filteredInstitutions = [];
78        }
79      });
80    }
81  }

```

Anexo 68. Función de Instituciones filtradas por país.

Para obtener los maestros de una institución, se crearon unas funciones para obtener los maestros de la institución.

```
export class ListTeachersInstitutionsComponent {
  teachers: getTeachers[] = [];
  InstitutionID: string | null = null;
  institution: Institutions | undefined;

  constructor(private router: Router, private route: ActivatedRoute,
    private userService: UserService, private catalogs: CatalogsService,
    private location: Location){}

  ngOnInit(): void {
    this.InstitutionID = this.route.snapshot.paramMap.get('id');
    if(this.InstitutionID){
      this.loadTeachersInst(this.InstitutionID);
      this.loadInstitution(this.InstitutionID);
    }
  }

  loadTeachersInst(institutionID: string): void {
    this.userService.getTeachersInst(institutionID).subscribe(response => {
      this.teachers = response.data;
    });
  }

  loadInstitution(InstitutionID: string): void {
    this.catalogs.getInstitution(InstitutionID).subscribe({
      next: (response: any) => {
        console.log('API Response:', response);
        this.institution = response.data.institution;
      },
      error: (error) => {
        console.error('Error al cargar la institución', error)
      }
    });
  }

  goBack(): void {
    this.location.back();
  }
}
```

Anexo 69. Funcionalidad de la lista de maestros por institución.

Para agregar un maestro a una institución se usa la misma estructura que el registro de alumnos, pero en este caso el formulario se tiene por preestablecido el tipo de usuario como Maestro y de igual manera el semestre se mantiene en nulo ya que un maestro no posee un semestre en específico.

```
this.formGroup = new FormGroup({
  nombre: new FormControl('', [nameValidator, makeRequired, lengthValidator({ min: 1, max: 30 })]),
  papellido: new FormControl('', [nameValidator, makeRequired, lengthValidator({ min: 1, max: 15 })]),
  sapellido: new FormControl('', [nameValidator, makeRequired, lengthValidator({ min: 1, max: 15 })]),
  matricula: new FormControl('', [makeRequired, lengthValidator({ min: 1, max: 15 })]),
  semestre: new FormControl('Null', []),
  tipo: new FormControl('Maestro', []),
  correo: new FormControl('', [emailValidator(), makeRequired, lengthValidator({ min: 1, max: 30 })]),
  contraseña: new FormControl('', [makeRequired, lengthValidator({ min: 8, max: 15 })]),
  confirmcontraseña: new FormControl('', [makeRequired, lengthValidator({ min: 8, max: 15 })]),
  edad: new FormControl('', [makeRequired]),
  genero: new FormControl('', [makeRequired]),
  institucion: new FormControl('', [makeRequired]),
  pais: new FormControl('', [makeRequired]),
  carrera: new FormControl('', [])
}, { validators: confirmPassword });

this.loadCountriesAndProcessParams();
this.formGroup.get('pais')?.valueChanges.subscribe(selectedCountryId => {
  this.onCountryChange(selectedCountryId);
});
this.ages = Array.from({ length: 99 - 17 + 1 }, (_, i) => i + 17);
}
```

Anexo 70. Código del formulario del maestro.

Para el caso de obtener la lista de integrantes del grupo del maestro, se inicializa el id del maestro y el id del grupo para obtener los miembros del grupo. Se crearon 2 funciones las cuales son para obtener los datos del maestro y cargar los integrantes del grupo a partir de los servicios “getTeachers” y “getIntegrantes”.

```
Complexity is 3 Everything is cool!
ngOnInit(): void {
  this.grupoId = this.route.snapshot.paramMap.get('grupoId');
  const usuarioId = this.route.snapshot.queryParamMap.get('usuarioId');

  if (this.grupoId) {
    this.loadGroupMembers(Number(this.grupoId));
    this.loadGroupDetails(Number(this.grupoId));
  }
  if (usuarioId) {
    this.loadTeacherDetails(Number(usuarioId));
  }
}

Complexity is 4 Everything is cool!
loadTeacherDetails(usuarioId: number): void {
  this.userService.getTeacher(usuarioId).subscribe({
    next: (response: any) => {
      this.teacher = response.data;
    },
    error: (error) => {
      console.error('Error al cargar los detalles del maestro', error);
    }
  });
}

Complexity is 5 Everything is cool!
loadGroupMembers(grupoId: number): void {
  Complexity is 4 Everything is cool!
  this.teacherGroupsService.getIntegrantes(grupoId).subscribe(response => {
    console.log('Respuesta completa del servicio:', response);
    this.students = response.data.map((student: any) => ({
      ...student,
      semestre: student.Semestre,
    }));

    const institutionID = response.data[0]?.Carrera?.Institucion?.InstitucionID;
    if (institutionID) {
      this.loadInstitution(institutionID.toString());
    } else {
      console.error('No se pudo encontrar el InstitucionID en los datos del grupo');
    }
  });
}
```

Anexo 71. Funcionalidad para obtener los datos del maestro y los miembros del grupo.

Para la lista de alumnos por institución se cargan los estudiantes que pertenecen a la institución a partir del servicio “getStudentsInst” de userService.

```
loadStudentInst(institutionID: string): void {
  this.userService.getStudentsInst(institutionID).subscribe(response => {
    this.students = response.data;
    this.filteredStudents = this.students; // Inicializa con todos los estudiantes
  });
}
```

Anexo 72. Funcionalidad de lista de alumnos de la institución.

Para obtener todos los alumnos del sistema, se inicializó el servicio “getStudents” del “userService”. En este caso se agregó una función para filtrar los estudiantes por carrera, institución o país. También se agregó una función para eliminar un estudiante del sistema.

```
ngOnInit(): void {
  this.userService.getStudents().subscribe(response => {
    this.students = response.data;
    this.filteredStudents = [...this.students];
    this.extractUniqueFilters();
  });
}
Complexity is 4 Everything is cool!
extractUniqueFilters(): void {
  this.uniqueCarreras = [...new Set(this.students.map(student => student.CarreraID.Nombre))];
  this.uniqueInstituciones = [...new Set(this.students.map(student => student.InstitucionID.Nombre))];
  this.uniquePaises = [...new Set(this.students.map(student => student.IDPais.Nombre))];
}
Complexity is 7 It's time to do something...
filterStudents(): void {
  Complexity is 6 It's time to do something...
  this.filteredStudents = this.students.filter(student =>
    (this.selectedCarrera ? student.CarreraID.Nombre === this.selectedCarrera : true) &&
    (this.selectedInstitucion ? student.InstitucionID.Nombre === this.selectedInstitucion : true) &&
    (this.selectedPais ? student.IDPais.Nombre === this.selectedPais : true)
  );
}
Complexity is 6 It's time to do something...
removeStudent(UsuarioID: number): void {
  if (confirm('¿Estás seguro de que deseas eliminar este estudiante?')) {
    this.userService.deleteStudent(UsuarioID).subscribe({
      Complexity is 3 Everything is cool!
      next: () => {
        // Actualizar la lista de estudiantes después de eliminar
        this.students = this.students.filter(student => student.UsuarioID !== UsuarioID);
        this.filteredStudents = this.filteredStudents.filter(student => student.UsuarioID !== UsuarioID);
      },
      error: (error) => {
        console.error('Error al eliminar el estudiante:', error);
        alert('Hubo un error al intentar eliminar el estudiante. Por favor, intenta de nuevo.');
```

Para la administración de maestros del sistema, se realizó de la misma forma que los alumnos por lo que se inicializó el servicio “getTeachers” del “userService”. Se agregó una función para filtrar los maestros por institución o país. También se agregó una función para eliminar un maestro del sistema.

```
ngOnInit(): void {
  this.userService.getTeachers().subscribe(response => {
    this.teachers = response.data;
    this.filteredTeachers = [...this.teachers];
    this.extractUniqueFilters();
  });
}
Complexity is 3 Everything is cool!
extractUniqueFilters(): void {
  this.uniqueInstituciones = [...new Set(this.teachers.map(teacher => teacher.InstitucionID.Nombre))];
  this.uniquePaises = [...new Set(this.teachers.map(teacher => teacher.IDPais.Nombre))];
}
Complexity is 5 Everything is cool!
filterTeachers(): void {
  Complexity is 4 Everything is cool!
  this.filteredTeachers = this.teachers.filter(teacher =>
    (this.selectedInstitucion ? teacher.InstitucionID.Nombre === this.selectedInstitucion : true) &&
    (this.selectedPais ? teacher.IDPais.Nombre === this.selectedPais : true)
  );
}
Complexity is 7 It's time to do something...
removeTeacher(UsuarioID: number): void {
  if (confirm('¿Estás seguro de que deseas eliminar este docente?')) {
    this.userService.deleteTeacher(UsuarioID).subscribe({
      Complexity is 3 Everything is cool!
      next: () => {
        this.teachers = this.teachers.filter(teacher => teacher.UsuarioID !== UsuarioID);
        this.filteredTeachers = this.filteredTeachers.filter(teacher => teacher.UsuarioID !== UsuarioID);
      },
      error: (error: any) => {
        console.error('Error al eliminar el docente:', error);
        alert('Hubo un error al intentar eliminar el docente. Por favor, intenta de nuevo.');
```

Se creó un componente para obtener la lista de cuestionarios existentes, en ella se creó una función para cargar los cuestionarios a partir del servicio “getQuizVersions” del cuestionarioService donde se obtienen todos los cuestionarios existentes. También se agregaron funciones para ver los detalles del cuestionario y crear el cuestionario.

```
ngOnInit() {
  this.cargarCuestionarios();
}

Complexity is 9 It's time to do something...
cargarCuestionarios() {
  Complexity is 8 It's time to do something...
  this.cuestionarioService.getQuizVersions().subscribe((data: any) => {
    Complexity is 6 It's time to do something...
    this.cuestionarios = data.cuestionarios.map((cuestionario: any) => ({
      ...cuestionario,
      fecha_publicacion:
        cuestionario.fecha && cuestionario.fecha.length > 0
        ? cuestionario.fecha[0].FechaPublicacion
        : null,
      fecha_cierre:
        cuestionario.fecha && cuestionario.fecha.length > 0
        ? cuestionario.fecha[0].FechaCierre
        : null,
    }));
  });
}

verDetallesCuestionario(cuestionarioID: number) {
  this.router.navigate(['administrador/quiz/${cuestionarioID}/detalles']);
}

irACrearCuestionario() {
  this.router.navigate(['administrador/create-quiz']);
}
}
```

Anexo 74. Funcionalidades de la lista de cuestionarios.

Para la funcionalidad de ver los detalles del cuestionario para obtener la información de los cuestionarios se creó la función “cargarCuestionario” a partir del servicio “getPreguntasRespuestas” llamando el id del cuestionario y almacenarlo en una variable data, posteriormente se agregó una función para obtener 10 preguntas por página para la función de paginado.

```
33 Complexity is 3 Everything is cool!
34 cargarCuestionario(cuestionarioID: number): void {
35   this.cuestionarioService.getPreguntasRespuestas(cuestionarioID).subscribe({
36     next: (data) => {
37       this.cuestionario = data;
38       this.preguntas = data.preguntas;
39       this.totalPaginas = Math.ceil(this.preguntas.length / this.preguntasPorPagina);
40     },
41     error: (err) => {
42       console.error('Error al cargar el cuestionario:', err);
43       alert('No se pudo cargar el cuestionario.');
```

```
44       this.router.navigate(['/quiz']);
45     }
46   });
47 }
48
49 Complexity is 3 Everything is cool!
50 obtenerPreguntasPagina(): any[] {
51   const inicio = (this.paginaActual - 1) * this.preguntasPorPagina;
52   const fin = this.paginaActual * this.preguntasPorPagina;
53   return this.preguntas.slice(inicio, fin);
54 }
55 siguientePagina(): void {
56   if (this.paginaActual < this.totalPaginas) {
57     this.paginaActual++;
58   }
59 }
60
```

Anexo 75. Funcionalidades para la vista detalles del cuestionario.

Para crear la nueva versión del cuestionario se agregaron funciones para poder realizar la funcionalidad. Se creó la función “cargarCuestionarioOriginal” el cual a partir de obtener el id del cuestionario con servicio “getPreguntasRespuestas” se obtiene la información del cuestionario original y lo almacena en una variable data para que solamente obtenga las preguntas, variables y signos.

```
ngOnInit(): void {  
  this.cargarCuestionarioOriginal();  
}  
  
Complexity is 5 Everything is cool!  
cargarCuestionarioOriginal(): void {  
  this.cuestionarioService.getPreguntasRespuestas(this.cuestionarioID).subscribe({  
    Complexity is 3 Everything is cool!  
    next: (data) => {  
      this.nombre = ''; // Nombre vacío por defecto  
      this.descripcion = ''; // Descripción vacía por defecto  
      this.preguntas = data.preguntas.map((pregunta: any) => ({  
        ...pregunta,  
        Texto: pregunta.Texto, // Copiar el texto original de la pregunta  
        Tipo: pregunta.Tipo, // Copiar el tipo original de la pregunta  
      }));  
    },  
    error: (err) => {  
      console.error('Error al cargar el cuestionario original:', err);  
      alert('Hubo un error al cargar el cuestionario original.');    },  
  });  
}
```

Anexo 76. Función “cargarCuestionarioOriginal”

Para la función de crear nuevo cuestionario se creó la función “crearCuestionario” en ella se asigna el nombre y descripción de forma predeterminada o por el usuario, luego de asignar las preguntas y el signo de la pregunta se procede a mandar el nuevo cuestionario al backend mediante el servicio “createQuiz” de “cuestionarioService”. Posteriormente muestra los mensajes si el cuestionario se ha creado exitosamente o Error al crear cuestionario.

```
Complexity is 8 It's time to do something...
crearCuestionario(): void {
  const nuevoCuestionario = {
    nombre: this.nombre || 'Nuevo Cuestionario',
    descripcion: this.descripcion || 'Sin descripción',
    preguntas: this.preguntas.map(pregunta => ({
      Texto: pregunta.Texto,
      Tipo: pregunta.Tipo,
      variables: pregunta.variables.map((variable: any) => variable.VariableID),
    })),
  };

  // Enviar el nuevo cuestionario al backend
  this.cuestionarioService.createQuiz(nuevoCuestionario).subscribe({
    next: () => {
      alert('Cuestionario creado exitosamente');
      this.router.navigate(['/quiz']);
    },
    error: (err) => {
      console.error('Error al crear el cuestionario:', err);
      alert('Hubo un error al crear el cuestionario.');
```

Anexo 77. Función “crearCuestionario”


```

1 <div class="crear-cuestionario">
2   <form (ngSubmit)="crearCuestionario()">
3     <h3>CREAR UN NUEVO CUESTIONARIO</h3>
4     <label>Nombre</label>
5     <input type="text" [(ngModel)]="nombre" name="nombre" required />
6     <label>Descripción</label>
7     <textarea [(ngModel)]="descripcion" name="descripcion"></textarea>
8     <h3>PREGUNTAS DEL CUESTIONARIO</h3>
9     <div *ngFor="let pregunta of obtenerPreguntasPagina(); let i = index" class="pregunta-row">
10      <div class="pregunta-col">
11        <label>Pregunta {{ (paginaActual - 1) * preguntasPorPagina + (i + 1) }}</label>
12        <input
13          type="text"
14          placeholder="Texto de la pregunta"
15          [(ngModel)]="pregunta.Texto"
16          name="textoPregunta{{ i }}"
17          required
18        />
19      </div>
20      <div class="variable-col">
21        <label>Variables</label>
22        <div class="variable-container">
23          <div *ngFor="let variable of pregunta.variables" class="variable-item">
24            {{ variable.Nombre }}
25          </div>
26        </div>
27      </div>
28      <div class="tipo-col">
29        <label>Signo</label>
30        <select [(ngModel)]="pregunta.Tipo" name="tipoPregunta{{ i }}">
31          <option value="Positiva">Positiva</option>
32          <option value="Negativa">Negativa</option>
33        </select>
34      </div>
35    </div>
36  </div>

```

Anexo 78. Código de la vista del formulario para crear nueva versión del cuestionario.

Modelos realizados

Para el funcionamiento de los servicios se realizó la creación de una serie de modelos que son importantes para el sistema la cual nos proporciona la estructura de los datos obtenidos a partir del backend.

Primero se creó un modelo interfaces el cual contiene tres interfaces cuya función es definir la estructura de los datos, la primera ResponseGet sirve para modelar las respuestas al recibir los datos del backend, la segunda ResponsePPD funciona de la misma forma para los procesos de los datos (en este caso actualizar, crear y eliminar datos) y finalmente la interfaz ErrorExcep se utiliza para modelar errores o excepciones que puedan ocurrir en el sistema.

```
You, el mes pasado | 1 author (You)
1  export interface ResponseGet<T> {
2      Institucion: any;
3      data: T;
4      status: number;
5      message: string;
6  }
7
You, hace 3 meses | 1 author (You)
8  export interface ResponsePPD {
9      status: number;
10     message: string;
11 }
12
You, hace 3 meses | 1 author (You)
13 export interface ErrorExcep {
14     status: number;
15     message: string;
16 }
```

Anexo 79. Modelo Responses

Para el modelo usuarios se añadieron las interfaces para la estructura de los datos del estudiante, maestro y administrador para diferentes funciones, que en este caso son para crear, obtener y actualizar.

```
1  import { Countries, Degrees, Institutions, } from "./catalogs";
2  import { Grupo} from "./gruops";
3
4  You, hace 3 meses | 1 author (You)
5  export interface createStudent{
6      Nombre: string,
7      Primer_apellido: string,
8      Segundo_apellido: string,
9      Matricula: string,
10     Semestre: string,
11     Tipo: string,
12     Correo: string,
13     Password: string,
14     edad: string,
15     genero: string,
16     InstitucionID: Institutions,
17     IDPais: Countries,
18     CarreraID: Degrees
19 }
20
21 You, hace 3 meses | 1 author (You)
22 export interface createTeacher{
23     Nombre: string,
24     Primer_apellido: string,
25     Segundo_apellido: string,
26     Matricula: string,
27     Semestre: null,
28     Tipo: string,
29     Correo: string,
30     Password: string,
31     edad: string,
32     genero: string,
33     InstitucionID: Institutions,
34     IDPais: Countries,
35     CarreraID: null,
36 }
```

Anexo 80. Modelo usuarios parte 1.

You, hace 3 meses | 1 author (You)

```
export interface createAdmi{
  Nombre: string,
  Primer_apellido: string,
  Segundo_apellido: string,
  Matricula: string,
  Semestre: null,
  Tipo: string,
  Correo: string,
  Password: string,
  edad: string,
  genero: string,
  InstitucionID: Institutions,
  IDPais: Countries,
  CarreraID: null,
}
```

You, el mes pasado | 1 author (You)

```
export interface UpdateAdmi {
  Nombre?: string;
  Primer_apellido?: string;
  Segundo_apellido?: string;
  Matricula?: string;
  Semestre?: string | null;
  Tipo?: string;
  Correo?: string;
  Password?: string;
  edad?: string;
  genero?: string;
  InstitucionID?: number;
  IDPais?: number;
  CarreraID?: number | null;
}
```

Anexo 81. Modelo usuarios parte 2.

```

71 export interface getStudents{
72     UsuarioID: number;
73     Nombre: string;
74     Primer_apellido: string;
75     Segundo_apellido: string;
76     Matricula: string;
77     Semestre: string;
78     Tipo: string;
79     Correo: string;
80     Password: string;
81     edad: string;
82     genero: string;
83     InstitucionID: Institutions;
84     IDPais: Countries;
85     CarreraID: Degrees;
86     created_at: string;
87     updated_at: string;
88     grupos: Grupo;
89 }
90
91
92
93 export interface getTeachers{
94     UsuarioID: number,
95     Nombre: string,
96     Primer_apellido: string,
97     Segundo_apellido: string,
98     Matricula: string,
99     Semestre: null,
100     Tipo: string,
101     Correo: string,
102     Password: string,
103     edad: string,
104     genero: string,
105     InstitucionID: Institutions,
106     IDPais: Countries,
107     CarreraID: null,
108 }

```

Anexo 82. Modelo usuarios parte 3.

El modelo catálogos contiene las interfaces para los datos de las instituciones, países, crear países, carreras y generaciones.

```
100, hace 3 meses | 1 author (You)
1  export interface Institutions{
2      InstitucionID: number;
3      Nombre: string;
4      IDPais: number;
5      created_at?: string;
6      updated_at?: string;
7  }
8
9  You, hace 3 meses | 1 author (You)
9  export interface Countries {
10     IDPais: number;
11     created_at?: string;
12     Nombre: string;
13     updated_at?: string;
14 }
15
16 You, el mes pasado | 1 author (You)
16 export interface CreateCountries {
17     IDPais: number;
18     created_at?: string;
19     Nombre: string;
20     updated_at?: string;
21 }
22
```

Anexo 83. Modelo catalogos parte 1.

```
23 export interface Degrees{
24     CarreraID: number;
25     Nombre: string;
26     InstitucionID: number;
27     created_at?: string;
28     updated_at?: string;
29 }
30
31 You, el mes pasado | 2 authors (Fernanda-Chable and one other)
31 export interface Generacion {
32     GeneracionID: number;
33     Nombre: string;
34     CarreraID: number;
35     created_at?: string;
36     updated_at?: string;
37 }
```

Anexo 84. Modelo catálogos parte 2.

Para el modelo cuestionarios se creó para obtener la estructura de las respuestas del backend.

```
Fernanda-Chable, el mes pasado | 1 author (Fernanda-Chable)
1 interface CuestionarioPendiente {
2   CuestionarioID: number;
3   Nombre: string;
4   Descripcion: string;
5   FechaPublicacion: string;
6   FechaCierre: string;
7 }
8
Fernanda-Chable, el mes pasado | 1 author (Fernanda-Chable)
9 interface GrupoConCuestionarios {
10  nombreGrupo: string;
11  cuestionarios: CuestionarioPendiente[];
12 }
13
Fernanda-Chable, el mes pasado | 1 author (Fernanda-Chable)
14 export interface Cuestionario {
15   CuestionarioID: number;
16   GrupoID: number;
17   Nombre: string;
18   Descripcion: string;
19   FechaPublicacion: string;
20   FechaCierre: string;
21   created_at: string;
22   updated_at: string;
23 }
```

Anexo 85. Modelo Cuestionarios.

Para el modelo grupos se crearon las interfaces para obtener las estructuras para los grupos junto con los alumnos.

```
export interface Grupo {
  GrupoID: number;
  Nombre: string;
  codigo_acceso: string;
  semestre: string;
  GeneracionID: Generacion;
  CarreraID: Degrees;
  created_at: string;
  updated_at: string;
  pivot: {
    UsuarioID: number;
    GrupoID: number;
    CarreraID: number;
  };
  carrera: Carrera;
}

You, hace 1 minuto | 2 authors (Fernanda-Chable and one other)

export interface Alumno {
  UsuarioID: number,
  Nombre: string,
  Primer_apellido: string,
  Segundo_apellido: string,
  Matricula: string,
  semestre: string,
  Tipo: string,
  Correo: string,
  Password: string,
  edad: string,
  genero: string,
  InstitucionID: Institutions,
  IDPais: Countries,
  CarreraID: Degrees,
  created_at: string;
  updated_at: string;
  grupos: Grupo[];
}
```

Anexo 86. Modelo grupos.

Servicios

Para la conexión de del backend con el frontend se realizaron una serie de servicios, para el caso del login se realizó el servicio para la autenticación de los resultados, a partir de la APIREST que se creó en el backend.

```
8   export class AuthService {
9     private apiUrl = 'http://127.0.0.1:8000';
10
11     constructor(private http: HttpClient) {}
12
13     Complexity is 4 Everything is cool!
14     login(data: any): Observable<any> {
15       return this.http.post(`${this.apiUrl}/login`, data);
16     }
17
18     logout(): void {
19       localStorage.removeItem('access_token');
20       localStorage.removeItem('user_id');
21     }
22
23     isLoggedIn(): boolean {
24       return !!localStorage.getItem('access_token');
25     }
26
27     getToken(): string | null {
28       return localStorage.getItem('access_token');
29     }
30     setUserId(id: number): void {
31       localStorage.setItem('user_id', id.toString());
32     }
33
34     Complexity is 3 Everything is cool!
35     getUserId(): number | null {
36       const userId = localStorage.getItem('user_id');
37       return userId ? parseInt(userId) : null;
38     }
39   }
```

Anexo 87. Servicio Autenticación

Para el manejo de los usuarios se creó el servicio el cual tiene las funciones para registrar a los usuarios, obtener la lista de los usuarios, crearlos y de igual manera eliminarlos.

```
export class UserService {
  private URL = 'http://127.0.0.1:8000/usuarios';

  constructor(private http: HttpClient) { }

  registerStudent(user: createStudent): Observable<ResponsePPD> {
    return this.http.post<ResponsePPD>(this.URL + '/register', user);
  }

  getStudents(): Observable<ResponseGet<getStudents[]>>{
    return this.http.get<ResponseGet<getStudents[]>>(this.URL + '/alumnos');
  }

  registerTeacher(user: createTeacher): Observable<ResponsePPD> {
    return this.http.post<ResponsePPD>(this.URL + '/register', user);
  }

  getTeachers(): Observable<ResponseGet<getTeachers[]>>{
    return this.http.get<ResponseGet<getTeachers[]>>(this.URL + '/maestros');
  }

  // Método para obtener datos de un maestro específico por su ID
  getTeacher(id: number): Observable<ResponseGet<getTeachers>>{
    return this.http.get<ResponseGet<getTeachers>>(`${this.URL}/maestro/${id}`);
  }
}
```

Anexo 88. “userService” parte 1.

```
39   registerAdmin(user: createAdmi): Observable<ResponsePPD> {
40     return this.http.post<ResponsePPD>(this.URL + '/register', user);
41   }
42
43   getAdmins(): Observable<ResponseGet<getAdmins[]>>{
44     return this.http.get<ResponseGet<getAdmins[]>>(this.URL + '/administradores');
45   }
46
47   // Método para obtener datos de un alumno específico por su ID
48   getStudent(id: number): Observable<ResponseGet<getStudents>> {
49     return this.http.get<ResponseGet<getStudents>>(`${this.URL}/alumnos/${id}`);
50   }
51
52   getAdministrator(id: number): Observable<ResponseGet<getAdmins>> {
53     return this.http.get<ResponseGet<getAdmins>>(`${this.URL}/administrador/${id}`);
54   }
55
56   //Metodo para obtener alumnos por institucionID
57   getStudentsInst(InstitutionID: string): Observable<ResponseGet<getStudents[]>>{
58     return this.http.get<ResponseGet<getStudents[]>>(`${this.URL}/alumnos/institucion/${InstitutionID}`)
59   }
60 }
```

Anexo 89. “userService” parte 2.

```

61  getTeachersInst(InstitutionID: string): Observable<ResponseGet<getTeachers[]>>{
62      return this.http.get<ResponseGet<getTeachers[]>>(`${this.URL}/maestros/institucion/${InstitutionID}`);
63  }
64
65  // Método para actualizar el perfil de administrador
66  updateAdmin(UsuarioID: string, data: UpdateAdmi): Observable<ResponsePPD> {
67      return this.http.put<ResponsePPD>(`${this.URL}/actualizar/${UsuarioID}`, data);
68  }
69
70  deleteStudent(UsuarioID: number): Observable<ResponsePPD> {
71      return this.http.delete<ResponsePPD>(`${this.URL}/eliminar/${UsuarioID}`);
72  }
73
74  deleteTeacher(UsuarioID: number): Observable<ResponsePPD> {
75      return this.http.delete<ResponsePPD>(`${this.URL}/eliminar/${UsuarioID}`);
76  }
77  }

```

Anexo 90. “userService” parte 3.

Para obtener todos los catálogos se realizó el servicio catalogs este contiene los servicios para obtener las instituciones, países, generaciones y carreras, de igual manera para crear un nuevo país.

```

10  export class CatalogsService {
11      private URL = "http://127.0.0.1:8000/catalogs";
12
13      constructor(private http: HttpClient) {}
14
15      getInstitutionsId(IDPais: string): Observable<ResponseGet<Institutions[]>> {
16          return this.http.get<ResponseGet<Institutions[]>>(`${this.URL}/instituciones?IDPais=${IDPais}`);
17      }
18
19      getInstitutions(): Observable<ResponseGet<Institutions[]>>{
20          return this.http.get<ResponseGet<Institutions[]>>(this.URL + "/ainstituciones");
21      }
22
23      getInstitution(InstitutionID: string): Observable<ResponseGet<Institutions[]>> {
24          return this.http.get<ResponseGet<Institutions[]>>(`${this.URL}/idinstituciones/${InstitutionID}`);
25      }
26
27      getCountries(): Observable<ResponseGet<Countries[]>>{
28          return this.http.get<ResponseGet<Countries[]>>(this.URL + "/paises")
29      }
30  }

```

Anexo 91. “catalogsService” parte 1.

```

31  getDegreesId(InstitutionID: string): Observable<ResponseGet<Degrees[]>>{
32      return this.http.get<ResponseGet<Degrees[]>>(`${this.URL}/carreras?InstitutionID=${InstitutionID}`);
33  }
34
35  getDegrees(): Observable<ResponseGet<Degrees[]>>{
36      return this.http.get<ResponseGet<Degrees[]>>(this.URL + "/acarreras")
37  }
38
39  getGeneraciones(): Observable<ResponseGet<Generacion[]>> {
40      return this.http.get<ResponseGet<Generacion[]>>(`${this.URL}/generaciones`);
41  }
42
43  createCountry(country: CreateCountries): Observable<ResponsePPD> {
44      return this.http.post<ResponsePPD>(this.URL + '/agregarpais', country);
45  }
46
47  }

```

Anexo 92. “catalogsService” parte 2.

Para obtener el alumno por grupo se creó el siguiente servicio.

```

9  export class AlumnoGruposService {
10     private apiUrl = 'http://127.0.0.1:8000/alumno';
11
12     constructor(private http: HttpClient) {}
13
14     Complexity is 3 Everything is cool!
15     getAlumnoGrupos(usuarioID: number): Observable<{ data: Grupo[], status: number, message: string }> {
16         const url = `${this.apiUrl}/${usuarioID}/grupos`;
17         const headers = new HttpHeaders({
18             'Authorization': `Bearer ${localStorage.getItem('access_token')}` || ''
19         });
20         return this.http.get<{ data: Grupo[], status: number, message: string }>(url, { headers });
21     }
22 }

```

Anexo 93. "AlumnosGruposService".

Para el servicio de los cuestionarios se crearon las funciones para obtener los cuestionarios, preguntas, versiones de los cuestionarios, publicar cuestionarios, etc.

```

8  export class CuestionarioService {
9     private apiUrl = 'http://127.0.0.1:8000';
10
11     constructor(private http: HttpClient) {}
12
13     Complexity is 4 Everything is cool!
14     getPreguntasRespuestas(cuestionarioID: number): Observable<any> {
15         return this.http.get<any>(`${this.apiUrl}/cuestionarios/${cuestionarioID}/preguntas`);
16     }
17     Complexity is 5 Everything is cool!
18     enviarRespuestas(usuarioID: number, respuestas: any): Observable<any> {
19         const payload = { respuestas };
20         return this.http.post<any>(`${this.apiUrl}/respuestas/${usuarioID}/guardarRespuestas`, payload, {
21             headers: { 'Content-Type': 'application/json' }
22         });
23     }
24
25     // Obtener todas Las preguntas
26     Complexity is 3 Everything is cool!
27     getAllQuestions(): Observable<any> {
28         return this.http.get(`${this.apiUrl}/preguntas`);
29     }
30 }

```

Anexo 94. "cuestionarioService" parte 1.

```

28 // Crear un nuevo cuestionario
29 Complexity is 4 Everything is cool!
30 createQuiz(data: any): Observable<any> {
31     return this.http.post(`${this.apiUrl}/cuestionarios`, data);
32 }
33 Complexity is 4 Everything is cool!
34 getQuizVersions(): Observable<any> {
35     return this.http.get<any>(`${this.apiUrl}/cuestionarios`);
36 }
37 Complexity is 5 Everything is cool!
38 publishQuiz(cuestionarioID: number, payload: any): Observable<any> {
39     console.log('Payload enviado:', payload); // Para verificar Los datos enviados
40     return this.http.post<any>(`${this.apiUrl}/cuestionarios/${cuestionarioID}/publicar`, payload);
41 }
42 // Método para obtener Los cuestionarios aplicados por el usuario (maestro)
43 Complexity is 4 Everything is cool!
44 obtenerCuestionariosPorUsuario(usuarioID: number): Observable<any> {
45     return this.http.get<any>(`${this.apiUrl}/cuestionarios/usuario/${usuarioID}`);
46 }
47 // Método para obtener todos Los cuestionarios
48 Complexity is 3 Everything is cool!
49 getAllCuestionarios(): Observable<any> {
50     return this.http.get(`${this.apiUrl}/cuestionarios/versions`);
51 }
52 }

```

Anexo 95. "cuestionarioService" parte 2.

```

50 Complexity is 4 Everything is cool!
51 getAllGroups(): Observable<any> {
52     return this.http.get<any>(`${this.apiUrl}/grupos`);
53 }
54 Complexity is 4 Everything is cool!
55 saveNewQuizVersion(quizId: number, data: any): Observable<any> {
56     return this.http.post(`${this.apiUrl}/cuestionarios/${quizId}/duplicar-editar`, data);
57 }
58 Complexity is 5 Everything is cool!
59 duplicarCuestionario(cuestionarioID: number, payload: any): Observable<any> {
60     return this.http.post(`${this.apiUrl}/cuestionarios/${cuestionarioID}/duplicar`, payload);
61 }
62 }
63

```

Anexo 96. "cuestionarioService" parte 3.

Servicio para obtener puntajes de las variables del cuestionario.

```

8 export class PuntajeService {
9     private apiUrl = 'http://127.0.0.1:8000/puntajes';
10
11     constructor(private http: HttpClient) { }
12
13     Complexity is 4 Everything is cool!
14     obtenerPuntajesVariables(usuarioID: number): Observable<any> {
15         return this.http.get<any>(`${this.apiUrl}/${usuarioID}`);
16     }
17 }
18

```

Anexo 97. "PuntajeService".

Servicio para obtener los puntajes por el cuestionario.

```

8 export class ResultsService {
9     private apiUrl = 'http://127.0.0.1:8000';
10
11     constructor(private http: HttpClient) { }
12
13     Complexity is 3 Everything is cool!
14     obtenerPuntajesPorCuestionario(usuarioID: number, cuestionarioID: number): Observable<any> {
15         return this.http.get(`${this.apiUrl}/puntajesporCuestionario/${usuarioID}/${cuestionarioID}`);
16     }
17 }
18

```

Anexo 98. "PuntajeService".

Servicio para obtener los cuestionarios resueltos.

```
8 export class SolvedQuizzesService {
9   private apiUrl = 'http://127.0.0.1:8000';
10
11   constructor(private http: HttpClient) { }
12
13   // Marcar un cuestionario como resuelto
14   // Complexity is 4 Everything is cool!
15   marcarCuestionarioResuelto(data: any): Observable<any> {
16     return this.http.post(`${this.apiUrl}/cuestionarios/marcar-resuelto`, data);
17   }
18
19   // Obtener cuestionarios resueltos por usuario y grupo
20   // Complexity is 3 Everything is cool!
21   obtenerCuestionariosResueltos(usuarioId: number, grupoId: number): Observable<any> {
22     return this.http.get(`${this.apiUrl}/cuestionarios/resueltos/${usuarioId}/${grupoId}`);
23   }
24
25   // Complexity is 4 Everything is cool!
26   getCuestionariosResueltosPorAlumno(alumnoId: number): Observable<any> {
27     return this.http.get<any>(`${this.apiUrl}/cuestionarios-resueltos/alumno/${alumnoId}`);
28   }
29
30   // Complexity is 3 Everything is cool!
31   obtenerCuestionariosPendientes(usuarioId: number, grupoId: number): Observable<any> {
32     return this.http.get(`${this.apiUrl}/cuestionarios/pendientes/${usuarioId}/${grupoId}`);
33   }
34 }
```

Anexo 99. "SolvedQuizzesService".

Servicio para obtener los grupos del maestro.

```
11 export class TeacherGroupsService {
12   private apiUrl = 'http://127.0.0.1:8000';
13
14   constructor(private http: HttpClient) { }
15
16   // Complexity is 4 Everything is cool!
17   getGruposDeMaestro(usuarioId: number): Observable<any> {
18     return this.http.get<any>(`${this.apiUrl}/maestro/${usuarioId}/grupos`);
19   }
20
21   getContadorGruposDeMaestro(usuarioId: number): Observable<Grupo[]> {
22     return this.http.get<Grupo[]>(`${this.apiUrl}/maestro/${usuarioId}/grupos`);
23   }
24
25   // Metodo para obtener los cuestionarios aplicados por grupo
26   getAppliedQuizzesByGroup(grupoId: number): Observable<{data: Cuestionario[]}> {
27     const url = `${this.apiUrl}/cuestionarios-aplicados/grupo/${grupoId}`;
28     return this.http.get<{data: Cuestionario[]}>(url);
29   }
30
31   // Nuevo método para obtener la información de un grupo específico
32   getGroupById(grupoId: number): Observable<{ data: Grupo }> {
33     const url = `${this.apiUrl}/grupos/${grupoId}`;
34     return this.http.get<{ data: Grupo }>(url);
35   }
36 }
```

Anexo 100. "TeacherGroupService" parte 1.

```

36 // Método para crear un nuevo grupo
    Complexity is 5 Everything is cool!
37 createGroup(grupoData: any): Observable<any> {
38     const url = `${this.apiUrl}/crear-grupos`;
39     return this.http.post<any>(url, grupoData);
40 }
41 // Método para obtener los integrantes de un grupo
    Complexity is 4 Everything is cool!
42 getIntegrantes(grupoId: number): Observable<any> {
43     return this.http.get<any>(`${this.apiUrl}/grupos/${grupoId}/integrantes`);
44 }
45
46 // Método para buscar alumnos
    Complexity is 3 Everything is cool!
47 buscarAlumno(nombre: string): Observable<any> {
48     const params = new HttpParams().set('nombre', nombre);
49     return this.http.get(`${this.apiUrl}/alumno/buscar`, { params });
50 }
51
52 // Método para añadir alumno al grupo
    Complexity is 3 Everything is cool!
53 agregarAlumnoAGrupo(grupoId: number, alumnoId: number): Observable<any> {
54     return this.http.post(`${this.apiUrl}/grupos/${grupoId}/alumnos/agregar`, { UsuarioID: alumnoId });
55 }
56 }

```

Anexo 101. "TeacherGroupService" parte 2.

Backend

Para el backend se realizaron ciertas funciones para el sistema, ya que es la base central para el manejo de los datos. Por lo que se realizó lo siguiente:

Migraciones

Las migraciones se realizaron con el fin de crear las tablas de países, instituciones, carreras, usuarios, variables, expresiones, categorías, cuestionarios y sobre todo definir las relaciones entre las tablas.

Tabla de Países

Se creó la tabla países para almacenar información básica de diferentes naciones. Incluye una clave primaria IDPais, un campo para el nombre (Nombre) y marcas de tiempo (created_at, updated_at). Su diseño facilita la administración eficiente y escalable de datos geográficos.

```
ackendCEP > database > migrations > 2024_07_12_223304_create_paises_table.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreatePaisesTable extends Migration
8  {
9      0 references|0 overrides
10     public function up(): void
11     {
12         Schema::create('paises', function (Blueprint $table): void {
13             $table->id(column: 'IDPais');
14             $table->string(column: 'Nombre');
15             $table->timestamps();
16         });
17     }
18     0 references|0 overrides
19     public function down(): void
20     {
21         Schema::dropIfExists('paises');
22     }
23 }
```

Anexo 102. Migración paises.

Tabla de Instituciones

La tabla instituciones se diseñó para registrar datos de instituciones educativas. Se conecta con la tabla de países mediante la clave foránea IDPais, garantizando integridad referencial y permitiendo la eliminación en cascada. Contiene un campo Nombre para identificar las instituciones.


```

BackendCEP > database > migrations > 2024_07_16_181714_create_instituciones_table.php > PHP IntelliSense > CreateInstitucionesTable
1
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateInstitucionesTable extends Migration
8 {
9     // references: 0 implementations
10    public function up(): void
11    {
12        Schema::create('instituciones', function (Blueprint $table) {
13            $table->id(column: 'InstitucionID');
14            $table->string(column: 'Nombre');
15            $table->unsignedInteger(column: 'IDPais');
16            $table->timestamps();
17
18            $table->foreign(column: 'IDPais')->references(column: 'IDPais')->on(table: 'paises')
19                ->onDelete(action: 'cascade');
20        });
21    }
22
23    // references: 0 overrides
24    public function down(): void
25    {
26        Schema::dropIfExists('instituciones');
27    }
28 }

```

Anexo 103. Migración Instituciones.

Tabla de Carreras

Esta tabla maneja información sobre carreras académicas, con un campo Nombre y una relación con la tabla instituciones a través de la clave foránea InstitucionID. La integridad se asegura mediante restricciones referenciales.

```

BackendCEP > database > migrations > 2024_07_17_212955_create_carreras_table.php > ...
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateCarrerasTable extends Migration
8 {
9     // references: 0 overrides
10    public function up(): void
11    {
12        Schema::create('carreras', function (Blueprint $table) {
13            $table->id(column: 'CarreraID'); // BIGINT UNSIGNED
14            $table->string(column: 'Nombre');
15            $table->foreignId(column: 'InstitucionID') // BIGINT UNSIGNED
16                ->constrained(table: 'instituciones', column: 'InstitucionID')
17                ->onDelete(action: 'cascade');
18            $table->timestamps();
19        });
20    }
21
22    // references: 0 overrides
23    public function down(): void
24    {
25        Schema::dropIfExists('carreras');
26    }
27 }

```

Anexo 104. Migración carreras.

Tabla de Generaciones

La tabla generaciones organiza información sobre generaciones académicas. Cada generación está asociada a una carrera mediante la clave foránea CarreraID. La eliminación en cascada asegura consistencia si se elimina una carrera.

```

BackendCEP > database > migrations > 2024_07_27_213013_create_generaciones_table.php > PHP IntelliSense
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateGeneracionesTable extends Migration
8  {
9      0 references | 0 overrides
10     public function up(): void
11     {
12         Schema::create('generaciones', function (Blueprint $table): void {
13             $table->id(column: 'GeneracionID'); // BIGINT UNSIGNED
14             $table->string(column: 'Nombre');
15             $table->foreignId(column: 'CarreraID') // BIGINT UNSIGNED
16                 ->constrained(table: 'carreras', column: 'CarreraID')
17                 ->onDelete(action: 'cascade');
18             $table->timestamps();
19         });
20     }
21
22     0 references | 0 overrides
23     public function down(): void
24     {
25         Schema::dropIfExists('generaciones');
26     }
27 }

```

Anexo 105. Migración Generaciones.

Tabla de Grupos

Los grupos académicos se registran en esta tabla, con un campo Nombre y una relación con generaciones a través de la clave foránea GeneracionID. Esto mantiene coherencia en la estructura académica.

```

BackendCEP > database > migrations > 2024_07_27_213026_create_grupos_table.php > PHP IntelliSense > C#
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateGruposTable extends Migration
8  {
9      0 references | 0 overrides
10     public function up(): void
11     {
12         Schema::create('grupos', function (Blueprint $table): void {
13             $table->id(column: 'GrupoID'); // BIGINT UNSIGNED
14             $table->string(column: 'Nombre');
15             $table->foreignId(column: 'GeneracionID') // BIGINT UNSIGNED
16                 ->constrained(table: 'generaciones', column: 'GeneracionID')
17                 ->onDelete(action: 'cascade');
18             $table->timestamps();
19         });
20     }
21
22     0 references | 0 overrides
23     public function down(): void
24     {
25         Schema::dropIfExists('grupos');
26     }
27 }

```

Anexo 106. Migración Grupos.

Tabla de Cuestionarios

Esta tabla gestiona evaluaciones asignadas a grupos. Incluye campos como Nombre, descripción, y fechas de publicación y cierre. Está vinculada a grupos mediante una clave foránea, permitiendo la eliminación en cascada.

```
BackendCEP > database > migrations > 2024_11_22_065212_create_cuestionarios_table.php > PHP IntelliSense
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  0 references | 0 implementations
8  class CreateCuestionariosTable extends Migration
9  {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create('cuestionarios', function (Blueprint $table): void {
14             $table->id(column: 'CuestionarioID');
15             $table->foreignId(column: 'GrupoID')
16                 ->nullable()
17                 ->constrained(table: 'grupos', column: 'GrupoID')
18                 ->onDelete(action: 'cascade');
19             $table->string(column: 'Nombre');
20             $table->text(column: 'Descripcion')->nullable();
21             $table->timestamps();
22         });
23     }
24
25     0 references | 0 overrides
26     public function down(): void
27     {
28         Schema::dropIfExists('cuestionarios');
29     }
30 }
```

Anexo 107. Migración Cuestionarios.

Tabla de Preguntas

Se utiliza para organizar preguntas relacionadas con un cuestionario específico. Cada registro incluye el texto de la pregunta y su tipo. La relación con cuestionarios garantiza integridad y permite la eliminación en cascada.

```
BackendCEP > database > migrations > 2024_07_27_213057_create_preguntas_table.php > PHP IntelliSense
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  0 references | 0 implementations
8  class CreatePreguntasTable extends Migration
9  {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create('preguntas', function (Blueprint $table): void {
14             $table->id(column: 'PreguntaID');
15             $table->text(column: 'Texto');
16             $table->enum(column: 'Tipo', allowed: ['Positiva', 'Negativa']);
17             $table->foreignId(column: 'CuestionarioID') // BIGINT UNSIGNED
18                 ->constrained(table: 'cuestionarios', column: 'CuestionarioID')
19                 ->onDelete(action: 'cascade');
20             $table->timestamps();
21         });
22     }
23
24     0 references | 0 overrides
25     public function down(): void
26     {
27         Schema::dropIfExists('preguntas');
28     }
29 }
```

Anexo 108. Migración Preguntas.

Tabla de Respuestas

Almacena opciones de respuesta, junto con su puntaje numérico. Se relaciona con preguntas a través de una clave foránea, lo que asegura consistencia en la evaluación.

```
BackendCEP > database > migrations > 2024_07_30_213831_create_respuestas_table.php > PHP IntelliSense
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  0 references | 0 implementations
8  class CreateRespuestasTable extends Migration
9  {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create('respuestas', function (Blueprint $table) { void {
14             $table->id(column: 'RespuestaID');
15             $table->foreignId(column: 'PreguntaID')
16                 ->constrained(table: 'preguntas', column: 'PreguntaID')
17                 ->onDelete(action: 'cascade');
18             $table->text(column: 'Texto');
19             $table->tinyInteger(column: 'Valor');
20             $table->timestamps();
21         });
22     }
23
24     0 references | 0 overrides
25     public function down(): void
26     {
27         Schema::dropIfExists('respuestas');
28     }
29 }
```

Anexo 109. Migración Respuestas.

Tabla de Usuarios

Centraliza la información de los usuarios, como datos personales, académicos y de acceso. Incluye relaciones con tablas como países, instituciones y carreras, permitiendo la gestión de roles (alumnos, maestros, administradores) dentro del sistema.

```
BackendCEP > database > migrations > 2024_08_19_013443_create_usuarios_table.php > PHP > CreateUsuariosTable > up()
7  0 references | 0 implementations
8  class CreateUsuariosTable extends Migration
9  {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create('usuarios', function (Blueprint $table) { void {
14             $table->id(column: 'UsuarioID'); // BIGINT UNSIGNED
15             $table->string(column: 'Nombre');
16             $table->string(column: 'Primer_apellido');
17             $table->string(column: 'Segundo_apellido')->nullable();
18             $table->string(column: 'Matricula')->unique();
19             $table->string(column: 'Semestre');
20             $table->enum(column: 'Tipo', allowed: ['Alumno', 'Maestro', 'Admin']);
21             $table->string(column: 'Correo')->unique();
22             $table->string(column: 'Password');
23             $table->integer(column: 'edad')->nullable();
24             $table->enum(column: 'genero', allowed: ['Masculino', 'Femenino', 'Otro'])->nullable();
25             $table->unsignedBigInteger(column: 'InstitucionID')->nullable();
26             $table->unsignedBigInteger(column: 'IDPaís')->nullable();
27             $table->unsignedBigInteger(column: 'CarreraID')->nullable();
28             $table->timestamps();
29
30             // Definición de claves foráneas
31             $table->foreign(column: 'InstitucionID')->references(column: 'InstitucionID')->on(table: 'instituciones')->onDelete(action: 'set null');
32             $table->foreign(column: 'IDPaís')->references(column: 'IDPaís')->on(table: 'países')->onDelete(action: 'set null');
33             $table->foreign(column: 'CarreraID')->references(column: 'CarreraID')->on(table: 'carreras')->onDelete(action: 'set null');
34         });
35     }
36
37     0 references | 0 overrides
38     public function down(): void
39     {
40     }
```

Anexo 110. Migración Usuarios.

Relación Cuestionario-Grupo

La tabla intermedia `cuestionario_grupo` conecta cuestionarios con grupos de estudiantes. Cada registro establece una asignación específica, respaldada por claves foráneas.

```
BackendCEP > database > migrations > 2024_11_22_065337_create_cuestionario_grupo_table.php > PHP > C
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  0 references | 0 implementations
8  class CreateCuestionarioGrupoTable extends Migration
9  {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create('cuestionario_grupo', function (Blueprint $table): void {
14             $table->id(column: 'CuestionarioGrupoID');
15             $table->foreignId(column: 'CuestionarioID')
16                 ->constrained(table: 'cuestionarios', column: 'CuestionarioID')
17                 ->onDelete(action: 'cascade');
18             $table->foreignId(column: 'GrupoID')
19                 ->constrained(table: 'grupos', column: 'GrupoID')
20                 ->onDelete(action: 'cascade');
21             $table->date(column: 'FechaPublicacion');
22             $table->date(column: 'FechaCierre');
23             $table->timestamps();
24         });
25     }
26     0 references | 0 overrides
27     public function down(): void
28     {
29         Schema::dropIfExists('cuestionario_grupo');
30     }
31 }
```

Anexo 111. Migración `cuestionario_grupo`.

Relación Usuario-Grupo

La tabla `grupo_usuario` define la participación de usuarios en grupos. Permite asociaciones múltiples entre usuarios y grupos, asegurando integridad mediante claves foráneas.

```

BackendCEP > database > migrations > 2024_08_30_214219_create_grupo_usuario_table.php > PHP IntelliSense > CreateGrupoUsuarioTable >
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  0 references | 0 implementations
8  class CreateGrupoUsuarioTable extends Migration
9  {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create('grupo_usuario', function (Blueprint $table): void {
14             $table->id();
15             $table->foreignId(column: 'GrupoID')->constrained(table: 'grupos', column: 'GrupoID');
16             $table->foreignId(column: 'UsuarioID')->constrained(table: 'usuarios', column: 'UsuarioID');
17             $table->timestamps();
18         });
19     }
20
21     0 references | 0 overrides
22     public function down(): void
23     {
24         Schema::dropIfExists('grupo_usuario');
25     }
26 }

```

Anexo 112. Migración usuario_grupo

Tabla de Categorías

Organiza las preguntas de los cuestionarios en diferentes categorías. Contiene un identificador y un campo Nombre para la clasificación, facilitando la búsqueda y organización.

```

BackendCEP > database > migrations > 2024_08_30_214219_create_grupo_usuario_table.php > PHP IntelliSense > CreateGrupoUsuarioTable >
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  0 references | 0 implementations
8  class CreateGrupoUsuarioTable extends Migration
9  {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create('grupo_usuario', function (Blueprint $table): void {
14             $table->id();
15             $table->foreignId(column: 'GrupoID')->constrained(table: 'grupos', column: 'GrupoID');
16             $table->foreignId(column: 'UsuarioID')->constrained(table: 'usuarios', column: 'UsuarioID');
17             $table->timestamps();
18         });
19     }
20
21     0 references | 0 overrides
22     public function down(): void
23     {
24         Schema::dropIfExists('grupo_usuario');
25     }
26 }

```

Anexo 113. Migración Categorías.

Tabla de Variables

Relaciona las variables con las categorías de preguntas. Incluye un identificador único, un campo Nombre, y una relación con categorías, permitiendo la eliminación en cascada.

```
BackendCEP > database > migrations > 2024_09_12_213047_create_variables_table.php > PHP > CreateVar
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateVariablesTable extends Migration
8  {
9      public function up(): void
10     {
11         Schema::create('variables', function (Blueprint $table): void {
12             $table->id(column: 'VariableID');
13             $table->string(column: 'Nombre');
14             $table->foreignId(column: 'CategoriaID')
15                 ->constrained(table: 'categorias', column: 'CategoriaID')
16                 ->onDelete(action: 'cascade');
17             $table->timestamps();
18         });
19     }
20
21     public function down(): void
22     {
23         Schema::dropIfExists('variables');
24     }
25 }
```

Anexo 114. Migración Variables.

Relación Pregunta-Variable

La tabla intermedia pregunta_variable conecta preguntas con variables mediante una relación de muchos a muchos. Esto asegura flexibilidad en la asignación de variables a preguntas.

```
BackendCEP > database > migrations > 2024_09_13_223633_create_pregunta_variable_table.php > PHP > Cre
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreatePreguntaVariableTable extends Migration
8  {
9      public function up(): void
10     {
11         Schema::create('pregunta_variable', function (Blueprint $table): void {
12             $table->id(column: 'PreguntaVariableID');
13             $table->foreignId(column: 'PreguntaID')
14                 ->constrained(table: 'preguntas', column: 'PreguntaID')
15                 ->onDelete(action: 'cascade');
16             $table->foreignId(column: 'VariableID')
17                 ->constrained(table: 'variables', column: 'VariableID')
18                 ->onDelete(action: 'cascade');
19             $table->timestamps();
20         });
21     }
22
23     public function down(): void
24     {
25         Schema::dropIfExists('pregunta_variable');
26     }
27 }
```

Anexo 115. Migración pregunta_Variable.

Tabla de Respuestas Enviadas

Registra las respuestas de los usuarios a los cuestionarios. Se relaciona con usuarios, cuestionarios y preguntas, asegurando integridad y seguimiento temporal.

```
BackendCEP > database > migrations > 2024_09_14_230523_create_respuestas_enviadas_table.php > PHP > Cre
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
0 references | 0 implementations
7 class CreateRespuestasEnviadasTable extends Migration
8 {
9
0 references | 0 overrides
10 public function up(): void
11 {
12     Schema::create('respuestas_enviadas', function (Blueprint $table): void {
13         $table->id(column: 'RespuestaEnviadaID');
14         $table->foreignId(column: 'CuestionarioID')
15             ->constrained(table: 'cuestionarios', column: 'CuestionarioID')
16             ->onDelete(action: 'cascade');
17         $table->foreignId(column: 'PreguntaID')
18             ->constrained(table: 'preguntas', column: 'PreguntaID')
19             ->onDelete(action: 'cascade');
20         $table->integer(column: 'respuesta_valor');
21         $table->foreignId(column: 'UsuarioID')->constrained(table: 'usuarios',
22             column: 'UsuarioID')->onDelete(action: 'cascade');
23         $table->timestamps();
24     });
25 }
26
0 references | 0 overrides
27 public function down(): void
28 {
29     Schema::dropIfExists('respuestas_enviadas');
30 }
31 };
```

Anexo 116. Migración respuestas_enviadas.

Tabla de Expresiones

Relación entre expresiones y variables. Incluye un identificador único y un campo descriptivo para cada expresión, con soporte para eliminación en cascada.

```
BackendCEP > database > migrations > 2024_09_15_244950_create_expresiones_table.php > PHP > Cre
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
0 references | 0 implementations
7 class CreateExpresionesTable extends Migration
8 {
9
0 references | 0 overrides
10 public function up(): void
11 {
12     Schema::create('expresiones', function (Blueprint $table): void {
13         $table->id(column: 'ExpresionesID');
14         $table->string(column: 'Nombre');
15         $table->foreignId(column: 'VariableID')
16             ->constrained(table: 'variables', column: 'VariableID')
17             ->onDelete(action: 'cascade');
18         $table->timestamps();
19     });
20 }
21
0 references | 0 overrides
22 public function down(): void
23 {
24     Schema::dropIfExists('expresiones');
25 }
26 };
```


Tabla de Cuestionarios Resueltos

Registra los cuestionarios completados por usuarios, indicando fecha de resolución y estado. Establece relaciones con usuarios, grupos y cuestionarios.

```
BackendCEP > database > migrations > 2024_09_24_224910_create_cuestionario_resuelto_table.php > PHP > Crea
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateCuestionarioResueltoTable extends Migration
8 {
9     public function up(): void
10     {
11         Schema::create('cuestionario_resuelto', function (Blueprint $table): void {
12             $table->id(column: 'CuestionarioResueltoID');
13             $table->foreignId(column: 'UsuarioID')
14                 ->constrained(table: 'usuarios', column: 'UsuarioID')
15                 ->onDelete(action: 'cascade');
16             $table->foreignId(column: 'CuestionarioID')
17                 ->constrained(table: 'cuestionarios', column: 'CuestionarioID')
18                 ->onDelete(action: 'cascade');
19             $table->foreignId(column: 'GrupoID')
20                 ->constrained(table: 'grupos', column: 'GrupoID')
21                 ->onDelete(action: 'cascade');
22             $table->timestamp(column: 'FechaResuelto');
23             $table->boolean(column: 'Resuelto')->default(value: false);
24             $table->timestamps();
25         });
26     }
27
28     public function down(): void
29     {
30         Schema::dropIfExists('cuestionario_resuelto');
31     }
32 }
```

Relación Cuestionario-Pregunta

Conecta cuestionarios con un conjunto de preguntas específicas, permitiendo eliminar preguntas o cuestionarios sin comprometer la integridad de la base de datos.

```

BackendCEP > database > migrations > 2024_10_30_025301_create_cuestionario_pregunta_table.php > PHP > Cre
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  0 references | 0 implementations
8  class CreateCuestionarioPreguntaTable extends Migration
9  {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create('cuestionario_pregunta', function (Blueprint $table): void {
14             $table->id(column: 'CuestionarioPreguntaID');
15             $table->foreignId(column: 'CuestionarioID')
16                 ->constrained(table: 'cuestionarios', column: 'CuestionarioID')
17                 ->onDelete(action: 'cascade');
18             $table->foreignId(column: 'PreguntaID')
19                 ->constrained(table: 'preguntas', column: 'PreguntaID')
20                 ->onDelete(action: 'cascade');
21             $table->timestamps();
22         });
23     }
24
25     0 references | 0 overrides
26     public function down(): void
27     {
28         Schema::dropIfExists('cuestionario_pregunta');
29     }
30 }

```

Anexo 119. Migración cuestionario_pregunta.

Relación Categoría-Expresión

La tabla categorias_expresion une categorías con expresiones mediante una relación muchos a muchos, con claves foráneas para asegurar consistencia.

```

BackendCEP > database > migrations > 2024_11_20_001757_create_categorias_expresion_table.php > PHP > Cre
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  0 references | 0 implementations
8  class CreateCategoriasExpresionTable extends Migration
9  {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create('categorias_expresion', function (Blueprint $table): void {
14             $table->id(column: 'CategoriaExpresionID');
15             $table->foreignId(column: 'CategoriaID')
16                 ->constrained(table: 'categorias', column: 'CategoriaID')
17                 ->onDelete(action: 'cascade');
18             $table->foreignId(column: 'ExpresionesID')
19                 ->constrained(table: 'expresiones', column: 'ExpresionesID')
20                 ->onDelete(action: 'cascade');
21             $table->timestamps();
22         });
23     }
24
25     0 references | 0 overrides
26     public function down(): void
27     {
28         Schema::dropIfExists('categorias_expresion');
29     }
30 }

```

Anexo 120. Migración categoria-expresión.

Tabla de Puntajes por Variable

Almacena los puntajes obtenidos por los usuarios en cada variable. Incluye relaciones con usuarios, grupos, cuestionarios y variables para mantener la referencialidad.

```

BackendCEP > database > migrations > 2024_11_20_004035_create_puntajes_variables_table.php > PHP > CreatePuntajesVariablesTable
5 use Illuminate\Support\Facades\Schema;
6
7 0 references | 0 implementations
8 class CreatePuntajesVariablesTable extends Migration
9 {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create('puntajes_por_variable', function (Blueprint $table): void {
14             $table->id(column: 'PuntajeID');
15             $table->unsignedBigInteger(column: 'UsuarioID');
16             $table->unsignedBigInteger(column: 'GrupoID');
17             $table->unsignedBigInteger(column: 'CuestionarioID');
18             $table->unsignedBigInteger(column: 'VariableID');
19             $table->integer(column: 'puntaje')->nullable();
20             $table->timestamps();
21
22             // Claves foráneas
23             $table->foreign(columns: 'UsuarioID')->references(columns: 'UsuarioID')
24             ->on(table: 'usuarios')->onDelete(action: 'cascade');
25             $table->foreign(columns: 'GrupoID')->references(columns: 'GrupoID')
26             ->on(table: 'grupos')->onDelete(action: 'cascade');
27             $table->foreign(columns: 'CuestionarioID')->references
28             (columns: 'CuestionarioID')->on(table: 'cuestionarios')->onDelete(action: 'cascade');
29             $table->foreign(columns: 'VariableID')->references(columns: 'VariableID')
30             ->on(table: 'variables')->onDelete(action: 'cascade');
31         });
32     }
33
34     0 references | 0 overrides
35     public function down(): void
36     {
37         Schema::dropIfExists('puntajes_por_variable');
38     }
39 }

```

Anexo 121. Migración `puntajes_por_variable`.

Tabla de Sumatorias de Puntuaciones

Registra resultados agregados como sumatorias y porcentajes. Incluye referencias a categorías, variables y expresiones, permitiendo un cálculo preciso de métricas.

```

BackendCEP > database > migrations > 2024_11_22_004516_create_sumatoria_puntuaciones_table.php > PHP > CreateSumatoriaPuntuacionesTable
7 0 references | 0 implementations
8 class CreateSumatoriaPuntuacionesTable extends Migration
9 {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create('sumatoria_puntuaciones', function (Blueprint $table): void {
14             $table->id(column: 'SumatoriaID');
15             $table->unsignedBigInteger(column: 'UsuarioID');
16             $table->unsignedBigInteger(column: 'CategoríaID');
17             $table->unsignedBigInteger(column: 'VariableID');
18             $table->unsignedBigInteger(column: 'ExpresionesID');
19             $table->unsignedBigInteger(column: 'CuestionarioID');
20             $table->decimal(column: 'total_puntuacion', total: 8, places: 2)->default(value: 0);
21             $table->decimal(column: 'puntaje_maximo', total: 8, places: 2)->default(value: 0);
22             $table->decimal(column: 'porcentaje', total: 5, places: 2)->default(value: 0);
23             $table->timestamps();
24
25             $table->foreign(columns: 'UsuarioID')->references(columns: 'UsuarioID')
26             ->on(table: 'usuarios')->onDelete(action: 'cascade');
27             $table->foreign(columns: 'CategoríaID')->references(columns: 'CategoríaID')
28             ->on(table: 'categorias')->onDelete(action: 'cascade');
29             $table->foreign(columns: 'VariableID')->references(columns: 'VariableID')
30             ->on(table: 'variables')->onDelete(action: 'cascade');
31             $table->foreign(columns: 'ExpresionesID')->references(columns: 'ExpresionesID')
32             ->on(table: 'expresiones')->onDelete(action: 'cascade');
33             $table->foreign(columns: 'CuestionarioID')->references(columns: 'CuestionarioID')
34             ->on(table: 'cuestionarios')->onDelete(action: 'cascade');
35         });
36     }
37
38     0 references | 0 overrides
39     public function down(): void
40     {
41     }
42 }

```

Anexo 122. Migración `sumatoria_de_puntuaciones`.

Modelos

Los modelos en este sistema definen la estructura y lógica de las tablas en la base de datos. Estos modelos representan entidades como usuarios, grupos, instituciones, países, carreras, cuestionarios, respuestas, entre otros, y facilitan operaciones como consultas, relaciones, y transacciones con el CRUD.

Modelo Usuario

Administra los datos de los usuarios, incluyendo relaciones con grupos, instituciones, países y carreras. Este modelo permite autenticación, gestiona campos sensibles y facilita la interacción con otras entidades del sistema.

```
BackendCEP > app > Models > Usuario.php > ...
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Foundation\Auth\User as Authenticatable;
7 use Laravel\Sanctum\HasApiTokens;
8 use App\Models\Grupos;
9
10 27 references | 0 implementations
11 class Usuario extends Authenticatable
12 {
13     use HasApiTokens, HasFactory;
14
15     0 references
16     protected $table = 'usuarios';
17
18     0 references
19     protected $primaryKey = 'UsuarioID';
20
21     0 references
22     protected $fillable = [
23         'Nombre',
24         'Primer_apellido',
25         'Segundo_apellido',
26         'Matricula',
27         'Semestre',
28         'Tipo',
29         'Correo',
30         'Password',
31         'edad',
32         'genero',
33         'InstitucionID',
34         'IDPais',
35         'CarreraID'
36     ];
37 }
```

```
BackendCEP > app > Models > Usuario.php > PHP > Usuario > grupos()
10 class Usuario extends Authenticatable
34 // Ocultar campos sensibles
35 0 references
36 protected $hidden = [
37     'Password',
38     'remember_token',
39 ];
40
41 // Casts para tipos de datos
42 0 references
43 protected $casts = [
44     'correo_verified_at' => 'datetime',
45 ];
46
47 // Relación muchos a muchos con grupos
48 0 references | 0 overrides
49 public function grupos(): BelongsToMany
50 {
51     return $this->belongsToMany(related: Grupos::class, table: 'grupo_usuario',
52     foreignPivotKey: 'UsuarioID', relatedPivotKey: 'GrupoID')
53     ->withPivot(columns: 'CarreraID', 'InstitucionID')
54     ->withTimestamps();
55 }
56
57 // Relación con la carrera
58 0 references | 0 overrides
59 public function carrera(): BelongsTo
60 {
61     return $this->belongsTo(related: Carreras::class, foreignKey: 'CarreraID');
62 }
```

Anexo 123. Modelo usuarios.

```

// Relación con la institución
0 references | 0 overrides
public function institucion(): BelongsTo
{
    return $this->belongsTo(related: Instituciones::class, foreignKey: 'InstitucionID');
}

// Relación con el país
0 references | 0 overrides
public function pais(): BelongsTo
{
    return $this->belongsTo(related: Países::class, foreignKey: 'IDPais');
}
}

```

Anexo 124. Relaciones con institución y país.

Modelo Países

Gestiona los datos relacionados con los países y establece una relación uno a muchos con usuarios, conectando cada país con los usuarios que pertenecen a él. Define atributos básicos como el nombre y utiliza una clave primaria personalizada (IDPais).

```

BackendCEP > app > Models > Países.php > PHP > Países > usuarios()
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  9 references | 0 implementations
8  class Países extends Model
9  {
10     0 references
11     protected $table = 'países';
12     0 references
13     protected $primaryKey = 'IDPais';
14     0 references
15     protected $fillable = ['Nombre'];
16     0 references
17     public $incrementing = true;
18     0 references
19     protected $keyType = 'int';
20     0 references
21     public $timestamps = true;
22
23     // Relación con Usuarios
24     0 references | 0 overrides
25     public function usuarios(): HasMany
26     {
27         return $this->hasMany(related: Usuario::class,
28                               foreignKey: 'IDPais', localKey: 'IDPais');
29     }
30 }

```

Anexo 125. Modelo Países.

Modelo Instituciones

Representa a las instituciones educativas, con atributos como Nombre e IDPais. Este modelo establece relaciones con usuarios, carreras y países, asegurando que cada institución esté correctamente asociada a un país y que pueda contener varias carreras y usuarios.

```
BackendCEP > app > Models > Instituciones.php > PHP > Instituciones
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  11 references | 0 implementations
8  class Instituciones extends Model
9  {
10 |
11 | 0 references
12 | protected $table = 'instituciones';
13 | 0 references
14 | protected $primaryKey = 'InstitucionID';
15 | 0 references
16 | protected $fillable = ['Nombre', 'IDPais'];
17 | 0 references
18 | public $incrementing = true;
19 | 0 references
20 | protected $keyType = 'int';
21 | 0 references
22 | public $timestamps = true;
23
24 | // Relación con Usuarios
25 | 0 references | 0 overrides
26 | public function usuarios(): HasMany
27 | {
28 |     return $this->hasMany(related: Usuario::class, foreignKey: 'InstitucionID', localKey: 'InstitucionID');
29 | }
```

Anexo 126. Modelo Instituciones.

```
7  class Instituciones extends Model
15 | 0 references
16 | public $timestamps = true;
17
18 | // Relación con Usuarios
19 | 0 references | 0 overrides
20 | public function usuarios(): HasMany
21 | {
22 |     return $this->hasMany(related: Usuario::class, foreignKey: 'InstitucionID', localKey: 'InstitucionID');
23 | }
24
25 | // Relación con Carreras
26 | 0 references | 0 overrides
27 | public function carreras(): HasMany
28 | {
29 |     return $this->hasMany(related: Carreras::class, foreignKey: 'InstitucionID', localKey: 'InstitucionID');
30 | }
31
32 | // Relación con Países
33 | 0 references | 0 overrides
34 | public function pais(): BelongsTo
35 | {
36 |     return $this->belongsTo(related: Países::class, foreignKey: 'IDPais', ownerKey: 'IDPais');
37 | }
```

Anexo 127. Relaciones del modelo instituciones.

Modelo Carreras

Se encarga de la información sobre carreras dentro de las instituciones educativas, vinculándolas a usuarios e instituciones. Incluye configuraciones para asociar cada carrera con su institución correspondiente.

```
BackendCEP > app > Models > Carreras.php > PHP > Carreras
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  11 references | 0 implementations
9  class Carreras extends Model
10 {
11     use HasFactory;
12
13     0 references
14     protected $table = 'carreras';
15     0 references
16     protected $primaryKey = 'CarreraID';
17     0 references
18     protected $fillable = [
19         'Nombre',
20         'InstitucionID',
21     ];
22
23     0 references
24     public $incrementing = true;
25     0 references
26     protected $keyType = 'int';
27     0 references
28     public $timestamps = true;
29 }
```

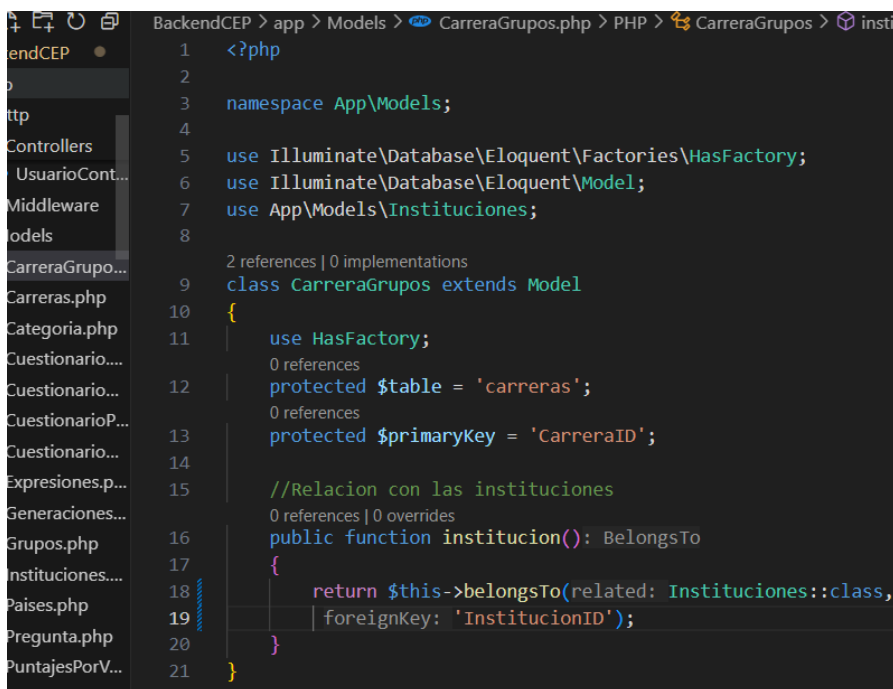
Anexo 128. Modelo Carreras.

```
// Definir la relación con Usuarios
0 references | 0 overrides
public function usuarios(): HasMany
{
    return $this->hasMany(related: Usuario::class, foreignKey: 'CarreraID', localKey: 'CarreraID');
}

// Definir la relación con Institución
0 references | 0 overrides
public function institucion(): BelongsTo
{
    return $this->belongsTo(related: Instituciones::class, foreignKey: 'InstitucionID', ownerKey: 'InstitucionID');
}
}
```

Anexo 129. Relaciones del modelo Carreras.

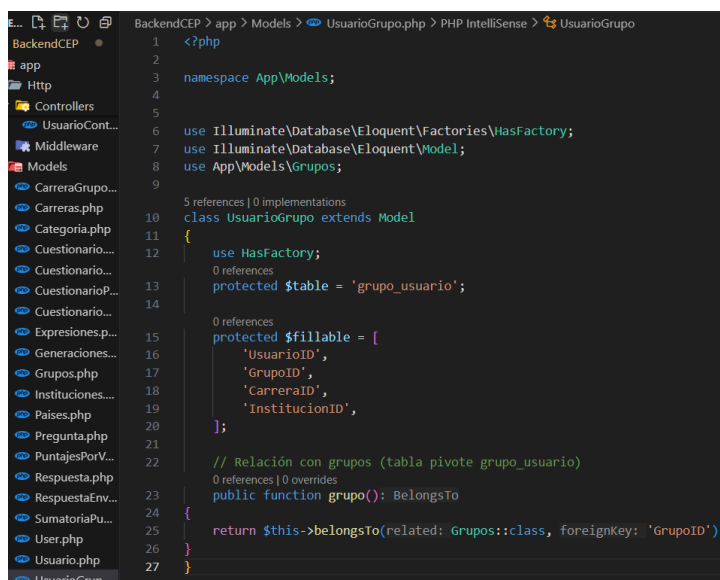
El modelo CarreraGrupos representa la tabla carreras y define una relación con el modelo Instituciones.



```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use App\Models\Instituciones;
8
9 class CarreraGrupos extends Model
10 {
11     use HasFactory;
12     protected $table = 'carreras';
13     protected $primaryKey = 'CarreraID';
14
15     //Relacion con las instituciones
16     public function institucion(): BelongsTo
17     {
18         return $this->belongsTo(related: Instituciones::class,
19             foreignKey: 'InstitucionID');
20     }
21 }
```

Anexo 130. Modelo Carrera grupos.

El modelo UsuarioGrupo permitió asociar a los usuarios con los grupos a los que están asignados, mediante la tabla pivote grupo_usuario.



```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use App\Models\Grupos;
8
9 class UsuarioGrupo extends Model
10 {
11     use HasFactory;
12     protected $table = 'grupo_usuario';
13
14     protected $fillable = [
15         'UsuarioID',
16         'GrupoID',
17         'CarreraID',
18         'InstitucionID',
19     ];
20
21     // Relacion con grupos (tabla pivote grupo_usuario)
22     public function grupo(): BelongsTo
23     {
24         return $this->belongsTo(related: Grupos::class, foreignKey: 'GrupoID');
25     }
26 }
27
```

Anexo 131. Modelo usuario_grupos.

Modelo Generaciones

Relaciona generaciones académicas con carreras específicas. Facilita la gestión y organización de las generaciones asociadas a un programa académico.

```
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use App\Models\Carreras;
8 6 references | 0 implementations
9 class Generaciones extends Model
10 {
11     use HasFactory;
12
13     0 references
14     protected $table = 'generaciones';
15     0 references
16     protected $primaryKey = 'GeneracionID';
17
18     0 references
19     protected $fillable = [
20         'Nombre',
21         'CarreraID',
22     ];
23
24     // Relación con Carrera
25     0 references | 0 overrides
26     public function carrera(): BelongsTo
27     {
28         return $this->belongsTo(related: Carreras::class, foreignKey: 'CarreraID');
29     }
30 }
```

Anexo 132. Modelo generaciones.

Modelo Grupos

Permite organizar a los estudiantes en grupos específicos, asociándolos con generaciones y carreras. Además, incluye funcionalidades como la generación de códigos de acceso para cada grupo, agilizando su gestión.

```

Backend/TP > app > Models > Grupos.php > PHP > Grupos > usuarios()
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use App\Models\CarreraGrupos;
8  use App\Models\Usuario;
9  use Illuminate\Support\Str;
10
11  22 references | 0 implementations
12  class Grupos extends Model
13  {
14      use HasFactory;
15      0 references
16      protected $table = 'grupos';
17      0 references
18      protected $primaryKey = 'GrupoID';
19
20      0 references
21      protected $fillable = ['Nombre', 'codigo_acceso', 'semestre', 'GeneracionID', 'CarreraID', 'UsuarioID'];
22
23      // Relación con carreras
24      0 references | 0 overrides
25      public function carrera(): BelongsTo
26      {
27          return $this->belongsTo(related: CarreraGrupos::class, foreignKey: 'CarreraID')->with(reactions: 'institucion');
28      }
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48

```

Anexo 133. Modelo grupos.

```

24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
// Relación con usuarios (muchos a muchos)
0 references | 0 overrides
public function usuarios(): BelongsToMany
{
    return $this->belongsToMany(related: Usuario::class, table: 'grupo_usuario',
    foreignPivotKey: 'GrupoID', relatedPivotKey: 'UsuarioID')->withPivot(columns: 'CarreraID');
}

// Método boot para añadir el evento creating
1 reference | 0 overrides
protected static function boot(): void
{
    parent::boot();

    // Evento que se dispara antes de crear un nuevo grupo
    static::creating(callback: function ($grupo): void {
        // Generar un código de acceso único de 8 caracteres
        $grupo->codigo_acceso = Str::random(8);
    });
}

0 references | 0 overrides
public function usuario(): BelongsTo
{
    return $this->belongsTo(related: Usuario::class, foreignKey: 'UsuarioID');
}
}

```

Anexo 134. Relaciones del modelo grupos

Modelo Pregunta

Organiza preguntas de cuestionarios, permitiendo relaciones con múltiples respuestas y variables. Facilita la asociación de preguntas con sus opciones de respuesta y contexto.

```

5 use App\Models\Variable;
6 use Illuminate\Database\Eloquent\Model;
7
8 // 11 references | 0 implementations
9 class Pregunta extends Model
10 {
11     // 0 references
12     protected $table = 'preguntas';
13     // 0 references
14     protected $primaryKey = 'PreguntaID';
15
16     // 0 references
17     public $timestamps = true;
18
19     // 0 references
20     protected $fillable = ['PreguntaID', 'Texto', 'Tipo'];
21
22     // Relación con las Respuesta
23     // 0 references | 0 overrides
24     public function respuestas(): HasMany
25     {
26         return $this->hasMany(related: Respuesta::class, foreignKey: 'PreguntaID', localKey: 'PreguntaID');
27     }
28
29     // Relación con las variables
30     // 0 references | 0 overrides
31     public function variables(): BelongsToMany
32     {
33         return $this->belongsToMany(related: Variable::class, table: 'pregunta_variable',
34             foreignPivotKey: 'PreguntaID', relatedPivotKey: 'VariableID');
35     }
36 }

```

Anexo 135. Modelo Pregunta

Modelo PuntajesPorVariable

Registra los puntajes obtenidos por los usuarios en cuestionarios para cada variable, permitiendo un análisis detallado de los resultados.

```

endCEP > app > Models > PuntajesPorVariable.php > PHP > PuntajesPorVariable > cuestionario()
class PuntajesPorVariable extends Model
{
    use HasFactory;
    // 0 references
    protected $table = 'puntajes_por_variable';
    // 0 references
    protected $primaryKey = 'PuntajeID';
    // 0 references
    protected $fillable = [
        'UsuarioID',
        'GrupoID',
        'CuestionarioID',
        'VariableID',
        'puntaje',
    ];
    // 0 references | 0 overrides
    public function usuario(): BelongsTo
    {
        return $this->belongsTo(related: User::class, foreignKey: 'UsuarioID');
    }
    // 0 references | 0 overrides
    public function grupo(): BelongsTo
    {
        return $this->belongsTo(related: Grupos::class, foreignKey: 'GrupoID');
    }
    // 0 references | 0 overrides
    public function cuestionario(): BelongsTo
    {
        return $this->belongsTo(related: Cuestionario::class, foreignKey: 'CuestionarioID');
    }
    // 0 references | 0 overrides
    public function variable(): BelongsTo
    {
        return $this->belongsTo(related: Variable::class, foreignKey: 'VariableID');
    }
}

```

Anexo 136. Modelo PuntajesPorVariable

Modelo RespuestaEnviada

Almacena las respuestas proporcionadas por los usuarios en los cuestionarios. Registra el valor de cada respuesta y su relación con la pregunta correspondiente.

```
BackendCEP > app > Models > RespuestaEnviada.php > PHP > RespuestaEnviada > usuario()
3 references | 0 implementations
10 class RespuestaEnviada extends Model
11 {
12     0 references
    protected $table = 'respuestas_enviadas';
13     0 references
    protected $primaryKey = 'RespuestaEnviadaID';
14     0 references
    protected $fillable = [
15         'CuestionarioID',
16         'PreguntaID',
17         'respuesta_valor',
18         'UsuarioID',
19     ];
20     // Relación con la tabla de Preguntas
    0 references | 0 overrides
21     public function pregunta(): BelongsTo
22     {
23         return $this->belongsTo(related: Pregunta::class, foreignKey: 'PreguntaID');
24     }
25     // Relación con la tabla de Usuario (Alumno)
    0 references | 0 overrides
26     public function usuario(): BelongsTo
27     {
28         return $this->belongsTo(related: Usuario::class, foreignKey: 'UsuarioID');
29     }
30     // Relación con la tabla de Cuestionarios
    0 references | 0 overrides
31     public function cuestionario(): BelongsTo
32     {
33         return $this->belongsTo(related: Cuestionario::class, foreignKey: 'CuestionarioID');
34     }
35 }
```

Anexo 137. Modelo RespuestaEnviada

Modelo SumatoriaPuntajes

Almacena datos agregados como sumatorias y porcentajes relacionados con evaluaciones. Se conecta con categorías, variables y expresiones para calcular resultados con precisión.

```
BackendCEP > app > Models > SumatoriaPuntajes.php > PHP > SumatoriaPuntajes > variable()
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 5 references | 0 implementations
9 class SumatoriaPuntajes extends Model
10 {
11     use HasFactory;
12     0 references
    protected $table = 'sumatoria_puntuaciones';
13     0 references
    protected $primaryKey = 'SumatoriaID';
14     0 references
    protected $fillable = [
15         'UsuarioID',
16         'CategoríaID',
17         'VariableID',
18         'ExpresionesID',
19         'CuestionarioID',
20         'total_puntuacion',
21         'puntaje_maximo',
22         'porcentaje'
23     ];
24     // Relación con Usuario
    0 references | 0 overrides
25     public function usuario(): BelongsTo
26     {
27         return $this->belongsTo(related: Usuario::class, foreignKey: 'UsuarioID');
28     }
29 }
```

Anexo 138. Modelo SumatoriaPuntajes

```
29 // Relación con Variable
30 0 references | 0 overrides
31 public function variable(): BelongsTo
32 {
33     return $this->belongsTo(related: Variable::class, foreignKey: 'VariableID');
34 }
35 // Relación con Cuestionario
36 0 references | 0 overrides
37 public function cuestionario(): BelongsTo
38 {
39     return $this->belongsTo(related: Cuestionario::class, foreignKey: 'CuestionarioID');
40 }
41 0 references | 0 overrides
42 public function categoria(): BelongsTo
43 {
44     return $this->belongsTo(related: Categoria::class, foreignKey: 'CategoriaID');
45 }
46 0 references | 0 overrides
47 public function expresion(): BelongsTo
48 {
49     return $this->belongsTo(related: Expresiones::class, foreignKey: 'ExpresionesID');
50 }
```

Anexo 139. Relaciones del modelo SumatoriaPuntajes

Modelo Variable

Define las variables utilizadas en los cuestionarios, vinculándolas a categorías y preguntas. Esto facilita la evaluación y organización de datos en contextos específicos.

```

BackendCEP > app > Models > Variable.php > PHP > Variable > preguntas()
6 use Illuminate\Database\Eloquent\Model;
7
8 class Variable extends Model
9 {
10     use HasFactory;
11     protected $table = 'variables';
12     protected $primaryKey = 'VariableID';
13     protected $fillable = ['Nombre', 'CategoriaID'];
14     public $timestamps = true;
15     // Relación con las preguntas
16     public function preguntas(): BelongsToMany
17     {
18         return $this->belongsToMany(related: Pregunta::class, table: 'pregunta_variable',
19             foreignPivotKey: 'VariableID', relatedPivotKey: 'PreguntaID');
20     }
21     public function categoria(): BelongsTo
22     {
23         return $this->belongsTo(related: Categoria::class, foreignKey: 'CategoriaID');
24     }
25     public function expresiones(): HasMany
26     {
27         return $this->hasMany(related: Expresiones::class, foreignKey: 'VariableID');
28     }
29 }
30

```

Anexo 140. Modelo Variable

Modelo Cuestionario

Gestiona cuestionarios asignados a grupos. Incluye relaciones con grupos y preguntas, permitiendo estructurar y vincular cuestionarios con sus elementos asociados a través de tablas pivote.

```

BackendCEP > app > Models > Cuestionario.php > ...
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use App\Models\Grupos;
8
9 class Cuestionario extends Model
10 {
11     use HasFactory;
12
13     protected $primaryKey = 'CuestionarioID';
14
15     protected $fillable = [
16         'GrupoID',
17         'Nombre',
18         'Descripcion',
19     ];
20     //Relacion con los grupos
21     public function grupo(): BelongsTo
22     {
23         return $this->belongsTo(related: Grupos::class, foreignKey: 'GrupoID');
24     }
25 }

```

Anexo 141. Modelo Cuestionario

```

0 references | 0 overrides
public function grupo(): BelongsTo
{
    return $this->belongsTo(related: Grupos::class, foreignKey: 'GrupoID');
}

//Relacion con las preguntas (tabla cuestionario_pregunta)
0 references | 0 overrides
public function preguntas(): BelongsToMany
{
    return $this->belongsToMany(related: Pregunta::class, table: 'cuestionario_pregunta',
    foreignPivotKey: 'CuestionarioID', relatedPivotKey: 'PreguntaID')
    ->with(relations: 'respuestas');
}

0 references | 0 overrides
public function cuestionarioGrupos(): HasMany
{
    return $this->hasMany(related: CuestionarioGrupo::class, foreignKey: 'CuestionarioID');
}

```

Anexo 142. Relaciones del modelo Cuestionario

Modelo CuestionarioGrupo

Establece una relación entre cuestionarios y grupos, facilitando la gestión de cuestionarios asignados a grupos específicos. Incluye control de fechas de publicación y cierre.

```

CuestionarioGrupo.php X Cuestionario.php CarreraGrupos.php Usuario.php SumatoriaPuntu
BackendCEP > app > Models > CuestionarioGrupo.php > ...
7 references | 0 implementations
10 class CuestionarioGrupo extends Model
11 {
12     use HasFactory;
13
14     0 references
15     protected $table = 'cuestionario_grupo';
16     0 references
17     protected $primaryKey = 'CuestionarioGrupoID';
18
19     0 references
20     protected $fillable = [
21         'CuestionarioID',
22         'GrupoID',
23         'FechaPublicacion',
24         'FechaCierre',
25     ];
26
27     // Relación con el cuestionario
28     0 references | 0 overrides
29     public function cuestionario(): BelongsTo
30     {
31         return $this->belongsTo(related: Cuestionario::class, foreignKey: 'CuestionarioID');
32     }
33
34     // Relación con los grupos
35     0 references | 0 overrides
36     public function grupo(): BelongsTo
37     {
38         return $this->belongsTo(related: Grupos::class, foreignKey: 'GrupoID');
39     }
40 }

```

Anexo 143. Modelo CuestionarioGrupo.

Modelo CuestionarioPregunta

Define la relación entre cuestionarios y preguntas a través de una tabla intermedia, permitiendo una asignación flexible y organizada de preguntas a diferentes cuestionarios.

```
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class CuestionarioPregunta extends Model
9 {
10     protected $table = 'cuestionario_pregunta';
11     protected $primaryKey = 'CuestionarioPreguntaID';
12     use HasFactory;
13     protected $fillable = [
14         'CuestionarioID',
15         'PreguntaID',
16     ];
17
18     // Relación con el cuestionario
19     public function cuestionario(): BelongsTo
20     {
21         return $this->belongsTo(related: Cuestionario::class, foreignKey: 'CuestionarioID');
22     }
23
24     // Relación con la pregunta
25     public function pregunta(): BelongsTo
26     {
27         return $this->belongsTo(related: Pregunta::class, foreignKey: 'PreguntaID', ownerKey: 'PreguntaID');
28     }
29 }
30 }
```

Anexo 144. Modelo CuestionarioGrupo.

Modelo CuestionarioResuelto

Registra información sobre los cuestionarios completados por los usuarios, incluyendo la fecha de resolución y su asociación con grupos y usuarios.

```

BackendCEP > app > Models > CuestionarioResuelto.php > PHP > CuestionarioResuelto > cuestionario()
8 class CuestionarioResuelto extends Model
9
10     use HasFactory;
11     0 references
12     protected $table = 'cuestionario_resuelto';
13     0 references
14     protected $primaryKey = 'CuestionarioResueltoID';
15     0 references
16     protected $fillable = [
17         'UsuarioID',
18         'CuestionarioID',
19         'GrupoID',
20         'FechaResuelto',
21         'Resuelto'
22     ];
23
24     // Relación con el usuario
25     0 references | 0 overrides
26     public function usuario(): BelongsTo
27     {
28         return $this->belongsTo(related: Usuario::class, foreignKey: 'UsuarioID');
29     }
30
31     // Relación con el cuestionario
32     0 references | 0 overrides
33     public function cuestionario(): BelongsTo
34     {
35         return $this->belongsTo(related: Cuestionario::class, foreignKey: 'CuestionarioID');
36     }
37
38     // Relación con el grupo
39     0 references | 0 overrides
40     public function grupo(): BelongsTo
41     {
42         return $this->belongsTo(related: Grupos::class, foreignKey: 'GrupoID');
43     }
44
45

```

Anexo 145. Modelo CuestionarioResuelto.

Modelo Expresiones

Administra expresiones relacionadas con variables, vinculándolas de forma estructurada para facilitar el manejo y análisis de datos en evaluaciones

```

BackendCEP > app > Models > Expresiones.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  3 references | 0 implementations
9  class Expresiones extends Model
10 {
11     use HasFactory;
12
13     0 references
14     protected $table = 'expresiones';
15     0 references
16     protected $primaryKey = 'ExpresionesID';
17     0 references
18     protected $fillable = ['Nombre', 'VariableID'];
19
20     0 references
21     public $timestamps = true;
22
23     // Relación con la variable
24     0 references | 0 overrides
25     public function variable(): BelongsTo
26     {
27         return $this->belongsTo(related: Variable::class, foreignKey: 'VariableID');
28     }
29 }

```

Anexo 146. Modelo CuestionariosResueltos.

Controladores

Los controladores son responsables de gestionar la interacción entre los usuarios y la base de datos, procesando las solicitudes y enviando respuestas adecuadas. Cada controlador maneja tareas específicas para garantizar que los datos sean procesados y devueltos correctamente al cliente.

Controlador Usuarios

Este controlador sirve para gestionar todas las operaciones relacionadas con los usuarios. En el caso del registro, valida los datos enviados por el cliente, garantizando que cumplan con las reglas establecidas, como la unicidad del correo y la matrícula. Si los datos son correctos, crea un nuevo usuario, encripta su contraseña y genera un token de autenticación, devolviendo el token junto con los datos del usuario registrado.

```

ndCEP > app > Http > Controllers > UsuarioController.php > ...
<?php

namespace App\Http\Controllers;

use App\Models\Usuario;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use App\Helpers\ResponseHelper;

14 references | 0 implementations
class UsuarioController extends Controller
{
    // Registro de usuario
    1 reference | 0 overrides
    public function register(Request $request): JsonResponse|mixed
    {
        $validator = Validator::make($request->all(), [
            'Nombre' => 'required|string|max:255',
            'Primer_apellido' => 'required|string|max:255',
            'Segundo_apellido' => 'nullable|string|max:255',
            'Matricula' => 'required|string|max:255|unique:usuarios',
            'Semestre' => 'required|string|max:255',
            'Tipo' => 'required|in:Alumno,Maestro,Admin',
            'Correo' => 'required|string|email|max:255|unique:usuarios',
            'Password' => 'required|string|min:8',
            'InstitucionID' => 'nullable|exists:instituciones,InstitucionID',
            'CarreraID' => 'nullable|exists:carreras,CarreraID',
            'edad' => 'nullable|integer|min:1|max:120',
            'genero' => 'nullable|in:Masculino,Femenino,Otro',
            'IDPais' => 'nullable|exists:países,IDPais',
        ]);
    }
}

```

Anexo 147. Validador de datos para el registro de usuarios.

```

BackendCEP > app > Http > Controllers > UsuarioController.php > ...
11 class UsuarioController extends Controller
14 public function register(Request $request): JsonResponse|mixed
21
22 if ($validator->fails()) {
23     return ResponseHelper::error(message: $validator->errors(), status: 422);
24 }
25
26 // Crear el usuario
27 $usuario = Usuario::create(attributes: [
28     'Nombre' => $request->Nombre,
29     'Primer_apellido' => $request->Primer_apellido,
30     'Segundo_apellido' => $request->Segundo_apellido,
31     'Matricula' => $request->Matricula,
32     'Semestre' => $request->Semestre,
33     'Tipo' => $request->Tipo,
34     'Correo' => $request->Correo,
35     'Password' => Hash::make($request->Password),
36     'InstitucionID' => $request->InstitucionID,
37     'CarreraID' => $request->CarreraID,
38     'edad' => $request->edad,
39     'genero' => $request->genero,
40     'IDPais' => $request->IDPais,
41 ]);
42
43 // Generar token de autenticación
44 $token = $usuario->createToken('auth_token')->plainTextToken;
45
46 return ResponseHelper::success(data: [
47     'access_token' => $token,
48     'token_type' => 'Bearer',
49     'usuario' => $usuario,
50 ], message: 'Usuario registrado correctamente', status: 201);
51 }

```

Anexo 148. Generación de token y Respuesta del Registro.

Para mostrar la información de un usuario específico, busca al usuario por su ID, incluyendo las relaciones con carrera, institución y país. Devuelve una respuesta que organiza los datos en un formato estructurado y fácil de interpretar.

```

BackendCEP > app > Http > Controllers > UsuarioController.php > ...
11 class UsuarioController extends Controller
{
    63 // 2 references | 0 overrides
    64 public function show($id): JsonResponse|mixed
    65 {
    66     // Cargar las relaciones de carrera, institución y país
    67     $usuario = Usuario::with('carrera', 'institucion', 'pais')->findOrFail($id);
    68     // Formatear la respuesta para incluir los datos como objetos
    69     $formattedUsuario = [
    70         'UsuarioID' => $usuario->UsuarioID,
    71         'Nombre' => $usuario->Nombre,
    72         'Primer_apellido' => $usuario->Primer_apellido,
    73         'Segundo_apellido' => $usuario->Segundo_apellido,
    74         'Matricula' => $usuario->Matricula,
    75         'Semestre' => $usuario->Semestre,
    76         'Tipo' => $usuario->Tipo,
    77         'Correo' => $usuario->Correo,
    78         'edad' => $usuario->edad,
    79         'genero' => $usuario->genero,
    80         'InstitucionID' => [
    81             'InstitucionID' => $usuario->institucion->InstitucionID,
    82             'Nombre' => $usuario->institucion->Nombre,
    83             'IDPais' => $usuario->institucion->IDPais,
    84             'created_at' => $usuario->institucion->created_at,
    85             'updated_at' => $usuario->institucion->updated_at,
    86         ],
    87         'CarreraID' => [
    88             'CarreraID' => $usuario->carrera->CarreraID,
    89             'Nombre' => $usuario->carrera->Nombre,
    90             'InstitucionID' => $usuario->carrera->InstitucionID,
    91             'created_at' => $usuario->carrera->created_at,
    92             'updated_at' => $usuario->carrera->updated_at,
    93         ],
    94     ];
    95     return Response::json($formattedUsuario);
    96 }

```

Anexo 149. obtención de detalles de un usuario.

```

BackendCEP > app > Http > Controllers > UsuarioController.php > ...
11 class UsuarioController extends Controller
{
    63 public function show($id): JsonResponse|mixed
    64 {
    65     $usuario = Usuario::findOrFail($id);
    66     $formattedUsuario = [
    67         'IDPais' => [
    68             'IDPais' => $usuario->pais->IDPais,
    69             'Nombre' => $usuario->pais->Nombre,
    70             'created_at' => $usuario->pais->created_at,
    71             'updated_at' => $usuario->pais->updated_at,
    72         ],
    73         'created_at' => $usuario->created_at,
    74         'updated_at' => $usuario->updated_at,
    75     ];
    76     return Response::json($formattedUsuario);
    77 }
    78 }

```

Anexo 150. Respuesta de los detalles del usuario

La función update permite editar los datos de un usuario existente. Primero, localiza al usuario por su ID y valida los datos enviados, verificando aspectos como la unicidad de correo y matrícula. Si hay errores, responde con un código 422 y detalla el problema. Si los datos son válidos, actualiza la información del usuario, incluyendo nombre, correo, institución y carrera.

```

BackendCEP > app > Http > Controllers > UsuarioController.php > PHP IntelliSense > UsuarioController
11 class UsuarioController extends Controller
{
    108 public function update(Request $request, $id): JsonResponse|mixed
    109 {
    110     $usuario = Usuario::findOrFail($id);
    111     $validator = Validator::make($request->all(), [
    112         'Nombre' => 'sometimes|string|max:255',
    113         'Primer_apellido' => 'sometimes|string|max:255',
    114         'Segundo_apellido' => 'sometimes|nullable|string|max:255',
    115         'Matricula' => 'sometimes|string|max:255|unique:usuarios,Matricula,' . $id . ',' . $usuario->UsuarioID,
    116         'Semestre' => 'sometimes|string|max:255',
    117         'Tipo' => 'sometimes|in:Alumno,Maestro,Admin',
    118         'Correo' => 'sometimes|string|email|max:255|unique:usuarios,Correo,' . $id . ',' . $usuario->UsuarioID,
    119         'Password' => 'sometimes|string|min:8',
    120         'edad' => 'sometimes|integer|min:1|max:120',
    121         'genero' => 'sometimes|in:Masculino,Femenino,Otro',
    122         'InstitucionID' => 'sometimes|nullable|exists:instituciones,InstitucionID',
    123         'CarreraID' => 'sometimes|nullable|exists:carreras,CarreraID',
    124         'IDPais' => 'sometimes|nullable|exists:países,IDPais',
    125     ]);
    126     if ($validator->fails()) {
    127         return Response::json(['errors' => $validator->errors()]);
    128     }
    129     // Actualizar el usuario
    130     $usuario->update($request->only([
    131         'Nombre', 'Primer_apellido', 'Segundo_apellido', 'Matricula', 'Semestre', 'Tipo',
    132         'Correo', 'InstitucionID', 'CarreraID', 'edad', 'genero', 'IDPais'
    133     ]));
    134     if ($request->has('Password')) {
    135         $usuario->update(['Password' => Hash::make($request->Password)]);
    136     }
    137     return Response::json(['usuario' => $usuario]);
    138 }
    139 }

```

Anexo 151. función actualizar usuario.

La función usuarios facilita la obtención de todos los usuarios registrados en el sistema, cargando sus relaciones con carrera, institución y país. La respuesta se organiza en un formato JSON detallado.

```
BackendCEP > app > Http > Controllers > UsuarioController.php > PHP IntelliSense > UsuarioController > update
11 class UsuarioController extends Controller
146 public function usuarios(): JsonResponse|mixed
147 {
148     // Cargar las relaciones de carrera, institución y país
149     $usuarios = Usuario::with('carrera', 'institucion', 'pais')->get();
150
151     // Formatear la respuesta para incluir los datos como objetos
152     $formattedUsuarios = $usuarios->map(callback: function (Usuario $usuario): array {
153         return [
154             'UsuarioID' => $usuario->UsuarioID,
155             'Nombre' => $usuario->Nombre,
156             'Primer_apellido' => $usuario->Primer_apellido,
157             'Segundo_apellido' => $usuario->Segundo_apellido,
158             'Matricula' => $usuario->Matricula,
159             'Semestre' => $usuario->Semestre,
160             'Tipo' => $usuario->Tipo,
161             'Correo' => $usuario->Correo,
162             'edad' => $usuario->edad,
163             'genero' => $usuario->genero,
164
165             // Manejar InstitucionID, si no está nulo
166             'InstitucionID' => $usuario->institucion ? [
167                 'InstitucionID' => $usuario->institucion->InstitucionID,
168                 'Nombre' => $usuario->institucion->Nombre,
169                 'IDPais' => $usuario->institucion->IDPais,
170                 'created_at' => $usuario->institucion->created_at,
171                 'updated_at' => $usuario->institucion->updated_at,
172             ] : null,
173
174             // Manejar CarreraID, si no está nulo
175             'CarreraID' => $usuario->carrera ? [
176                 'CarreraID' => $usuario->carrera->CarreraID,
177                 'Nombre' => $usuario->carrera->Nombre,
178                 'InstitucionID' => $usuario->carrera->InstitucionID,
179                 'created_at' => $usuario->carrera->created_at,
180                 'updated_at' => $usuario->carrera->updated_at,
181             ] : null,
```

Anexo 152. función usuarios parte 1.

```

] : null,

// Manejar IDPais, si no está nulo
'IDPais' => $usuario->pais ? [
    'IDPais' => $usuario->pais->IDPais,
    'Nombre' => $usuario->pais->Nombre,
    'created_at' => $usuario->pais->created_at,
    'updated_at' => $usuario->pais->updated_at,
] : null,

'created_at' => $usuario->created_at,
'updated_at' => $usuario->updated_at,
]);

return ResponseHelper::success(data: $formattedUsuarios, message: 'Lista de usuarios obtenida correctamente');
}

```

Anexo 153. Función usuarios parte 2.

Para eliminar un usuario, busca el registro por su ID y lo elimina, devolviendo un mensaje de confirmación.

```

// Eliminar un usuario
1 reference|0 overrides
public function eliminar($id): JsonResponse|mixed
{
    $usuario = Usuario::findOrFail(id: $id);
    $usuario->delete();
    return ResponseHelper::success(data: null, message: 'Usuario eliminado correctamente');
}

```

Anexo 154. Función eliminar usuario.

Se crearon varias funciones que permiten obtener listas específicas de usuarios, como alumnos, maestros o administradores. Estas funciones filtran a los usuarios según su rol y cargan sus relaciones asociadas, devolviendo los datos en un formato organizado. También permite buscar alumnos o maestros por institución, devolviendo los registros que coincidan con el InstitucionID proporcionado.

```

BackendCEP > app > Http > Controllers > UsuarioController.php > PHP IntelliSense > UsuarioController > obtenerAlumnos
11 class UsuarioController extends Controller
209 public function obtenerAlumnos(): JsonResponse|mixed
210 {
211     // Cargar las relaciones de carrera, institución y país
212     $alumnos = Usuario::with(relations: 'carrera', 'institucion', 'pais')
213         ->where(column: 'Tipo', operator: 'Alumno')
214         ->get();
215
216     // Formatear la respuesta para incluir los datos como objetos
217     $formattedAlumnos = $alumnos->map(callback: function (Usuario $alumno): array {
218         return [
219             'UsuarioID' => $alumno->UsuarioID,
220             'Nombre' => $alumno->Nombre,
221             'Primer_apellido' => $alumno->Primer_apellido,
222             'Segundo_apellido' => $alumno->Segundo_apellido,
223             'Matricula' => $alumno->Matricula,
224             'Semestre' => $alumno->Semestre,
225             'Tipo' => $alumno->Tipo,
226             'Correo' => $alumno->Correo,
227             'edad' => $alumno->edad,
228             'genero' => $alumno->genero,
229             'InstitucionID' => [
230                 'InstitucionID' => $alumno->institucion->InstitucionID,
231                 'Nombre' => $alumno->institucion->Nombre,
232                 'IDPais' => $alumno->institucion->IDPais,
233                 'created_at' => $alumno->institucion->created_at,
234                 'updated_at' => $alumno->institucion->updated_at,
235             ],
236             'CarreraID' => [
237                 'CarreraID' => $alumno->carrera->CarreraID,
238                 'Nombre' => $alumno->carrera->Nombre,
239                 'InstitucionID' => $alumno->carrera->InstitucionID,
240                 'created_at' => $alumno->carrera->created_at,
241                 'updated_at' => $alumno->carrera->updated_at,
242             ],

```

Anexo 155. Función obtenerAlumnos parte 1.

```

243             'IDPais' => [
244                 'IDPais' => $alumno->pais->IDPais,
245                 'Nombre' => $alumno->pais->Nombre,
246                 'created_at' => $alumno->pais->created_at,
247                 'updated_at' => $alumno->pais->updated_at,
248             ],
249             'created_at' => $alumno->created_at,
250             'updated_at' => $alumno->updated_at,
251         ];
252     });
253
254     return ResponseHelper::success(data: $formattedAlumnos, message: 'Lista de alumnos obtenida correctamente');
255 }

```

Anexo 156. Función obtenerAlumnos parte 2.

```

BackendCIP > app > Http > Controllers > UsuarioController.php > PHP IntelliSense > UsuarioController > obtenerMaestros
11 class UsuarioController extends Controller
258 public function obtenerMaestros(): JsonResponse|mixed
259 {
260     // Cargar las relaciones de carrera, institución y país
261     $maestros = Usuario::with('carrera', 'institucion', 'pais')
262         ->where('column': 'tipo', operator: 'Maestro')
263         ->get();
264
265     // Formatear la respuesta para incluir los datos como objetos
266     $formattedMaestros = $maestros->map(callback: function (Usuario $maestro): array {
267         return [
268             'UsuarioID' => $maestro->UsuarioID,
269             'Nombre' => $maestro->Nombre,
270             'Primer_apellido' => $maestro->Primer_apellido,
271             'Segundo_apellido' => $maestro->Segundo_apellido,
272             'Matricula' => $maestro->Matricula,
273             'Semestre' => $maestro->Semestre,
274             'Tipo' => $maestro->Tipo,
275             'Correo' => $maestro->Correo,
276             'edad' => $maestro->edad,
277             'genero' => $maestro->genero,
278
279             // Manejar InstitucionID, si no está nulo
280             'InstitucionID' => $maestro->institucion ? [
281                 'InstitucionID' => $maestro->institucion->InstitucionID,
282                 'Nombre' => $maestro->institucion->Nombre,
283                 'IDPais' => $maestro->institucion->IDPais,
284                 'created_at' => $maestro->institucion->created_at,
285                 'updated_at' => $maestro->institucion->updated_at,
286             ] : null,
287
288             // Manejar CarreraID, si no está nulo
289             'CarreraID' => $maestro->carrera ? [
290                 'CarreraID' => $maestro->carrera->CarreraID,
291                 'Nombre' => $maestro->carrera->Nombre,
292                 'InstitucionID' => $maestro->carrera->InstitucionID,
293                 'created_at' => $maestro->carrera->created_at,

```

Anexo 157. Función obtenerMaestros parte 1.

```

        ] : null,

        // Manejar IDPais, si no está nulo
        'IDPais' => $maestro->pais ? [
            'IDPais' => $maestro->pais->IDPais,
            'Nombre' => $maestro->pais->Nombre,
            'created_at' => $maestro->pais->created_at,
            'updated_at' => $maestro->pais->updated_at,
        ] : null,

        'created_at' => $maestro->created_at,
        'updated_at' => $maestro->updated_at,
    ];
});

return ResponseHelper::success(data: $formattedMaestros, message: 'Lista de maestros obtenida correctamente');
}

```

Anexo 158. función obtenerMaestros parte 2.

```

BackendCIP > app > Http > Controllers > UsuarioController.php > PHP IntelliSense > UsuarioController > obtenerAdministradores
11 class UsuarioController extends Controller
314 public function obtenerAdministradores(): JsonResponse|mixed
315 {
316     // Cargar las relaciones de carrera, institución y país
317     $administradores = Usuario::with('carrera', 'institucion', 'pais')
318         ->where('column': 'tipo', operator: 'Admin')
319         ->get();
320
321     // Formatear la respuesta para incluir los datos como objetos
322     $formattedAdministradores = $administradores->map(callback: function (Usuario $admin): array {
323         return [
324             'UsuarioID' => $admin->UsuarioID,
325             'Nombre' => $admin->Nombre,
326             'Primer_apellido' => $admin->Primer_apellido,
327             'Segundo_apellido' => $admin->Segundo_apellido,
328             'Matricula' => $admin->Matricula,
329             'Semestre' => $admin->Semestre,
330             'Tipo' => $admin->Tipo,
331             'Correo' => $admin->Correo,
332             'edad' => $admin->edad,
333             'genero' => $admin->genero,
334
335             // Manejar InstitucionID, si no está nulo
336             'InstitucionID' => $admin->institucion ? [
337                 'InstitucionID' => $admin->institucion->InstitucionID,
338                 'Nombre' => $admin->institucion->Nombre,
339                 'IDPais' => $admin->institucion->IDPais,
340                 'created_at' => $admin->institucion->created_at,
341                 'updated_at' => $admin->institucion->updated_at,
342             ] : null,
343
344             // Manejar CarreraID, si no está nulo
345             'CarreraID' => $admin->carrera ? [
346                 'CarreraID' => $admin->carrera->CarreraID,
347                 'Nombre' => $admin->carrera->Nombre,
348                 'InstitucionID' => $admin->carrera->InstitucionID,
349                 'created_at' => $admin->carrera->created_at,

```

Anexo 159. Función obtenerAdministradores parte 1.


```

        'updated_at' => $admin->carrera->updated_at,
    ] : null,

    // Manejar IDPais, si no está nulo
    'IDPais' => $admin->pais ? [
        'IDPais' => $admin->pais->IDPais,
        'Nombre' => $admin->pais->Nombre,
        'created_at' => $admin->pais->created_at,
        'updated_at' => $admin->pais->updated_at,
    ] : null,

    'created_at' => $admin->created_at,
    'updated_at' => $admin->updated_at,
    ];
});

return ResponseHelper::success(data: $formattedAdministradores, message: 'Lista de Administradores obtenida correctamente');
}

```

Anexo 160. Función obtenerAdministradores Parte 2.

```

BackendCEP > app > Http > Controllers > UsuarioController.php > PHP IntelliSense > UsuarioController > buscarMaestroPorInstitucion
11 class UsuarioController extends Controller
12 {
13     public function buscarMaestroPorInstitucion($institucionId): JsonResponse|mixed
14     {
15         $maestros = Usuario::with(relations: 'carrera', 'institucion', 'pais')
16             ->where(column: 'Tipo', operator: 'Maestro')
17             ->where(column: 'InstitucionID', operator: $institucionId)
18             ->get();
19
20         if ($maestros->isEmpty()) {
21             return ResponseHelper::error(message: 'No se encontraron maestros en esta institución', status: 404);
22         }
23
24         // Formatear la respuesta para incluir los datos como objetos
25         $formattedMaestros = $maestros->map(callback: function (Usuario $maestro): array {
26             return [
27                 'UsuarioID' => $maestro->UsuarioID,
28                 'Nombre' => $maestro->Nombre,
29                 'Primer_apellido' => $maestro->Primer_apellido,
30                 'Segundo_apellido' => $maestro->Segundo_apellido,
31                 'Matricula' => $maestro->Matricula,
32                 'Semestre' => $maestro->Semestre,
33                 'Tipo' => $maestro->Tipo,
34                 'Correo' => $maestro->Correo,
35                 'edad' => $maestro->edad,
36                 'genero' => $maestro->genero,
37
38                 'InstitucionID' => $maestro->institucion ? [
39                     'InstitucionID' => $maestro->institucion->InstitucionID,
40                     'Nombre' => $maestro->institucion->Nombre,
41                     'IDPais' => $maestro->institucion->IDPais,
42                     'created_at' => $maestro->institucion->created_at,
43                     'updated_at' => $maestro->institucion->updated_at,
44                 ] : null,
45             ];
46         });
47     }
48 }

```

Anexo 161. Función buscarMaestrosPorInstitución parte 1.

```

        'CarreraID' => $maestro->carrera ? [
            'CarreraID' => $maestro->carrera->CarreraID,
            'Nombre' => $maestro->carrera->Nombre,
            'InstitucionID' => $maestro->carrera->InstitucionID,
            'created_at' => $maestro->carrera->created_at,
            'updated_at' => $maestro->carrera->updated_at,
        ] : null,

        'IDPais' => $maestro->pais ? [
            'IDPais' => $maestro->pais->IDPais,
            'Nombre' => $maestro->pais->Nombre,
            'created_at' => $maestro->pais->created_at,
            'updated_at' => $maestro->pais->updated_at,
        ] : null,

        'created_at' => $maestro->created_at,
        'updated_at' => $maestro->updated_at,
    ];
});

return ResponseHelper::success(data: $formattedMaestros, message: 'Lista de maestros obtenida correctamente');
}

```

Anexo 162. Función buscarMaestrosPorInstitución parte 2.

```

BackendCEP > app > Http > Controllers > UsuarioController.php > PHP IntelliSense > UsuarioController > buscarAlumnoPorInstitucion
11 class UsuarioController extends Controller
428 public function buscarAlumnoPorInstitucion($institucionId): JsonResponse|mixed
429 {
430     $alumnos = Usuario::with(relations: 'carrera', 'institucion', 'pais')
431         ->where(column: 'Tipo', operator: 'Alumno')
432         ->where(column: 'InstitucionID', operator: $institucionId)
433         ->get();
434
435     if ($alumnos->isEmpty()) {
436         return ResponseHelper::error(message: 'No se encontraron alumnos en esta institución', status: 404);
437     }
438
439     // Formatear la respuesta para incluir los datos como objetos
440     $formattedAlumnos = $alumnos->map(callback: function (Usuario $alumno): array {
441         return [
442             'UsuarioID' => $alumno->UsuarioID,
443             'Nombre' => $alumno->Nombre,
444             'Primer_apellido' => $alumno->Primer_apellido,
445             'Segundo_apellido' => $alumno->Segundo_apellido,
446             'Matricula' => $alumno->Matricula,
447             'Semestre' => $alumno->Semestre,
448             'Tipo' => $alumno->Tipo,
449             'Correo' => $alumno->Correo,
450             'edad' => $alumno->edad,
451             'genero' => $alumno->genero,
452
453             'InstitucionID' => $alumno->institucion ? [
454                 'InstitucionID' => $alumno->institucion->InstitucionID,
455                 'Nombre' => $alumno->institucion->Nombre,
456                 'IDPais' => $alumno->institucion->IDPais,
457                 'created_at' => $alumno->institucion->created_at,
458                 'updated_at' => $alumno->institucion->updated_at,
459             ] : null,

```

Anexo 163. Función buscarAlumnosPorInstitución parte 1.

```

460             'CarreraID' => $alumno->carrera ? [
461                 'CarreraID' => $alumno->carrera->CarreraID,
462                 'Nombre' => $alumno->carrera->Nombre,
463                 'InstitucionID' => $alumno->carrera->InstitucionID,
464                 'created_at' => $alumno->carrera->created_at,
465                 'updated_at' => $alumno->carrera->updated_at,
466             ] : null,
467
468             'IDPais' => $alumno->pais ? [
469                 'IDPais' => $alumno->pais->IDPais,
470                 'Nombre' => $alumno->pais->Nombre,
471                 'created_at' => $alumno->pais->created_at,
472                 'updated_at' => $alumno->pais->updated_at,
473             ] : null,
474
475             'created_at' => $alumno->created_at,
476             'updated_at' => $alumno->updated_at,
477         ];
478     });
479
480     return ResponseHelper::success(data: $formattedAlumnos, message: 'Lista de alumnos obtenida correctamente');
481 }

```

Anexo 164. Función buscarAlumnosPorInstitución parte 2.

La función login comprueba los datos ingresados para iniciar sesión, asegurándose de que se incluya una combinación válida de correo o matrícula junto con la contraseña. A continuación, busca al usuario correspondiente en la base de datos. Si no encuentra coincidencias o la contraseña no es correcta, devuelve un mensaje de error indicando que las credenciales no son válidas. En caso de autenticación exitosa, genera un token de acceso utilizando el método adecuado, permitiendo que el usuario se identifique en futuras solicitudes. Finalmente, responde con un JSON que incluye el token y la información del usuario autenticado.

```
Reference [0] overrides
public function login(Request $request): JsonResponse|mixed
{
    // Validar los datos de la solicitud
    $request->validate(rules: [
        'correo' => 'required|string',
        'password' => 'required|string',
    ]);

    // Buscar al usuario por Correo o Matrícula
    $usuario = Usuario::where(column: 'Correo', operator: $request->Correo)
        ->orWhere(column: 'Matricula', operator: $request->Correo)
        ->first();

    // Verificar si el usuario existe y la contraseña es correcta
    if (!$usuario || !Hash::check($request->Password, $usuario->Password)) {
        return response()->json(['message' => 'Credenciales incorrectas'], 401);
    }

    // Crear un token para el usuario autenticado
    $token = $usuario->createToken('auth_token')->plainTextToken;

    return response()->json([
        'access_token' => $token,
        'token_type' => 'Bearer',
        'usuario' => $usuario,
    ]);
}
```

Anexo 165. Función Login.

Controlador Países

El controlador de países se enfoca en gestionar las operaciones relacionadas con los registros de países en la base de datos. por ende, crearon las siguientes funciones.

Se creó la función agregarPais el cual se encarga de agregar un nuevo país, valida que el nombre sea único y cumpla con los requisitos establecidos. Si los datos son correctos, crea y guarda el registro, devolviendo una respuesta con el país creado.

```

BackendCEP > app > Http > Controllers > PaísesController.php > PHP > PaísesController > agregarPais()
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Helpers\ResponseHelper;
7 use App\Models\Países;
8 use Illuminate\Support\Facades\Validator;
9
10 class PaísesController extends Controller
11 {
12     // Método para crear un nuevo país
13     public function agregarPais(Request $request): JsonResponse|mixed
14     {
15         // Validación de los datos entrantes
16         $validator = Validator::make($request->all(), [
17             'Nombre' => 'required|string|max:255|unique:países,Nombre',
18         ]);
19
20         // Si la validación falla, devolver errores
21         if ($validator->fails()) {
22             return ResponseHelper::error(message: $validator->errors()->first(), status: 400);
23         }
24
25         // Crear el nuevo país
26         $pais = new Países();
27         $pais->Nombre = $request->input(key: 'Nombre');
28
29         // Guardar el país en la base de datos
30         if ($pais->save()) {
31             return ResponseHelper::success(data: $pais, message: 'País creado exitosamente', status: 201);
32         } else {
33             return ResponseHelper::error(message: 'Error al crear el país', status: 500);
34         }
35     }
36 }

```

Anexo 166. Función agregarPais.

Para actualizar un país, se creó una función la cual valida que el nuevo nombre no se duplique con otros registros, excepto con el país que está siendo modificado. Luego, busca el registro por su ID y, si lo encuentra, actualiza el nombre y guarda los cambios. Si todo se realiza correctamente, devuelve un mensaje de éxito.

```

BackendCEP > app > Http > Controllers > PaísesController.php > PHP > PaísesController > actualizarPais()
9 class PaísesController extends Controller
37 public function actualizarPais(Request $request, $id): JsonResponse|mixed
38 {
39     // Validación de los datos entrantes
40     $validator = Validator::make($request->all(), [
41         'Nombre' => 'required|string|max:255|unique:países,Nombre,' . $id . ',IDPaís',
42     ]);
43
44     // Si la validación falla, devolver errores
45     if ($validator->fails()) {
46         return ResponseHelper::error(message: $validator->errors()->first(), status: 400);
47     }
48
49     // Buscar el país por ID
50     $pais = Países::find(id: $id);
51
52     if (!$pais) {
53         return ResponseHelper::error(message: 'País no encontrado', status: 404);
54     }
55
56     // Actualizar el nombre del país
57     $pais->Nombre = $request->input(key: 'Nombre');
58
59     // Guardar los cambios en la base de datos
60     if ($pais->save()) {
61         return ResponseHelper::success(data: $pais, message: 'País actualizado correctamente');
62     } else {
63         return ResponseHelper::error(message: 'Error al actualizar el país', status: 500);
64     }
65 }

```

Anexo 167. Función actualizarPais.

Se creó la función países el cual sirve para obtener todos los países registrados, este controlador recupera los datos y los organiza en una respuesta que incluye un mensaje de confirmación.

```

// Método para obtener todas los paises
1 reference | 0 overrides
public function paises(): JsonResponse|mixed
{
    $paises = Paises::all();

    // Devolver una respuesta exitosa con los datos
    return ResponseHelper::success(data: $paises, message: 'Paises obtenidos correctamente');
}

```

Anexo 168. Función paises.

Se agrego la función país para buscar un país específico, el controlador lo localiza por su ID y devuelve su información detallada. En caso de que el país no exista, envía un mensaje de error.

```

// Método para obtener una institución específica por ID
1 reference | 0 overrides
public function pais($id): JsonResponse|mixed
{
    $paises = Paises::find(id: $id);

    if (!$paises) {
        // Devolver un error si no se encuentra la institución
        return ResponseHelper::error(message: 'Pais no encontrado', status: 404);
    }

    // Devolver una respuesta exitosa con la institución
    return ResponseHelper::success(data: $paises, message: 'Pais obtenido correctamente');
}

```

Anexo 169. Función pais.

Se creo la función eliminarPais la cual permite eliminar un país, buscándolo por su ID y eliminándolo si existe.

```

BackendCEP > app > Http > Controllers > PaisesController.php > PHP > PaisesController > buscarPaises()
9 class PaisesController extends Controller
93 public function eliminarPais($id): JsonResponse|mixed
94 {
95     $pais = Paises::findOrFail(id: $id);
96     $pais->delete();
97
98     return ResponseHelper::success(data: null, message: 'Pais eliminado correctamente');
99 }
100

```

Anexo 170. Función eliminarPais.

La función buscarPaises permite encontrar países cuyo nombre coincida parcial o totalmente con un término específico. Si encuentra resultados, devuelve una lista con los países correspondientes; de lo contrario, informa que no se encontraron coincidencias.

```

BackendCEP > app > http > Controllers > PaisController.php > PHP > PaisController > buscarPaises()
9 class PaisController extends Controller
102 public function buscarPaises(Request $request): JsonResponse{mixed
103 {
104     // Validar que se envíe el parámetro de búsqueda
105     $validator = Validator::make($request->all(), [
106         'query' => 'required|string|max:255',
107     ]);
108
109     // Si la validación falla, devolver errores
110     if ($validator->fails()) {
111         return ResponseHelper::error(message: $validator->errors()->first(), status: 400);
112     }
113
114     // Obtener el parámetro de búsqueda
115     $query = $request->input(key: 'query');
116
117     // Buscar países que coincidan parcialmente con el nombre
118     $paises = Paises::where(column: 'nombre', operator: 'LIKE', value: '%' . $query . '%')->get();
119
120     // Comprobar si se encontraron resultados
121     if ($paises->isEmpty()) {
122         return ResponseHelper::error(message: 'No se encontraron países que coincidan con la búsqueda', status: 404);
123     }
124
125     // Devolver los resultados de la búsqueda
126     return ResponseHelper::success(data: $paises, message: 'Resultados de la búsqueda obtenidos correctamente');
127 }
128 }

```

Anexo 171. Función buscarPaises.

Controlador de instituciones

En el controlador de instituciones se crearon funciones para la administración de las entidades educativas, por ende, se realizaron las siguientes funciones.

El método `crearInstitucion` valida los datos de entrada y permite crear una nueva institución asociada a un país, devolviendo un error si la validación falla o guardando la institución en caso de éxito.

```
BackendCEP > app > Http > Controllers > InstitucionesController.php > ...
5 use App\Models\Instituciones;
6 use Illuminate\Support\Facades\Validator;
7 use Illuminate\Http\Request;
8 use App\Helpers\ResponseHelper;
9
10 // 7 references | 0 implementations
11 class InstitucionesController extends Controller
12 {
13     // Método para crear una nueva institución
14     // 1 reference | 0 overrides
15     public function crearInstitucion(Request $request): JsonResponse|mixed
16     {
17         // Validación de los datos entrantes
18         $validator = Validator::make($request->all(), [
19             'Nombre' => 'required|string|max:255',
20             'IDPais' => 'required|integer|exists:países,IDPais',
21         ]);
22
23         // Si la validación falla, devolver errores
24         if ($validator->fails()) {
25             return ResponseHelper::error(message: $validator->errors()->first(), status: 400);
26         }
27
28         // Crear la nueva institución
29         $institucion = new Instituciones();
30         $institucion->Nombre = $request->input(key: 'Nombre');
31         $institucion->IDPais = $request->input(key: 'IDPais');
32
33         // Guardar la institución en la base de datos
34         if ($institucion->save()) {
35             return ResponseHelper::success(data: [
36                 'institucion' => $institucion
37             ], message: 'Institución creada exitosamente', status: 201);
38         } else {
39             return ResponseHelper::error(message: 'Error al crear la institución', status: 500);
40         }
41     }
42 }
```

Anexo 172. función `crearInstitucion`.

A través del método `instituciones` se obtienen todas las instituciones, mientras que `institucion` permite buscar una en particular por su ID, manejando errores en caso de que no exista.

```
BackendCEP > app > Http > Controllers > InstitucionesController.php > ...
10 class InstitucionesController extends Controller
11 {
12     public function crearInstitucion(Request $request): JsonResponse|mixed
13     {
14     }
15 }
16
17 // Método para obtener todas las instituciones
18 // 1 reference | 0 overrides
19 public function instituciones(): JsonResponse|mixed
20 {
21     $instituciones = Instituciones::all();
22
23     // Devolver una respuesta exitosa con los datos
24     return ResponseHelper::success(data: $instituciones, message: 'Instituciones obtenidas correctamente');
25 }
26
27 // Método para obtener una institución específica por ID
28 // 1 reference | 0 overrides
29 public function institucion($id): JsonResponse|mixed
30 {
31     $institucion = Instituciones::find(id: $id);
32
33     if (!$institucion) {
34         // Devolver un error si no se encuentra la institución
35         return ResponseHelper::error(message: 'Institución no encontrada', status: 404);
36     }
37
38     // Devolver una respuesta exitosa con la institución
39     return ResponseHelper::success(data: [
40         'institucion' => $institucion
41     ], message: 'Institución obtenida correctamente');
42 }
43 }
```

Anexo 173. Funciones `instituciones` e `institucion`.

Se creó la función `institucionesPorPais` el cual filtra las instituciones por país.

```
6 // Método para obtener instituciones filtradas por país
7 1 reference | 0 overrides
8 public function institucionesPorPais(Request $request): JsonResponse|mixed
9 {
10     $IDPais = $request->query(key: 'IDPais');
11
12     // Validar si se proporciona el IDPais
13     if (!$IDPais) {
14         return ResponseHelper::error(message: 'IDPais es requerido', status: 400);
15     }
16
17     // Filtrar instituciones por el ID del país
18     $instituciones = Instituciones::where(column: 'IDPais', operator: $IDPais)->get();
19
20     // Devolver una respuesta exitosa con las instituciones filtradas
21     return ResponseHelper::success(data: $instituciones, message: 'Instituciones obtenidas correctamente');
22 }
```

Anexo 174. Función `institucionesPorPais`.

Para actualizar una institución, el método `actualizarInstitucion` permite modificar su nombre y gestiona posibles errores.

```
1 reference | 0 overrides
public function actualizarInstitucion(Request $request, $id): JsonResponse|mixed
{
    // Validación de los datos entrantes
    $validator = Validator::make($request->all(), [
        'Nombre' => 'required|string|max:255',
    ]);

    // Si la validación falla, devolver errores
    if ($validator->fails()) {
        return ResponseHelper::error(message: $validator->errors()->first(), status: 400);
    }

    // Buscar la institución por el campo InstitucionID en lugar de 'id'
    $institucion = Instituciones::find(id: $id);

    if (!$institucion) {
        return ResponseHelper::error(message: 'Institución no encontrada', status: 404);
    }

    // Actualizar el nombre de la institución
    $institucion->Nombre = $request->input(key: 'Nombre');

    // Guardar los cambios en la base de datos
    if ($institucion->save()) {
        return ResponseHelper::success(data: [
            'institucion' => $institucion
        ], message: 'Nombre de la institución actualizado correctamente');
    } else {
        return ResponseHelper::error(message: 'Error al actualizar el nombre de la institución', status: 500);
    }
}
```

Anexo 175. Función `actualizarInstitución`.

Se creó la función `destroy` el cual elimina una institución por su ID asegurando que la operación se realice correctamente.


```

// Método para eliminar una institución
1 reference | 0 overrides
public function destroy($id): JsonResponse|mixed
{
    $institucion = Instituciones::findOrFail(id: $id);
    $institucion->delete();

    return ResponseHelper::success(data: null, message: 'Institución eliminada correctamente');
}

```

Anexo 176. Función destroy (institución).

Controlador carreras

En el controlador de carreras, se implementan métodos relacionados a si administración.

El método agregarCarrera valida los datos ingresados y crea una nueva carrera asociada a una institución, devolviendo una respuesta exitosa y el método editarCarrera permite modificar una carrera existente mediante validaciones y actualizaciones en la base de datos.

```

BackendCEP > app > Http > Controllers > CarrerasController.php > ...
3 namespace App\Http\Controllers;
4
5 use App\Models\Carreras;
6 use Illuminate\Http\Request;
7 use App\Helpers\ResponseHelper;
8
9 7 references | 0 implementations
class CarrerasController extends Controller
10 {
11
12     // Método para agregar una nueva carrera
13     1 reference | 0 overrides
    public function agregarCarrera(Request $request): JsonResponse|mixed
14     {
15         $validatedData = $request->validate(rules: [
16             'Nombre' => 'required|string|max:255',
17             'InstitucionID' => 'required|exists:instituciones,InstitucionID',
18         ]);
19         $carrera = Carreras::create(attributes: $validatedData);
20         return ResponseHelper::success(data: $carrera, message: 'Carrera creada correctamente', status: 201);
21     }
22
23     // Método para editar una carrera existente
24     1 reference | 0 overrides
    public function editarCarrera(Request $request, $id): JsonResponse|mixed
25     {
26         $carrera = Carreras::find(id: $id);
27         if (!$carrera) {
28             return ResponseHelper::error(message: 'Carrera no encontrada', status: 404);
29         }
30         $validatedData = $request->validate(rules: [
31             'Nombre' => 'sometimes|required|string|max:255',
32             'InstitucionID' => 'sometimes|required|exists:instituciones,InstitucionID',
33         ]);
34         $carrera->update($validatedData);
35         return ResponseHelper::success(data: $carrera, message: 'Carrera actualizada correctamente');
36     }

```

Anexo 177. Función agregarCarrera y editarCarrera.

Además, el método carreras obtiene todas las carreras disponibles, mientras que carrera busca una específica por su ID y maneja posibles errores en caso de no hallarla.

```

BackendCEP > app > Http > Controllers > CarrerasController.php > ...
9 class CarrerasController extends Controller
24 public function editarCarrera(Request $request, $id): JsonResponse|mixed
36 }
37
38 // Método para obtener todas las carreras
39 public function carreras(): JsonResponse|mixed
40 {
41     $carreras = Carreras::all();
42
43     // Devolver una respuesta exitosa con los datos
44     return ResponseHelper::success(data: $carreras, message: 'Carreras obtenidas correctamente');
45 }
46
47 // Método para obtener una carrera específica por ID
48 public function carrera($id): JsonResponse|mixed
49 {
50     $carrera = Carreras::find(id: $id);
51
52     if (!$carrera) {
53         // Devolver un error si no se encuentra la carrera
54         return ResponseHelper::error(message: 'Carrera no encontrada', status: 404);
55     }
56
57     // Devolver una respuesta exitosa con la carrera
58     return ResponseHelper::success(data: $carrera, message: 'Carrera obtenida correctamente');
59 }

```

Anexo 178. Funciones carreras y carrera.

Otros dos métodos que se crearon fueron el método `carrerasPorInstitucion`, que filtra las carreras en función de la institución a la que pertenecen, y el método `eliminar`, que se encarga de eliminar una carrera existente por su ID.

```

61 // Método para obtener carreras filtradas por institución
62 public function carrerasPorInstitucion(Request $request): JsonResponse|mixed
63 {
64     $InstitucionID = $request->query(key: 'InstitucionID');
65
66     // Validar si se proporciona el InstitucionID
67     if (!$InstitucionID) {
68         return ResponseHelper::error(message: 'InstitucionID es requerido', status: 400);
69     }
70
71     // Filtrar carreras por el ID de la institución
72     $carreras = Carreras::where(column: 'InstitucionID', operator: $InstitucionID)->get();
73
74     // Devolver una respuesta exitosa con las carreras filtradas
75     return ResponseHelper::success(data: $carreras, message: 'Carreras obtenidas correctamente');
76 }
77
78 // Eliminar una carrera
79 public function eliminar($id): JsonResponse|mixed
80 {
81     $carrera = Carreras::findOrFail(id: $id);
82     $carrera->delete();
83     return ResponseHelper::success(data: null, message: 'Carrera eliminada correctamente');
84 }
85
86
87 }

```

Anexo 179. Funciones carrerasPorInstitución y eliminar.

Controlador cuestionario

El controlador de cuestionarios incluye un método denominado `getPreguntasRespuestas`, que permite buscar un cuestionario por su ID junto con sus grupos, preguntas y respuestas asociadas. Si el cuestionario no se encuentra, retorna un error 404. El

método organiza las preguntas y respuestas en un formato JSON estructurado para facilitar su uso.

```
BackendCEP > app > Http > Controllers > CuestionarioController.php > ...
5 use Illuminate\Http\Request;
6 use App\Models\Pregunta;
7 use App\Models\CuestionarioPregunta;
8 use App\Models\Cuestionario;
9 use App\Models\Grupos;
10 use App\Models\Respuesta;
11 use App\Models\CuestionarioGrupo;
12
13 10 references | 0 implementations
14 class CuestionarioController extends Controller
15 {
16     1 reference | 0 overrides
17     public function getPreguntasRespuestas($cuestionarioID): JsonResponse|mixed
18     {
19         // Metodo para encontrar el cuestionario y sus preguntas, incluyendo las respuestas y las variables.
20         $cuestionario = Cuestionario::with(relations: ['grupo', 'preguntas.respuestas', 'preguntas.variables'])
21             ->where(column: 'CuestionarioID', operator: $cuestionarioID)
22             ->first();
23
24         if (!$cuestionario) {
25             return response()->json(['message' => 'Cuestionario no encontrado'], 404);
26         }
27
28         // Tomar solo las primeras 55 preguntas
29         $preguntas = $cuestionario->preguntas->take(55);
30
31         // Estructura de la respuesta para incluir el nombre, la descripción, las preguntas, las respuestas y las variables.
32         return response()->json([
33             'CuestionarioID' => $cuestionario->CuestionarioID,
34             'nombre' => $cuestionario->Nombre,
35             'descripcion' => $cuestionario->Descripcion,
36             'fechas' => $cuestionario->cuestionarioGrupos->map(function ($grupo): array {
37                 return [
38                     'GrupoNombre' => $grupo->grupo->Nombre ?? 'sin grupo',
39                     'FechaPublicacion' => $grupo->FechaPublicacion,
40                     'FechaCierre' => $grupo->FechaCierre,
41                 ];
42             });
43         ]);
44     }
45 }
```

Anexo 180. Funcion *getPreduntasRespuestas* parte 1.

```
BackendCEP > app > Http > Controllers > CuestionarioController.php > ...
13 class CuestionarioController extends Controller
14 {
15     public function getPreguntasRespuestas($cuestionarioID): JsonResponse|mixed
16     {
17         // Metodo para encontrar el cuestionario y sus preguntas, incluyendo las respuestas y las variables.
18         $cuestionario = Cuestionario::with(relations: ['grupo', 'preguntas.respuestas', 'preguntas.variables'])
19             ->where(column: 'CuestionarioID', operator: $cuestionarioID)
20             ->first();
21
22         if (!$cuestionario) {
23             return response()->json(['message' => 'Cuestionario no encontrado'], 404);
24         }
25
26         // Tomar solo las primeras 55 preguntas
27         $preguntas = $cuestionario->preguntas->take(55);
28
29         // Estructura de la respuesta para incluir el nombre, la descripción, las preguntas, las respuestas y las variables.
30         return response()->json([
31             'CuestionarioID' => $cuestionario->CuestionarioID,
32             'nombre' => $cuestionario->Nombre,
33             'descripcion' => $cuestionario->Descripcion,
34             'fechas' => $cuestionario->cuestionarioGrupos->map(function ($grupo): array {
35                 return [
36                     'GrupoNombre' => $grupo->grupo->Nombre ?? 'sin grupo',
37                     'FechaPublicacion' => $grupo->FechaPublicacion,
38                     'FechaCierre' => $grupo->FechaCierre,
39                 ];
40             });
41             'preguntas' => $preguntas->map(function ($pregunta): array {
42                 return [
43                     'PreguntaID' => $pregunta->PreguntaID,
44                     'Texto' => $pregunta->Texto,
45                     'Tipo' => $pregunta->Tipo,
46                     'variables' => $pregunta->variables->map(function ($variable): array {
47                         return [
48                             'VariableID' => $variable->VariableID,
49                             'Nombre' => $variable->Nombre,
50                         ];
51                     });
52                 ],
53             'respuestas' => $pregunta->respuestas->map(function ($respuesta): array {
54                 return [
55                     'RespuestaID' => $respuesta->RespuestaID,
56                     'Texto' => $respuesta->Texto,
57                     'Valor' => $respuesta->Valor,
58                 ];
59             });
60             ],
61         ]);
62     }
63 }
```

Anexo 181. Funcion *getPreduntasRespuestas* parte 2.

Para crear un cuestionario, se valida primero la entrada del usuario asegurándose de que datos como el nombre, descripción y preguntas sean correctos. A continuación, se limita el número de preguntas a 55 y se asocian con variables y respuestas preexistentes en la base de datos, devolviendo un mensaje de éxito y los detalles del cuestionario.

```

BackendCIP > app > Http > Controller > CuestionarioController.php > -
13 class CuestionarioController extends Controller
14 {
15     // Método de entrada
16     public function crearCuestionario(Request $request): JsonResponse|mixed
17     {
18         // Validar datos de entrada
19         $validatedData = $request->validate(rules: [
20             'nombre' => 'required|string|max:255',
21             'descripcion' => 'nullable|string',
22             'preguntas' => 'required|array',
23             'preguntas.*.texto' => 'required|string',
24             'preguntas.*.tipo' => 'required|string|in:positiva,negativa',
25             'preguntas.*.variables' => 'required|array',
26             'preguntas.*.variables.*' => 'exists:variables,variableID',
27         ]);
28
29         // Limitar las preguntas a las primeras 55
30         $preguntas = array_slice(array: $validatedData['preguntas'], offset: 0, length: 55);
31
32         // Crear el cuestionario
33         $cuestionario = Cuestionario::create(attributes: [
34             'nombre' => $validatedData['nombre'],
35             'descripcion' => $validatedData['descripcion'],
36         ]);
37
38         // Procesar preguntas y asociarlas al cuestionario
39         foreach ($preguntas as $preguntaData) {
40             // Crear una nueva pregunta
41             $nuevaPregunta = Pregunta::create(attributes: [
42                 'texto' => $preguntaData['texto'],
43                 'tipo' => $preguntaData['tipo'],
44             ]);
45
46             // Asociar las variables a la nueva pregunta
47             $nuevaPregunta->variables()->sync($preguntaData['variables']);
48
49             // Asociar las respuestas existentes a la nueva pregunta
50             $respuestasExistentes = Respuesta::where(column: 'PreguntaID', operator: 1)->get(); // ID de la pregunta de referencia
51         }
52     }
53 }

```

Anexo 182. Función crearCuestionario parte 1.

```

13 class CuestionarioController extends Controller
14 {
15     // Método de entrada
16     public function crearCuestionario(Request $request): JsonResponse|mixed
17     {
18         // Validar datos de entrada
19         $validatedData = $request->validate(rules: [
20             'nombre' => 'required|string|max:255',
21             'descripcion' => 'nullable|string',
22             'preguntas' => 'required|array',
23             'preguntas.*.texto' => 'required|string',
24             'preguntas.*.tipo' => 'required|string|in:positiva,negativa',
25             'preguntas.*.variables' => 'required|array',
26             'preguntas.*.variables.*' => 'exists:variables,variableID',
27         ]);
28
29         // Limitar las preguntas a las primeras 55
30         $preguntas = array_slice(array: $validatedData['preguntas'], offset: 0, length: 55);
31
32         // Crear el cuestionario
33         $cuestionario = Cuestionario::create(attributes: [
34             'nombre' => $validatedData['nombre'],
35             'descripcion' => $validatedData['descripcion'],
36         ]);
37
38         // Procesar preguntas y asociarlas al cuestionario
39         foreach ($preguntas as $preguntaData) {
40             // Crear una nueva pregunta
41             $nuevaPregunta = Pregunta::create(attributes: [
42                 'texto' => $preguntaData['texto'],
43                 'tipo' => $preguntaData['tipo'],
44             ]);
45
46             // Asociar las variables a la nueva pregunta
47             $nuevaPregunta->variables()->sync($preguntaData['variables']);
48
49             // Asociar las respuestas existentes a la nueva pregunta
50             $respuestasExistentes = Respuesta::where(column: 'PreguntaID', operator: 1)->get(); // ID de la pregunta de referencia
51             foreach ($respuestasExistentes as $respuesta) {
52                 $respuesta->create(attributes: [
53                     'texto' => $respuesta->texto,
54                     'valor' => $respuesta->valor,
55                     'PreguntaID' => $nuevaPregunta->PreguntaID,
56                 ]);
57             }
58
59             // Asociar la nueva pregunta al cuestionario
60             $cuestionarioPregunta::create(attributes: [
61                 'CuestionarioID' => $cuestionario->CuestionarioID,
62                 'PreguntaID' => $nuevaPregunta->PreguntaID,
63             ]);
64         }
65
66         return response()->json([
67             'message' => 'Cuestionario creado exitosamente',
68             'cuestionario' => $cuestionario,
69         ], 201);
70     }
71 }

```

Anexo 183. Función crearCuestionario parte 2.

Se creo la función getQuizVersions para obtener versiones de los cuestionarios junto con sus preguntas y respuestas, además de los grupos asociados. Cada cuestionario incluye detalles como nombre, descripción y fechas de publicación y cierre, y la información se estructura en formato JSON. El método también permite publicar un cuestionario asignándolo a un grupo específico con fechas de publicación y cierre, validando los datos de entrada y estableciendo relaciones entre cuestionarios y grupos en caso de éxito.

```

BackendCEP > app > Http > Controllers > CuestionarioController.php > ...
13 class CuestionarioController extends Controller
125 public function getQuizVersions(): JsonResponse|mixed
126 {
127     $cuestionarios = Cuestionario::with(relations: ['preguntas.respuestas', 'cuestionarioGrupos.grupo'])
128     ->get();
129
130     return response()->json([
131         'cuestionarios' => $cuestionarios->map(callback: function (Cuestionario $cuestionario): array {
132             return [
133                 'cuestionarioID' => $cuestionario->CuestionarioID,
134                 'nombre' => $cuestionario->Nombre,
135                 'descripcion' => $cuestionario->Descripcion,
136                 'fechas' => $cuestionario->cuestionarioGrupos->map(function ($grupo): array {
137                     return [
138                         'GrupoNombre' => $grupo->grupo->Nombre ?? 'sin grupo', // Si el grupo no existe
139                         'FechaPublicacion' => $grupo->FechaPublicacion,
140                         'FechaCierre' => $grupo->FechaCierre,
141                     ];
142                 })),
143                 'preguntas' => $cuestionario->preguntas->map(function ($pregunta): array {
144                     return [
145                         'PreguntaID' => $pregunta->PreguntaID,
146                         'Texto' => $pregunta->Texto,
147                         'Tipo' => $pregunta->Tipo,
148                         'respuestas' => $pregunta->respuestas->map(function ($respuesta): array {
149                             return [
150                                 'RespuestaID' => $respuesta->RespuestaID,
151                                 'Texto' => $respuesta->Texto,
152                                 'Valor' => $respuesta->Valor,
153                             ];
154                         })),
155                     ];
156                 })),
157             ];
158         })),
159     ]);

```

Anexo 184. Función getQuizVersions.

```

BackendCEP > app > Http > Controllers > CuestionarioController.php > ...
13 class CuestionarioController extends Controller
164 //Metodo para publicar un cuestionario
165 public function publicarCuestionario($cuestionarioID, Request $request): JsonResponse|mixed
166 {
167     // Validar entrada
168     $validatedData = $request->validate(rules: [
169         'grupo_id' => 'required|integer|exists:grupos,GrupoID',
170         'fecha_publicacion' => 'required|date',
171         'fecha_cierre' => 'required|date|after_or_equal:fecha_publicacion',
172     ]);
173
174     // Buscar cuestionario y grupo
175     $cuestionario = Cuestionario::find(id: $cuestionarioID);
176     $grupo = Grupos::find(id: $validatedData['grupo_id']);
177
178     if (!$cuestionario || !$grupo) {
179         return response()->json(['message' => 'Cuestionario o grupo no encontrado', 404]);
180     }
181
182     // Asignar el cuestionario al grupo con fechas de publicación y cierre
183     CuestionarioGrupo::create(attributes: [
184         'CuestionarioID' => $cuestionarioID,
185         'GrupoID' => $grupo->GrupoID,
186         'FechaPublicacion' => $validatedData['fecha_publicacion'],
187         'FechaCierre' => $validatedData['fecha_cierre'],
188     ]);
189
190     return response()->json([
191         'message' => 'Cuestionario publicado exitosamente en el grupo especificado',
192         'cuestionario' => $cuestionario,
193         'grupo' => $grupo,
194     ]);
195 }

```

Anexo 185. Método para publicar cuestionario.

Se creó la función obtenerCuestionariosPorUsuario para la obtención de cuestionarios aplicados a los grupos creados por un maestro, recopilando información como el nombre, descripción, fechas y datos de los grupos y carreras relacionadas.

```

BackendCEP > app > Http > Controllers > CuestionarioController.php > ...
13 class CuestionarioController extends Controller
198 // Método para obtener los cuestionarios aplicados para el maestro
199 public function obtenerCuestionariosPorUsuario($usuarioId): JsonResponse|mixed
200 {
201     try {
202         // Obtener los grupos asociados al maestro (grupos creados por el maestro)
203         $gruposMaestro = Grupos::where(column: 'UsuarioID', operator: $usuarioId) // Buscar los grupos creados por el maestro
204         ->pluck(column: 'GrupoID'); // Obtener solo los IDs de los grupos
205
206         // Obtener los cuestionarios asignados a esos grupos junto con sus relaciones
207         $cuestionarios = CuestionarioGrupo::whereIn(column: 'GrupoID', values: $gruposMaestro) // Buscar en la tabla intermedia 'cuestionario_grupo'
208         ->with(relations: ['cuestionario', 'grupo.carrera']) // Cargar las relaciones necesarias
209         ->get();
210
211         // Mapear los resultados de los cuestionarios
212         $resultados = $cuestionarios->map(callback: function (CuestionarioGrupo $cuestionarioGrupo): array {
213             // Formatear la información de cada cuestionario y del grupo al que pertenece
214             return [
215                 'CuestionarioID' => $cuestionarioGrupo->cuestionario->CuestionarioID, // Obtener el ID del cuestionario
216                 'Nombre' => $cuestionarioGrupo->cuestionario->Nombre, // Obtener el nombre del cuestionario
217                 'Descripcion' => $cuestionarioGrupo->cuestionario->Descripcion, // Obtener la descripción del cuestionario
218                 'FechaPublicacion' => $cuestionarioGrupo->FechaPublicacion, // Fecha de publicación desde cuestionario_grupo
219                 'FechaCierre' => $cuestionarioGrupo->FechaCierre, // Fecha de cierre desde cuestionario_grupo
220                 // Datos del grupo
221                 'GrupoID' => $cuestionarioGrupo->grupo->GrupoID, // ID del grupo
222                 'GrupoNombre' => $cuestionarioGrupo->grupo->Nombre, // Nombre del grupo
223                 'GrupoSemestre' => $cuestionarioGrupo->grupo->Semestre, // Semestre del grupo
224                 'GrupoCarreraID' => $cuestionarioGrupo->grupo->CarreraID, // Carrera del grupo
225                 // Acceder al nombre de la carrera a través de la relación
226                 'CarreraNombre' => $cuestionarioGrupo->grupo->carrera->Nombre, // Nombre de la carrera asociada al grupo
227             ];
228         });
229

```

Anexo 186. Función obtenerCuestionariosPorUsuario parte 1.

```

BackendCEP > app > Http > Controllers > CuestionarioController.php > ...
13 class CuestionarioController extends Controller
199 public function obtenerCuestionariosPorUsuario($usuarioId): JsonResponse|mixed
225         // Acceder al nombre de la carrera a través de la relación
226         'CarreraNombre' => $cuestionarioGrupo->grupo->carrera->Nombre, // Nombre de la carrera asociada al grupo
227     ];
228 });
229
230 // Responder con los cuestionarios encontrados
231 return response()->json([
232     'message' => 'Cuestionarios obtenidos para los grupos del usuario',
233     'cuestionarios' => $resultados,
234 ]);
235
236 } catch (\Exception $e) {
237     // Manejo de error en caso de que la consulta falle
238     return response()->json([
239         'message' => 'Error al obtener los cuestionarios',
240         'error' => $e->getMessage(),
241     ], 500);
242 }
243

```

Anexo 187. Función obtenerCuestionariosPorUsuario parte 2.

De igual manera se crearon funciones para obtener información detallada de un cuestionario específico por su ID, verificando su existencia en la base de datos y devolviendo los detalles en formato JSON o un error 404 si no se encuentra. Finalmente, es posible duplicar y editar cuestionarios existentes, creando nuevas versiones basadas en las modificaciones ingresadas por el usuario. La operación incluye la validación de datos, la

creación de una nueva versión con las preguntas asociadas, y la respuesta con un mensaje confirmando la creación de la nueva versión junto con los detalles correspondientes.

```
//Obtener la informacion de un solo cuestionario
1 reference | 0 overrides
public function getCuestionario($cuestionarioId): JsonResponse|mixed
{
    try {
        $cuestionario = Cuestionario::find(id: $cuestionarioId);

        if (!$cuestionario) {
            return response()->json([
                'success' => false,
                'message' => 'Cuestionario no encontrado.',
            ], 404);
        }

        return response()->json([
            'success' => true,
            'data' => $cuestionario,
        ], 200);
    } catch (\Exception $e) {
        return response()->json([
            'success' => false,
            'message' => 'Error al obtener el cuestionario.',
            'error' => $e->getMessage(),
        ], 500);
    }
}
```

Anexo 188. Función `getCuestionario`.

```
BackendCEP > app > Http > Controllers > CuestionarioController.php > ...
13 class CuestionarioController extends Controller
2 references | 0 overrides
273 public function duplicarYeditarCuestionario($cuestionarioID, Request $request): JsonResponse|mixed
274 {
275     // Usar el ID 1 como predeterminado si no se proporciona uno válido
276     if (!$cuestionarioID || $cuestionarioID <= 0) {
277         $cuestionarioID = 1;
278     }
279
280     $cuestionarioOriginal = Cuestionario::with(relations: ['preguntas.respuestas'])->find(id: $cuestionarioID);
281
282     if (!$cuestionarioOriginal) {
283         return response()->json(['message' => 'Cuestionario no encontrado'], 404);
284     }
285
286     // Validar los datos de entrada
287     $validatedData = $request->validate(rules: [
288         'nombre' => 'required|string|max:255',
289         'descripcion' => 'nullable|string',
290         'fecha_publicacion' => 'required|date',
291         'fecha_cierre' => 'required|date|after_or_equal:fecha_publicacion',
292         'preguntas' => 'required|array',
293         'preguntas.*.PreguntaID' => 'exists:preguntas,PreguntaID',
294         'preguntas.*.Texto' => 'required|string',
295         'preguntas.*.Tipo' => 'required|string|in:Positiva,Negativa'
296     ]);
297
298     // Crear la nueva versión del cuestionario
299     $nuevaVersion = Cuestionario::create(attributes: [
300         'GrupoID' => $cuestionarioOriginal->GrupoID,
301         'Nombre' => $validatedData['nombre'],
302         'Descripcion' => $validatedData['descripcion'],
303         'FechaPublicacion' => $validatedData['fecha_publicacion'],
304         'FechaCierre' => $validatedData['fecha_cierre'],
305     ]);
```

Anexo 189. Función `duplicarYeditarCuestionario` parte 1.

```

// Asociar preguntas a la nueva versión
foreach ($validatedData['preguntas'] as $preguntaData) {
    $pregunta = Pregunta::find(id: $preguntaData['PreguntaID']);
    if ($pregunta) {
        // Crear una nueva asociación con la pregunta editada
        CuestionarioPregunta::create(attributes: [
            'CuestionarioID' => $nuevaVersion->CuestionarioID,
            'PreguntaID' => $pregunta->PreguntaID,
        ]);

        // Actualizar el texto y el tipo de la pregunta para la nueva versión
        $pregunta->Texto = $preguntaData['Texto'];
        $pregunta->Tipo = $preguntaData['Tipo'];
        $pregunta->save();
    }
}

return response()->json([
    'message' => 'Nueva versión del cuestionario creada exitosamente',
    'cuestionario_original' => $cuestionarioOriginal,
    'nueva_version' => $nuevaVersion,
]);
}
}

```

Anexo 190. Funcion duplicarYEditarCuestionario parte 2.

Controlador Cuestionario resuelto.

El controlador para la gestión de cuestionarios resueltos permite manejar la funcionalidad de marcar un cuestionario como completado por un alumno y recuperar los cuestionarios resueltos asociados a él. Al marcar un cuestionario como resuelto, se registra la relación entre el cuestionario y el alumno, incluyendo detalles como su estado. Cuando se solicita la lista de cuestionarios resueltos por un alumno en un grupo específico, el sistema obtiene estos registros junto con información del cuestionario asociado, asegurando que las relaciones sean consistentes y devolviendo la información estructurada con los detalles necesarios.

```
BackendCIP > app > Http > Controllers > CuestionarioResueltoController.php > ...
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use App\Models\CuestionarioResuelto;
8 use App\Helpers\ResponseHelper;
9 use App\Models\Cuestionario;
10 use App\Models\CuestionarioGrupo;
11 class CuestionarioResueltoController extends Controller
12 {
13     //Método para marcar un cuestionario como resuelto
14     public function marcarComoResuelto(Request $request): JsonResponse|mixed
15     {
16         $resuelto = CuestionarioResuelto::create(attributes: [
17             'UsuarioID' => $request->UsuarioID,
18             'CuestionarioID' => $request->CuestionarioID,
19             'GrupoID' => $request->GrupoID,
20             'FechaResuelto' => now(),
21             'Resuelto' => true,
22         ]);
23         return ResponseHelper::success(data: $resuelto, message: 'Cuestionario marcado como resuelto');
24     }
25
26     // Método para obtener cuestionarios resueltos
27     public function obtenerCuestionariosResueltos($usuarioId, $grupoId): JsonResponse|mixed
28     {
29         // Obtener cuestionarios resueltos con la relación cuestionario
30         $cuestionariosResueltos = CuestionarioResuelto::with('cuestionario')
31             ->where('column': 'UsuarioID', operator: $usuarioId)
32             ->where('column': 'GrupoID', operator: $grupoId)
33             ->where('column': 'Resuelto', operator: true)
34             ->get();
35
36         // Mapear los resultados con validación de relaciones
37         $resultados = $cuestionariosResueltos->map(callback: function (CuestionarioResuelto $resuelto): array {
```

Anexo 191. Función marcarComoResuelto.

Este método recupera todos los cuestionarios resueltos por un alumno en distintos grupos, incluyendo detalles del cuestionario y del grupo. Para cada cuestionario resuelto, se extrae información como el nombre, descripción del cuestionario, y fechas de publicación y cierre, si están disponibles.

Este controlador permite recuperar todos los cuestionarios resueltos por un alumno en distintos grupos, mostrando datos como el nombre y descripción del cuestionario, además de las fechas de publicación y cierre si están disponibles. Para garantizar la consistencia, cualquier relación nula se reemplaza con un valor predeterminado, devolviendo los resultados mapeados en un formato claro y organizado. También es posible obtener cuestionarios pendientes de resolución para un alumno en un grupo específico, filtrando los cuestionarios ya resueltos y devolviendo los restantes con detalles como nombre, descripción y fechas.

```

BackendCEP > app > Http > Controllers > CuestionarioResueltoController.php > ...
11 class CuestionarioResueltoController extends Controller
28 public function obtenerCuestionariosResueltos($usuarioId, $grupoId): JsonResponse|mixed
29 {
30     // Mapear los resultados con validación de relaciones
31     $resultados = $cuestionariosResueltos->map(callback: function (CuestionarioResuelto $resuelto): array {
32         $cuestionario = $resuelto->cuestionario; // Relación con cuestionario
33
34         return [
35             'CuestionarioResueltoID' => $resuelto->CuestionarioResueltoID,
36             'UsuarioID' => $resuelto->UsuarioID,
37             'CuestionarioID' => $resuelto->CuestionarioID,
38             'GrupoID' => $resuelto->GrupoID,
39             'FechaResuelto' => $resuelto->FechaResuelto,
40             'Resuelto' => $resuelto->Resuelto,
41             'Nombre' => $cuestionario->Nombre ?? 'Cuestionario no disponible', // Validar relación nula
42         ];
43     });
44
45     return ResponseHelper::success(data: $resultados, message: 'Cuestionarios resueltos obtenidos correctamente');
46 }
47
48 // Método para obtener cuestionarios resueltos por alumno
49 1 reference | 0 overrides
50 public function obtenerCuestionariosResueltosPorAlumno($usuarioId): JsonResponse|mixed
51 {
52     // Obtener todos los cuestionarios resueltos del alumno en diferentes grupos
53     $cuestionariosResueltos = CuestionarioResuelto::with(relations: ['cuestionario.cuestionarioGrupos', 'grupo'])
54         ->where(column: 'UsuarioID', operator: $usuarioId)
55         ->where(column: 'Resuelto', operator: true)
56         ->get();
57
58     // Mapear los datos para incluir la información del cuestionario, grupo y las fechas
59     $resultados = $cuestionariosResueltos->map(callback: function (CuestionarioResuelto $resuelto): array {
60         $cuestionario = $resuelto->cuestionario; // Relación con cuestionario
61         $grupo = $resuelto->grupo; // Relación con grupo

```

Anexo 192. Función obtenerCuestionariosResueltosPorAlumno pate 1.

```

BackendCEP > app > Http > Controllers > CuestionarioResueltoController.php > PHP > CuestionarioResueltoController > obtenerCuestionariosResueltos
11 class CuestionarioResueltoController extends Controller
57 public function obtenerCuestionariosResueltosPorAlumno($usuarioId): JsonResponse|mixed
58 {
59     // Obtener cuestionario_grupo para las fechas, si existe
60     $cuestionarioGrupo = $cuestionario->cuestionarioGrupos->firstWhere('GrupoID', $resuelto->GrupoID);
61
62     return [
63         'CuestionarioResueltoID' => $resuelto->CuestionarioResueltoID,
64         'CuestionarioID' => $resuelto->CuestionarioID,
65         'GrupoID' => $resuelto->GrupoID,
66         'FechaResuelto' => $resuelto->FechaResuelto,
67         'Resuelto' => $resuelto->Resuelto,
68         'NombreCuestionario' => $cuestionario->Nombre ?? 'Cuestionario no disponible',
69         'DescripcionCuestionario' => $cuestionario->Descripcion ?? 'Sin descripción',
70         'NombreGrupo' => $grupo->Nombre ?? 'Grupo no disponible',
71         'FechaPublicacion' => $cuestionarioGrupo->FechaPublicacion ?? 'Fecha no asignada',
72         'FechaCierre' => $cuestionarioGrupo->FechaCierre ?? 'Fecha no asignada',
73     ];
74
75     return ResponseHelper::success(data: $resultados, message: 'Cuestionarios resueltos obtenidos correctamente');
76 }
77
78 }
79

```

Anexo 193. Función obtenerCuestionariosResueltosPorAlumno pate 2.

```

89
90 // Obtener los cuestionarios pendientes del alumno
91 1 reference | 0 overrides
92 public function obtenerCuestionariosPendientes($usuarioId, $grupoId): JsonResponse|mixed
93 {
94     // Obtener los ID de los cuestionarios que ya han sido resueltos por el alumno en el grupo
95     $cuestionariosResueltosIds = CuestionarioResuelto::where(column: 'UsuarioID', operator: $usuarioId)
96         ->where(column: 'GrupoID', operator: $grupoId)
97         ->pluck(column: 'CuestionarioID');
98
99     // Obtener los cuestionarios aplicados al grupo desde 'cuestionario_grupo'
100     $cuestionariosPendientes = CuestionarioGrupo::where(column: 'GrupoID', operator: $grupoId)
101         ->whereNotIn(column: 'CuestionarioID', values: $cuestionariosResueltosIds)
102         ->with(relations: 'cuestionario') // Relación con el modelo cuestionario
103         ->get();

```

Anexo 194. Función ObtenerCuestionariosPendientes

Esta Función facilita la obtención de cuestionarios asignados a un grupo, cargando las relaciones correspondientes y estructurando los datos para incluir información del cuestionario y del grupo. Estos datos son devueltos con un mensaje que asegura su correcta recuperación.

```

Backend/CEP > app > http > Controllers > CuestionarioResueltoController.php > PHP > CuestionarioResueltoController > obtenerCuestionariosResueltosPorAlumno()
11 class CuestionarioResueltoController extends Controller
92 public function obtenerCuestionariosPendientes($usuarioId, $grupoId): JsonResponse{mixed
103     ->get()
104     ->map(callback: function (CuestionarioGrupo $cuestionarioGrupo): array {
105         return [
106             'CuestionarioID' => $cuestionarioGrupo->CuestionarioID,
107             'Nombre' => $cuestionarioGrupo->Cuestionario->Nombre,
108             'Descripcion' => $cuestionarioGrupo->Cuestionario->Descripcion,
109             'FechaPublicacion' => $cuestionarioGrupo->FechaPublicacion,
110             'FechaCierre' => $cuestionarioGrupo->FechaCierre,
111             'GrupoID' => $cuestionarioGrupo->GrupoID,
112         ];
113     });
114
115     return ResponseHelper::success(data: $cuestionariosPendientes, message: 'Cuestionarios pendientes obtenidos correctamente');
116 }
117
118
119 T reference [0 overrides]
119 public function obtenerCuestionariosAplicadosPorGrupo($grupoId): JsonResponse{mixed
120 {
121     // Obtener los cuestionarios que han sido asignados al grupo a través de la tabla 'cuestionario_grupo'
122     $cuestionarios = CuestionarioGrupo::where(column: 'GrupoID', operator: $grupoId)
123         ->with(relations: ['cuestionario', 'grupo']) // Cargar relaciones con cuestionario y grupo
124         ->get();
125
126     $resultados = $cuestionarios->map(callback: function (CuestionarioGrupo $cuestionarioGrupo): array {
127         return [
128             'CuestionarioID' => $cuestionarioGrupo->Cuestionario->CuestionarioID,
129             'Nombre' => $cuestionarioGrupo->Cuestionario->Nombre,
130             'Descripcion' => $cuestionarioGrupo->Cuestionario->Descripcion,
131             'GrupoNombre' => $cuestionarioGrupo->grupo->Nombre ?? 'Sin nombre',
132             'FechaPublicacion' => $cuestionarioGrupo->FechaPublicacion,
133             'FechaCierre' => $cuestionarioGrupo->FechaCierre,
134         ];
135     });
136 }

```

Anexo 195. Función cuestionariosAplicadosPorGrupo.

Controlador generaciones

En el controlador de generaciones, se gestionan operaciones para agregar y editar generaciones, validando datos como el nombre y el ID de la carrera asociada. Si los datos son válidos, se crea o actualiza la generación, devolviendo una respuesta con la información generada. Este controlador también permite recuperar todas las generaciones registradas, buscar una generación específica por su ID, o eliminar una generación de la base de datos, asegurando respuestas claras para cada operación.

```
BackendCEP > app > Http > Controllers > GeneracionController.php > ...
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use App\Models\Generaciones;
8 use App\Helpers\ResponseHelper;
9 class GeneracionController extends Controller
10 {
11     //Metodo para agregar una nueva generación
12     public function agregarGeneracion(Request $request): JsonResponse|mixed
13     {
14         $validatedData = $request->validate(rules: [
15             'Nombre' => 'required|string|max:255',
16             'CarreraID' => 'required|exists:carreras,CarreraID',
17         ]);
18         $generacion = Generaciones::create(attributes: $validatedData);
19         return ResponseHelper::success(data: $generacion, message: 'Generación creada correctamente', status: 201);
20     }
21
22     // Editar una generación existente
23     public function editarGeneracion(Request $request, $GeneracionID): JsonResponse|mixed
24     {
25         $generacion = Generaciones::find(id: $GeneracionID);
26         if (!$generacion) {
27             return ResponseHelper::error(message: 'Generación no encontrada', status: 404);
28         }
29         $validatedData = $request->validate(rules: [
30             'Nombre' => 'sometimes|required|string|max:255',
31             'CarreraID' => 'sometimes|required|exists:carreras,CarreraID',
32         ]);
33         $generacion->update($validatedData);
34         return ResponseHelper::success(data: $generacion, message: 'Generación actualizada correctamente');
35     }
36 }
```

Anexo 196. Funciones agregarGeneración y editarGeneración.

```
BackendCEP > app > Http > Controllers > GeneracionController.php > ...
9 class GeneracionController extends Controller
30
37 // Obtener todas las generaciones
38 public function generaciones(): JsonResponse|mixed
39 {
40     $generaciones = Generaciones::all();
41     return ResponseHelper::success(data: $generaciones, message: 'Generaciones obtenidas correctamente');
42 }
43
44 // Obtener una generación específica por ID
45 public function generacion($GeneracionID): JsonResponse|mixed
46 {
47     $generacion = Generaciones::find(id: $GeneracionID);
48     if (!$generacion) {
49         return ResponseHelper::error(message: 'Generación no encontrada', status: 404);
50     }
51     return ResponseHelper::success(data: $generacion, message: 'Generación obtenida correctamente');
52 }
53
54 // Eliminar una generación
55 public function eliminarGeneracion($GeneracionID): JsonResponse|mixed
56 {
57     $generacion = Generaciones::findOrFail(id: $GeneracionID);
58     $generacion->delete();
59     return ResponseHelper::success(data: null, message: 'Generación eliminada correctamente');
60 }
61 }
```

Anexo 197. Funciones generaciones y generación (búsqueda por id) y eliminarGeneración.

Controlador grupo

El controlador de grupos maneja funcionalidades relacionadas con los grupos de alumnos. Permite obtener los grupos asociados a un alumno, cargando detalles sobre la carrera e institución. También incluye métodos para buscar un grupo utilizando un código de acceso, agregar alumnos a un grupo con validaciones estrictas, y crear grupos para maestros. Al crear un grupo, se valida la información proporcionada y se genera un código único de acceso, devolviendo detalles del grupo creado.

```
BackendCEP > app > Http > Controllers > GrupoController.php > ...
5 use App\Models\UsuarioGrupo;
6 use App\Models\Grupos;
7 use App\Models\Usuario;
8 use Illuminate\Http\Request;
9 use App\Helpers\ResponseHelper;
10
11 11 references | 0 implementations
12 class GrupoController extends Controller
13 {
14     1 reference | 0 overrides
15     public function getAlumnoGrupos($id): JsonResponse|mixed
16     {
17         // Se busca al alumno y carga sus grupos, carrera e institución
18         $alumno = Usuario::with(relations: ['grupos.carrera.institucion'])
19             ->find(id: $id);
20
21         if (!$alumno || $alumno->grupos->isEmpty()) {
22             return ResponseHelper::error(message: 'El alumno no tiene grupos asignados o no existe', status: 404);
23         }
24
25         return ResponseHelper::success(data: $alumno->grupos);
26     }
27
28     // Método para obtener un grupo por código de acceso
29     1 reference | 0 overrides
30     public function getGrupoByCodigo(Request $request): JsonResponse|mixed
31     {
32         $codigo = $request->input(key: 'codigo_acceso');
33
34         $grupo = Grupos::where(column: 'codigo_acceso', operator: $codigo)->with(relations: 'carrera.institucion')->first();
35
36         if (!$grupo) {
37             return ResponseHelper::error(message: 'Grupo no encontrado', status: 404);
38         }
39
40         return ResponseHelper::success(data: $grupo, message: 'Grupo encontrado');
```

Anexo 198. Funciones *getAlumnosGrupos* y *getGrupoByCodigo*

Este controlador asegura que los maestros solo puedan acceder a sus propios grupos y estudiantes, filtrando la información relevante. También es posible obtener información completa de un grupo, incluyendo sus relaciones con la carrera e institución asociadas. Para los grupos ya existentes, se puede consultar su lista de integrantes con detalles completos como matrícula, semestre y datos de carrera e institución, devolviendo los resultados en formato estructurado. Finalmente, se permite buscar alumnos por nombre dentro de un grupo y agregar nuevos estudiantes validando su existencia y las relaciones asociadas. Este sistema también facilita la recuperación de todos los grupos registrados, devolviendo datos completos en caso de éxito o indicando la ausencia de información si no se encuentran registros.

```

BackendCEP > app > Http > Controllers > GrupoController.php > ...
11 class GrupoController extends Controller
...
41 // Método para agregar un alumno a un grupo
42 public function joinGrupo(Request $request): JsonResponse|mixed
43 {
44     $validated = $request->validate(rules: [
45         'usuarioID' => 'required|exists:usuarios,usuarioID',
46         'codigo_acceso' => 'required|string|exists:grupos,codigo_acceso'
47     ]);
48
49     $grupo = Grupos::where(column: 'codigo_acceso', operator: $validated['codigo_acceso'])->first();
50
51     if (!$grupo) {
52         return ResponseHelper::error(message: 'Grupo no encontrado', status: 404);
53     }
54
55     // Se verifica si el alumno ya está en el grupo
56     $exists = UsuarioGrupo::where(column: 'UsuarioID', operator: $validated['usuarioID'])
57         ->where(column: 'GrupoID', operator: $grupo->GrupoID)
58         ->exists();
59
60     if ($exists) {
61         return ResponseHelper::error(message: 'El alumno ya pertenece a este grupo', status: 400);
62     }
63
64     // Añadimos al alumno al grupo
65     try {
66         UsuarioGrupo::create(attributes: [
67             'UsuarioID' => $validated['usuarioID'],
68             'GrupoID' => $grupo->GrupoID,
69             'CarreraID' => $grupo->carrera->CarreraID,
70             'InstitucionID' => $grupo->carrera->institucion->InstitucionID
71         ]);
72     } catch (\Exception $e) {
73         \Log::error('Error al agregar alumno al grupo: ' . $e->getMessage());
74         return ResponseHelper::error(message: 'Error al añadir alumno al grupo', status: 500);
75     }

```

Anexo 199. Función joinGrupo.

```

BackendCEP > app > Http > Controllers > GrupoController.php > ...
11 class GrupoController extends Controller
12 public function joinGrupo(Request $request): JsonResponse|mixed
...
75 }
76
77 return ResponseHelper::success(data: 'Alumno añadido al grupo exitosamente');
78 }
79
80 //Metodo de maestros para crear grupos
81 public function crearGrupo(Request $request): JsonResponse|mixed
82 {
83     // Validar los datos recibidos
84     $validated = $request->validate(rules: [
85         'nombre' => 'required|string|max:255',
86         'semestre' => 'required|string|max:10',
87         'generacion_id' => 'required|integer',
88         'carrera_id' => 'required|integer',
89         'usuario_id' => 'required|integer',
90     ]);
91
92     // Crear un nuevo grupo
93     $grupo = Grupos::create(attributes: [
94         'Nombre' => $validated['nombre'],
95         'Semestre' => $validated['semestre'],
96         'GeneracionID' => $validated['generacion_id'],
97         'CarreraID' => $validated['carrera_id'],
98         'UsuarioID' => $validated['usuario_id'],
99         // 'codigo_acceso' se genera automáticamente en el modelo
100     ]);
101
102     return ResponseHelper::success(data: $grupo, message: 'Grupo creado exitosamente. Código de acceso: ' . $grupo->codigo_acceso);
103 }
104

```

Anexo 200. Función crearGrupo.

```
BackendCEP > app > Http > Controllers > GrupoController.php > ...
11 class GrupoController extends Controller
1 reference | 0 overrides
105 public function obtenerGruposDeMaestro($usuarioId): JsonResponse|mixed
106 {
107     // Verifica si el usuario existe y es un maestro
108     $usuario = Usuario::find(id: $usuarioId);
109
110     if (!$usuario) {
111         return response()->json(['error' => 'Usuario no encontrado'], 404);
112     }
113
114     if (strcasecmp(string1: $usuario->Tipo, string2: 'Maestro') != 0) {
115         return response()->json(['error' => 'El usuario no es un maestro'], 403);
116     }
117
118     // Obtener los grupos donde el usuario es el creador
119     $grupos = Grupos::where(column: 'UsuarioID', operator: $usuarioId) // Verifica que se esté buscando por UsuarioID
120     ->with(relations: ['carrera.institucion', 'usuarios' => function ($query): void {
121         $query->where('Tipo', 'Alumno'); // Filtrar solo a los alumnos
122     }])
123     ->get();
124
125     if ($grupos->isEmpty()) {
126         return response()->json(['message' => 'El maestro no tiene grupos creados.'], 404);
127     }
128
129     return response()->json($grupos);
130 }
```

Anexo 201. Función obtenerGruposDeMaestro.

Función show sirve para obtener la información de un grupo en específico haciendo la búsqueda por el ID.

```
BackendCEP > app > Http > Controllers > GrupoController.php > ...
11 class GrupoController extends Controller
1 reference | 0 overrides
133 public function show($id): JsonResponse|mixed{
134     // Cargar el grupo junto con carrera e institución
135     $grupo = Grupos::with(relations: 'carrera.institucion')->find(id: $id);
136
137     if (!$grupo) {
138         return response()->json(['message' => 'Grupo no encontrado'], 404);
139     }
140
141     return response()->json(['data' => $grupo]);
142
143
144 }
```

Anexo 202. Función show.

```

BackendCEP > app > Http > Controllers > GrupoController.php > PHP > GrupoController > getIntegrantes()
11 class GrupoController extends Controller
145 //Metodo para obtener los integrantes del grupo
146 public function getIntegrantes($id): JsonResponse|mixed
147 {
148     // Obtener el grupo junto con los usuarios, carrera e institución
149     $grupo = Grupos::with(relations: ['usuarios.carrera.institucion'])->find(id: $id);
150
151     if (!$grupo) {
152         return response()->json(['message' => 'Grupo no encontrado'], 404);
153     }
154
155     // Mapear los integrantes del grupo para incluir la información de carrera e institución
156     $integrantes = $grupo->usuarios->map(function ($usuario): array {
157         return [
158             'UsuarioID' => $usuario->UsuarioID,
159             'Nombre' => $usuario->Nombre,
160             'Primer_apellido' => $usuario->Primer_apellido,
161             'Segundo_apellido' => $usuario->Segundo_apellido,
162             'Matricula' => $usuario->Matricula,
163             'Semestre' => $usuario->Semestre,
164             'Tipo' => $usuario->Tipo,
165             'Correo' => $usuario->Correo,
166             'Edad' => $usuario->Edad,
167             'Genero' => $usuario->Genero,
168             'Carrera' => $usuario->carrera ? [
169                 'CarreraID' => $usuario->carrera->CarreraID,
170                 'Nombre' => $usuario->carrera->Nombre,
171                 'Institucion' => $usuario->carrera->institucion ? [
172                     'InstitucionID' => $usuario->carrera->institucion->InstitucionID,
173                     'Nombre' => $usuario->carrera->institucion->Nombre
174                 ] : null
175             ] : null,
176         ];
177     });
178     return response()->json(['data' => $integrantes]);
179 }

```

Anexo 203. Función *getIntegrantes*.

```

BackendCEP > app > Http > Controllers > GrupoController.php > PHP > GrupoController > getIntegrantes()
11 class GrupoController extends Controller
180
181 // Método para buscar a un alumno por su nombre
182 public function buscarAlumnoPorNombre(Request $request): JsonResponse|mixed
183 {
184     $nombre = $request->input(key: 'nombre'); // Nombre, apellidos o nombre completo a buscar
185
186     // Realizar una búsqueda en los campos Nombre, ApellidoPaterno y ApellidoMaterno
187     $alumnos = Usuario::where(column: 'Tipo', operator: 'Alumno')
188         ->where(column: function(Usuario $query) use ($nombre): void {
189             $query->where(column: 'Nombre', operator: 'like', value: '%'. $nombre . '%')
190                 ->orWhere(column: 'Primer_apellido', operator: 'like', value: '%'. $nombre . '%')
191                 ->orWhere(column: 'Segundo_apellido', operator: 'like', value: '%'. $nombre . '%')
192                 ->orWhereRaw(sql: "CONCAT(Nombre, ' ', Primer_apellido, ' ', Segundo_apellido) LIKE ?", bindings: ['%' . $nombre . '%']);
193         })
194         ->with(relations: 'carrera.institucion') // Obtener la información de la carrera e institución
195         ->get();
196
197     if ($alumnos->isEmpty()) {
198         return ResponseHelper::error(message: 'No se encontraron alumnos con ese nombre', status: 404);
199     }
200
201     return ResponseHelper::success(data: $alumnos);
202 }

```

Anexo 204. Función *buscarAlumnoPorNombre*.


```

BackendCEP > app > Http > Controllers > GrupoController.php > PHP > GrupoController > agregarAlumnoAGrupo()
11 class GrupoController extends Controller

204 // Método para agregar un alumno al grupo
    1 reference | 0 overrides
205 public function agregarAlumnoAGrupo(Request $request, $grupoId): JsonResponse|mixed
206 {
207     $validated = $request->validate(rules: [
208         'UsuarioID' => 'required|exists:usuarios,UsuarioID',
209     ]);
210
211     $grupo = Grupos::find(id: $grupoId);
212
213     if (!$grupo) {
214         return ResponseHelper::error(message: 'Grupo no encontrado', status: 404);
215     }
216
217     // Verificar si el alumno ya está en el grupo
218     $exists = UsuarioGrupo::where(column: 'UsuarioID', operator: $validated['UsuarioID'])
219         ->where(column: 'GrupoID', operator: $grupo->GrupoID)
220         ->exists();
221
222     if ($exists) {
223         return ResponseHelper::error(message: 'El alumno ya pertenece a este grupo', status: 400);
224     }
225     // Añadir al alumno al grupo
226     try {
227         UsuarioGrupo::create(attributes: [
228             'UsuarioID' => $validated['UsuarioID'],
229             'GrupoID' => $grupo->GrupoID,
230             'CarreraID' => $grupo->CarreraID,
231             'InstitucionID' => $grupo->carrera->InstitucionID
232         ]);
233     } catch (\Exception $e) {
234         \Log::error('Error al agregar alumno al grupo: ' . $e->getMessage());
235         return ResponseHelper::error(message: 'Error al añadir alumno al grupo', status: 500);
236     }
237     return ResponseHelper::success(data: 'Alumno añadido al grupo exitosamente');
238 }

```

Anexo 205. Función *agregarAlumnoAGrupo*.

```

BackendCEP > app > Http > Controllers > GrupoController.php > PHP > GrupoController > agregarAlumnoAGrupo()
11 class GrupoController extends Controller

239
240 1 reference | 0 overrides
241 public function getAllGrupos(): JsonResponse|mixed
242 {
243     // Obtener todos los grupos con relaciones adicionales si las necesitas
244     $grupos = Grupos::with(relations: 'carrera.institucion')->get();
245
246     if ($grupos->isEmpty()) {
247         return response()->json(['message' => 'No hay grupos disponibles.'], 404);
248     }
249
250     return response()->json(['data' => $grupos]);
251 }

```

Anexo 206. Función *getAllGrupos*.

Controlador pregunta

Este método permite recuperar todas las preguntas almacenadas en la base de datos junto con sus relaciones asociadas a las variables. Para lograrlo, se emplea el modelo Pregunta en combinación con la función `with`, lo que asegura la carga eficiente de las relaciones. Finalmente, los datos se devuelven en formato JSON, lo que facilita su procesamiento o uso en la interfaz del sistema. En caso de no existir preguntas, retorna un arreglo vacío.

```

BackendCEP > app > Http > Controllers > PreguntController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use App\Models\Pregunta;
8
9
10 2 references | 0 implementations
11 class PreguntController extends Controller
12 {
13     //Se obtienen las pregunta con su relacion por variables
14     1 reference | 0 overrides
15     public function obtenerPreguntas(): JsonResponse|mixed
16     {
17         $preguntas = Pregunta::with(relations: 'variables')->get();
18         return response()->json($preguntas);
19     }
20 }

```

Anexo 207. Función obtenerPreguntas.

Controlador puntaje de variable

Este método calcula los puntajes de diversas variables con base en un cuestionario y un usuario específicos. Se define un arreglo que asocia las preguntas a cada variable utilizando sus IDs, organizados por variable. Un ciclo foreach recorre estas variables y utiliza el método calcularPuntajePorVariable para obtener el puntaje correspondiente de cada conjunto de preguntas. Los resultados se almacenan en un arreglo llamado \$puntajes.

```

BackendCEP > app > Http > Controllers > PuntajeVariableController.php > PHP IntelliSense > PuntajeVariableController
2
3 namespace App\Http\Controllers;
4
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use Illuminate\Support\Facades\DB;
8
9 2 references | 0 implementations
10 class PuntajeVariableController extends Controller
11 {
12     1 reference | 0 overrides
13     public function obtenerPuntajesPorCuestionario($UsuarioID, $CuestionarioID): JsonResponse|mixed
14     {
15         // IDs de las preguntas por variable
16         $preguntaVariables = [
17             1 => [1, 12, 23, 45, 34],
18             2 => [2, 13, 24, 46, 35],
19             3 => [3, 14, 25, 47, 36],
20             4 => [4, 15, 26, 48, 37],
21             5 => [5, 16, 27, 49, 38],
22             6 => [6, 17, 28, 50, 39],
23             7 => [7, 18, 29, 51, 40],
24             8 => [8, 19, 30, 52, 41],
25             9 => [9, 20, 31, 53, 42],
26             10 => [10, 21, 32, 54, 43],
27             11 => [11, 22, 33, 55, 44]
28         ];
29
30         // Calcular puntajes de las variables
31         $puntajes = [];
32         foreach ($preguntaVariables as $variable => $preguntaIDs) {
33             $puntajes["puntaje_variable_{$variable}"] = $this->calcularPuntajePorVariable(
34                 UsuarioID: $UsuarioID,
35                 CuestionarioID: $CuestionarioID,
36                 preguntaIDs: $preguntaIDs
37             );
38         }
39     }
40 }

```

Anexo 208. Función *obtenerPuntajesPorCuestionario*.

```

BackendCEP > app > Http > Controllers > PuntajeVariableController.php > PHP IntelliSense > PuntajeVariableController
9 class PuntajeVariableController extends Controller
11 public function obtenerPuntajesPorCuestionario($UsuarioID, $CuestionarioID): JsonResponse|mixed
12 {
13     // Calcular puntaje condicional para la variable 11
14     $puntaje_condicional = $this->aplicarCondicionPuntaje(puntaje: $puntajes["puntaje_variable_11"]);
15
16     // Ajustar puntajes de las variables 1 a 10 aplicando la condición
17     foreach (range(start: 1, end: 10) as $variable) {
18         $puntajes["puntaje_variable_{$variable}"] -= $puntaje_condicional;
19     }
20
21     // Guardar los puntajes en la base de datos
22     $this->guardarPuntajesEnBD(UsuarioID: $UsuarioID, CuestionarioID: $CuestionarioID, puntajeVariable0: $puntaje_condicional, $puntajes);
23
24     // Retornar solo los puntajes ajustados
25     return response()->json(['puntajes' => $puntajes]);
26 }
27
28 1 reference
29 private function calcularPuntajePorVariable($UsuarioID, $CuestionarioID, $preguntaIDs): float|int
30 {
31     // Obtener las preguntas con sus tipos (Positiva o Negativa)
32     $preguntas = DB::table('preguntas')
33         ->whereIn(column: 'PreguntaID', values: $preguntaIDs)
34         ->pluck(column: 'Tipo', key: 'PreguntaID');
35
36     $puntaje = 0;
37
38     // Calcular el puntaje según el tipo de cada pregunta
39     foreach ($preguntas as $preguntaID => $tipo) {
40         $valor = DB::table('respuestas_enviadas')
41             ->where(column: 'UsuarioID', operator: $UsuarioID)
42             ->where(column: 'CuestionarioID', operator: $CuestionarioID)
43             ->where(column: 'PreguntaID', operator: $preguntaID)
44             ->value(column: 'respuesta_valor');
45     }
46 }

```

Anexo 209. Función *calcularPuntajePorVariable* parte 1.

```

BackendCEP > app > Http > Controllers > PuntajeVariableController.php > PHP IntelliSense > PuntajeVariableController
9 class PuntajeVariableController extends Controller
53 private function calcularPuntajePorVariable($UsuarioID, $CuestionarioID, $preguntaIDs): float|int
69
70     if ($valor !== null) {
71         $puntaje += ($Tipo === 'Positiva' ? $valor : -$valor);
72     }
73 }
74
75 // Sumar 6 al puntaje de la variable
76 $puntaje += 6;
77
78 return $puntaje;
79 }
80
81 1 reference
82 private function aplicarCondicionPuntaje($puntaje): int
83 {
84     if ($puntaje <= 20) {
85         return 0;
86     } elseif ($puntaje <= 22) {
87         return 3;
88     } elseif ($puntaje <= 24) {
89         return 5;
90     } else {
91         return 7;
92     }
93 }
94 // Los otros métodos no requieren modificaciones adicionales.
95
96 1 reference
97 private function guardarPuntajesEnBD($UsuarioID, $CuestionarioID, $puntajeVariable0, $puntajes): void
98 {
99     $grupoID = $this->obtenerGrupoID(UsuarioID: $UsuarioID, CuestionarioID: $CuestionarioID);
100     if (!$grupoID) {
101         throw new \Exception(message: 'No se pudo encontrar el grupo correspondiente.');
```

Anexo 210. Función *calcularPuntajePorVariable* parte 2.

El método `guardarPuntajesEnBD` almacena estos puntajes en la base de datos. Primero, identifica el grupo relacionado con el usuario y el cuestionario, verificando si existen registros previos para evitar duplicaciones. Luego, asigna cada puntaje a su variable correspondiente en la tabla `puntajes_por_variable`, asegurándose de registrar correctamente el puntaje de la variable 11.

```

BackendCEP > app > Http > Controllers > PuntajeVariableController.php > PHP IntelliSense > PuntajeVariableController
9 class PuntajeVariableController extends Controller
96 private function guardarPuntajesEnBD($usuarioID, $cuestionarioID, $puntajeVariable0, $puntajes): void
103 // Verificar si los puntajes ya están guardados
104 $puntajesExistentes = DB::table('puntajes_por_variable')
105     ->where(column: 'UsuarioID', operator: $usuarioID)
106     ->where(column: 'GrupoID', operator: $grupoID)
107     ->where(column: 'CuestionarioID', operator: $cuestionarioID)
108     ->exists();
109 if ($puntajesExistentes) {
110     // Si ya existen, no realizar nuevas inserciones
111     return;
112 }
113
114 foreach ($puntajes as $variable => $puntaje) {
115     $variableID = $this->obtenerVariableID(variable: $variable);
116     DB::table('puntajes_por_variable')->insert(values: [
117         'UsuarioID' => $usuarioID,
118         'GrupoID' => $grupoID,
119         'CuestionarioID' => $cuestionarioID,
120         'VariableID' => $variableID,
121         'puntaje' => $puntaje,
122         'created_at' => now(),
123         'updated_at' => now(),
124     ]);
125 }
126
127 // Guardar puntaje de la variable 0
128 DB::table('puntajes_por_variable')->insert(values: [
129     'UsuarioID' => $usuarioID,
130     'GrupoID' => $grupoID,
131     'CuestionarioID' => $cuestionarioID,
132     'VariableID' => 11,
133     'puntaje' => $puntajeVariable0,
134     'created_at' => now(),
135     'updated_at' => now(),
136 ]);

```

Anexo 211. Función guardarPuntajesEnBD.

```

BackendCEP > app > Http > Controllers > PuntajeVariableController.php > PHP IntelliSense > PuntajeVariableController
9 class PuntajeVariableController extends Controller
139 private function obtenerGrupoID($usuarioID, $cuestionarioID): mixed
140 {
141     return DB::table('grupo_usuario')
142         ->join(table: 'cuestionario_grupo', first: 'grupo_usuario.GrupoID', operator: '=', second: 'cuestionario_grupo.GrupoID')
143         ->where(column: 'grupo_usuario.UsuarioID', operator: $usuarioID)
144         ->where(column: 'cuestionario_grupo.CuestionarioID', operator: $cuestionarioID)
145         ->value(column: 'grupo_usuario.GrupoID');
146 }
147
148 private function obtenerVariableID($variable): int
149 {
150     return (int)str_replace(search: 'puntaje_variable_', replace: '', subject: $variable);
151 }
152 }

```

Anexo 212. Función obtenerGrupoID.

Controlador respuestas enviadas

Esta función valida y registra las respuestas enviadas por los alumnos, asegurando que coincidan con las referencias de las tablas de cuestionarios y preguntas. Las respuestas son procesadas en un bucle y almacenadas en la tabla RespuestaEnviada, junto con el ID del usuario. Además, se registran logs para monitorear el proceso. Al finalizar, devuelve un mensaje confirmando el éxito del registro.

```

BackendCEP > app > Http > Controllers > RespuestaEnviadaController.php > ...
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Models\RespuestaEnviada;
7 use Illuminate\Support\Facades\Log;
8
9 class RespuestaEnviadaController extends Controller
10 {
11     1 reference | 0 overrides
12     public function store(Request $request, $UsuarioID): JsonResponse|mixed
13     {
14         Log::info('Solicitud recibida desde Angular', $request->all());
15         // Validar los datos entrantes
16         $request->validate(rules: [
17             'respuestas' => 'required|array',
18             'respuestas.*.CuestionarioID' => 'required|integer|exists:cuestionarios,CuestionarioID',
19             'respuestas.*.PreguntaID' => 'required|integer|exists:preguntas,PreguntaID',
20             'respuestas.*.respuesta_valor' => 'required|integer',
21         ]);
22
23         // Registrar respuestas recibidas
24         Log::info('Respuestas recibidas:', $request->input(key: 'respuestas'));
25         // Recorrer las respuestas y guardarlas
26         foreach ($request->input(key: 'respuestas') as $respuesta) {
27             RespuestaEnviada::create(attributes: [
28                 'CuestionarioID' => $respuesta['CuestionarioID'],
29                 'PreguntaID' => $respuesta['PreguntaID'],
30                 'respuesta_valor' => $respuesta['respuesta_valor'],
31                 'UsuarioID' => $UsuarioID,
32             ]);
33         }
34
35         return response()->json(['message' => 'Respuestas guardadas correctamente'], 200);
36     }
37 }

```

Anexo 213. Controlador respuestasEnviadas.

Controlador sumatoria puntajes

Esta función procesa y almacena los puntajes categorizados por variables en un cuestionario. Primero, se obtienen las categorías, variables y expresiones relacionadas. Luego, se calcula el puntaje total de cada variable desde la tabla puntajes_por_variable, y se calcula el porcentaje basado en el puntaje máximo posible. Los datos se guardan en la tabla sumatoria_puntuaciones, incluyendo información del usuario, categoría y variable. Se puede recuperar el puntaje de los alumnos de un grupo específico para un cuestionario. Si no hay alumnos, devuelve un error; de lo contrario, para cada alumno se obtienen los puntajes de la tabla sumatoria_puntuaciones, incluyendo detalles por categoría y variable. La respuesta final es un JSON con los datos de los alumnos y sus puntajes.

```

BackendCEP > app > Http > Controllers > SumatoriaPuntuacionesController.php > PHP > SumatoriaPuntuacionesController
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use App\Models\RespostaEnviada;
8 use Illuminate\Support\Facades\DB;
9 use App\Models\SumatoriaPuntajes;
10 use App\Models\Variable;
11 use App\Models\Categoria;
12 use App\Models\Expresiones;
13
14
15 class SumatoriaPuntuacionesController extends Controller
16 {
17     1 reference | 0 overrides
18     public function guardarPuntajesPorCategoriaVariable($usuarioID, $cuestionarioID): JsonResponse|mixed
19     {
20         // Obtener todas las categorías
21         $categorias = Categoria::with(relations: ['variables.expresiones'])->get();
22
23         foreach ($categorias as $categoria) {
24             foreach ($categoria->variables as $variable) {
25                 // Obtener el puntaje de la variable desde la tabla puntajes_por_variable
26                 $puntajeVariable = DB::table('puntajes_por_variable')
27                     ->where(column: 'UsuarioID', operator: $usuarioID)
28                     ->where(column: 'CuestionarioID', operator: $cuestionarioID)
29                     ->where(column: 'VariableID', operator: $variable->VariableID)
30                     ->value(column: 'puntaje'); // Obtener el valor de la columna "puntaje"
31
32                 // Calcular el puntaje máximo
33                 $puntajeMaximo = DB::table('pregunta_variable')
34                     ->where(column: 'VariableID', operator: $variable->VariableID)
35                     ->count() * 5; // considerando que cada respuesta tiene un máximo de 5 puntos
36
37                 $porcentaje = $puntajeMaximo > 0 ? ($puntajeVariable / $puntajeMaximo) * 100 : 0;
38
39                 foreach ($variable->expresiones as $expresion) {
40                     // Guardar en la tabla sumatoria.puntajes

```

Anexo 214. Función guardarPuntajesPorCategoriaYVariable

```

BackendCEP > app > Http > Controllers > SumatoriaPuntuacionesController.php > PHP > SumatoriaPuntuacionesController
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use App\Models\RespostaEnviada;
8 use Illuminate\Support\Facades\DB;
9 use App\Models\SumatoriaPuntajes;
10 use App\Models\Variable;
11 use App\Models\Categoria;
12 use App\Models\Expresiones;
13
14
15 class SumatoriaPuntuacionesController extends Controller
16 {
17     1 reference | 0 overrides
18     public function guardarPuntajesPorCategoriaVariable($usuarioID, $cuestionarioID): JsonResponse|mixed
19     {
20         // Obtener todas las categorías
21         $categorias = Categoria::with(relations: ['variables.expresiones'])->get();
22
23         foreach ($categorias as $categoria) {
24             foreach ($categoria->variables as $variable) {
25                 // Obtener el puntaje de la variable desde la tabla puntajes_por_variable
26                 $puntajeVariable = DB::table('puntajes_por_variable')
27                     ->where(column: 'UsuarioID', operator: $usuarioID)
28                     ->where(column: 'CuestionarioID', operator: $cuestionarioID)
29                     ->where(column: 'VariableID', operator: $variable->VariableID)
30                     ->value(column: 'puntaje'); // Obtener el valor de la columna "puntaje"
31
32                 // Calcular el puntaje máximo
33                 $puntajeMaximo = DB::table('pregunta_variable')
34                     ->where(column: 'VariableID', operator: $variable->VariableID)
35                     ->count() * 5; // considerando que cada respuesta tiene un máximo de 5 puntos
36
37                 $porcentaje = $puntajeMaximo > 0 ? ($puntajeVariable / $puntajeMaximo) * 100 : 0;
38
39                 foreach ($variable->expresiones as $expresion) {
40                     // Guardar en la tabla sumatoria.puntajes

```

Anexo 215. Función obtenerPuntajesPorGrupo.

Puntajes por categorías y expresiones

Desde los IDs del usuario y el cuestionario, se agrupan puntajes por categorías, calculando su porcentaje basado en los puntajes totales y máximos. El resultado es un JSON con los detalles del usuario, cuestionario, categorías y porcentajes calculados.

También se agrupan puntajes por expresiones dentro de una categoría. Se recuperan las expresiones asociadas y se calculan sus porcentajes. El resultado es un JSON con el detalle del usuario, cuestionario, categoría y expresiones con sus puntajes.

```

BackendCEP > app > Http > Controllers > SumatoriaPuntuacionesController.php > PHP > SumatoriaPuntuacionesController
15 class SumatoriaPuntuacionesController extends Controller
106 public function obtenerPuntajesPorCategoria($usuarioID, $cuestionarioID): JsonResponse|mixed
107 {
108     // Obtener información del usuario
109     $usuario = DB::table('usuarios')
110         ->where(column: 'UsuarioID', operator: $usuarioID)
111         ->select(columns: 'Nombre', 'Primer_apellido', 'Segundo_apellido')
112         ->first();
113
114     if (!$usuario) {
115         return response()->json(['message' => 'Usuario no encontrado.'], 404);
116     }
117
118     // Obtener información del cuestionario
119     $cuestionario = DB::table('cuestionarios')
120         ->where(column: 'CuestionarioID', operator: $cuestionarioID)
121         ->select(columns: 'Nombre as cuestionario_nombre', 'Descripcion as cuestionario_descripcion')
122         ->first();
123
124     if (!$cuestionario) {
125         return response()->json(['message' => 'Cuestionario no encontrado.'], 404);
126     }
127
128     // Obtener puntajes agrupados por categoría
129     $categorias = SumatoriaPuntajes::with(relations: 'categoria')
130         ->select(columns: 'CategoriaID', DB::raw('SUM(total_puntuacion) as total_puntuacion'), DB::raw('SUM(puntaje_maximo) as puntaje_maximo'))
131         ->where(column: 'UsuarioID', operator: $usuarioID)
132         ->where(column: 'CuestionarioID', operator: $cuestionarioID)
133         ->groupBy(groups: 'CategoriaID')
134         ->get()
135         ->map(callback: function (SumatoriaPuntajes $item): SumatoriaPunt_ {
136             $item->porcentaje = $item->puntaje_maximo > 0 ? ($item->total_puntuacion / $item->puntaje_maximo) * 100 : 0;
137             return $item;
138         });
139
140     return response()->json([
141         'message' => 'Puntajes por categoría obtenidos correctamente.',

```

Anexo 216. Función obtener puntajes por categoría.

```

BackendCEP > app > Http > Controllers > SumatoriaPuntuacionesController.php > PHP > SumatoriaPuntuacionesController > obtenerExpresionesPorCategoria
15 class SumatoriaPuntuacionesController extends Controller
156 public function obtenerExpresionesPorCategoria($usuarioID, $cuestionarioID, $categoriaID): JsonResponse|mixed
174 if (!$cuestionario) {
175     return response()->json(['message' => 'Cuestionario no encontrado.'], 404);
176 }
177 // Obtener información de la categoría
178 $categoria = DB::table('categorias')
179     ->where(column: 'CategoriaID', operator: $categoriaID)
180     ->select(columns: 'Nombre as categoria_nombre')
181     ->first();
182
183 if (!$categoria) {
184     return response()->json(['message' => 'Categoría no encontrada.'], 404);
185 }
186
187 // Obtener puntajes agrupados por expresiones dentro de la categoría
188 $expresiones = SumatoriaPuntajes::with(relations: 'expresion')
189     ->select(columns: 'ExpresionesID', DB::raw('SUM(total_puntuacion) as total_puntuacion'),
190         DB::raw('SUM(puntaje_maximo) as puntaje_maximo'))
191     ->where(column: 'UsuarioID', operator: $usuarioID)
192     ->where(column: 'CuestionarioID', operator: $cuestionarioID)
193     ->where(column: 'CategoriaID', operator: $categoriaID)
194     ->groupBy(groups: 'ExpresionesID')
195     ->get()
196     ->map(callback: function (SumatoriaPuntajes $item): SumatoriaPunt_ {
197         $item->porcentaje = $item->puntaje_maximo > 0 ? ($item->total_puntuacion / $item->puntaje_maximo) * 100 : 0;
198         return $item;
199     });
200
201     return response()->json([
202         'message' => 'Puntajes por categoría obtenidos correctamente.',
203         'usuario' => [
204             'nombre' => $usuario->Nombre,
205             'primer_apellido' => $usuario->Primer_apellido,
206             'segundo_apellido' => $usuario->Segundo_apellido,
207         ],

```

Anexo 217. Función obtener expresiones por categoría.

Preguntas por expresión

Este método obtiene preguntas vinculadas a una expresión específica en un cuestionario. Recupera datos del usuario, cuestionario, categoría y expresión asociada, junto con las respuestas enviadas. Devuelve un JSON con esta información y los puntajes asociados.


```

BackendCEP > app > Http > Controllers > SumatoriaPuntuacionesController.php > PHP > SumatoriaPuntuacionesController > obtenerExpresionesPorCategoria()
15 class SumatoriaPuntuacionesController extends Controller
156 public function obtenerExpresionesPorCategoria($UsuarioID, $CuestionarioID, $CategoriaID): JsonResponse|mixed
208     'cuestionario' => [
209         'nombre' => $cuestionario->cuestionario_nombre,
210         'descripcion' => $cuestionario->cuestionario_descripcion,
211     ],
212     'categoria' => [
213         'nombre' => $categoria->categoria_nombre,
214     ],
215     'data' => $expresiones,
216 ], 200);
217 }
218
219
220 // Obtener las preguntas por categoria, expresion
221 1 reference|0 overrides
221 public function obtenerPreguntasPorExpresion($UsuarioID, $CuestionarioID, $ExpresionesID): JsonResponse|mixed
222 {
223     // Obtener información del usuario
224     $usuario = DB::table('usuarios')
225         ->where(column: 'UsuarioID', operator: $UsuarioID)
226         ->select(columns: 'Nombre', 'Primer_apellido', 'Segundo_apellido')
227         ->first();
228
229     if (!$usuario) {
230         return response()->json(['message' => 'Usuario no encontrado.'], 404);
231     }
232
233     // Obtener información del cuestionario
234     $cuestionario = DB::table('cuestionarios')
235         ->where(column: 'CuestionarioID', operator: $CuestionarioID)
236         ->select(columns: 'Nombre as cuestionario_nombre', 'Descripcion as cuestionario_descripcion')
237         ->first();
238
239     if (!$cuestionario) {
240         return response()->json(['message' => 'Cuestionario no encontrado.'], 404);
241     }

```

Anexo 218. Función obtenerPreguntas por Expresión.

```

BackendCEP > app > Http > Controllers > SumatoriaPuntuacionesController.php > PHP > SumatoriaPuntuacionesController > obtenerPreguntasPorExpresion()
15 class SumatoriaPuntuacionesController extends Controller
221 public function obtenerPreguntasPorExpresion($UsuarioID, $CuestionarioID, $ExpresionesID): JsonResponse|mixed
239 {
240     if (!$cuestionario) {
241         return response()->json(['message' => 'Cuestionario no encontrado.'], 404);
242     }
243
244     // Obtener información de la expresión
245     $expresion = DB::table('expresiones')
246         ->where(column: 'ExpresionesID', operator: $ExpresionesID)
247         ->select(columns: 'Nombre as expresion_nombre', 'VariableID')
248         ->first();
249
250     if (!$expresion) {
251         return response()->json(['message' => 'Expresión no encontrada.'], 404);
252     }
253
254     // Obtener información de la categoría asociada
255     $categoria = DB::table('categorias')
256         ->join(table: 'variables', first: 'categorias.CategoriaID', operator: '=', second: 'variables.CategoriaID')
257         ->where(column: 'variables.VariableID', operator: $expresion->VariableID)
258         ->select(columns: 'categorias.Nombre as categoria_nombre')
259         ->first();
260
261     if (!$categoria) {
262         return response()->json(['message' => 'Categoría no encontrada.'], 404);
263     }
264
265     // Obtener preguntas relacionadas a la expresión
266     $preguntas = DB::table('respuestas_enviadas')
267         ->join(table: 'preguntas', first: 'respuestas_enviadas.PreguntaID', operator: '=', second: 'preguntas.PreguntaID')
268         ->where(column: 'respuestas_enviadas.UsuarioID', operator: $UsuarioID)
269         ->where(column: 'respuestas_enviadas.CuestionarioID', operator: $CuestionarioID)
270         ->whereIn(column: 'respuestas_enviadas.PreguntaID', values: function ($query) use ($expresion) {
271             $query->select('PreguntaID')
272                 ->from('pregunta_variable')
273                 ->where('VariableID', $expresion->VariableID);
274         });

```

Anexo 219. Función obtenerPreguntasPorExpresion.

```
BackendCEP > app > Http > Controllers > SumatoriaPuntuacionesController.php > PHP > SumatoriaPuntuacionesController > obtenerPreguntas
15 class SumatoriaPuntuacionesController extends Controller
221 public function obtenerPreguntasPorExpresion($UsuarioID, $CuestionarioID, $ExpresionesID): JsonResponse|mixed
274     ->select(columns: 'preguntas.Texto as pregunta', 'respuestas_enviadas.respuesta_valor as puntaje',
275     DB::raw('5 as puntaje_maximo')) // Puntaje máximo fijo
276     ->get();
277
278     return response()->json([
279         'message' => 'Preguntas por expresión obtenidas correctamente.',
280         'usuario' => [
281             'nombre' => $usuario->Nombre,
282             'primer_apellido' => $usuario->Primer_apellido,
283             'segundo_apellido' => $usuario->Segundo_apellido,
284         ],
285         'cuestionario' => [
286             'nombre' => $cuestionario->cuestionario_nombre,
287             'descripcion' => $cuestionario->cuestionario_descripcion,
288         ],
289         'categoria' => [
290             'nombre' => $categoria->categoria_nombre,
291         ],
292         'expresion' => [
293             'nombre' => $expresion->expresion_nombre,
294         ],
295         'data' => $preguntas,
296     ], 200);
297 }
298
```

Anexo 220. Respuesta de la obtención de puntajes.

Se obtienen los puntajes de un usuario para un cuestionario, con información agrupada por categorías, variables y expresiones. Si no se encuentran resultados, se devuelve un mensaje indicando la ausencia de datos; en caso contrario, la respuesta es un JSON con los puntajes obtenidos.

```
1 reference | 0 overrides
public function obtenerPuntajes($UsuarioID, $CuestionarioID): JsonResponse|mixed
{
    // Obtener los puntajes del usuario para el cuestionario específico
    $puntajes = SumatoriaPuntajes::with(relations: ['categoria', 'variable', 'expresion'])
        ->where(column: 'UsuarioID', operator: $UsuarioID)
        ->where(column: 'CuestionarioID', operator: $CuestionarioID)
        ->get();

    // Verificar si se encontraron puntajes
    if ($puntajes->isEmpty()) {
        return response()->json(['message' => 'No se encontraron puntajes para este usuario y cuestionario.'], 404);
    }

    // Devolver los puntajes obtenidos
    return response()->json([
        'message' => 'Puntajes obtenidos correctamente.',
        'data' => $puntajes,
    ], 200);
}
}
```

Anexo 221. Función obtenerPuntajes.

Controlador de estadísticas grupales

Este método se encarga de calcular las estadísticas de las puntuaciones obtenidas por un grupo en un cuestionario específico. Genera valores como promedio, desviación estándar, mínimo y máximo, agrupando los resultados por categoría y variable para los usuarios del grupo. Si no hay datos disponibles, devuelve un mensaje indicando la ausencia de resultados. En caso de éxito, proporciona las estadísticas calculadas en formato JSON.

```
BackendCIP > app > Http > Controllers > EstadisticosGrupalesController.php > PHP IntelliSense > EstadisticosGrupalesController
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use Illuminate\Support\Facades\DB;
8
9 class EstadisticosGrupalesController extends Controller
10 {
11     1 reference | 0 overrides
12     public function calcularEstadisticosPorGrupo($GrupoID, $CuestionarioID): JsonResponse|mixed
13     {
14         $estadisticos = DB::table('sumatoria_puntuaciones')
15             ->join('grupo_usuario', 'first: sumatoria_puntuaciones.UsuarioID', operator: '=', second: 'grupo_usuario.UsuarioID')
16             ->where('column: grupo_usuario.GrupoID', operator: $GrupoID)
17             ->where('column: sumatoria_puntuaciones.CuestionarioID', operator: $CuestionarioID)
18             ->select(
19                 columns: 'sumatoria_puntuaciones.CategoriaID',
20                 'sumatoria_puntuaciones.VariableID',
21                 DB::raw("AVG(total_puntuacion) as promedio"),
22                 DB::raw("STD(total_puntuacion) as desviacion_estandar"),
23                 DB::raw("MIN(total_puntuacion) as minimo"),
24                 DB::raw("MAX(total_puntuacion) as maximo")
25             )
26             ->groupBy('groups: sumatoria_puntuaciones.CategoriaID', 'sumatoria_puntuaciones.VariableID')
27             ->get();
28
29         if ($estadisticos->isEmpty()) {
30             return response()->json(['message' => 'No se encontraron datos para el grupo y cuestionario especificados.'], 404);
31         }
32
33         return response()->json([
34             'message' => 'Estadísticos calculados correctamente.',
35             'data' => $estadisticos,
36         ], 200);
37     }
38 }
```

Anexo 222. Función *calcularEstadisticosPorGrupo*.

Se calculan los cuartiles de las puntuaciones de un grupo en un cuestionario, ordenando los datos por categoría y variable. Los resultados incluyen los valores del primer, segundo (mediana) y tercer cuartil, además del promedio, desviación estándar, mínimo y máximo. Si no se encuentran datos, se genera un mensaje de error. Los cuartiles se obtienen utilizando funciones auxiliares que calculan con precisión los percentiles deseados.

```

BackEndCEP > app > http > Controllers > EstadisticosGruposController.php > PHP IntelliSense > EstadisticosGruposController
9 class EstadisticosGruposController extends Controller
10
11 public function calcularCuartiles($grupoID, $cuestionarioID): JsonResponse|mixed
12 {
13     $datos = DB::table('sumatoria_puntuaciones')
14     ->join(table: 'grupo_usuario', first: 'sumatoria_puntuaciones.UsuarioID', operator: '=', second: 'grupo_usuario.UsuarioID')
15     ->where(column: 'grupo_usuario.GrupoID', operator: $grupoID)
16     ->where(column: 'sumatoria_puntuaciones.CuestionarioID', operator: $cuestionarioID)
17     ->select(columns: 'CategoriaID', 'VariableID', 'total_puntuacion')
18     ->orderBy(column: 'total_puntuacion')
19     ->get();
20
21     if ($datos->isEmpty()) {
22         return response()->json(['message' => 'No se encontraron datos para calcular los cuartiles.'], 404);
23     }
24
25     $resultados = [];
26
27     foreach ($datos->groupBy(['CategoriaID', 'VariableID']) as $categoriaVariable => $valores) {
28         $puntuaciones = $valores->pluck('total_puntuacion')->sort()->values();
29         $total = $puntuaciones->count();
30
31         if ($total > 1) {
32             $resultados[$categoriaVariable] = [
33                 'promedio' => $puntuaciones->avg(),
34                 'desviacion_estandar' => $this->calcularDesviacionEstandar(valores: $puntuaciones),
35                 'minimo' => $puntuaciones->first(),
36                 'maximo' => $puntuaciones->last(),
37                 'cuartil_1' => $this->calcularCuartil(valores: $puntuaciones, percentil: 0.25),
38                 'mediana' => $this->calcularCuartil(valores: $puntuaciones, percentil: 0.50),
39                 'cuartil_3' => $this->calcularCuartil(valores: $puntuaciones, percentil: 0.75),
40             ];
41         } else {
42             $resultados[$categoriaVariable] = [
43                 'promedio' => $total > 0 ? $puntuaciones->avg() : null,
44                 'desviacion_estandar' => null,
45                 'minimo' => $puntuaciones->first(),
46             ];
47         }
48     }
49
50     return response()->json($resultados);
51 }

```

Anexo 223. Función calcularCuartiles.

Este método permite calcular la desviación estándar de un conjunto de valores. Si la cantidad de datos es mayor a uno, se obtiene el promedio y se calcula la suma de las diferencias al cuadrado entre cada valor y el promedio. Posteriormente, se divide entre el número de valores menos uno y se aplica la raíz cuadrada al resultado. Si hay menos de dos valores, el método devuelve null, ya que no es posible realizar el cálculo.

```

83
84 1 reference
85 private function calcularDesviacionEstandar($valores): float|null
86 {
87     $n = $valores->count();
88     if ($n <= 1) {
89         return null; // No se puede calcular la desviación estándar con menos de 2 valores
90     }
91
92     $promedio = $valores->avg();
93     $suma = $valores->reduce(function ($carry, $item) use ($promedio): float|int {
94         return $carry + pow(num: $item - $promedio, exponent: 2);
95     });
96
97     return sqrt(num: $suma / ($n - 1));
98 }

```

Anexo 224. Función calcularDesviaciónEstandar.

El cálculo de cuartiles se basa en determinar un percentil específico dentro de un conjunto de valores. Se calcula el índice correspondiente al percentil deseado, y dependiendo de si es un valor entero o no, se devuelve directamente el valor o se interpola entre los valores adyacentes. Si el conjunto de datos está vacío, el método retorna null.

```

BackendCEP > app > Http > Controllers > EstadisticosGrupalesController.php > PHP IntelliSense > EstadisticosGrupalesController
9 class EstadisticosGrupalesController extends Controller
100
101 3 references
102 private function calcularCuartil($valores, $percentil): mixed
103 {
104     $n = $valores->count();
105     if ($n == 0) {
106         return null; // No se puede calcular cuartiles con un conjunto vacío
107     }
108     $index = $percentil * ($n - 1);
109     $lower = (int) floor(num: $index);
110     $upper = (int) ceil(num: $index);
111     if ($lower == $upper) {
112         return $valores[$lower];
113     }
114     $lowerValue = $valores[$lower];
115     $upperValue = $valores[$upper];
116     return $lowerValue + ($upperValue - $lowerValue) * ($index - $lower);
117 }
118 }
119

```

Anexo 225. Función calcularCuartil.

Rutas

En Laravel, las rutas son responsables de definir los puntos de acceso y los métodos que manejan las solicitudes HTTP. Cada ruta se vincula a un controlador encargado de procesar la lógica correspondiente. Por ejemplo, una ruta como `/cuestionarios/resueltos/{usuarioId}` está asociada a un controlador que devuelve los cuestionarios completados por un usuario específico. Estas rutas facilitan una comunicación eficiente y estructurada entre las solicitudes del usuario y la base de datos, permitiendo acceder y manipular los datos de manera organizada.

```
BackendCEP > routes > web.php
1 <?php
2
3 use App\Http\Controllers\PuntajePorCuestionarioController;
4 use Illuminate\Support\Facades\Route;
5 use App\Http\Controllers\CuestionarioController;
6 use App\Http\Controllers\InstitucionesController;
7 use App\Http\Controllers\PaisesController;
8 use App\Http\Controllers\RespuestaEnviadaController;
9 use App\Http\Controllers\PuntajeController;
10 use App\Http\Controllers\UsuarioController;
11 use App\Http\Controllers\CarrerasController;
12 use App\Http\Controllers\GrupoController;
13 use App\Http\Controllers\CuestionarioResueltoController;
14 use App\Http\Controllers\PreguntaController;
15 use App\Http\Controllers\GeneracionController;
16 use App\Http\Controllers\SumatoriaPuntuacionesController;
17 use App\Http\Controllers\PuntajeVariableController;
18 use App\Http\Controllers\EstadisticosGrupalesController;
19
20 // Cuestionarios
21 Route::get('/cuestionarios/{cuestionarioID}/preguntas', [CuestionarioController::class, 'getPreguntasRespuestas']);
22 Route::get('/preguntas', [PreguntaController::class, 'obtenerPreguntas']);
23 Route::post('/respuestas/{UsuarioID}/guardarRespuestas', [RespuestaEnviadaController::class, 'store']);
24
25
26 //borrar esto (es del primer cuestionario del home)
27 Route::get('/puntajes/{UsuarioID}', [PuntajeController::class, 'obtenerPuntajesVariables']);
28
29
30 //Usuarios
31 Route::post('/usuarios/register', [UsuarioController::class, 'register']);
32 Route::put('/usuarios/actualizar/{id}', [UsuarioController::class, 'update']);
33 Route::delete('/usuarios/eliminar/{id}', [UsuarioController::class, 'eliminar']);
34 Route::get('/usuarios/alumnos', [UsuarioController::class, 'obtenerAlumnos']);
35 Route::get('/usuarios/maestros', [UsuarioController::class, 'obtenerMaestros']);
36 Route::get('/usuarios/administradores', [UsuarioController::class, 'obtenerAdministradores']);
37 Route::get('/usuarios/all', [UsuarioController::class, 'usuarios']);
```

Anexo 226. Rutas parte 1.

```
BackendCEP > routes > web.php
1 <?php
2
38 Route::get('/usuarios/alumnos/{id}', [UsuarioController::class, 'show']);
39 Route::get('/usuarios/administrador/{id}', [UsuarioController::class, 'show']);
40 Route::get('/usuarios/maestro/{id}', [UsuarioController::class, 'obtenerMaestro']);
41 Route::get('/usuarios/maestros/institucion/{institucionID}', [UsuarioController::class, 'buscarMaestroPorInstitucion']);
42 Route::get('/usuarios/alumnos/institucion/{institucionID}', [UsuarioController::class, 'buscarAlumnoPorInstitucion']);
43 Route::post('/login', [UsuarioController::class, 'login']);
44
45
46 //Instituciones
47 Route::post('catalogs/añadirInstitucion', [InstitucionesController::class, 'crearInstitucion']);
48 Route::get('catalogs/ainstituciones', [InstitucionesController::class, 'instituciones']);
49 Route::get('catalogs/idinstituciones/{id}', [InstitucionesController::class, 'institucion']);
50 Route::get('catalogs/instituciones', [InstitucionesController::class, 'institucionesPorPais']);
51 Route::delete('catalogs/deleteInstitucion/{id}', [InstitucionesController::class, 'destroy']);
52 Route::put('catalogs/instituciones/{id}/actualizar', [InstitucionesController::class, 'actualizarInstitucion']);
53
54 //Paises
55 Route::post('catalogs/agregarpais', [PaisesController::class, 'agregarPais']);
56 Route::put('catalogs/paises/actualizar/{id}', [PaisesController::class, 'actualizarPais']);
57 Route::delete('catalogs/paises/eliminar/{id}', [PaisesController::class, 'eliminarPais']);
58 Route::get('catalogs/paises', [PaisesController::class, 'paises']);
59 Route::get('catalogs/paises/{id}', [PaisesController::class, 'pais']);
60 Route::get('catalogs/paises/buscar', [PaisesController::class, 'buscarPaises']);
61
62
63 //Carreras
64 Route::post('catalogs/carrera/agregar', [CarrerasController::class, 'agregarCarrera']);
65 Route::put('catalogs/carrera/editar/{id}', [CarrerasController::class, 'editarCarrera']);
66 Route::get('catalogs/acarreras', [CarrerasController::class, 'carreras']);
67 Route::get('catalogs/carreras/{id}', [CarrerasController::class, 'carrera']);
68 Route::get('catalogs/carreras', [CarrerasController::class, 'carrerasPorInstitucion']);
69 Route::delete('catalogs/carreras/eliminar/{id}', [CarrerasController::class, 'eliminar']);
70
71 //Generacion
72 Route::post('catalogs/generaciones/agregar', [GeneracionController::class, 'agregarGeneracion']);
73 Route::put('catalogs/generaciones/{id}', [GeneracionController::class, 'editarGeneracion']);
```

Anexo 227. Rutas parte 2.

```

BackendCEP > routes > web.php
1  <?php
71 //Generacion
72 Route::post('catalogs/generaciones/agregar', [GeneracionController::class, 'agregarGeneracion']);
73 Route::put('catalogs/generaciones/{id}', [GeneracionController::class, 'editarGeneracion']);
74 Route::get('catalogs/generaciones', [GeneracionController::class, 'generaciones']);
75 Route::get('catalogs/generaciones/{id}', [GeneracionController::class, 'generacion']);
76 Route::delete('catalogs/generaciones/eliminar/{id}', [GeneracionController::class, 'eliminarGeneracion']);
77
78 //Grupos
79 Route::get('/alumno/{id}/grupos', [GrupoController::class, 'getAlumnoGrupos']);
80 Route::get('/grupos', [GrupoController::class, 'getAllGrupos']);
81
82
83 // Rutas para gestionar el acceso a grupos
84 Route::post('/grupos/buscar-por-codigo', [GrupoController::class, 'getGrupoByCodigo']);
85 Route::post('/grupos/unirse', [GrupoController::class, 'joinGrupo']);
86
87 //Cuestionarios Resueltos
88 Route::post('/cuestionarios/marcar-resuelto', [CuestionarioResueltoController::class, 'marcarComoResuelto']);
89 Route::get('/cuestionarios/resueltos/{usuarioId}/{grupoId}', [CuestionarioResueltoController::class, 'obtenerCuestionariosResueltos']);
90 Route::get('/cuestionarios-resueltos/alumno/{usuarioId}', [CuestionarioResueltoController::class, 'obtenerCuestionariosResueltosPorAlumno']);
91 Route::get('/cuestionarios/{cuestionarioId}', [CuestionarioController::class, 'getCuestionario']);
92
93 //cuestionarios pendientes
94 Route::get('/cuestionarios/pendientes/{usuarioId}/{grupoId}', [CuestionarioResueltoController::class, 'obtenerCuestionariosPendientes']);
95
96 //Maestro
97 //Crear grupo
98 Route::post('/crear-grupos', [GrupoController::class, 'crearGrupo']);
99 Route::get('/maestro/{usuarioId}/grupos', [GrupoController::class, 'obtenerGruposDeMaestro']);
100 Route::get('/grupos/{id}', [GrupoController::class, 'show']);
101 Route::get('/grupos/{id}/integrantes', [GrupoController::class, 'getIntegrantes']);
102 //agregar alumnos
103 Route::get('/alumno/buscar', [GrupoController::class, 'buscarAlumnoPorNombre']);
104 Route::post('/grupos/{grupoId}/alumnos/agregar', [GrupoController::class, 'agregarAlumnoAGrupo']);

```

Anexo 228. Rutas parte 3.

```

BackendCEP > routes > web.php
1  <?php
106 //cuestionarios
107 Route::get('/cuestionarios-aplicados/grupo/{grupoId}', [CuestionarioResueltoController::class, 'obtenerCuestionariosAplicadosPorGrupo']);
108 Route::get('/cuestionarios/usuario/{usuarioId}', [CuestionarioController::class, 'obtenerCuestionariosPorUsuario']);
109
110 Route::post('/cuestionarios/{cuestionarioID}/duplicar-editar', [CuestionarioController::class, 'duplicarYEditarCuestionario']);
111
112
113 //Admin
114 //Crear nuevo cuestionario
115 Route::post('/cuestionarios', [CuestionarioController::class, 'crearCuestionario']);
116 Route::get('/cuestionarios', [CuestionarioController::class, 'getQuizVersions']);
117 Route::post('/cuestionarios/{cuestionarioID}/publicar', [CuestionarioController::class, 'publicarCuestionario']);
118 Route::post('/cuestionarios/{cuestionarioID}/duplicar', [CuestionarioController::class, 'duplicarYEditarCuestionario']);
119 Route::get('/cuestionarios/versions', [CuestionarioController::class, 'getQuizVersions']);
120
121 //Puntuaciones por variable
122 Route::post('/puntajes/sumar-puntuaciones/{UsuarioID}/{CuestionarioID}', [SumatoriaPuntuacionesController::class,
123 'guardarPuntajesPorCategoriaYVariable']);
124 Route::get('/puntajes/sumar-puntuaciones/{UsuarioID}/{CuestionarioID}', [SumatoriaPuntuacionesController::class, 'obtenerPuntajes']);
125 //Para el maestro(sumatoria por grupo)
126 Route::get('/puntajes/consultar/{GrupoID}/{CuestionarioID}', [SumatoriaPuntuacionesController::class, 'obtenerPuntajesPorGrupo']);
127 Route::get('/puntajes/categorias/{UsuarioID}/{CuestionarioID}', [SumatoriaPuntuacionesController::class, 'obtenerPuntajesPorCategoria']);
128 Route::get('/puntajes/expresiones/{UsuarioID}/{CuestionarioID}/{CategoriaID}', [SumatoriaPuntuacionesController::class,
129 'obtenerExpresionesPorCategoria']);
130 Route::get('/puntajes/preguntas/{UsuarioID}/{CuestionarioID}/{ExpresionesID}', [SumatoriaPuntuacionesController::class,
131 'obtenerPreguntasPorExpresion']);
132
133 //Puntuaciones generales guardadas
134 Route::get('/puntajes/cuestionario/{UsuarioID}/{CuestionarioID}', [PuntajeVariableController::class, 'obtenerPuntajesPorCuestionario']);
135
136 //Estadisticos grupales
137 Route::get('/grupos/{GrupoID}/estadisticos/{CuestionarioID}', [EstadisticosGrupalesController::class, 'calcularEstadisticosPorGrupo']);
138 Route::get('/grupos/{GrupoID}/cuartiles/{CuestionarioID}', [EstadisticosGrupalesController::class, 'calcularCuartiles']);

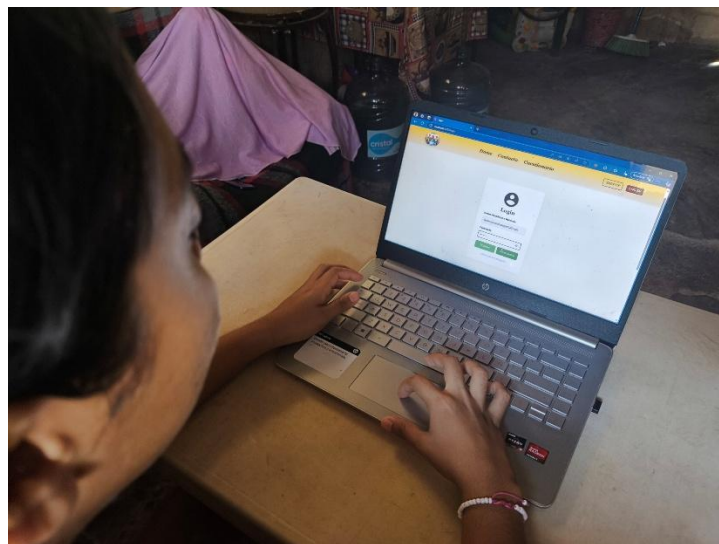
```

Anexo 229. Rutas parte 3.

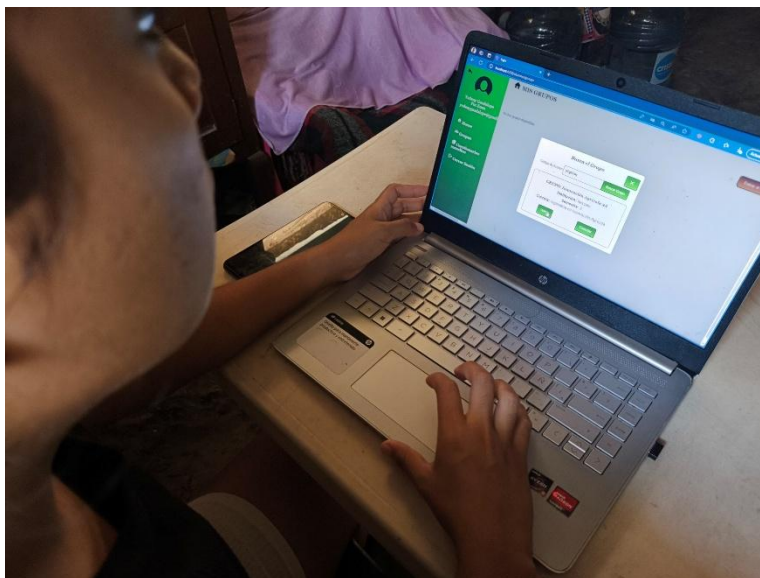
Pruebas de usabilidad del sistema.



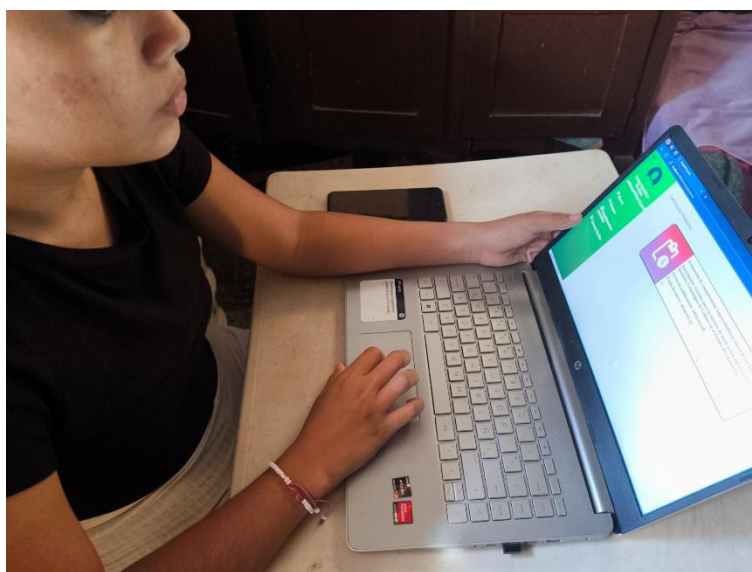
Anexo 230. Evidencia de pruebas de usabilidad.



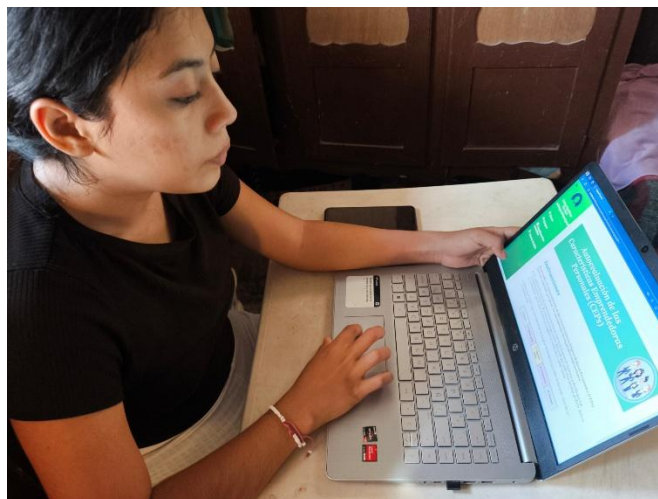
Anexo 231. Evidencia de pruebas de usabilidad.



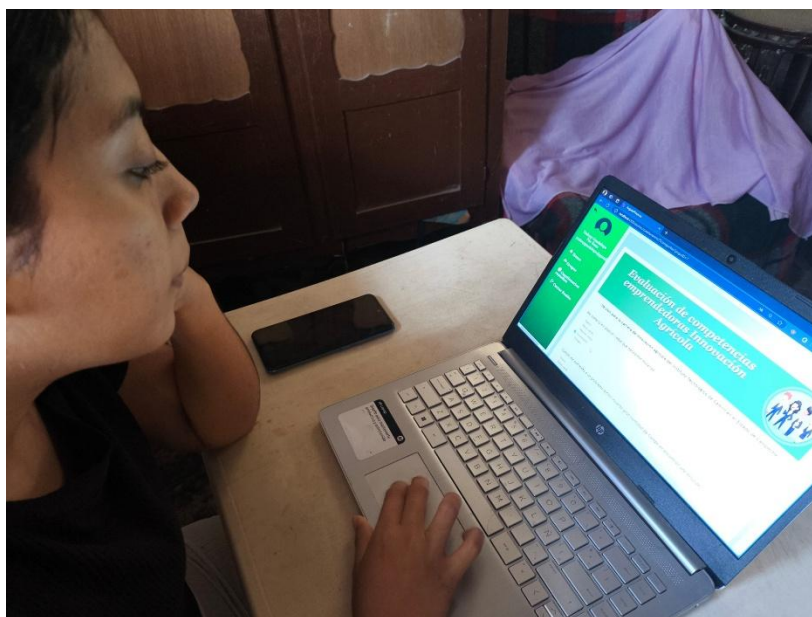
Anexo 232. Evidencia de pruebas de usabilidad.



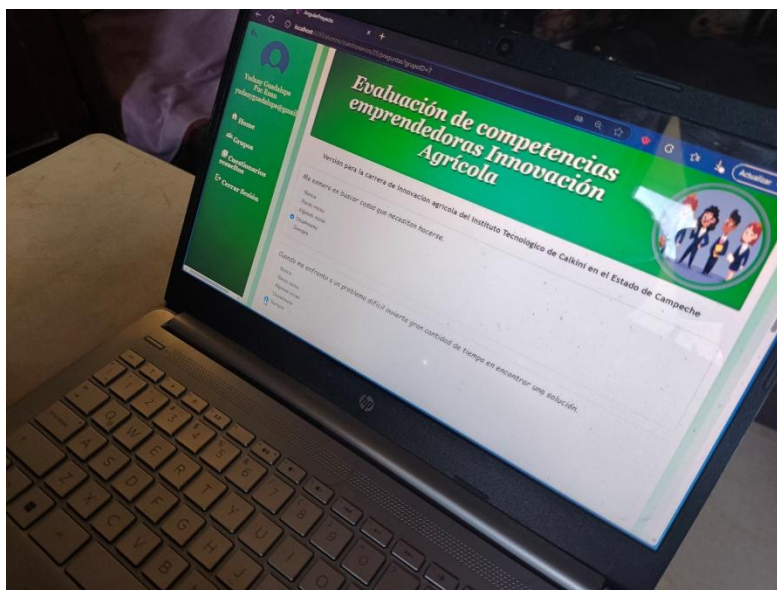
Anexo 233. Evidencia de pruebas de usabilidad.



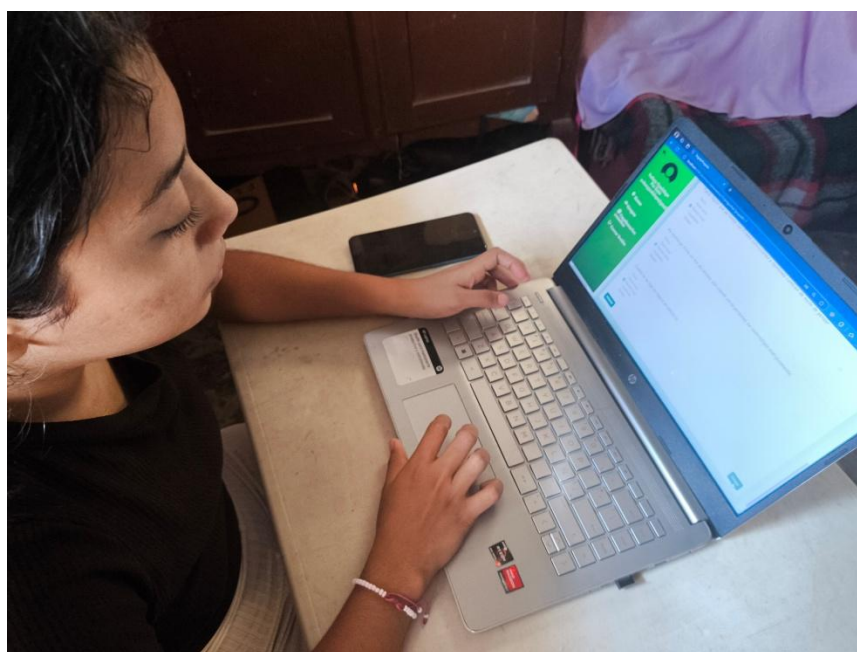
Anexo 234. Evidencia de pruebas de usabilidad.



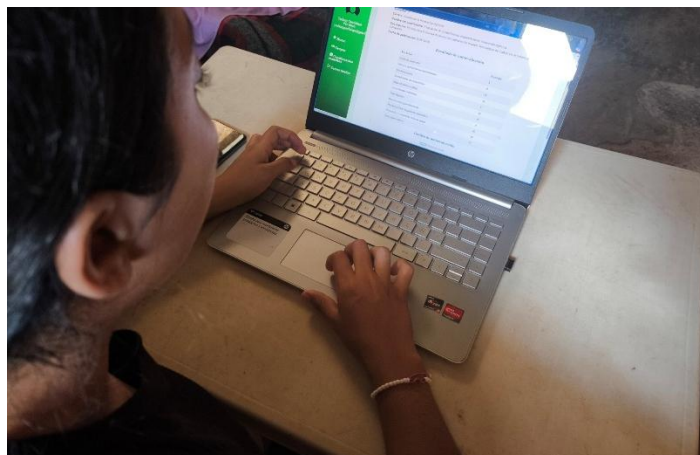
Anexo 235. Evidencia de pruebas de usabilidad.



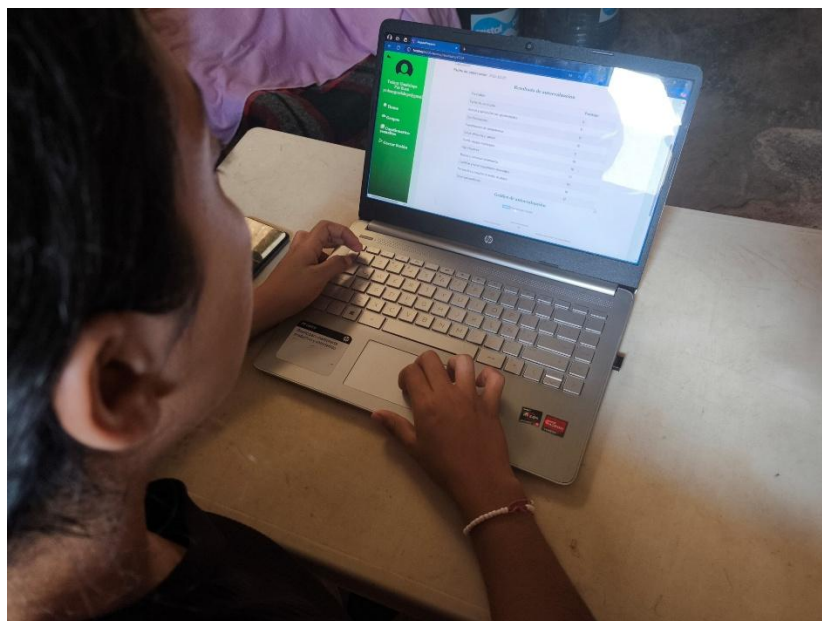
Anexo 236. Evidencia de pruebas de usabilidad.



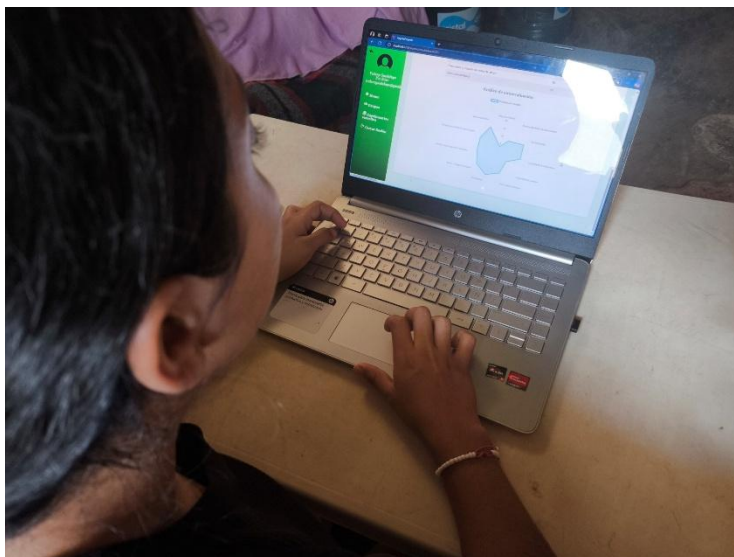
Anexo 237. Evidencia de pruebas de usabilidad.



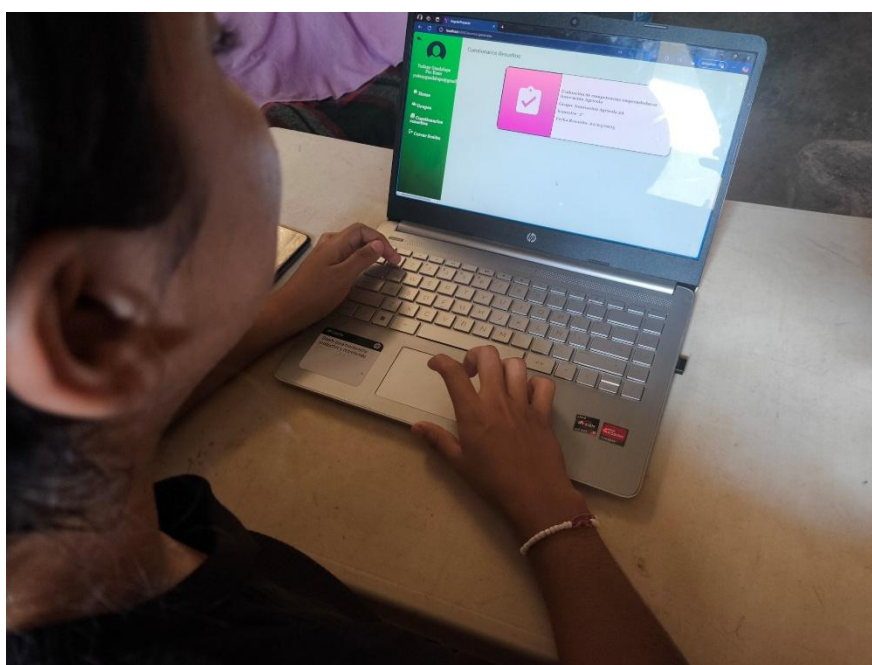
Anexo 238. Evidencia de pruebas de usabilidad.



Anexo 239. Evidencia de pruebas de usabilidad.



Anexo 240. Evidencia de pruebas de usabilidad.



Anexo 241. Evidencia de pruebas de usabilidad.