

**“ALGORITMOS PARA ANÁLISIS ESTADÍSTICO
PARAMÉTRICO Y NO PARAMÉTRICO DE DATOS DE
ACTITUDES DE SERVICIO COMUNITARIO BAJO LA ESCALA
CSAS”**

TESIS PARA TITULACIÓN INTEGRAL

INGENIERÍA SISTEMAS COMPUTACIONALES

**EGRESADO:
FELIPE ISMAEL MAAS CANUL**

**DIRECTOR Y CO-DIRECTOR:
DR. GONZALO MIGUEL QUETZ AGUIRRE
DRA. MARLENE MÉNDEZ MORENO**

Calkiní, Campeche, México 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/03/25/652

INGENIERÍA EN SISTEMAS COMPUTACIONALES

CONSTANCIA DE NO INCONVENIENCIA

MODALIDAD: TESIS

C. FELIPE ISMAEL MAAS CANUL 7264
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 DR GONZALO MIGUEL QUETZ AGUIRRE, y el co-director DRA. MARLENE MÉNDEZ MORENO, respectivamente, este Instituto consta de no tener inconveniencia para la realización del acto protocolario de titulación integral de su trabajo profesional titulado: **"ALGORITMOS PARA EL ANÁLISIS ESTADÍSTICO PARAMÉTRICO Y NO PARAMÉTRICO DE DATOS DE ACTITUDES DE SERVICIO COMUNITARIO BAJO LA ESCALA CSAS"**.

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, MEXICO.



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_calkini@tecnm.mx | itescam.edu.mx



Dedicatoria.

Este documento es un agradecimiento principalmente a todos los maestros, por quienes obtuve los conocimientos necesarios para lograr realizar este proyecto, así como por esos consejos valiosos que me brindaron a lo largo del camino. Su dedicación y paciencia han sido fundamentales para mi crecimiento académico y personal.

A mi familia y amigos, que estuvieron apoyándome incondicionalmente, no solo durante la realización de este proyecto, sino en cada etapa de mi vida. Su aliento, comprensión y motivación han sido pilares esenciales para superar los desafíos y celebrar los logros.

También quiero expresar mi gratitud a todas aquellas personas que, de una forma u otra, contribuyeron directa o indirectamente a la culminación de este trabajo.

Finalmente, quiero reconocer el esfuerzo y dedicación que he puesto en este proyecto, el cual representa no solo un logro académico, sino también un testimonio de superación y compromiso. Este es el resultado de un camino lleno de aprendizajes.

Agradecimientos.

Palabras nunca sobran en el momento de agradecer, aunque a las personas a quienes van dirigidas sí.

Principalmente, quiero agradecer a mi abuela y tías, que siempre han estado presentes, incluso cuando estuve a punto de dejarlo todo; cuando no sabía si podría avanzar con toda la presión y estrés que ha llevado toda la carrera, así como este proyecto.

Hablando de personas a quienes agradecer, hay dos que, a pesar de que no estuvieron al principio, estuvieron en la recta final y en momentos de importancia, siendo estas: Una amiga que aprecio tanto, Coranda Calán Calán, que cuando pensé que iba a caer o que lo iba a dejar todo, no dejó de apoyarme, escucharme y aconsejarme, volviéndose una persona importante.

De igual forma, a toda su familia, que me pudo abrir sus puertas para acogerme como uno más de ellos, motivándome cada vez más.

Y a mi pareja, Fabiola, que en estos últimos meses se volvió uno de los pilares fundamentales, ya que fue una de las personas que me estuvo motivando a seguir cuando ya estaba a punto de dejarlo todo.

Y, aún así, a compañeros que estuvieron ahí presentes, cuando tuve dudas o incluso cuando pude ayudarlos en algo, y por los momentos vividos, sean en momentos de alegría o tristeza.

Índice.

Contenido

Dedicatoria.	3
Agradecimientos.	4
Índice.	5
Índice Figuras.	6
Índice Tablas.	7
Resumen.	8
Capítulo 1	9
1.1 ANTECEDENTES	9
1.2 INTRODUCCION	10
1.3 JUSTIFICACION	11
1.4 OBJETIVOS	12
1.5 ALCANCES Y LIMITACIONES.	13
1.6 ORGANIGRAMA	14
Capítulo 2	19
2.1 MARCO TEORICO.	19
Capítulo 3.	22
3.1 ATERIALES Y METODOS	22
3.2 IMPREMENTACION DE LA METODOLOGIA	31
3.3 REQUERIMEINTOS.	35
3.4 MODELO RELACIONAL	39
Capítulo 4.	40
Resultados y Discusión.	40
4.1 IMPLEMENTACION	40
4.2 PRUEBAS DE FUNCIONALIDAD DEL SISTEMA	44
4.3 PRUEBAS DE USABILIDAD Y RECORECCION DE RETROALIMENTACION	49
4.4 COMPARACION DEL METODO DE RECORRECCION	51
4.5 PANTALLAS DE LOS USUARIOS QUE CONTESTARON LA ENCUESTA	53
4.6 RESPUESTAS DE LOS DATOS DEMOGRAFICOS	54
4.7 RESPUESTAS DE LOS DATOS OBTENIDOS	56

Capítulo 5.	66
Conclusiones	66
Recomendaciones.	67
Referencias Bibliográficas.....	68
Anexos.....	70

Índice Figuras.

Imagen My SQL 3.1.1.1.1	22
Imagen ANGULAR 3.1.1.2.1	24
Imagen GITHUB 3.1.1.3.1.....	25
Imagen LARAVEL 3.1.1.4.1	27
Imagen PHP 3.1.1.5.1	28
Imagen JWT 3.1.1.6.1	29
Imagen JS 3.1.1.7.1.....	30
Imagen TS 3.1.1.8.1	31
Imagen 3.3.1 Diagrama de uso para el estudiante	35
Imagen 3.3.2 Diagrama de inicio de seccion	36
Imagen 3.3.3 Diagrama de uso para el administrador	37
Imagen 3.3.4 Diagrama de uso para el evaluador	38
Modelo relacional	39
Pantalla 1.....	44
Pantalla 2.....	45
Pantalla 3.....	46
Pantalla 4.....	47
Pantalla 5.....	48
Imagen prueba 1 y 2.....	49
Imagen prueba 3 y 4.....	50
Resultado 1 y 2.....	53
Resultado 3.....	55
Grafica 1 y 2.....	54
Grafica 3 y 4.....	56
Grafica 5 y 6.....	57
Grafica 7 y 8.....	58
Grafica 9 y 10.....	59
Grafica 11 y 12.....	60
Grafica 13 y 14.....	61
Grafica 15 y 16.....	62
Grafica 17 y 18.....	63
Grafica 19 y 20.....	64
Grafica 21 y 22.....	65

Imagen anexo 1.....	70
Imagen anexo 2.....	71
Imagen anexo 3.....	74
Imagen anexo 4.....	75
Imagen anexo 5 y 6.....	76
Imagen anexo 7.....	80
Imagen anexo 8.....	96
Imagen anexo 9.....	97
Imagen anexo 10.....	159
Imagen anexo 11.....	195
Imagen anexo 12.....	202
Imagen anexo 13.....	206
Imagen anexo 14.....	214
Imagen anexo 15.....	287

Índice Tablas.

<u>Tabla organigrama</u>	15
Tabla 3.2.1 Equipo Scrum	32
Tabla 3.2.2 Tipos de usuarios del sistema	34
Tabla Comparativa	52

Resumen.

El presente proyecto tiene el objetivo de analizar las actitudes con respecto a los servicios comunitarios utilizando la escala de CSAS, aplicando técnicas de estadística paramétricas y no paramétricas junto con la minería de datos, esto con el propósito principal de identificar patrones, tendencias, así como factores predictivos que puedan mejorar la comprensión del impacto sobre la participación comunitaria teniendo la percepción de los individuos. Además de llevarse a cabo una investigación sobre las técnicas de visualización matemática y análisis gráficos, que podrían facilitar la interpretación de los resultados para usuarios sin experiencia.

De igual forma se ha realizado un modelo de base de datos relacional para poder recolectar y almacenar las respuestas con respecto al cuestionario CSAS, así como el uso de algoritmos estadísticos para poder validar la consistencia interna de los datos y se puedan realizar comparaciones entre sí.

En cuanto a la visualización de los resultados, se exploran diversas técnicas matemáticas para representar los análisis de manera clara, como podrían ser las Gráficos de barras, y dashboards interactivos que son de suma importancia para facilitar la vistas para todo aquel que desee usarlo como podrían ser, los administradores, así como los evaluadores y estudiantes, que no tengan conocimientos técnicos muy avanzados para poder interpretar los resultados con facilidad.

Capítulo 1

1.1 ANTECEDENTES

El servicio comunitario ha emergido como una práctica fundamental en el ámbito educativo, promoviendo no solo el compromiso social de los estudiantes, sino también su desarrollo personal y profesional. Diversas instituciones educativas han integrado programas de servicio comunitario en sus currículos, reconociendo su potencial para cultivar habilidades cívicas y fomentar una ciudadanía activa (Rondón, 2020). Esta práctica tiene como objetivo aplicar los conocimientos adquiridos durante la formación académica, beneficiando tanto a las comunidades como a los estudiantes, quienes adquieren experiencia educativa y fortalecen actitudes positivas hacia el compromiso social y el trabajo en equipo (¿Qué es servicio comunitario?, 2024).

Las actitudes de los estudiantes hacia el servicio comunitario son un factor crítico que influye en su disposición para participar en estas actividades. Definidas como las evaluaciones y sentimientos de una persona hacia un objeto o situación, las actitudes pueden impactar significativamente el compromiso en iniciativas de servicio comunitario (Concepto y naturaleza de las actitudes, s.f.). La medición de estas actitudes resulta fundamental para comprender los factores que motivan a los estudiantes a involucrarse en el servicio comunitario y, al mismo tiempo, desarrollar estrategias efectivas que incrementen su participación.

En este contexto, se ha desarrollado la Escala de Actitudes hacia el Servicio Comunitario (CSAS), la cual ha sido ampliamente utilizada en investigaciones para evaluar las actitudes de los estudiantes hacia el servicio comunitario. Esta herramienta permite obtener una visión cuantitativa de las percepciones y motivaciones de los individuos involucrados en actividades de servicio, abordando dimensiones como el interés en el servicio, la responsabilidad cívica y el compromiso social (Tumino, 2012). Además, el uso de la escala CSAS facilita la recolección de datos y proporciona una base sólida para el análisis estadístico.

El análisis de los datos obtenidos a través de la CSAS puede realizarse mediante métodos estadísticos paramétricos y no paramétricos. Los métodos paramétricos, que requieren suposiciones como la normalidad y la homogeneidad de varianzas, son útiles cuando se cumplen estas condiciones (Velázquez, 2023). Sin embargo, en casos donde estas suposiciones no se cumplen, los métodos no paramétricos, como la prueba de Kruskal-Wallis o la de Mann-Whitney, ofrecen alternativas efectivas para evaluar las diferencias en actitudes (González, 2023). La elección del método adecuado es crucial para asegurar la validez y fiabilidad de los resultados obtenidos, contribuyendo así a un entendimiento más profundo de las actitudes hacia el servicio comunitario en contextos educativos.

Dada esta situación, el presente estudio sobre las actitudes hacia el servicio comunitario, respaldado por métodos estadísticos apropiados, es esencial para fomentar la participación estudiantil en estas iniciativas y contribuir a su desarrollo integral.

1.2 INTRODUCCIÓN.

En ocasiones, la participación en el servicio comunitario es esencial para la formación integral de los estudiantes, lo que promueve una cultura de participación social y tiene un gran impacto en la comunidad.

En el presente proyecto se ha llevado a cabo una investigación sobre las actitudes hacia el servicio comunitario para comprender ciertos factores que podrían motivar o desmotivar la participación de los estudiantes. Por lo cual decidió utilizar como herramienta principal el cuestionario de escalas CSAS, que nos permite medir estas actitudes de manera estandarizada.

Sin embargo, los estudios actuales sobre las aptitudes sobre el servicio comunitario son limitados porque no exploran de manera exhaustiva las diferencias entre los diversos grupos poblacionales, que no se pueden identificar ciertos patrones complejos al representar los datos. Ya que la mayoría de estos estudios se enfocan en el análisis estadístico tradicional, pero no exploran el potencial de otras técnicas de minería de datos o análisis estadístico más avanzadas.

El objetivo del presente proyecto es la implementación de distintos algoritmos mediante el uso de un software para evaluar las ventajas y desventajas que ofrece la automatización en el análisis de datos paramétricos y no paramétricos. Se integrarán enfoques estadísticos para realizar un análisis más profundo de las comparaciones de actitudes entre los grupos y se utilizarán algoritmos de minería de datos para analizar patrones específicos.

Como resultado, se prevé la creación de una página interactiva que facilite la visualización y la comprensión de los resultados obtenidos. Esta plataforma permitirá a los usuarios que no son expertos acceder a los resultados de los cuestionarios a través de gráficos dinámicos e intuitivos, lo que podría ser una herramienta útil para interpretar los datos que requieren conocimientos avanzados.

1.3 JUSTIFICACIÓN

Importancia Académica y Necesidad de la Investigación

La investigación sobre las actitudes hacia el servicio comunitario es esencial en el ámbito académico, ya que proporciona una comprensión profunda de los factores que motivan o desmotivan la participación activa en actividades de servicio. Este proyecto propone el desarrollo de un software que facilite el análisis de estas actitudes mediante el uso de la escala CSAS, integrando metodologías avanzadas de análisis estadístico paramétrico y no paramétrico junto con técnicas de minería de datos. Esta herramienta permitirá no solo identificar patrones y tendencias complejas, sino también descubrir factores predictivos que influyen en las actitudes hacia el servicio comunitario.

La creación de este software llenará un vacío metodológico al proporcionar un análisis robusto y detallado de los datos, permitiendo a investigadores y educadores contar con una plataforma integral que evalúe y visualice los resultados de manera precisa y estandarizada. Además, este software servirá como una herramienta útil para futuras investigaciones y programas educativos, permitiendo optimizar estrategias que promuevan actitudes positivas hacia el servicio comunitario.

Por lo tanto, una vez implementados estos algoritmos, nos surge la siguiente pregunta: ¿Qué técnicas de visualización matemática y análisis gráfico pueden facilitar la interpretación de los resultados obtenidos en la plataforma? Responder a esta pregunta es clave para optimizar el acceso, la interpretación y el uso de los datos procesados por parte de investigadores y educadores.

1.4 OBJETIVOS GENERALES Y ESPECIFICOS.

Objetivo General

Desarrollar un software para el análisis y la aplicación de técnicas estadísticas paramétricas y no paramétricas para identificar patrones, tendencias y factores predictivos, facilitando una comprensión más profunda del impacto de la participación comunitaria en la percepción individual y grupal.

Objetivos Particulares

Diseñar y optimizar un modelo de datos estructurado que permita la recolección, almacenamiento y análisis eficiente de las respuestas de la escala CSAS a nivel individual y grupal. Teniendo en cuenta que se implementación de una base de datos relacional con estructuras normalizadas que permitan un análisis robusto de los datos.

Implementar algoritmos de análisis estadístico paramétrico y no paramétrico para evaluar las actitudes de servicio comunitario, tanto a nivel individual como comparativo entre grupos, garantizando la validez estadística de los resultados. Mediante la ejecución y validación de los análisis estadísticos que incluyan ANOVA, Cronbach's Alpha

Desarrollar y automatizar informes personalizables que presenten los resultados del análisis de forma clara y accesible para diferentes audiencias, utilizando técnicas de visualización efectivas, automatizando informes que incluyan gráficos y resúmenes estadísticos relevantes a través de un software.

Implementar técnicas de minería de datos para descubrir patrones y tendencias en las actitudes hacia el servicio comunitario, identificando posibles grupos o subgrupos con respuestas similares, identificación de patrones significativos mediante algoritmos de clustering o análisis de componentes principales.

Evaluar la efectividad de diferentes visualizaciones de datos para mejorar la comprensión y comunicación de los resultados del análisis, asegurando que los hallazgos sean claros y útiles para la toma de decisiones, mediante la comparación y validación de visualización (gráficos de barras, dashboards interactivos) en términos de comprensión y aplicabilidad de los datos.

1.5 ALCANCES Y LIMITACIONES.

1.5.1 Alcance del proyecto.

Este proyecto beneficiará a varias partes interesadas:

- **Instituciones educativas:** Obtendrán un marco analítico más sólido para evaluar los programas de servicio comunitario y ajustar sus currículos en función de los resultados.
- **Organizaciones no gubernamentales y comunitarias:** Podrán utilizar los hallazgos para diseñar mejores estrategias que promuevan la participación activa en el servicio comunitario, basándose en datos específicos sobre las actitudes y motivaciones de diferentes grupos poblacionales.
- **Investigadores académicos:** El enfoque metodológico que combina estadísticas avanzadas con minería de datos contribuirá al cuerpo de conocimiento en áreas relacionadas con el servicio comunitario, actitudes y comportamiento sociales.
- **Participantes del servicio comunitario:** A través de la implementación de programas más efectivos, los participantes podrán experimentar un mayor sentido de satisfacción y motivación, lo que puede incrementar su compromiso con el servicio comunitario a largo plazo.

1.5.2 Limitaciones.

- **Muestra limitada o sesgada:** Si el tamaño de la muestra no es lo suficientemente grande o representativa de la población, los resultados podrían no ser generalizables. Además, si los participantes pertenecen a un grupo con experiencias similares (por ejemplo, estudiantes de una sola universidad), esto podría limitar la diversidad de respuestas.
- **Autoinforme y sesgo de deseabilidad social:** Dado que la escala CSAS se basa en las respuestas auto informadas, los participantes pueden modificar sus respuestas para alinearse con lo que consideran socialmente deseable, lo que podría no reflejar sus verdaderas actitudes hacia el servicio comunitario.
- **Limitaciones del CSAS:** Aunque la escala CSAS es una herramienta validada, puede que no capte todas las dimensiones del servicio comunitario o que su aplicabilidad se limite a ciertos contextos culturales o sociales. Los factores contextuales que influyen en las actitudes hacia el servicio comunitario podrían variar entre culturas y no ser completamente capturados por esta escala.
- **Interpretación de datos no paramétricos:** Aunque el estudio pretende utilizar tanto análisis paramétricos como no paramétricos, los métodos no paramétricos tienen la limitación de proporcionar menos información estructurada sobre la relación entre variables, lo que podría dificultar la interpretación de algunos resultados.
- **Limitaciones en la minería de datos:** La minería de datos es poderosa para identificar patrones, pero también puede generar asociaciones espurias o sin relevancia práctica si no se interpretan cuidadosamente. Además, la complejidad de los datos puede llevar a

sobreajustes de los modelos utilizados, lo que podría dificultar la generalización de los hallazgos.

1.6 ORGANIGRAMA

Fase	Actividad	Fecha de inicio	Fecha de final
Investigación Inicial	Definir objetivos y alcance del proyecto	12/08/2024	16/08/2024
	Revisión de literatura y casos similares	17/08/2024	23/08/2024
	Reunión con asesores y definición de requerimientos técnicos	24/08/2024	30/08/2024
Planificación del Proyecto	Elaborar cronograma detallado y asignación de recursos	31/08/2024	05/09/2024
Diseño del Frontend	Diseño de la interfaz de usuario	06/09/2024	12/09/2024
	Implementación de formularios para el test CSAS	13/09/2024	18/09/2024
	Diseño e implementación de visualizaciones gráficas	19/09/2024	25/09/2024
Diseño del Backend	Modelado de la base de datos	26/09/2024	01/10/2024
	Diseño de la API y endpoints	02/10/2024	07/10/2024
	Desarrollo del módulo de autenticación y gestión de usuarios	08/10/2024	14/10/2024
	Implementación del módulo de	15/10/2024	21/10/2024

	administración de grupos y encuestas		
	Desarrollo de almacenamiento de resultados y seguimiento longitudinal	22/10/2024	28/10/2024
Integración	Integración del sistema frontend con backend	29/10/2024	05/11/2024
Pruebas y Validación	Pruebas de funcionalidad del sistema	06/11/2024	12/11/2024
	Pruebas de usabilidad y recolección de retroalimentación de usuarios	13/11/2024	19/11/2024
	Ajustes y corrección de errores	20/11/2024	26/11/2024
Documentación y Entrega	Elaboración de documentación técnica del sistema	27/11/2024	01/12/2024
	Redacción del informe final del proyecto	02/12/2024	08/12/2024
	Preparación de la presentación final y defensa del proyecto	09/12/2024	14/12/2024

Tabla de organigrama

1.6.1 DESCRIPCIÓN DETALLADA DE LAS ACTIVIDADES

1. Investigación Inicial

Definir objetivos y alcance del proyecto:

Definir el objetivo general de analizar las actitudes hacia el servicio comunitario usando la escala CSAS, aplicando técnicas estadísticas avanzadas.

Establecer los objetivos particulares como la implementación de algoritmos paramétricos y no paramétricos, la automatización de informes, y la evaluación de visualizaciones.

Delimitar el alcance en términos de participantes, así como las asignaciones de roles en la plataforma y el tipo de análisis a realizar.

Revisión de literatura y casos similares:

Revisar estudios previos sobre actitudes hacia el servicio comunitario, el uso de la escala CSAS, y metodologías de análisis estadístico aplicadas en estudios similares.

Analizar investigaciones que implementen minería de datos y técnicas de visualización aplicadas a estudios de actitudes o comportamiento social.

1.Reunión con asesores y definición de requerimientos técnicos:

- 1.1 Colaborar con asesores académicos y técnicos para definir los lineamientos metodológicos y estadísticos.
- 1.2 Definir los requerimientos del sistema, incluyendo las tecnologías que se usarán para el frontend, backend, y la gestión de datos.
- 1.3 Identificar los recursos y las herramientas necesarias, como bases de datos, lenguajes de programación.

2. Planificación del Proyecto

- **Elaborar cronograma detallado y asignación de recursos:**

- Crear un cronograma que cubra todas las fases del proyecto desde la investigación hasta la integración, pruebas, y documentación.
- Asignar recursos a cada tarea específica, incluyendo tiempos de trabajo, herramientas y tecnologías, y la participación de los asesores.

3. Diseño del Frontend

Diseño de la interfaz de usuario:

- Crear una interfaz sencilla e intuitiva para la recolección de datos, dirigida a los participantes que responderán a la escala de Likert

- Diseñar pantallas de visualización de resultados que sean fáciles de interpretar para diferentes audiencias.

Implementación de formularios para el test CSAS:

- Crear formularios interactivos que permitan a los usuarios responder de manera eficiente, asegurando una experiencia de usuario fluida.
- Implementar validaciones en los formularios para asegurar la consistencia y completitud de las respuestas.

Diseño e implementación de visualizaciones gráficas:

- Diseñar visualizaciones claras y efectivas que representen los resultados del análisis de datos, como gráficos de barras, diagramas de dispersión, y mapas de calor.
- Utilizar librerías para integrar visualizaciones interactivas en el frontend.
- 4. Diseño del Backend
- Modelado de la base de datos:
- Diseñar una base de datos relacional optimizada que almacene las respuestas individuales y las características de los participantes.
- Definir las tablas necesarias para registrar información demográfica, grupos de participantes, resultados de análisis, y metadatos.
- Diseño de la API y endpoints:
- Crear una API RESTful que permita la comunicación entre el frontend y el backend.
- Definir endpoints que gestionen las respuestas a los formularios, el almacenamiento en la base de datos, y la recuperación de datos para el análisis estadístico.
- Desarrollo del módulo de autenticación y gestión de usuarios:
- Implementar un sistema de autenticación segura para usuarios, permitiendo el acceso de administradores, asesores y participantes.
- Gestionar la creación y administración de cuentas de usuario y roles (administrador, maestro, alumno).
- Implementación del módulo de administración de grupos y encuestas:

- Desarrollar un módulo que permita crear, modificar y asignar encuestas a grupos específicos de participantes.
- Implementar la capacidad de administrar grupos de manera flexible, permitiendo la comparación de resultados entre diferentes cohortes.
- 5. Integración
- Integración del sistema frontend con backend:
- Asegurar que la interfaz de usuario del frontend esté correctamente conectada con el backend a través de la API, permitiendo una experiencia fluida para los usuarios.
- Pruebas de funcionalidad del sistema:
- Realizar pruebas funcionales para asegurarse de que todas las características del sistema, desde la recolección de datos hasta la generación de informes, funcionen correctamente.
- Pruebas de usabilidad y recolección de retroalimentación de usuarios:
- Ejecutar pruebas de usabilidad con usuarios finales para identificar posibles áreas de mejora en la interfaz y en la experiencia general.
- Recoger retroalimentación de asesores y participantes para realizar ajustes en el sistema.
- Ajustes y corrección de errores:
- Realizar los ajustes necesarios y corregir los errores detectados durante las pruebas, mejorando la estabilidad y funcionalidad del sistema.
- 7. Documentación y Entrega
- Elaboración de documentación técnica del sistema:
- Crear documentación detallada sobre el diseño y desarrollo del sistema, incluyendo diagramas de la arquitectura, especificaciones de la API, y la estructura de la base de datos.
- Incluir guías para la instalación, despliegue y uso del sistema, así como procedimientos de mantenimiento.
- Incluir análisis de datos estadísticos, visualizaciones, patrones descubiertos mediante minería de datos y recomendaciones basadas en los hallazgos.

Capítulo 2

2.1 MARCO TEÓRICO

2.1.1. El Servicio Comunitario en el Contexto Educativo

El servicio comunitario es una actividad clave en la formación integral de los estudiantes, promoviendo el desarrollo de habilidades técnicas y sociales. A través de esta experiencia, los estudiantes aplican sus conocimientos académicos en situaciones reales, beneficiando tanto a la comunidad como a su propio desarrollo personal y profesional (Rondón, 2020).

Las instituciones educativas han implementado el servicio comunitario en sus planes de estudio como una estrategia para fomentar una ciudadanía activa, ética y socialmente responsable. Esta práctica refuerza el aprendizaje experiencial, generando un impacto positivo en las comunidades al tiempo que desarrolla competencias clave en los estudiantes, como el trabajo en equipo, la empatía y la resolución de problemas (¿Qué es servicio comunitario?, 2024).

2.1.2. Actitudes y su Importancia en el Servicio Comunitario

Las actitudes, definidas como evaluaciones subjetivas hacia un objeto, persona o situación, desempeñan un papel crítico en el comportamiento humano (Eagly & Chaiken, 1993). En el contexto del servicio comunitario, las actitudes de los estudiantes hacia estas actividades influyen directamente en su disposición a participar y su compromiso con las iniciativas. Según Algor Education (s.f.), estas actitudes están moldeadas por factores como experiencias previas, valores culturales y percepciones personales.

Medir estas actitudes es crucial para entender las motivaciones y barreras que enfrentan los estudiantes. Los resultados permiten a los educadores y diseñadores de programas desarrollar estrategias efectivas para fomentar una mayor participación en actividades de servicio comunitario (Concepto y naturaleza de las actitudes, s.f.).

2.1.3. Escala CSAS: Herramienta para la Medición de Actitudes

La Escala de Actitudes hacia el Servicio Comunitario (CSAS) es una herramienta validada que mide las actitudes de los estudiantes en tres dimensiones principales: interés en el servicio, responsabilidad cívica y compromiso social (Tumino, 2012).

Esta escala permite obtener datos cuantitativos que facilitan el análisis estadístico, proporcionando una base sólida para identificar patrones y tendencias en las respuestas de los participantes. Además, la CSAS es adaptable a diferentes contextos culturales y educativos, lo que la convierte en una herramienta versátil para la investigación social.

2.1.4 Comparación entre la Recolección de Datos Manual y Digital

La recolección de datos es el proceso de recopilación y medición de información sobre variables establecidas de una manera sistemática, que permita obtener respuestas relevantes, probar

hipótesis y evaluar resultados. La recolección de datos en el proceso de investigación es común a todos los campos de estudio. Si bien los métodos varían según la disciplina, el énfasis es garantizar una recopilación precisa y confiable.

El uso de herramientas digitales especializadas, como el software diseñado para la recolección de datos en la escala CSAS, ha permitido optimizar el proceso al reducir errores humanos, asegurar la integridad de la información y automatizar el análisis estadístico. Además, la digitalización facilita la aplicación de métodos paramétricos y no paramétricos sin necesidad de convertir manualmente los datos. Estas ventajas han sido documentadas en estudios sobre la automatización de encuestas y el impacto de la digitalización en la confiabilidad de los resultados (Velázquez, 2023).

2.1.5. Análisis Estadístico de Datos

2.1.5.1 Métodos Paramétricos

Los métodos paramétricos, como el análisis de varianza (ANOVA) y la prueba t de Student, son ampliamente utilizados en la investigación cuantitativa debido a su solidez estadística y capacidad de detectar diferencias significativas entre grupos. El ANOVA permite comparar múltiples grupos mediante la evaluación de la variabilidad dentro y entre ellos, mientras que la prueba t de Student se emplea cuando se analizan dos grupos de manera independiente o pareada (Velázquez, 2023).

Otro enfoque paramétrico relevante es la regresión lineal, utilizada para modelar la relación entre una variable dependiente y una o más variables independientes. Este método es útil para predecir tendencias y evaluar la influencia de diferentes factores en un fenómeno determinado (Martínez, 2022).

2.1.5.2 Métodos No Paramétricos

Cuando las variables no siguen una distribución normal o el tamaño muestral es reducido, se pueden emplear métodos no paramétricos. En este contexto, la prueba de Chi-cuadrada se utiliza para evaluar la independencia entre dos variables categóricas, analizando si la distribución observada difiere significativamente de la esperada. Este método es común en estudios de comportamiento y ciencias sociales, ya que permite determinar asociaciones sin requerir suposiciones sobre la distribución de los datos (González, 2023).

Por otro lado, la regresión logística es un modelo estadístico adecuado para predecir variables categóricas, como la probabilidad de que un individuo participe en actividades de servicio comunitario en función de sus características demográficas y actitudes previas. Esta técnica es ampliamente utilizada en estudios de predicción y clasificación, ya que permite modelar relaciones entre variables sin necesidad de asumir una distribución normal en los datos (Ruiz, 2023).

2.1.5.3 Técnicas de Minería de Datos

La minería de datos complementa el análisis estadístico al identificar patrones complejos en grandes conjuntos de datos. Técnicas como el clustering y el análisis de componentes principales

son valiosas para segmentar grupos de estudiantes con actitudes similares y descubrir factores predictivos de participación en servicio comunitario (Han et al., 2012).

2.1.6 Herramientas Tecnológicas para Análisis y Visualización

2.1.6.1 MySQL

Como sistema de gestión de bases de datos relacionales, MySQL permite almacenar y procesar grandes volúmenes de datos de manera eficiente, garantizando la integridad y seguridad de la información.

2.1.6.2 Angular

Angular es un framework de desarrollo frontend que facilita la creación de interfaces dinámicas e interactivas, como dashboards que visualizan resultados estadísticos de forma intuitiva (Hiberus, s.f.).

2.1.6.3 Laravel

Laravel es un framework backend de PHP que permite construir aplicaciones web robustas, con especial énfasis en la seguridad y la escalabilidad (Axarnet, s.f.).

2.1.6.4 JSON Web Tokens (JWT)

JWT es una tecnología estándar que asegura la autenticación de usuarios y la transmisión de datos entre sistemas, garantizando la integridad de la información (Auth0, s.f.).

2.1.7 Metodología Scrum: Un Enfoque Ágil para el Desarrollo

La metodología Scrum es un marco ágil que permite gestionar proyectos de manera iterativa y flexible. Este enfoque fomenta la colaboración entre equipos y asegura entregas continuas y de alta calidad. En el presente proyecto, Scrum ha facilitado la integración de múltiples tecnologías y el cumplimiento de los objetivos planteados (APD, s.f.).

Capítulo 3

3.1 MATERIALES Y METODOS.

En el presente capítulo se realizará la descripción breve pero detallada sobre los programas, técnicas, así como los lenguajes de programación que se aplicaron durante dicho proyecto. Esto teniendo en cuenta su enfoque principal el cual nos garantiza el entendimiento con respecto a los resultados.

A lo largo del desarrollo del sistema utilizamos varias herramientas, así como lenguajes de programación para consolidar el proyecto.

3.1.1 recursos y software

3.1.1.1 Mysql



Imagen MySQL 3.1.1.1

MySQL es una de las bases de datos relacionales de software libre más populares del mundo. Debe esta popularidad a su uso generalizado en sitios web de comercio electrónico, redes sociales y aplicaciones, entre los que se incluyen Drupal, Joomla, Magento y WordPress. También es una parte esencial de la pila de aplicaciones web Linux-Apache-MySQL-PHP/Perl/Python (LAMP), de uso generalizado, que pone en marcha muchas aplicaciones, sitios web y servicios populares.

Ventajas de uso

- **Alta disponibilidad:** en MySQL hace referencia a la capacidad que tiene el motor de la base de datos para funcionar durante un periodo prolongado sin fallos. La alta disponibilidad en MySQL puede ser muy compleja de configurar y depende de los requisitos de disponibilidad específicos de cada usuario y de cómo despliegue MySQL. Para configurar la alta disponibilidad en MySQL, los usuarios tendrán que tener en cuenta la replicación de datos, la detección de fallos, los mecanismos de conmutación por error o por recuperación y el redireccionamiento del tráfico de la base de datos a la instancia secundaria después de una conmutación por error. Cloud SQL para MySQL consigue que la alta disponibilidad sea un proceso sencillo.
- **Seguridad:** las consideraciones de seguridad en MySQL pueden abarcar una amplia gama de factores. Algunos de los más importantes están relacionados con la protección

de datos, como la protección de datos contra la corrupción mediante mecanismos de redundancia de datos y procesos de seguridad general, como contraseñas y privilegios, control de acceso y seguridad de red. Descubre más información sobre la seguridad en Cloud SQL para MySQL

- **Copia de seguridad y restauración:** MySQL admite la copia de seguridad y la recuperación de datos a través de varios mecanismos, como las herramientas de terceros. Algunas de las formas de crear una copia de seguridad y restaurar MySQL son la utilidad `mysqldump`, las copias de seguridad incrementales con registros binarios y la replicación. Cloud SQL para MySQL ofrece sus propias funciones de copia de seguridad y restauración para las copias de seguridad automatizadas o bajo demanda.
- **Flexibilidad:** Es fácil añadir, actualizar o eliminar tablas y relaciones y hacer otros cambios en los datos siempre que lo necesites sin tener que modificar la estructura general de la base de datos y sin que afecte a las aplicaciones que ya tienes.
- **Facilidad de uso:** Es fácil ejecutar consultas complejas con SQL, por lo que los usuarios con menos experiencia pueden interactuar de forma intuitiva con la base de datos.
- **Rendimiento:** la optimización del rendimiento es un aspecto fundamental a la hora de gestionar cualquier base de datos. MySQL proporciona muchas funciones y opciones de ajuste para facilitar el desarrollo de aplicaciones con un rendimiento excelente. Si quieres obtener más información sobre estas funciones, consulta el artículo sobre consejos para optimizar el rendimiento de MySQL. En este otro artículo sobre el ajuste de consultas también se proporcionan prácticas recomendadas específicas para mejorar el rendimiento.
- **Mejora en la seguridad y la gestión de accesos:** Con controles precisos sobre quién puede ver y modificar el código, protege los proyectos contra accesos no autorizados y posibles amenazas. La capacidad de configurar permisos detallados y recibir alertas de seguridad contribuye a mantener la integridad y la seguridad del código, siendo esencial para proteger la información y los activos del proyecto.

3.1.1.2 Angular:



Imagen Angular 3.1.1.2.1

Es una plataforma de desarrollo, construida sobre TypeScript. Es un framework basado en componentes para crear aplicaciones web escalables. Una colección de bibliotecas bien integradas que cubren una amplia variedad de características, que incluyen enrutamiento, administración de formularios, comunicación cliente-servidor y más. Un conjunto de herramientas para desarrolladores que permiten desarrollar, compilar, probar y actualizar el código fuente de la aplicación.

Con Angular, estás aprovechando una plataforma que puede escalar desde proyectos de un solo desarrollador hasta aplicaciones de nivel empresarial. Angular está diseñado para que la actualización sea lo más sencilla posible, y así aprovechar los últimos desarrollos con un mínimo de esfuerzo.

Ventajas de uso:

- **Componentes personalizados:** Los usuarios tienen total libertad de construir sus propios componentes que pueden empaquetar la funcionalidad junto con la lógica de renderizado en piezas reutilizables en el código
- **Enlace de datos:** Una de las grandes ventajas es que permite a los usuarios mover datos sin esfuerzo desde el código JavaScript a la vista y reaccionar a los eventos del usuario sin tener que escribir ningún código manualmente.
- **Inyección de dependencia:** Angular permite a los usuarios escribir servicios modulares e inyectarlos donde sea que se necesiten. Esto mejora la capacidad de prueba y la reutilización de los mismos servicios.
- **Pruebas:** Las pruebas son herramientas de primera clase y Angular no es la excepción en esto, se ha creado desde cero teniendo en cuenta la capacidad de prueba. Este permite realizar pruebas a cada parte de su aplicación, lo cual es muy recomendable.
- **Integral:** Angular es un framework completo y proporciona soluciones listas para usar para la comunicación del servidor, el enrutamiento dentro de su aplicación y más.
- **Compatibilidad del navegador:** Angular es multiplataforma y compatible con muchos navegadores. Una aplicación angular normalmente se puede ejecutar en todos los navegadores (por ejemplo; Chrome, Firefox) y sistemas operativos, como Windows, macOS y Linux.

3.1.1.3 GitHub



Imagen Github 3.1.1.3.1

GitHub es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprada por Microsoft en junio del 2018. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargar la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

Como su nombre indica, la web utiliza el sistema de control de versiones Git diseñado por Linus Torvalds. Un sistema de gestión de versiones es ese con el que los desarrolladores pueden administrar su proyecto, ordenando el código de cada una de las nuevas versiones que sacan de sus aplicaciones para evitar confusiones. Así, al tener copias de cada una de las versiones de su aplicación, no se perderán los estados anteriores cuando se va a actualizar.

Así pues, Git es uno de estos sistemas de control, que permite comparar el código de un archivo para ver las diferencias entre las versiones, restaurar versiones antiguas si algo sale mal, y fusionar los cambios de distintas versiones. También permite trabajar con distintas ramas de un proyecto, como la de desarrollo para meter nuevas funciones al programa o la de producción para depurar los bugs.

Ventajas de uso

- **Control total sobre el proyecto:** GitHub proporciona un control total sobre el proyecto para gestionar y seguir el historial completo del código. Esta visibilidad mantiene todo organizado eliminando errores y potenciando el entendimiento de cómo y por qué se hicieron ciertos cambios.
- **Mejora en la colaboración del equipo:** El soporte facilita la comunicación y la coordinación mediante herramientas como pull requests y revisiones de código para que los cambios propuestos sean revisados y aprobados antes de integrarse en el proyecto principal, lo que reduce la probabilidad de errores y conflictos.
- **Aceleración del ciclo de desarrollo:** Al reducir la necesidad de intervención manual en estos procesos, GitHub ayuda a que se concentren en el desarrollo de nuevas características y la solución de problemas, acelerando así el tiempo de entrega del software y mejorando la eficiencia general del equipo.
- **Facilitación de la contribución abierta:** Los desarrolladores contribuyen a proyectos de otros y comparten su propio código con una comunidad global. Esta apertura no solo promueve la colaboración y el aprendizaje, sino que también abre puertas a beneficios de

las contribuciones de otros, enriqueciendo así la calidad del software y acelerando la innovación en el campo tecnológico.

- **Reducción de costos:** Al ofrecer una plataforma que integra control de versiones, gestión de proyectos y herramientas de colaboración en un solo lugar, el sistema reduce la necesidad de múltiples herramientas separadas. Asimismo, la automatización de tareas y la mejora en la eficiencia del equipo ayudan a reducir el tiempo de desarrollo, lo que a su vez disminuye los costos asociados con el desarrollo de software.

4.1.1.4 Laravel



Imagen Laravel 4.1.1.4.1

Laravel es un framework de PHP y es utilizado para desarrollar aplicaciones web, creando un entorno de trabajo y proporciona herramientas a los desarrolladores para ayudarles a desarrollar en PHP.

Lo que se busca con Laravel es construir aplicaciones sólidas y estables, que sean fáciles de desarrollar y la utilización de parte del código preprogramada, para que pueda aprovecharse y reutilizarse, evitando así la reescritura del código en la misma aplicación.

Gracias a esto se consiguen aplicaciones con un código estable, sencillo de actualizar y con la posibilidad de añadir nuevas funcionalidades sin necesidad de modificar el código base, por medio de un sistema de paquetes modulares.

Ventajas de uso

- **Laravel es sencillo de utilizar y aprender:** Una de las cualidades más apreciadas de Laravel es que resulta sencillo de aprender, respecto a otros frameworks. No por esto significa que sea menos potente que otras opciones, pero sí que es más fácil de aprender, así que muchos desarrolladores elijen Laravel como su principal opción a la hora de crear sus proyectos.
- **Extensa documentación y comunidad enorme:** Otro punto a favor de Laravel es la extensa documentación que el desarrollador tiene disponible en la red. También existen tutoriales para todo tipo de usuarios, ya sean principiantes como más avanzados. Si tienes alguna duda o problema con algo, posiblemente ya le habrá pasado antes a alguien y muy posiblemente ya esté documentado.
- **Librerías y paquetes disponibles:** Laravel dispone de una gran cantidad de librerías y paquetes, ya sean los oficiales, como los que aporta la comunidad, que puedes aprovechar para añadir en tus distintos proyectos. Esto simplifica el tiempo que se tarda en el proceso de desarrollo de una
- **Seguridad y fortaleza:** La seguridad es muy importante en cualquier proyecto web y Laravel es muy sólido en este aspecto. Todas las aplicaciones web desarrolladas con Laravel tienen una defensa muy alta frente a las amenazas más comunes, como el malware o las inyecciones de SQL maliciosas.

3.1.1.5 PHP



Imagen PHP 3.1.1.5.1

El Preprocesador de Hipertexto (PHP - Hypertext Preprocessor) que antes significaba Páginas Personales es un lenguaje de programación del lado del servidor gratuito y de código abierto muy utilizado en el desarrollo de sitios web estáticos, dinámicos o aplicaciones web. De hecho, alrededor del 78% de todos los sitios, incluidos los de gran tráfico como Facebook y Wikipedia están contruidos con PHP.

La primera versión de PHP se lanzó hace 26 años. Ahora está en la versión 8.2, lanzada en diciembre de 2022. Este lenguaje se ejecuta en el motor Zend, la cual es la implementación más popular. PHP se utiliza sobre todo para crear servidores web. Se ejecuta en el navegador y también es capaz de ejecutarse en la línea de comandos. Por tanto, si prefieres no mostrar la salida de tu código en el navegador, puedes mostrarlo en la terminal.

Ventajas de uso

- **Multiplataforma:** El lenguaje PHP es independiente de cualquier plataforma. Esto significa que no es necesario tener un sistema operativo concreto para utilizarlo, ya que funciona en todas las plataformas, como Mac, Windows o Linux.
- **Código abierto:** PHP es de código abierto. El código original se pone a disposición de todo aquel que quiera basarse en este. De hecho, esta es una de las razones por las que uno de sus frameworks, Laravel, es tan popular.
- **Fácil de aprender:** Aunque es un lenguaje de programación, PHP no es difícil de aprender para los principiantes absolutos. Por tanto, si tienes conocimientos básicos, podrías aprenderlo muy bien. Asimismo, hay diversos cursos de PHP desde cero que podrías estudiar.
- **Sincronización con todas las Bases de Datos:** Puedes conectar fácilmente PHP a todas las bases de datos, relacionales y no relacionales. Por tanto, se puede conectar en cualquier momento a MySQL, Postgress, MongoDB o cualquier otra base de datos.

Comunidades de Apoyo: PHP tiene una comunidad en línea de gran apoyo. La documentación oficial proporciona guías sobre cómo utilizar las características y puedes conseguir fácilmente tu problema fijado y resuelto.

3.1.1.6 JSON Web Tokens



Imagen JSON Web Tokens 3.1.1.6.1

JWT (JSON Web Token) es un estándar que está dentro del documento RFC 7519.

En el mismo se define un mecanismo para poder propagar entre dos partes, y de forma segura, la identidad de un determinado usuario, además con una serie de claims o privilegios.

Estos privilegios están codificados en objetos de tipo JSON, que se incrustan dentro de del payload o cuerpo de un mensaje que va firmado digitalmente.

Ventajas de uso

- **Autenticación:** Esta es la situación típica para el uso de JWT, una vez que el usuario ha iniciado sesión, cada solicitud posterior incluirá el JWT, permitiendo al usuario acceder a las rutas, servicios y recursos que están permitidos con ese token. El inicio de sesión único es una característica que utiliza ampliamente JWT hoy en día, debido a su pequeña sobrecarga y su capacidad para ser utilizado fácilmente entre los sistemas de diferentes dominios.
- **Intercambio de información:** Los JWT son una buena forma de transmitir información de forma segura entre las partes, porque como se pueden firmar, por ejemplo utilizando un par de claves públicas/privadas, se puede estar seguro de que el remitente es quien dice ser. Además, como la *firma se calcula utilizando el encabezado y la carga útil, también puedes verificar que el contenido no ha cambiado.*

3.1.1.7 JavaScript



Imagen JavaScript 3.1.1.7.1

JavaScript es un lenguaje de programación, de secuencias de comandos, capaz de aportar soluciones eficaces en la mayoría de los ámbitos de la tecnología. Te permite crear contenido de actualización dinámica, controlar multimedia, animar imágenes, etc.

Ventajas de uso

- **Simplicidad:** Posee una estructura sencilla que lo vuelve más fácil de aprender e implementar.
- **Velocidad:** Se ejecuta más rápido que otros lenguajes y favorece la detección de los errores.
- **Versatilidad:** Es compatible con otros lenguajes, como: PHP, Perl y Java. Además, hace que la ciencia de datos y el aprendizaje automático sean accesibles.
- **Popularidad:** Existen numerosos recursos y foros disponibles para ayudar a los principiantes con habilidades y conocimientos limitados.
- **Carga del servidor:** La validación de datos puede realizarse a través del navegador web y las actualizaciones solo se aplican a ciertas secciones de la página web.
- **Actualizaciones:** Se actualiza de forma continua con nuevos frameworks y librerías, esto le asegura relevancia dentro del sector.

3.1.1.8 TypeScript



Imagen TypeScript 3.1.1.8.1

TypeScript es un lenguaje de programación libre y de código abierto. Fue creado por Anders Hejlsberg, diseñador de C# y creador de Delphi y Turbo Pascal, y es desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript que añade tipado estático opcional y funciones avanzadas a JavaScript. TypeScript es usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor, o extensiones para programas (Node.js y Deno). Además, TypeScript admite conceptos de OOP como clases, interfaces y herencia, lo que permite un código estructurado y organizado. En resumen, TypeScript es una herramienta valiosa para los desarrolladores que buscan mejorar la calidad y la escalabilidad de sus aplicaciones web, y que desean aprovechar las características avanzadas de JavaScript.

Ventajas de uso

- **Detecta errores en una fase temprana:** A medida que se va escribiendo el código, TypeScript va a proporcionar un reporte con errores o mejoras que se pueden realizar.
- **Amplía la funcionalidad de JavaScript:** TypeScript permite utilizar funciones y conceptos de programación avanzados de manera sencilla, como clases, interfaces, enumeraciones, entre otros.
- **Proporciona una mejor experiencia de desarrollo para grandes proyectos:** TypeScript es especialmente útil para proyectos grandes con muchos desarrolladores, ya que puede ayudar a mantener el código organizado y fácil de entender.
- **Integración con librerías y frameworks:** TypeScript es compatible con muchas librerías y frameworks de JavaScript, lo que puede facilitar la integración de estas en tus proyectos.

3.2 Implementación de la metodología del proyecto

En dicho proceso decidimos utilizar lo que vendrían siendo la metodología de scrum, el cual nos permitiría gestionar hasta cierto punto de forma ágil y flexible, asegurando con esto la entrega continua en cada interacción, teniendo en cuenta que con esta metodología nos estaría favoreciendo la colaboración, así como la comunicación constante con el resto del equipo

A lo largo del proyecto se fueron realizando diversas reuniones para ir dando a conocer los avances del proyecto, esto con el fin de que todos los miembros del equipo pudieran compartir avances, obstáculos y contratiempos obtenido en cada intervalo de reunión o de entrega de avances

Rol	Descripción	Responsable
Scrum Master	Responsable del proyecto	DR Marlene Méndez Moreno
Development Team	Desarrollador del proyecto	Ing.Felipe Ismael Maas Canul

Tabla 3.2.1

Para plasmar los requerimientos necesarios en dicho proyecto, fue necesario el tenernos que reunir varias veces con el resto del equipo de trabajo, dichas reuniones fueron de suma importancia para llegar a dichas necesidades, así como ciertas expectativas hacia los usuarios finales

A continuación, demostraremos una lista de procesos contemplados

1. Gestión de Usuarios:

Esta sección se encarga de todo lo relacionado con los usuarios que interactúan con la plataforma, ya sea como administradores o estudiantes que responden los cuestionarios:

- **Registro y autenticación:** Los usuarios deben poder registrarse e iniciar sesión en el sistema. Esto para asignar ciertos puntos específicos como podrían ser, generación, grupo, institución etc, etc
- **Roles y permisos:** Definir roles como administrador, creador de cuestionarios o encuestados, y asegurarse de que cada rol tenga los permisos adecuados (por ejemplo, los administradores pueden crear, editar y eliminar cuestionarios, mientras que los estudiantes solo pueden responder).
- **Gestión de perfiles:** Los usuarios deben poder editar su información personal, cambiar su contraseña, y ver el historial de cuestionarios completados.

2. Gestión de Cuestionarios:

Aquí es donde se manejan todos los aspectos relacionados con la creación, edición y administración del cuestionario CSAS.

- **Creación de cuestionarios:** Los administradores pueden crear nuevos cuestionarios, añadiendo preguntas con escalas de Likert

- **Edición y personalización:** Los cuestionarios pueden ser editados, permitiendo cambiar preguntas, ajustar las escalas o definir nuevos criterios, teniendo en cuenta ciertos factores que podrían afectar las encuestas, como podrían ser, la institución, el tiempo de servicio realizado, incluso el tipo de servicio.
- **Publicación de cuestionarios:** Una vez creados, los cuestionarios se pueden publicar para que los estudiantes accedan y respondan.
- **Borradores y plantillas:** Los administradores pueden guardar cuestionarios como borradores para terminarlos más tarde o reutilizar plantillas de cuestionarios previos.

3. Gestión de Grupos:

Esta funcionalidad permite organizar a los encuestados o estudiantes en grupos para facilitar el análisis y la administración de los cuestionarios. En una plataforma de cuestionarios CSAS, puede incluir:

- **Creación de grupos:** Agrupar a los encuestados por clases, secciones o áreas de interés para aplicar cuestionarios específicos.
- **Asignación de cuestionarios a grupos:** Asignar cuestionarios a grupos específicos, permitiendo que diferentes grupos de encuestados accedan a diferentes cuestionarios.
- **Monitoreo del progreso:** Los administradores pueden ver el progreso de cada grupo en términos de cuestionarios completados o pendientes.
- **Comunicación con grupos:** Opciones para enviar notificaciones o recordatorios a grupos de encuestados acerca de la fecha límite para completar los cuestionarios.

4. Análisis y Reportes:

Esta sección es crucial para la interpretación de los datos obtenidos a través de los cuestionarios CSAS. Las principales funcionalidades incluyen:

- **Análisis descriptivo:** Ofrecer estadísticas básicas como la media, mediana, moda, varianza, y distribución de las respuestas para cada pregunta del cuestionario.
- **Reportes personalizados:** Generar reportes detallados sobre los resultados de los cuestionarios, desglosados por grupos o a nivel individual.
- **Visualización de datos:** Proveer gráficos y visualizaciones (como gráficos de barras, tortas o diagramas de dispersión) para representar los resultados de manera clara y comprensible.
- **Comparación entre grupos:** Permitir comparar los resultados de diferentes grupos, por ejemplo, para evaluar diferencias entre grupos de estudiantes de distintas clases o secciones.
- **Exportación de reportes:** Opción para exportar los datos en diferentes formatos, facilitando la generación de informes y la presentación de resultados.

Tipos de usuario	Descripción
Estudiantes	Son los que tendrán la tarea de responder las encuestas y visualizar su avance
Evaluadores	Son los que tendrán la tarea de diseñar y configurar los cuestionarios CSAS, y los que analizarán los resultados y tendrán que generar reportes
Administrativos	Son los que tendrán la tarea de gestionar la plataforma, administraran usuarios, proyectos y cuestionarios, así como generar reportes de acuerdo a lo necesario

Tabla 3.2.2

3.3 Requerimientos

En este apartado ya podemos dar una descripción detallada con respecto a los requerimientos propuestos, esto con ayuda de diagramas, de igual forma se clasificarán en dos puntos. Los requerimientos funcionales el cual describen las características y funciones específicas que el sistema, mientras que los requerimientos no funcionales se refieren a las propiedades y restricciones que debe cumplir el sistema en términos de rendimiento.

3.3.1 Diagrama de uso del estudiante

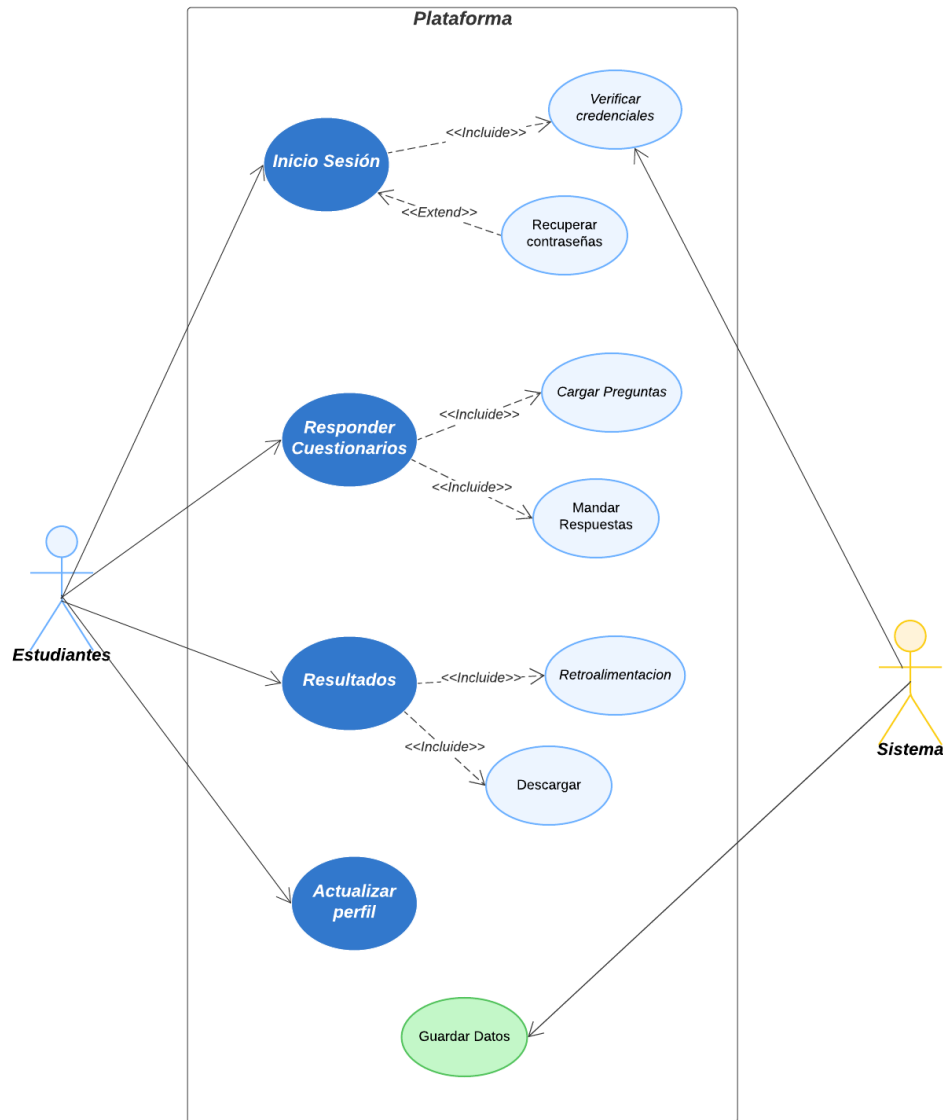


Imagen 3.3.1 Diagrama de uso del estudiante

Este diagrama muestra cómo un estudiante utiliza la plataforma para autenticarse, responder cuestionarios, revisar sus resultados y actualizar su perfil, mientras que el sistema apoya en tareas de verificación y almacenamiento de datos de manera automatizada.

3.3.2 Diagrama de inicio de sesión

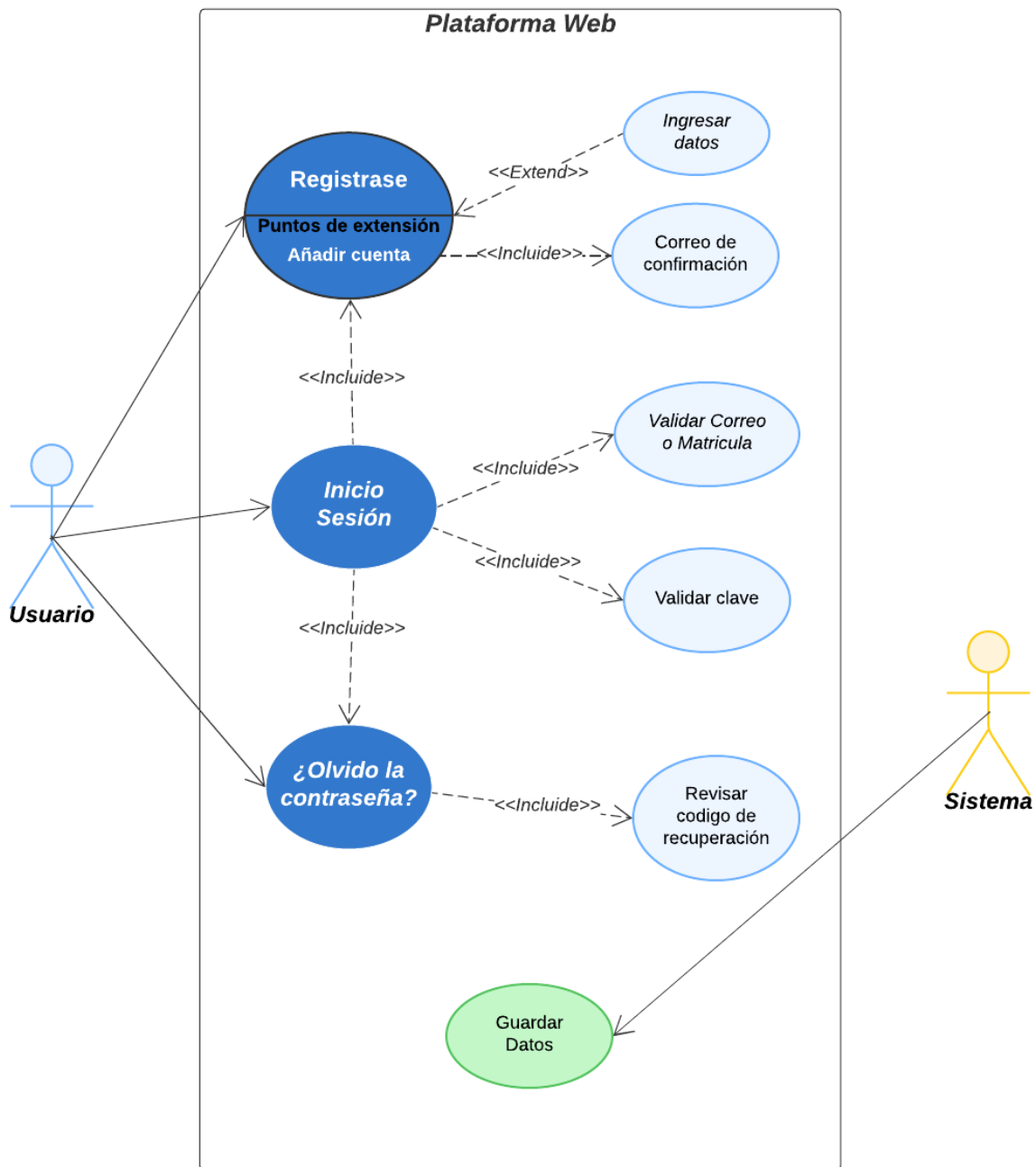


Imagen 3.3.2 Diagrama de inicio de sesión

Este diagrama muestra cómo un estudiante utiliza la plataforma para autenticarse, responder el cuestionario

3.3.3 Diagrama de uso del administrador

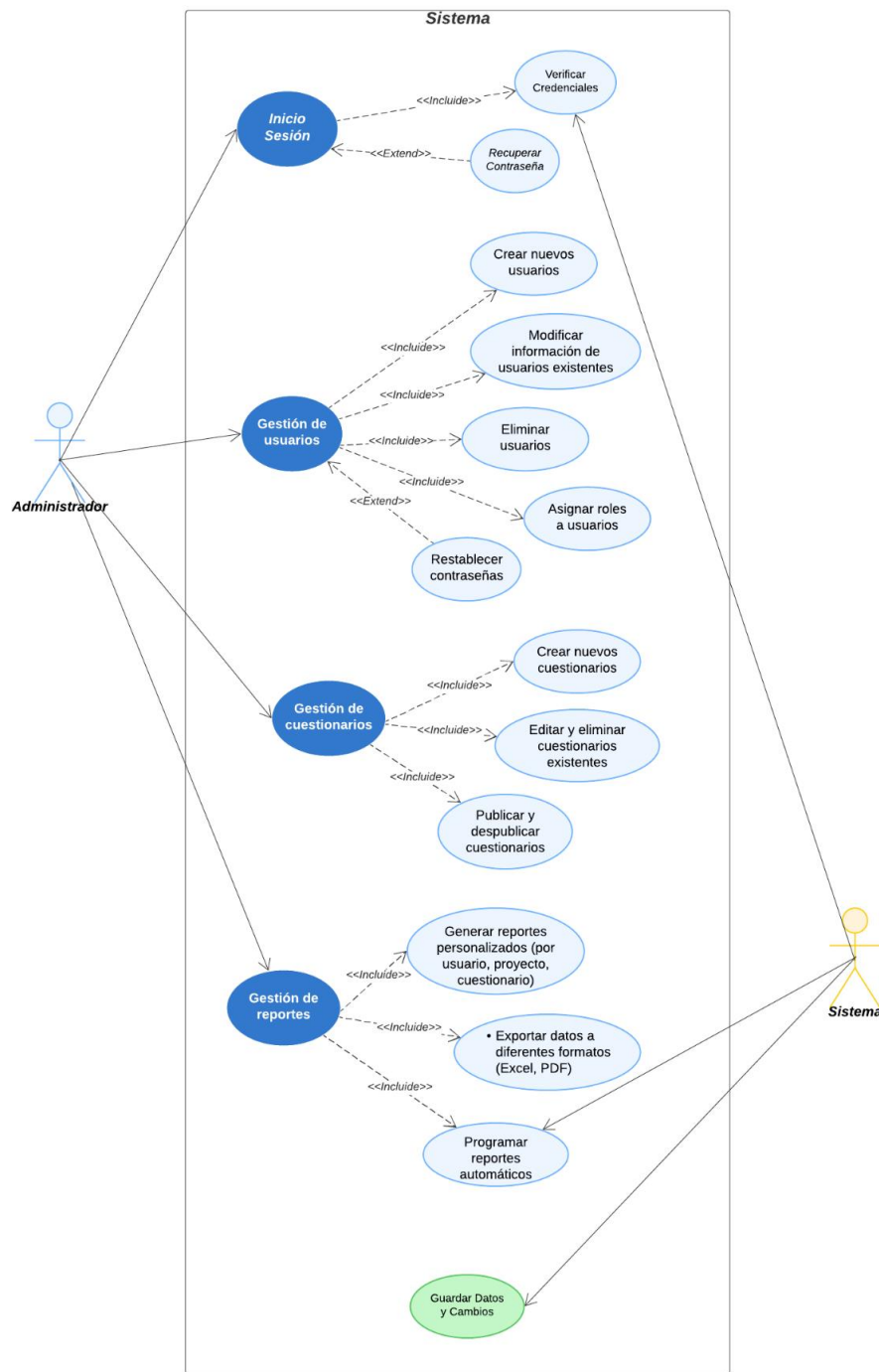


Imagen 2.3.3 Diagrama de uso del administrador

Este diagrama muestra cómo un administrador interactúa con la plataforma para gestionar cuestionarios, estudiantes, resultados e informes, mientras que el sistema proporciona soporte en tareas automatizadas como verificación de las cuentas de los usuarios y almacenamiento de datos.

2.3.4 Diagrama de uso del evaluador

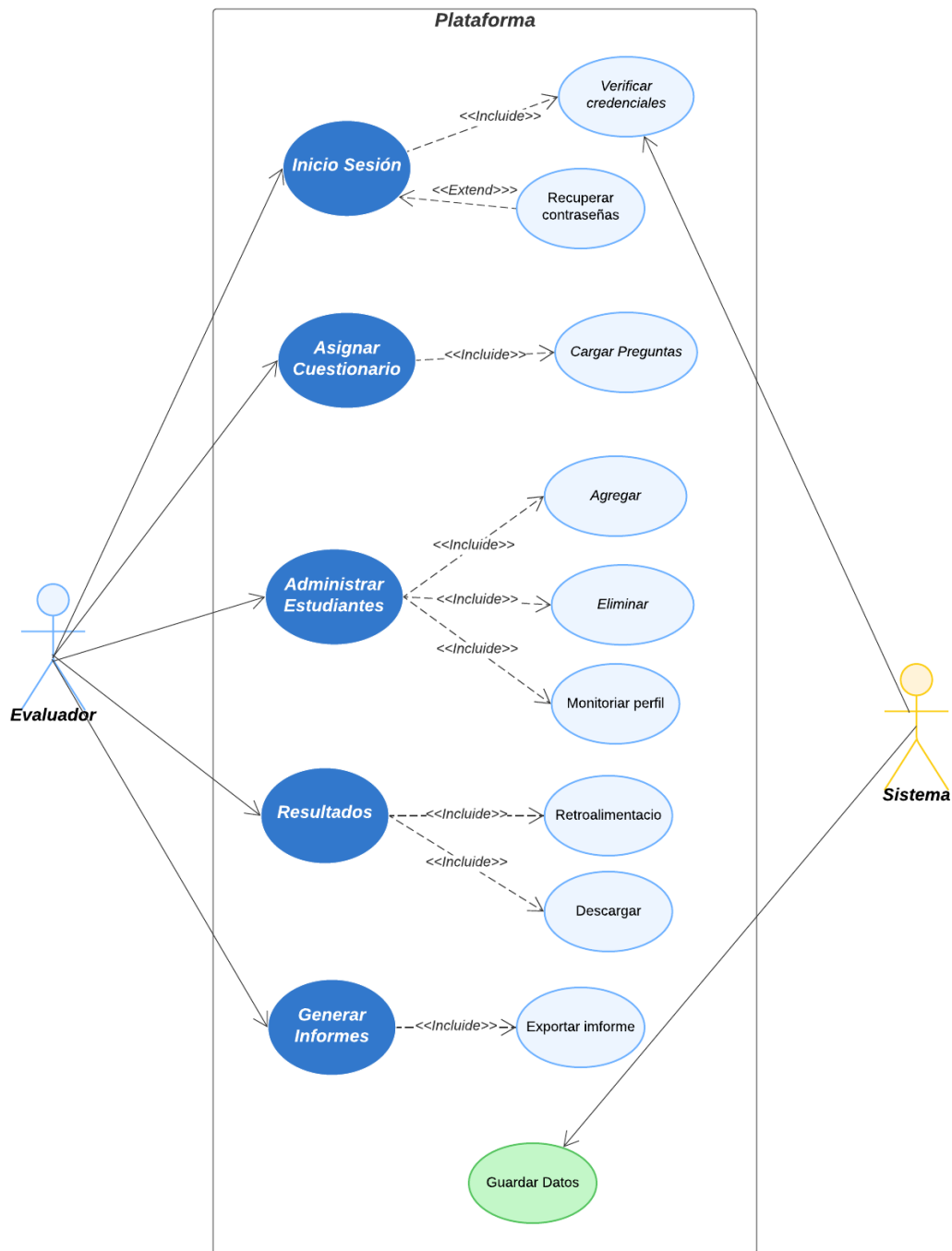


Imagen 2.3.4 Diagrama de uso del evaluador

Este diagrama muestra cómo un evaluador interactúa con la plataforma para gestionar cuestionarios, estudiantes, resultados e informes, mientras que el sistema proporciona soporte en tareas automatizadas como verificación de las cuentas de los usuarios y almacenamiento de datos.

3.4 Modelo relacional

Después de tener de forma concreta lo que se busca tener como requerimiento, se procede a diseñar un modelo ER (Entidad – Relación), para lo que sería la base de datos del sistema

De igual forma es importante destacar que la base de datos esta desarrollada en MySQL, lo cual es una herramienta muy versátil, así como su fácil manejo

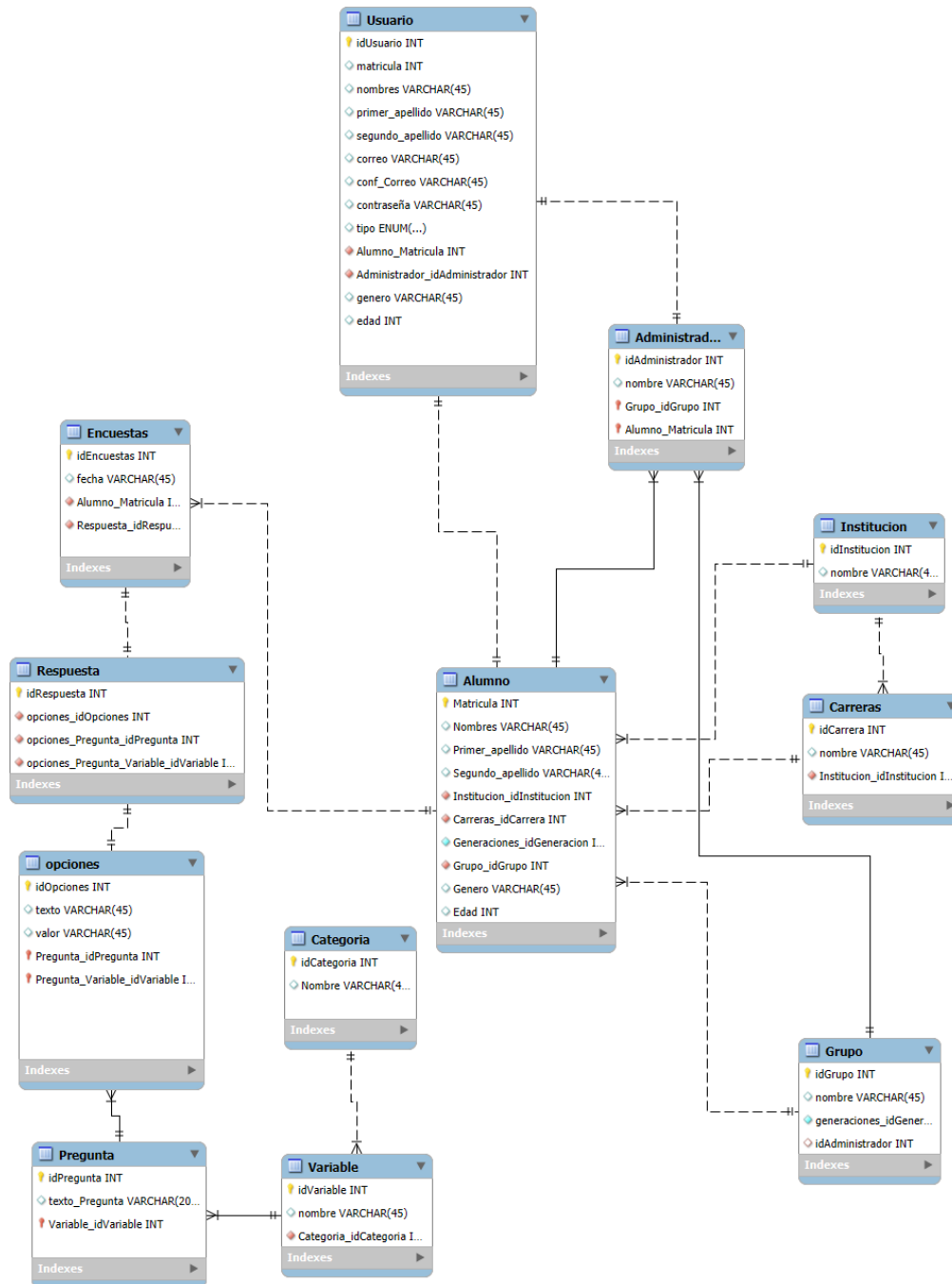


Imagen 3.4.1 Modelo relacional

Capítulo 4.

Resultados y Discusión.

En esta sección se presenta lo logrado tras seguir la metodología establecida, la cual incluyó el diseño y programación del proyecto. Este fue desarrollado con el propósito de recabar información que permita analizar las actitudes hacia el servicio comunitario, utilizando como herramienta la Escala de Actitudes hacia el Servicio Comunitario (CSAS).

4.1 Implementación

El proyecto se centró en el desarrollo de un sistema interactivo para recolectar, analizar y visualizar datos relacionados con las actitudes hacia el servicio comunitario. La arquitectura del sistema incluyó un frontend dinámico desarrollado en Angular, un backend robusto basado en Laravel, y una base de datos eficiente utilizando MySQL.

Backend: Laravel

1. Modelado de la Base de Datos:

- Se diseñaron tablas relacionales para gestionar las respuestas, preguntas, opciones, usuarios y categorías. Este diseño se implementó siguiendo el enfoque **Entidad-Relación (ER)**.
- Cada tabla incluye claves primarias y relaciones foráneas que permiten la consistencia y la integridad referencial de los datos.

2. Desarrollo de Controladores:

- Laravel facilitó la creación de controladores que manejan las interacciones entre la aplicación y la base de datos. Ejemplo:
 - **Controlador de Encuestas:** Permite crear, modificar y publicar encuestas.
 - **Controlador de Respuestas:** Procesa y almacena las respuestas de los participantes.

3. Autenticación con JSON Web Tokens (JWT):

- Se implementó JWT para garantizar un sistema seguro de autenticación y gestión de sesiones, protegiendo así los datos sensibles de los usuarios.

4. APIs RESTful:

- Se diseñaron endpoints para que el frontend pudiera interactuar con el backend, facilitando operaciones como:
 - Registro y autenticación de usuarios.

- Envío y almacenamiento de respuestas.
- Obtención de estadísticas y resultados para visualización.

Base de Datos: MySQL

1. Creación de la Base de Datos:

- Se configuró un servidor MySQL para almacenar los datos recolectados de manera eficiente. La base de datos incluyó tablas normalizadas para evitar redundancias y garantizar la integridad de los datos.

2. Migraciones y Seeders:

- Laravel se utilizó para gestionar las migraciones y garantizar que las estructuras de las tablas se mantuvieran consistentes a lo largo del desarrollo.
- Los seeders permitieron llenar la base de datos con datos iniciales de prueba para validar las funcionalidades del sistema.

Frontend: Angular

1. Diseño de la Interfaz de Usuario:

- Se desarrollaron componentes modulares para garantizar una experiencia de usuario dinámica e intuitiva. Ejemplo:
 - Formulario interactivo para responder encuestas.
 - Dashboard para visualizar los resultados.

2. Visualización de Datos:

- Gráficas dinámicas (barras, líneas, y mapas de calor) se generaron utilizando bibliotecas como **Chart.js**.
- Las visualizaciones permiten a los administradores comparar resultados entre diferentes grupos.

3. Enrutamiento y Estados:

- Angular Router se utilizó para gestionar las rutas dentro de la aplicación, garantizando una navegación fluida.
- Los estados del usuario y las encuestas se controlaron utilizando **Servicios de Angular**.

Colaboración y Gestión del Código: GitHub

1. Control de Versiones:

- Todo el código fuente fue gestionado en un repositorio GitHub, lo que facilitó la colaboración y el seguimiento de cambios entre los miembros del equipo.
- Las ramas separadas para desarrollo y producción permitieron implementar nuevas características sin interrumpir la funcionalidad existente.

2. Revisiones de Código:

- Los cambios en el código pasaron por revisiones utilizando pull requests, garantizando calidad y consistencia.

PHP como Lenguaje Base

1. Integración en Laravel:

- PHP fue el lenguaje base para desarrollar toda la lógica del backend.
- Su flexibilidad permitió implementar algoritmos personalizados para el análisis de datos.

4.1.1 El cuestionario CSAS estaría midiendo:

- **Interés en el servicio comunitario:** Nivel de atracción personal hacia actividades de servicio.
- **Responsabilidad cívica:** Percepción del deber de contribuir al bienestar de la comunidad.
- **Compromiso social:** Grado de implicación activa en actividades de servicio.

4.1.2 Técnicas de Análisis

1. Estadística descriptiva:

- Se calcularon medidas de tendencia central y dispersión (media, mediana, desviación estándar) para cada dimensión de la escala.

2. Análisis paramétrico:

- Se utilizó ANOVA para determinar si existían diferencias significativas en las dimensiones de la escala según el género, la carrera y la generación.

3. Análisis no paramétrico:

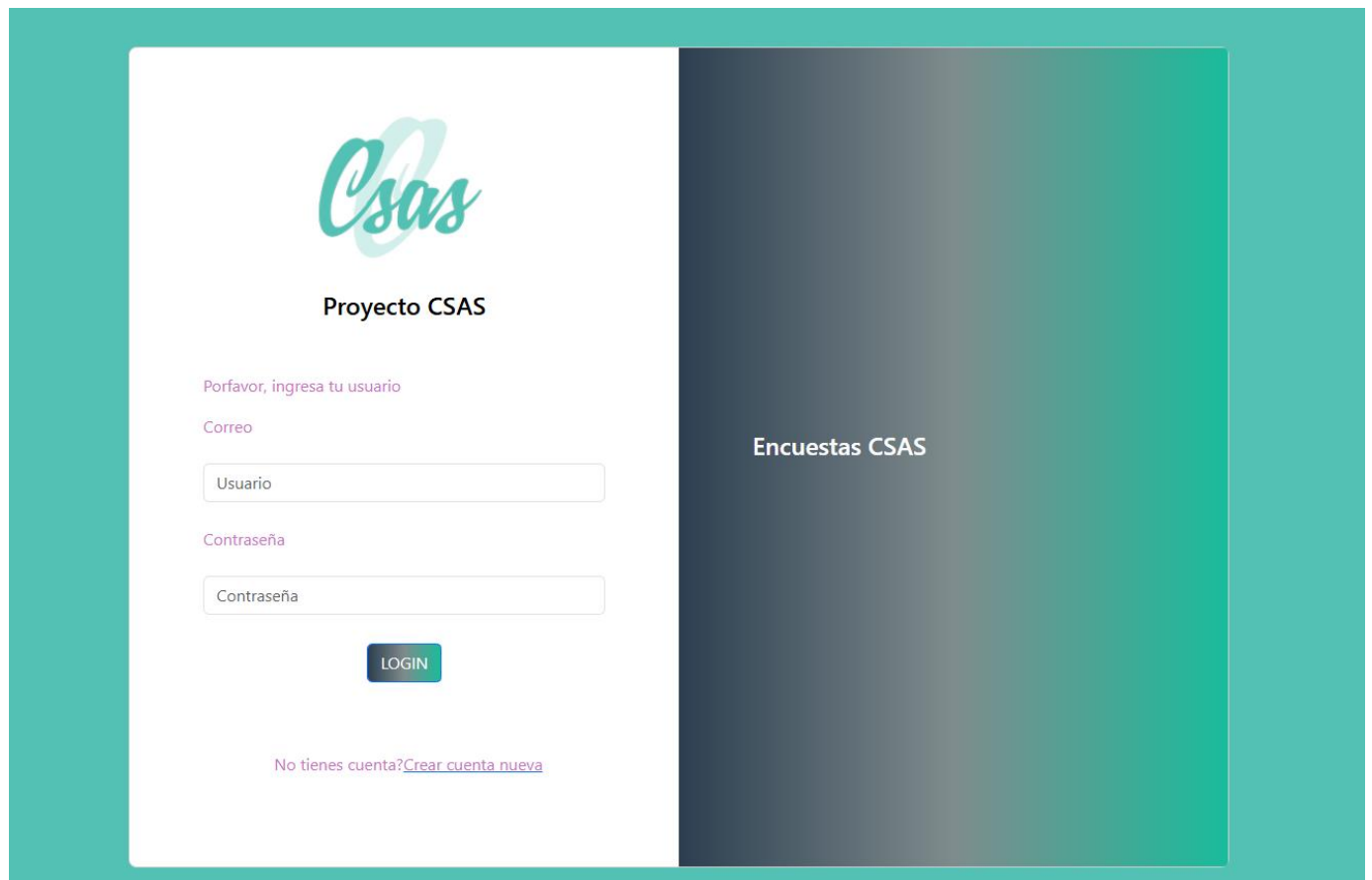
- Para los datos no normalmente distribuidos, se emplearon pruebas con el alfa de cronbach. Esto permitió analizar grupos con tamaños desiguales o distribuciones heterogéneas.

4.2 Pruebas de funcionalidad del sistema



Pantalla 1

Como primera pantalla tenemos lo que vendría siendo nuestro home, el cual esto nos recibirá cuando estemos iniciando el proyecto, el cual de la misma forma nos mandara a nos debe redirigir al apartado del login.



Pantalla 2

Segunda pantalla login, en este apartado una vez ya teniendo la cuenta creada, nos debería permitir ingresar al cuestionario en dado caso de no tener una cuenta en el apartado de crea cuenta nos redirige a lo que sea el formulario para crearlo.

Csas
Proyecto CSAS

Porfavor, ingresa tus datos

Nombre

Nombre

Correo

Correo

Contraseña

Contraseña

Confirmar Contraseña

Confirmar Contraseña

Seleccionar Rol

Estudiante

REGISTRARSE

Encuestas CSAS

Pantalla 3

Formulario para la creación de nuestra cuenta, en este apartado debemos llenar los apartados, con los datos que nos piden, en este caso siendo, nombre, correo, contraseña y otro apartado para confirmarlo, una vez registrado en el botón de registrarse nos redirige directamente a la ventana de login.



Datos Demográficos

1. Matrícula

2. Género

- ☐ Masculino
☐ Femenino
☐ Otro

3. Edad

 años

4. Lo que te describe mejor:

- ☐ Estudiante actual de secundaria
☐ Estudiante universitario actual
☐ Universidad graduada en el último año
☐ Estudiante graduado
☐ Otro

5. ¿Es usted miembro actual o alumno de una institución?

- ☐ Sí ☐ No

Pantalla 4

Una vez teniendo la cuenta creada y logueada, debemos llenar un pequeño formulario en el cual debemos llenar unos datos que serán importantes, esto para saber la matrícula la de institución en la que estudia el encuestado, grado de estudio, edad entre otras preguntas.



Cuestionario de Actitudes hacia el Servicio Comunitario

Cuestionario de actitudes de servicio comunitario

Lea cuidadosamente cada oración y seleccione la opción que mejor describa su reacción a cada afirmación.

10. Los grupos comunitarios necesitan nuestra ayuda.

Totalmente en
desacuerdo



En
desacuerdo



Algo en
desacuerdo



Ni estoy de acuerdo ni en
desacuerdo



Parcialmente de
acuerdo



De
acuerdo



Totalmente de
acuerdo



11. Hay muchas personas en la comunidad que necesitan ayuda.

Totalmente en
desacuerdo



En
desacuerdo



Algo en
desacuerdo



Ni estoy de acuerdo ni en
desacuerdo



Parcialmente de
acuerdo



De
acuerdo



Totalmente de
acuerdo



12. Hay necesidades en la comunidad.

Totalmente en
desacuerdo



En
desacuerdo



Algo en
desacuerdo



Ni estoy de acuerdo ni en
desacuerdo



Parcialmente de
acuerdo



De
acuerdo



Totalmente de
acuerdo



17. El voluntariado en proyectos comunitarios puede mejorar enormemente los recursos de la comunidad.

Totalmente en
desacuerdo



En
desacuerdo



Algo en
desacuerdo



Ni estoy de acuerdo ni en
desacuerdo



Parcialmente de
acuerdo



De
acuerdo



Totalmente de
acuerdo



18. Cuanta más gente ayude, mejor será la situación.

Totalmente en
desacuerdo



En
desacuerdo



Algo en
desacuerdo



Ni estoy de acuerdo ni en
desacuerdo



Parcialmente de
acuerdo



De
acuerdo



Totalmente de
acuerdo



19. Contribuir con mis habilidades hará de la comunidad un lugar mejor.

Totalmente en
desacuerdo



En
desacuerdo



Algo en
desacuerdo



Ni estoy de acuerdo ni en
desacuerdo



Parcialmente de
acuerdo



De
acuerdo



Totalmente de
acuerdo



Anterior 1 2 3 4 Siguiente

Página 1 de 4

Pantalla 5

Siguientes pantallas, estas siendo de la encuesta en la cual tendremos un total de 4 paginas para un total de 45 preguntas el cual nos ayudara a evaluar a los encuestados, esto para recabar la información necesaria y posteriormente realizar el análisis.

4.3 Pruebas de usabilidad y recolección de retroalimentación



Imagen prueba 1



Imagen prueba 2

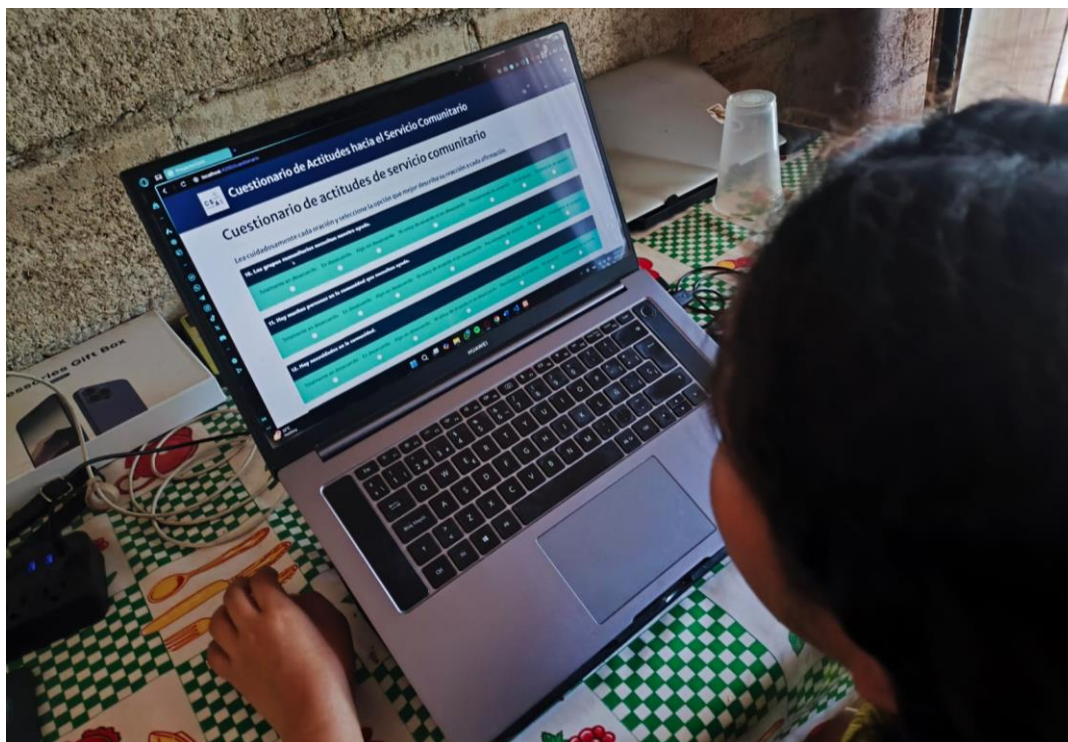


Imagen prueba 3



Imagen prueba 4

4.4 Comparación del Método de Recolección y su Impacto en el Análisis

Para evaluar las diferencias entre el método tradicional de papel y lápiz y la implementación de un software automatizado, se analizaron los siguientes aspectos clave:

Característica	Papel y Lápiz	Software del cuestionario CSAS
Tiempo destinado a la recopilación de datos	Lento: Requiere tiempo para distribuir, recoger y transcribir los datos.	Rápido y automatizado: Los datos se recogen y almacenan en tiempo real.
Precisión de Datos	Propenso a errores: Errores humanos en la transcripción y recolección.	Alta precisión: Validación automática de datos y reducción de errores humanos.
Tiempo de Análisis	Días o semanas: Requiere tiempo para la transcripción y el análisis manual.	Segundos o minutos: Análisis automático y generación de resultados en tiempo real.
Visualización	Manual y limitada: Gráficos y tablas creados manualmente, con poca interactividad.	Dinámica e interactiva: Gráficos y dashboards interactivos que se actualizan automáticamente.
Comparación de Datos	Difícil y laboriosa: Requiere comparación manual entre grupos o periodos.	Automática y personalizable: Comparaciones instantáneas y personalizables según criterios.
Disponibilidad	Archivos físicos: Limitado acceso y riesgo de pérdida o daño.	Acceso remoto: Datos disponibles en la nube, accesibles desde cualquier lugar y momento.
Seguridad de Datos	Baja: Riesgo de pérdida, daño o acceso no autorizado a los archivos físicos.	Alta: Encriptación y control de acceso para proteger los datos.
Escalabilidad	Limitada: Dificultad para manejar grandes volúmenes de datos.	Alta: Capacidad para manejar grandes volúmenes de datos y crecer con las necesidades.
Colaboración	Limitada: Dificultad para compartir y colaborar en tiempo real.	Facilitada: Colaboración en tiempo real y acceso compartido a los datos.
Costo a largo plazo	Alto: Costos recurrentes de papel, impresión y almacenamiento físico.	Bajo: Reducción de costos operativos y de almacenamiento físico.
Flexibilidad	Baja: Dificultad para modificar o actualizar encuestas y análisis.	Alta: Fácil modificación de encuestas, análisis y visualizaciones según necesidades.

Característica	Papel y Lápiz	Software del cuestionario CSAS
Integración con otras herramientas	Limitada: Dificultad para integrar datos con otras herramientas de análisis.	Alta: Integración sencilla con otras herramientas de análisis y visualización de datos.

Tabla de comparación

La digitalización no solo optimiza la velocidad y precisión de la recolección de datos, sino que también permite aplicar de manera inmediata pruebas estadísticas paramétricas y no paramétricas, como ANOVA y pruebas de Kruskal-Wallis. En el método tradicional, la transcripción manual de los datos supone un retraso significativo en la obtención de resultados y aumenta la posibilidad de errores humanos.

Análisis de Datos de la Escala de Actitudes de Servicio Comunitario

A partir de los datos recopilados en la **Escala de Actitudes de Servicio Comunitario**, se identificaron tendencias en la percepción de los participantes sobre la importancia del servicio comunitario. Entre los resultados más relevantes:

- El 65% de los participantes se identifican como mujeres y el 35% como hombres.
- Solo el 15% de los encuestados ha participado en actividades de voluntariado en los últimos 12 meses.
- Un 85% de los participantes reconocen que hay necesidades en la comunidad que requieren atención inmediata.
- El 50% de los encuestados está de acuerdo en que los estudiantes universitarios pueden ayudar a mejorar la comunidad local.
- El 55% cree que el servicio comunitario es fundamental para mejorar la sociedad.

4.5 Pantallas de los usuarios que contestaron la encuesta

**Cuestionario de Actitudes hacia el Servicio Comunitario**



Carlos Perez
carlos.perez@example.com

- Dashboard
- Usuarios
- Crear Encuestas
- Reportes
- Cerrar Sesión

Lista de usuarios

Agregar

ID Usuario	Tipo	Matricula o ID	Nombres	Apellidos	Correo	Acciones
1	Alumno	8198	Jose daniel	Mex Naal	8198@itescam.edu.mx	Modificar Eliminar
2	Alumno	8173	Diego Fabricio	Zi Cauich	8173@itescam.edu.mx	Modificar Eliminar
3	Alumno	8186	Ricardo Jesús	Salcedo Horna	8186@itescam.edu.mx	Modificar Eliminar
4	Alumno	8130	Karla Marinthia	Gutiérrez Huchin	8130@itescam.edu.mx	Modificar Eliminar
5	Alumno	8131	Martha Irene	Chi Cauich	8131@itescam.edu.mx	Modificar Eliminar

Anterior

1

2

3

4

Siguiente

Resultado 1

**Cuestionario de Actitudes hacia el Servicio Comunitario**



Carlos Perez
carlos.perez@example.com

- Dashboard
- Usuarios
- Crear Encuestas
- Reportes
- Cerrar Sesión

Lista de usuarios

Agregar

ID Usuario	Tipo	Matricula o ID	Nombres	Apellidos	Correo	Acciones
6	Alumno	8159	Joel Daniel	Pat Sanches	8159@itescam.edu.mx	Modificar Eliminar
7	Alumno	8196	Jesus Antonio	Moo Kantun	8196@itescam.edu.mx	Modificar Eliminar
8	Alumno	8187	Rodrigo Ema.	Cohuo Ucan	8187@itescam.edu.mx	Modificar Eliminar
9	Alumno	8160	Mariana Isabel	Uc Puc	8160@itescam.edu.mx	Modificar Eliminar
10	Alumno	8106	Maira Yaneli	Puc Canul	8061@itescam.edu.mx	Modificar Eliminar

Anterior

1

2

3

4

Siguiente

Resultado 2

4.6 Respuestas de los datos demográficos



Grafica 1



Grafica 2



Cuestionario de Actitudes hacia el Servicio Comunitario



Carlos Perez
carlos.perez@example.com

- Dashboard
- Usuarios
- Crear Encuestas
- Reportes
- Cerrar Sesión

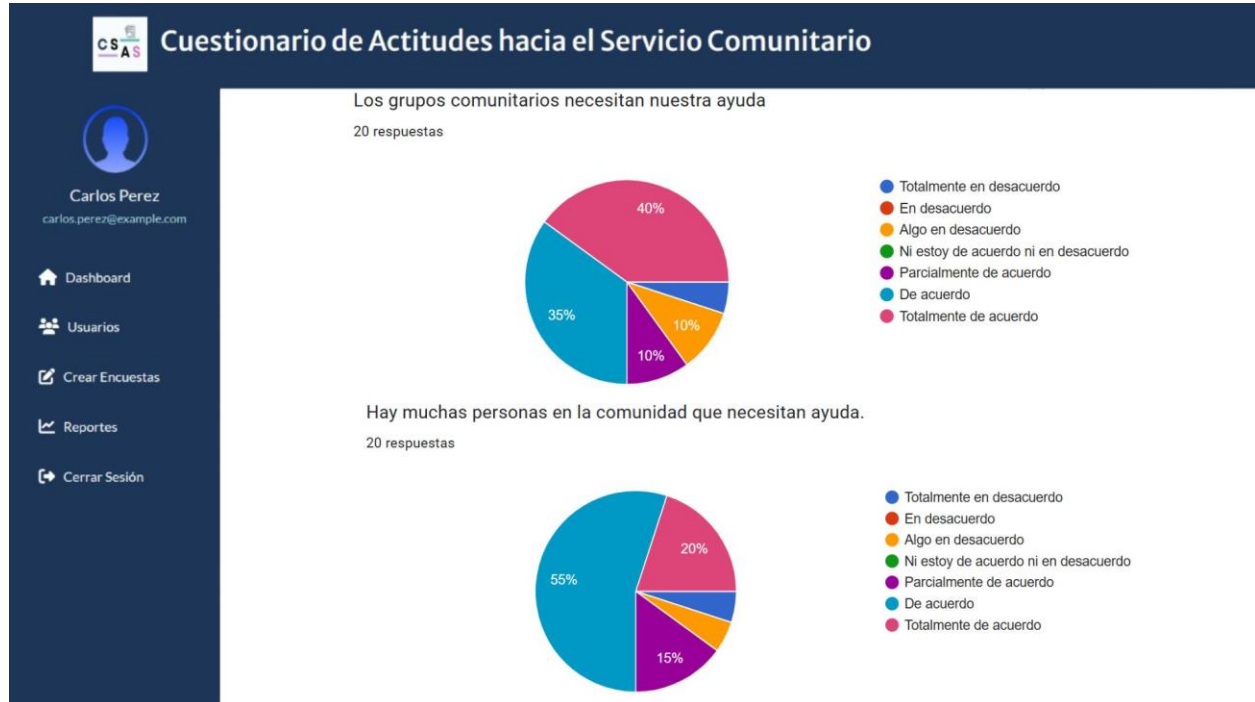
Si ha trabajado como voluntario en los últimos doce meses, ¿Cuántas horas de trabajo voluntario por semana en promedio?

20 respuestas

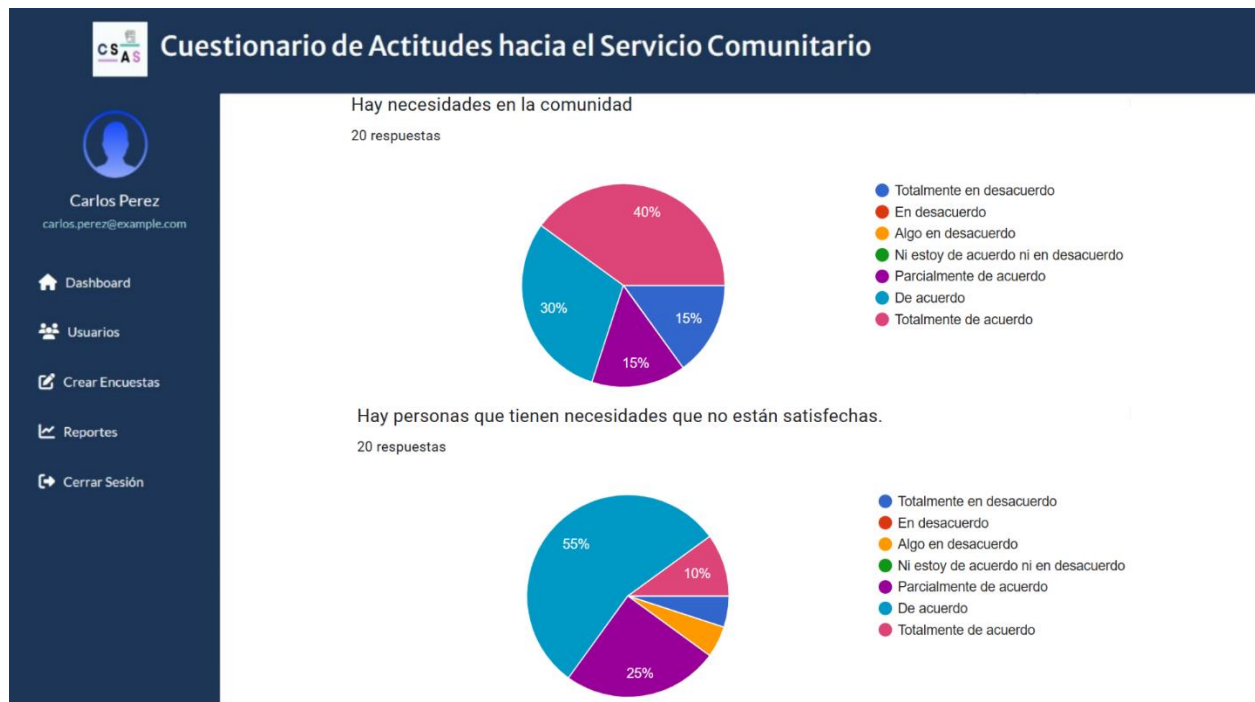
10
6
12
374
3
4
12 horas
5
40
1 hora por día
12 horas
20
8
7
0
No he trabajado como voluntario

Resultado 3

4.7 Graficas de los resultados obtenidos



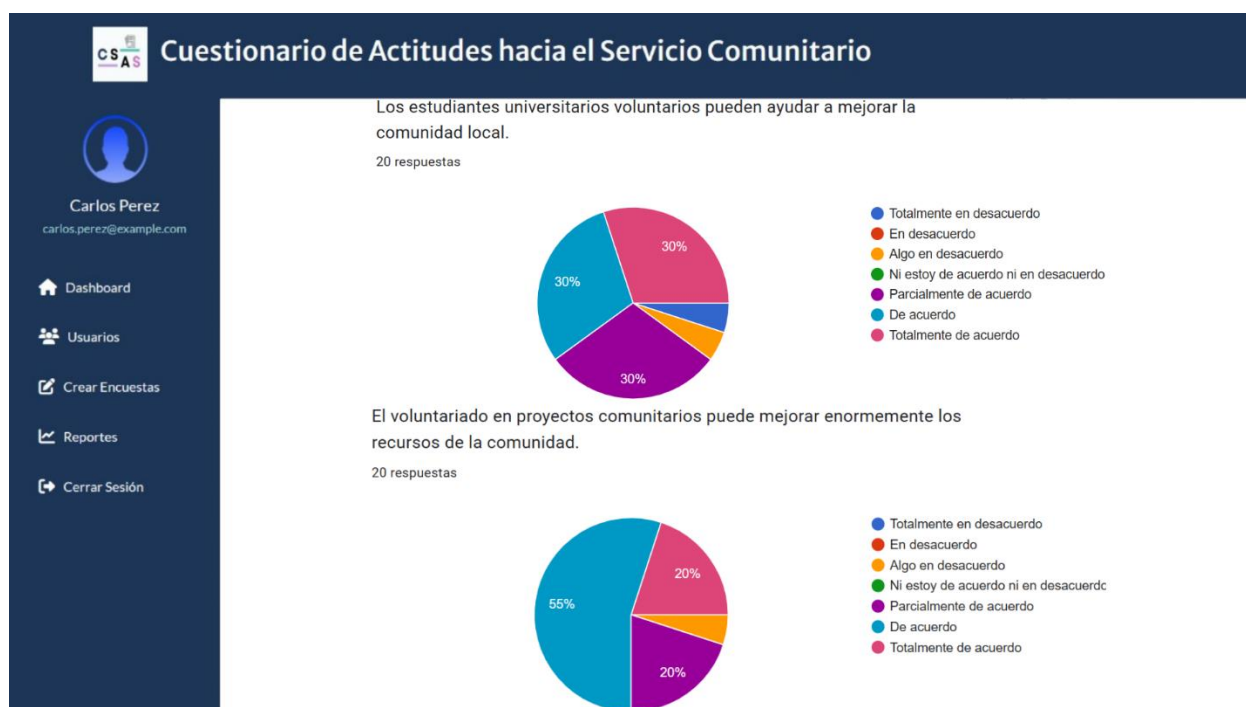
Grafica 3



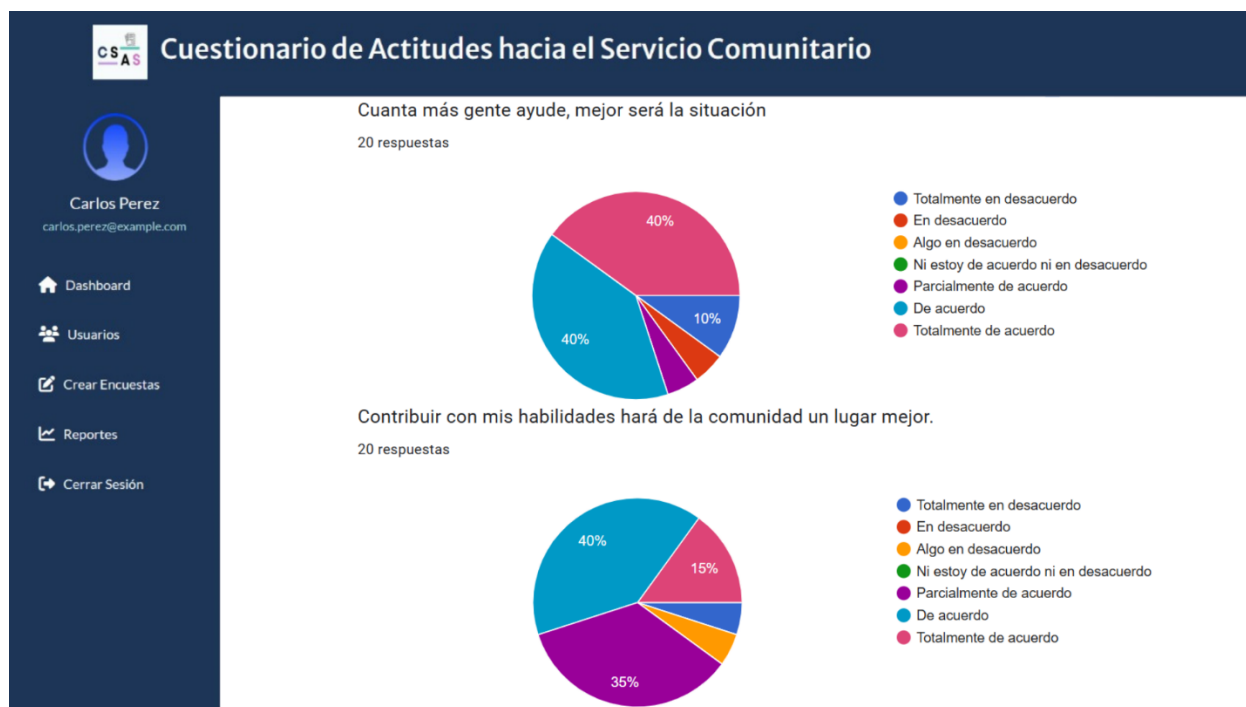
Grafica 4



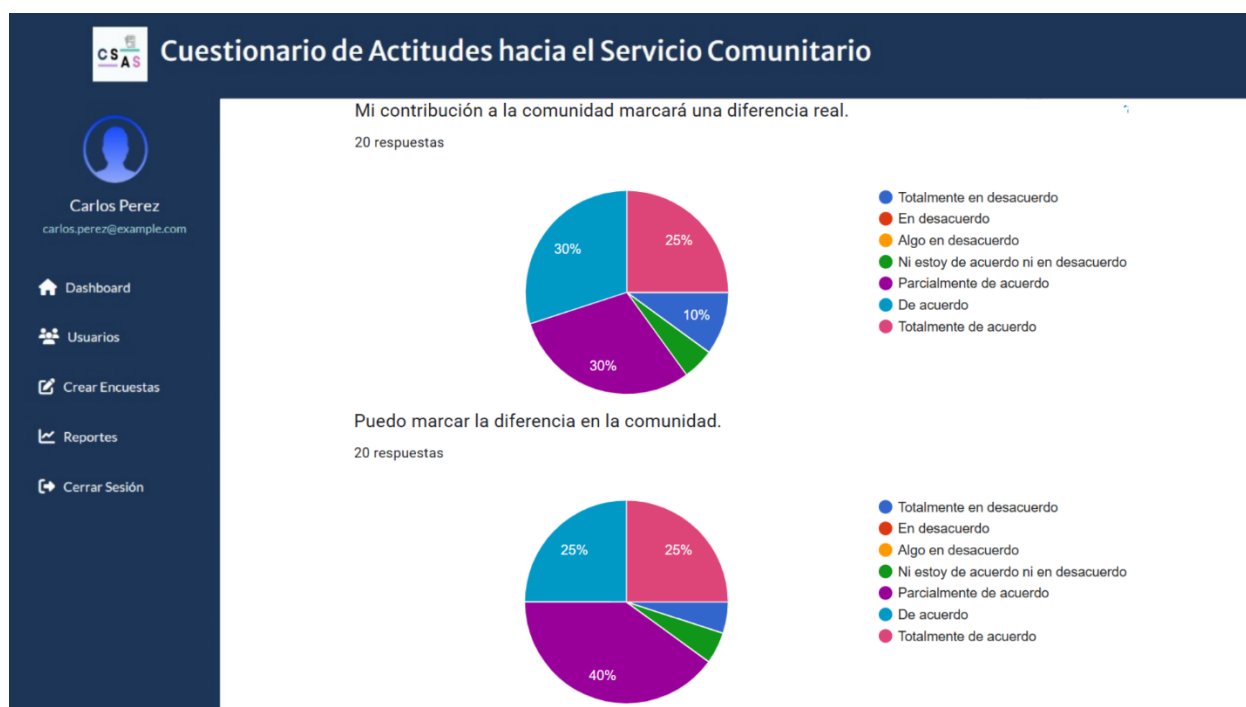
Grafica 5



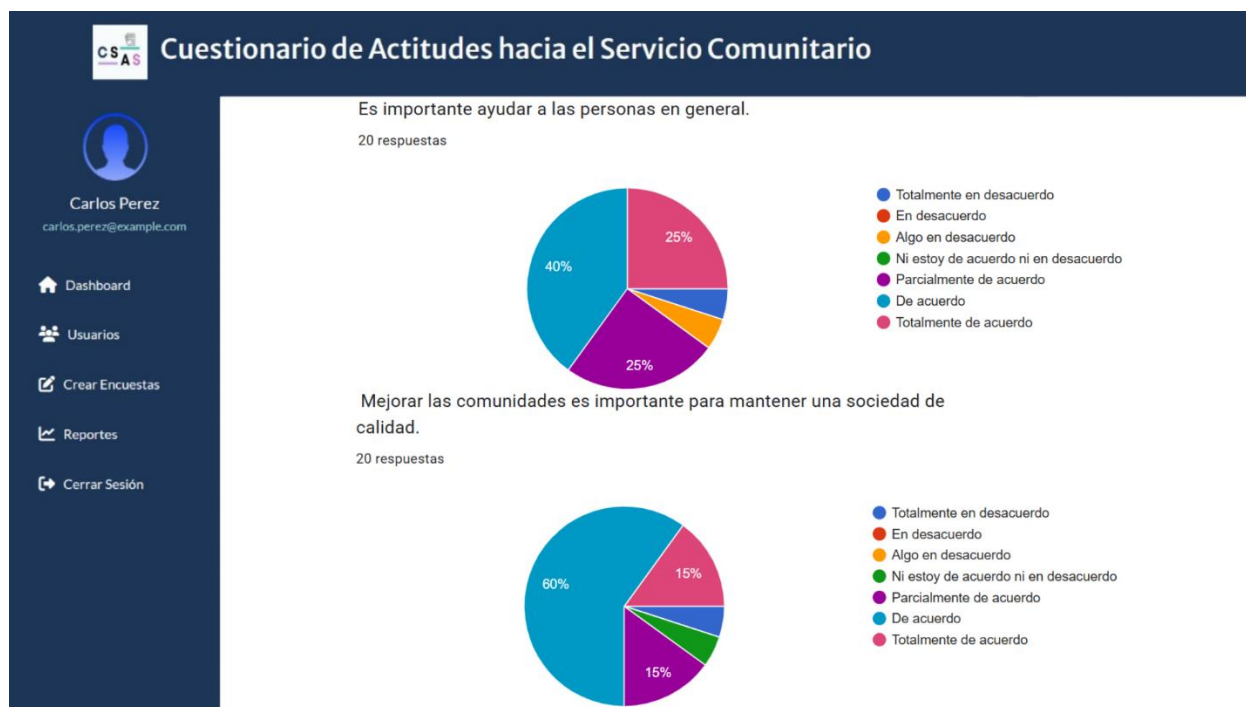
Grafica 6



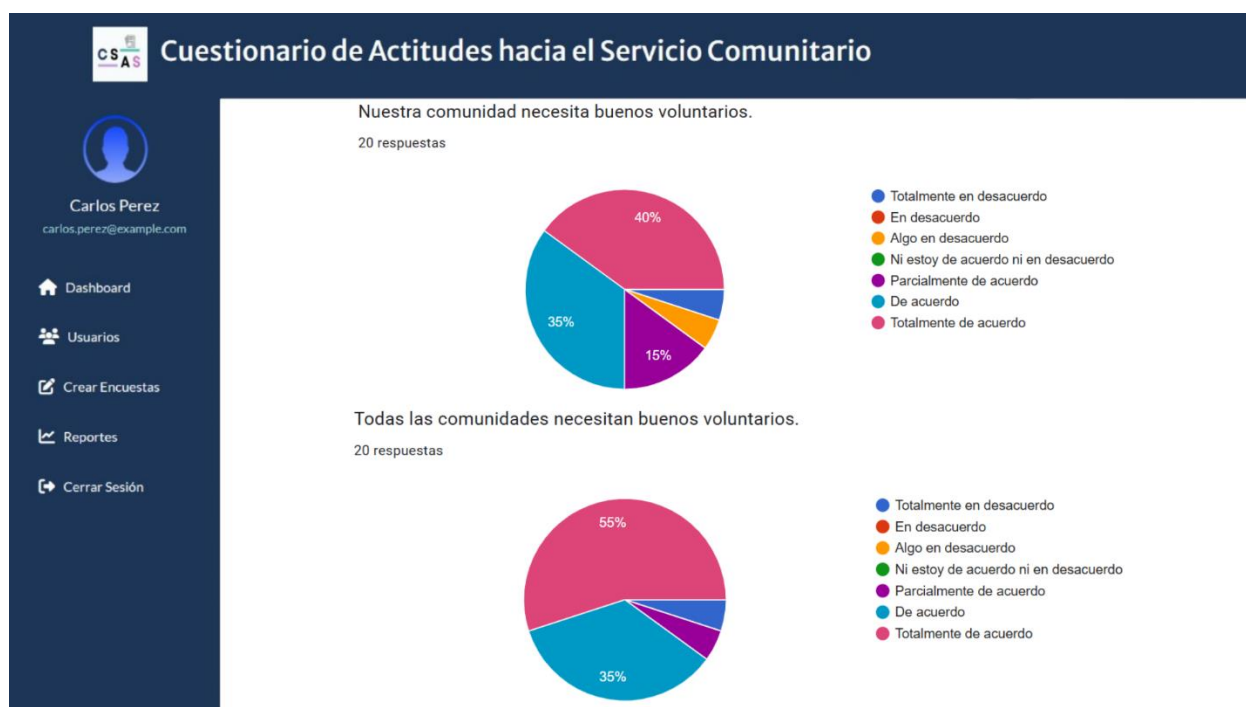
Grafica 7



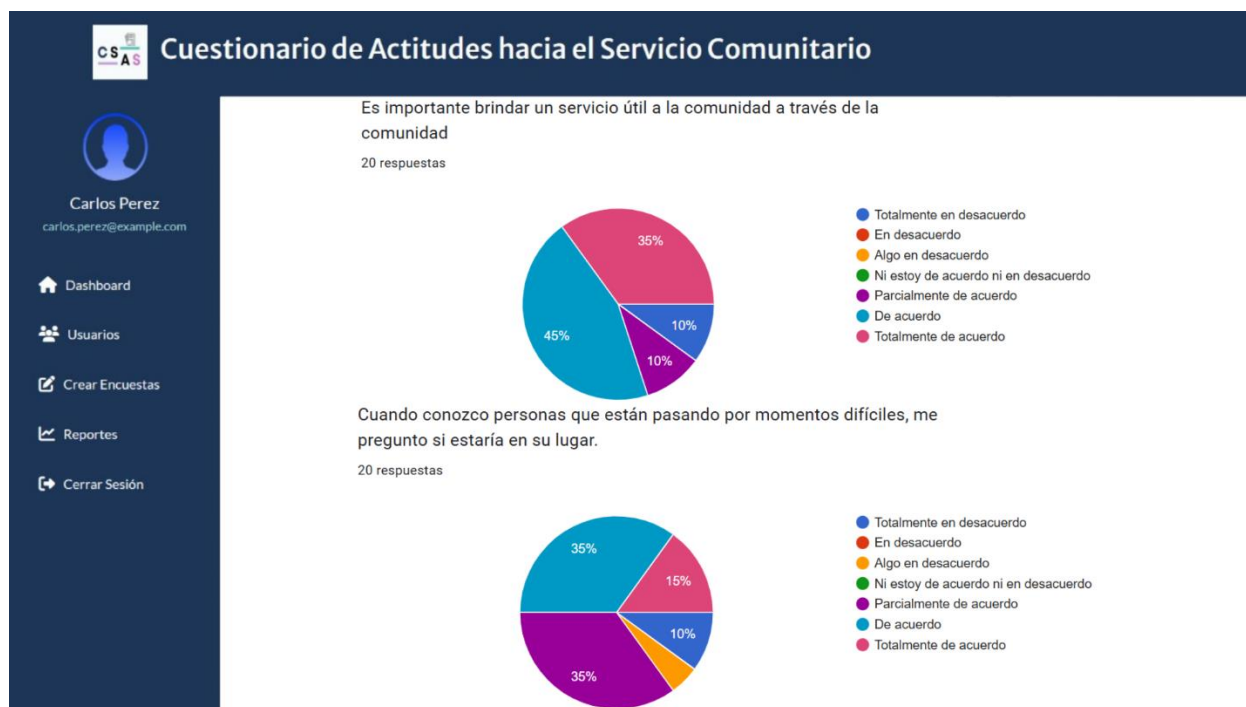
Grafica 8



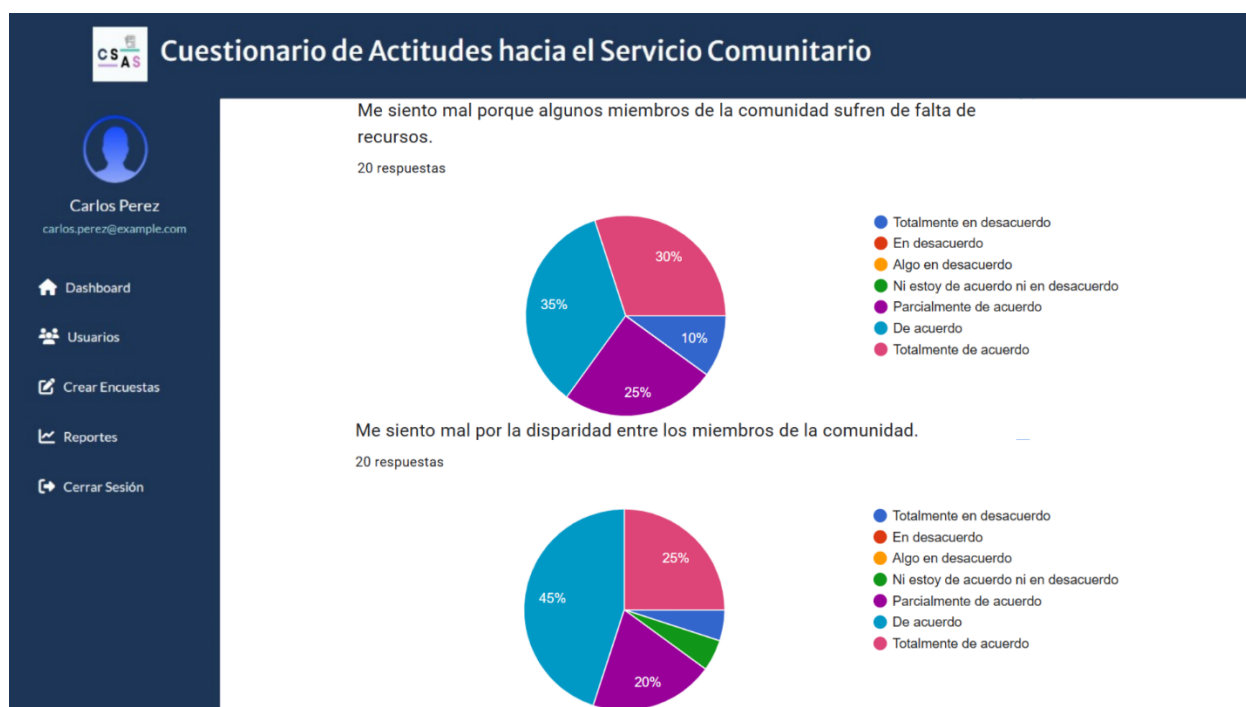
Grafica 9



Grafica 10



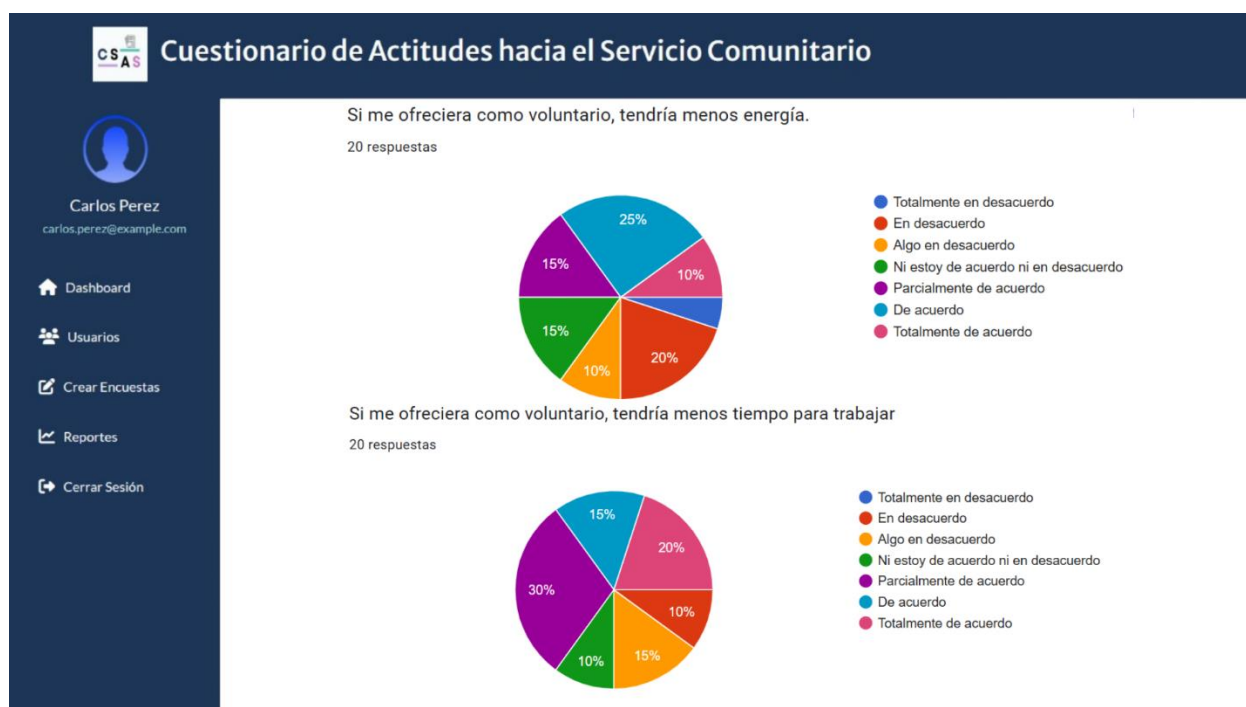
Grafica 11



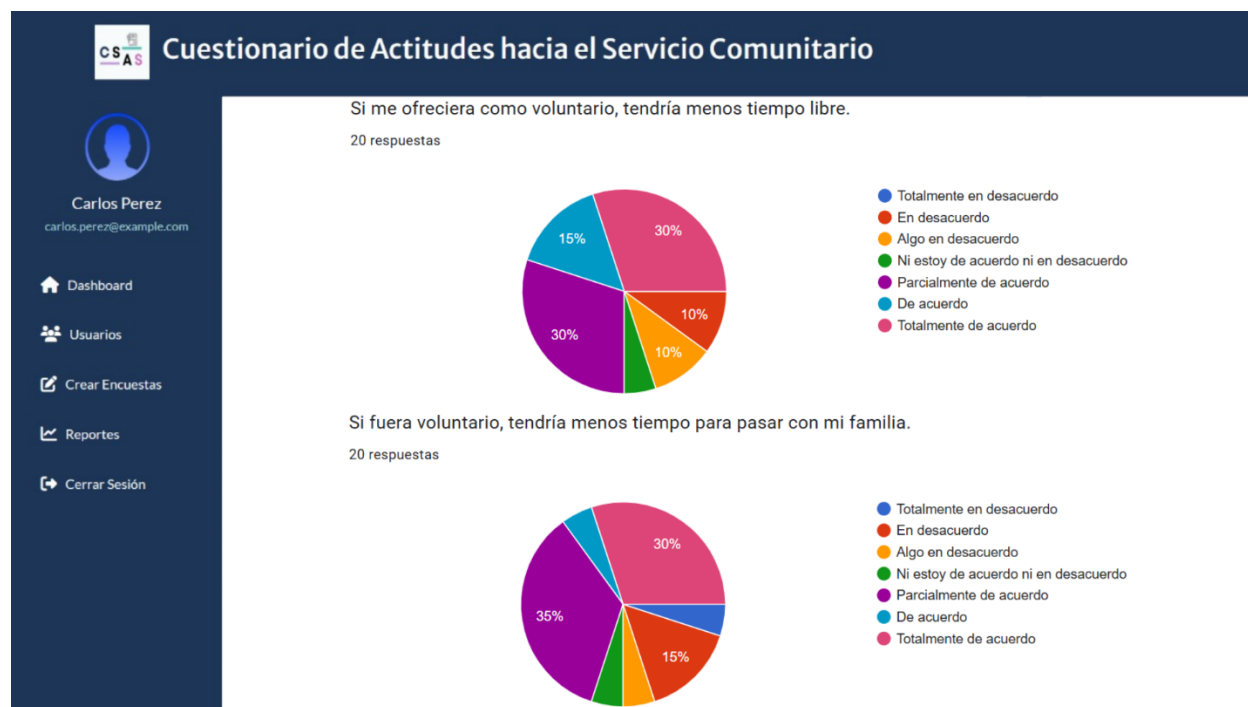
Grafica 12



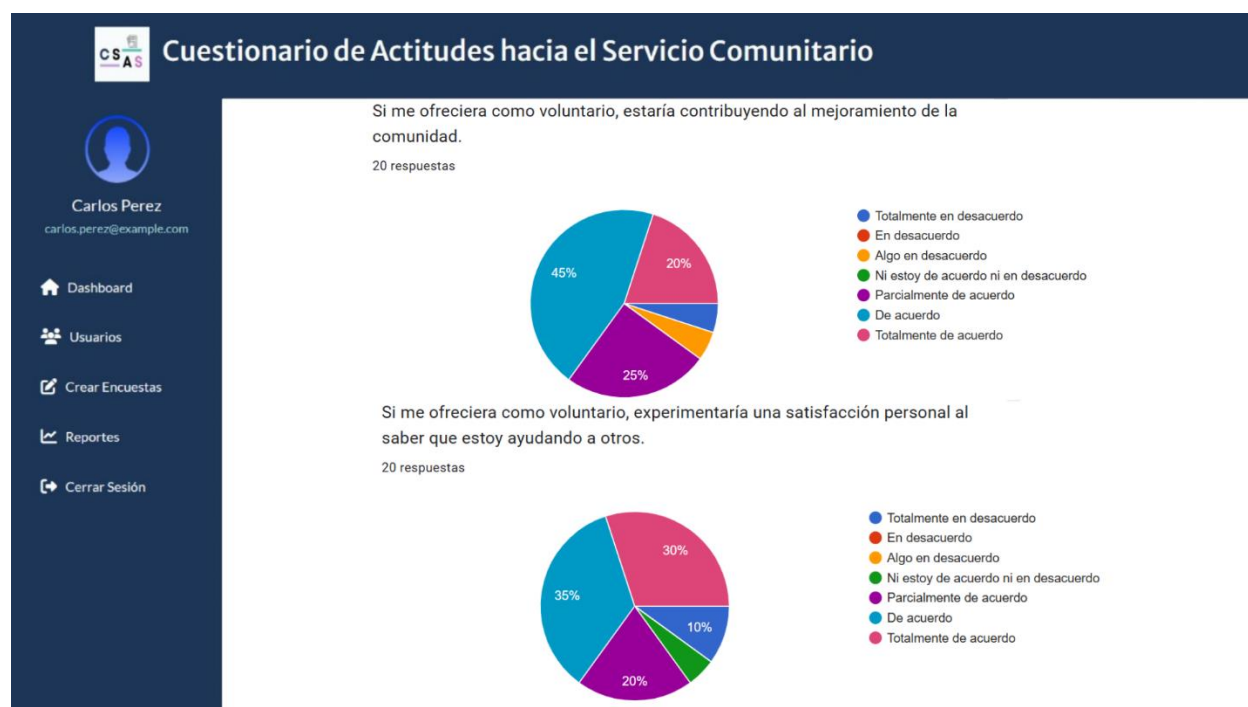
Grafica 13



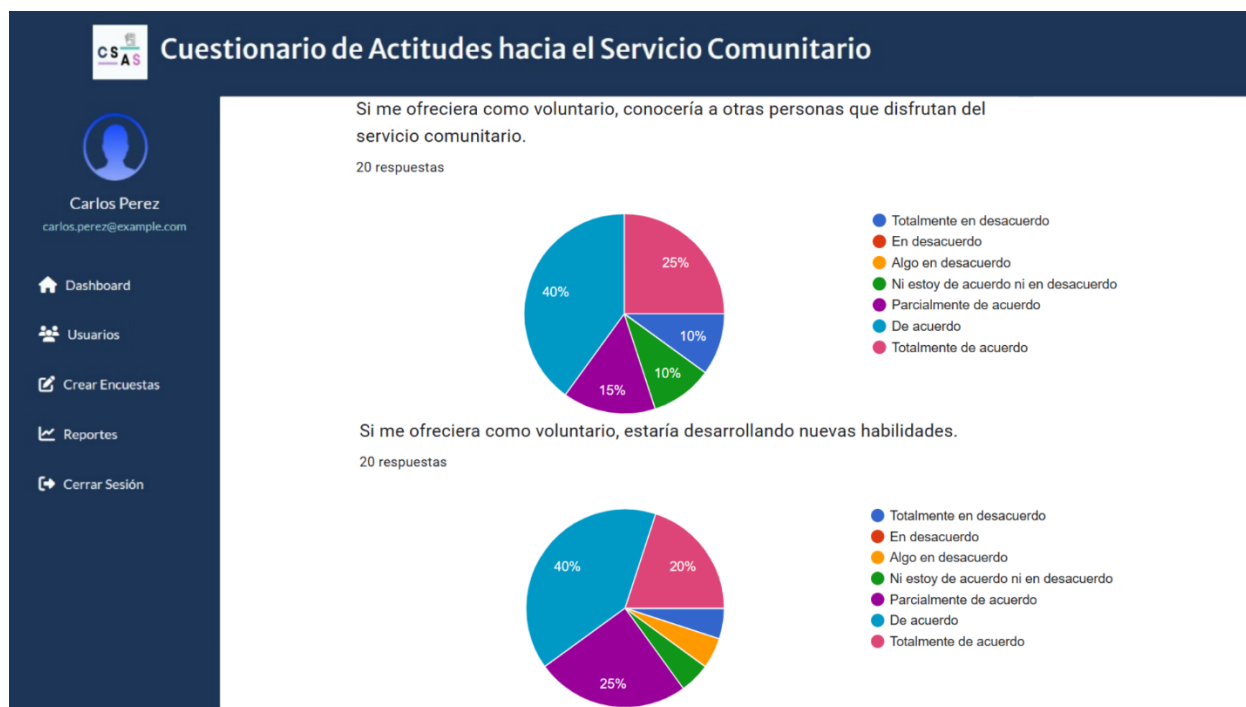
Grafica 14



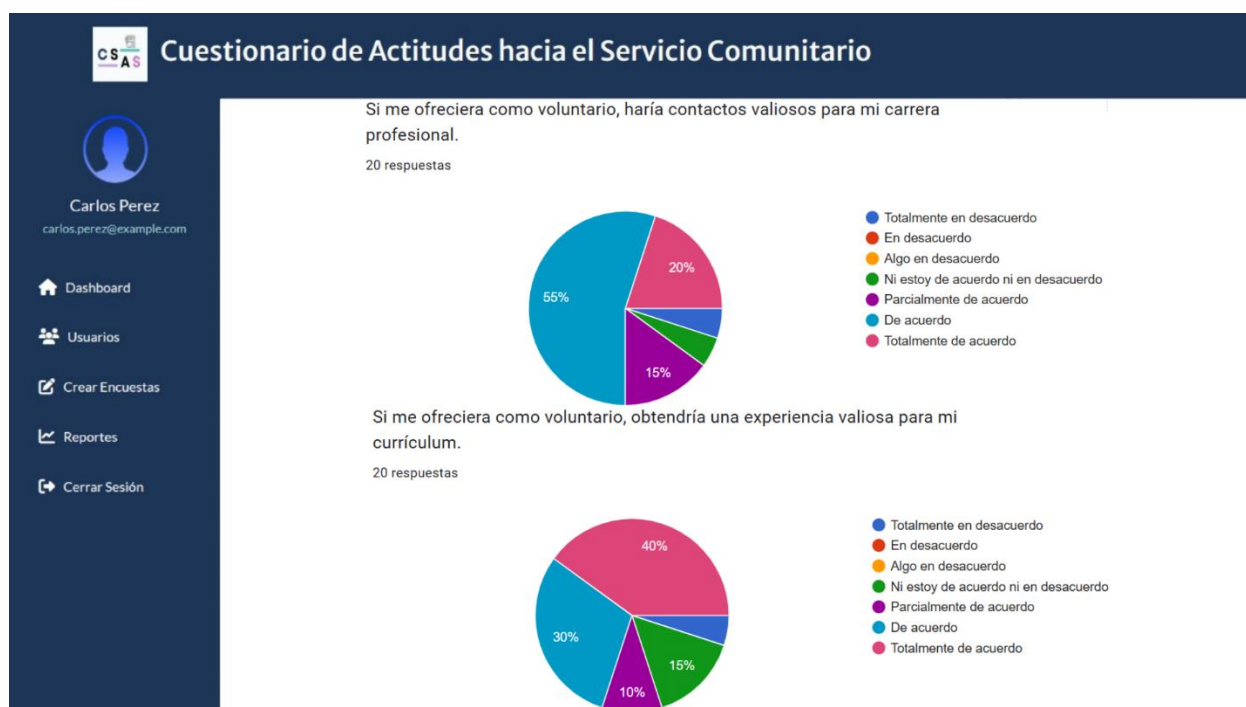
Grafica 15



Grafica 16



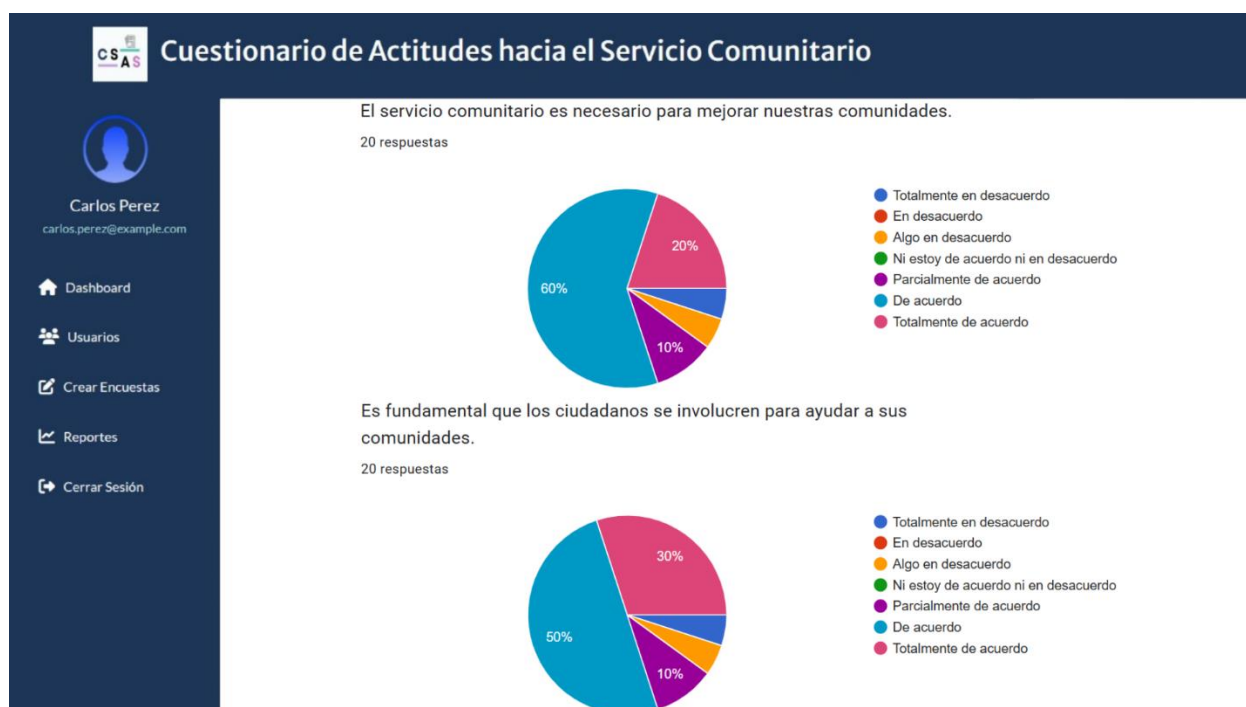
Grafica 17



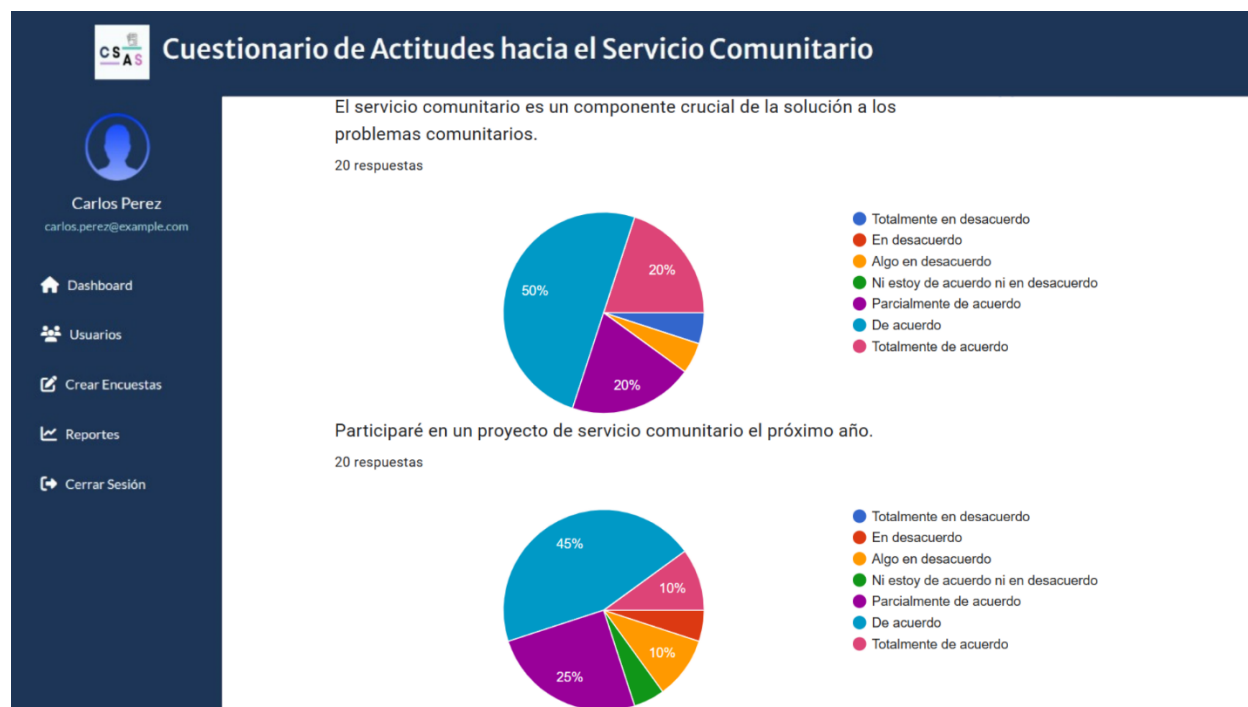
Grafica 18



Grafica 19



Grafica 20



Grafica 21



Grafica 22

Capítulo 5

Conclusiones.

El proyecto desarrollado implementó una serie de técnicas de visualización matemática y análisis gráfico que facilitan significativamente la interpretación de los resultados obtenidos a través de los algoritmos estadísticos paramétricos y no paramétricos aplicados a los datos de la escala CSAS. Estas técnicas incluyen **gráficos de barras, dashboards interactivos, gráficos circulares, diagramas de dispersión, mapas de calor, gráficos de líneas y visualización de datos demográficos**, las cuales fueron diseñadas para hacer que los datos sean accesibles y comprensibles, incluso para usuarios sin conocimientos técnicos avanzados.

Los **gráficos de barras y circulares** permiten comparar respuestas y distribuciones de manera intuitiva, mientras que los **dashboards interactivos** ofrecen una experiencia dinámica y personalizable para explorar los datos en tiempo real. Los **diagramas de dispersión y mapas de calor** ayudan a identificar patrones y relaciones entre variables, y los **gráficos de líneas** son útiles para analizar tendencias a lo largo del tiempo. Además, la **visualización de datos demográficos** contextualiza los resultados, permitiendo análisis comparativos entre diferentes grupos.

Estas herramientas no solo optimizan la velocidad y precisión en la recolección y análisis de datos, sino que también permiten una interpretación más clara y efectiva de los resultados, lo que facilita la toma de decisiones informadas. La implementación de estas técnicas en la plataforma asegura que los hallazgos sean útiles para una variedad de usuarios, desde administradores y evaluadores hasta estudiantes, contribuyendo así a una mejor comprensión de las actitudes hacia el servicio comunitario y a la optimización de los programas relacionados.

Recomendaciones.

1. Mejoras en la Plataforma

- Optimización de la Interfaz de Usuario:
 - Realizar pruebas de usabilidad adicionales con un mayor número de usuarios para identificar áreas de mejora en la interfaz gráfica y garantizar una experiencia más intuitiva.

2. Ampliación del Alcance del Proyecto

- Inclusión de Más Grupos Poblacionales:
 - Ampliar la muestra de estudio para incluir a más instituciones educativas, comunidades o regiones, lo que permitiría obtener resultados más generalizables y representativos.

3. Capacitación y Difusión

- Capacitación para Usuarios:
 - Ofrecer talleres o capacitaciones a los usuarios finales (administradores, evaluadores y estudiantes) para maximizar el aprovechamiento de la plataforma.

Referencias Bibliográficas.

- ¿Qué es servicio comunitario? (09 de 02 de 2024). Obtenido de Cáritas de Monterrey:
<https://www.caritas.org.mx/que-es-servicio-comunitario/>
- Concepto y naturaleza de las actitudes. (s.f.). Obtenido de Algor cards:
<https://cards.algoreducation.com/es/content/jot3iM5n/naturaleza-actitudes-comportamiento>
- Gonzalez, F. (14 de 02 de 2023). Análisis Paramétricos vs No Paramétricos, ¿cómo y cuando utilizarlos? Obtenido de inLinkedIn: <https://es.linkedin.com/pulse/an%C3%A1lisis-param%C3%A9tricos-vs-c%C3%B3mo-y-cuando-utilizarlos-francisco-gonz%C3%A1lez>
- Marisa Tumino, V. K. (28 de junio de 2012). Acción y reflexión: actitud del estudiante hacia el aprendizaje y el servicio. Obtenido de https://www.scielo.org.ar/scielo.php?pid=S1669-27212012000100005&script=sci_arttext
- Rondón, L. (16 de 12 de 2020). El Hacer del Servicio Comunitario en la Formación Ética Social de los Estudiantes del Sector Universitario. Revista Científica CIENCIAEDUC.
- Velázquez, A. (17 de 04 de 2023). ¿Qué son las pruebas paramétricas? Obtenido de QuestionPro: <https://www.questionpro.com/blog/es/pruebas-parametricas/>
- Hiberus. (s. f.). ¿Qué es Angular y para qué sirve? Hiberus. <https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/>
- Axarnet. (s. f.). ¿Qué es Laravel? Axarnet Blog. <https://axarnet.es/blog/que-es-laravel#:~:text=Qu%C3%A9%20es%20Laravel,-Si%20nos%20ce%C3%B1imos&text=PHP%20es%20el%20lenguaje%20de,en%20PHP%20sus%20aplicaciones%20web>
- Google Cloud. (s. f.). Cloud SQL para MySQL. Google Cloud. <https://cloud.google.com/mysql?hl=es>
- Rubio, J. (2020, 28 mayo). Qué es GitHub y qué les ofrece a los desarrolladores. Xataka. <https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores>
- EBIS Education. (s. f.). GitHub: Qué es, para qué sirve, ventajas y cómo usarlo. EBIS Education. <https://www.ebiseducation.com/github-que-es-para-que-sirve-ventajas-y-como-usarlo>
- OpenWebinars. (2022, 27 mayo). ¿Qué es JSON Web Token y cómo funciona? OpenWebinars. <https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/>
- Auth0. (s. f.). JSON Web Tokens: lo que necesitas saber. Auth0. <https://auth0.com/es/learn/json-web-tokens>
- Ceei Valencia. (2023, 27 marzo). “Tokens” y su uso en aplicaciones web. Emprenemjunts. <https://ceeivalencia.emprenemjunts.es/?op=8&n=28660>
- Aledo, A. (2021, 30 marzo). Introducción a TypeScript. Medium. <https://medium.com/@alejodev95/introducci%C3%B3n-a-typescript-9740a71aaee>
- CodersFree. (s. f.). ¿Qué es PHP? Programación, ventajas y por qué usarlo. CodersFree. <https://codersfree.com/posts/que-es-php-programacion-ventajas-usarlo>
- Asociación para el Progreso de la Dirección (APD). (s. f.). Metodología Scrum: ¿qué es? APD. <https://www.apd.es/metodologia-scrum-que-es/>
- Eagly, A. H., & Chaiken, S. (1993). The psychology of attitudes. Harcourt Brace Jovanovich College Publishers.

- Few, S. (2012). *Show me the numbers: Designing tables and graphs to enlighten*. Analytics Press.
- Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques*. Elsevier.
- Hiberus. (s. f.). ¿Qué es Angular y para qué sirve? Recuperado de <https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/>
- Tumino, M. (2012). Acción y reflexión: actitud del estudiante hacia el aprendizaje y el servicio. Recuperado de https://www.scielo.org.ar/scielo.php?pid=S1669-27212012000100005&script=sci_arttext
- KIO Networks. (s.f.). *Técnicas y beneficios de recolección de datos digitales*. KIO Networks. Recuperado el 25 de febrero de 2025, de <https://www.kio.tech/blog/tecnicas-y-beneficios-de-recoleccion-de-datos-digitales>
- Field, A. (2018). *Discovering statistics using IBM SPSS statistics* (5th ed.). SAGE Publications.
- Hair, J. F., Black, W. C., Babin, B. J., & Anderson, R. E. (2019). *Multivariate data analysis* (8th ed.). Cengage Learning.
- Montgomery, D. C. (2019). *Design and analysis of experiments* (10th ed.). Wiley.

Anexos.

En este apartado colocaremos las capturas de pantalla de código el cual dividiremos en dos apartados, que estaría así, Backend (Crud) y frontend (src) el cual tenemos los diferentes códigos a usar, de igual forma, después la explicación de lo que vendría siendo nuestro back, estaríamos mencionando lo que vendría siendo nuestra base de datos, explicando la relación que tienen

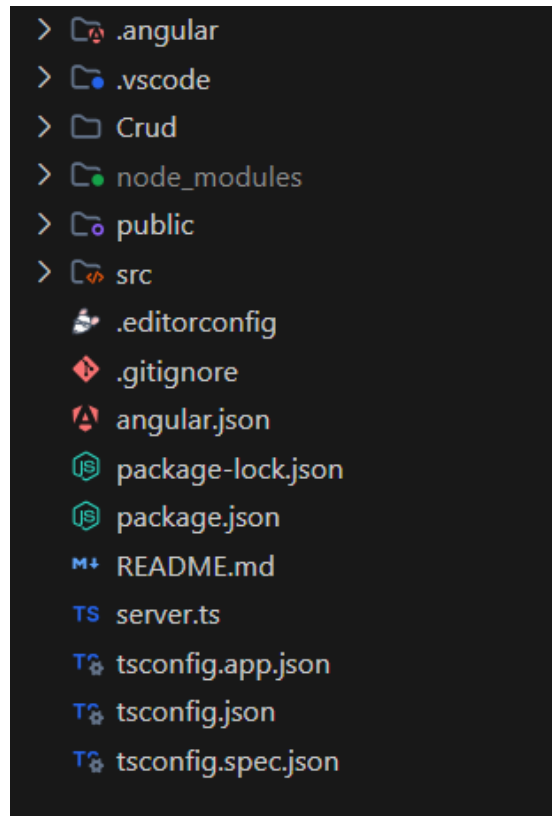


Imagen Anexo 1

Como se puede mostrar en la imagen, la estructura web organizada en dos partes principales: el frontend y el backend.

1. Backend (Carpeta Crud): Esta carpeta contiene el código del backend del proyecto. En el backend, se maneja la lógica, las interacciones con la base de datos, y las APIs que el frontend consume. Aquí podrías tener rutas, controladores, y configuraciones para interactuar con tu servidor.
2. Frontend (Carpeta src): La carpeta src contiene el código del frontend del proyecto, que está desarrollado en Angular. Aquí se encuentran los componentes, servicios, módulos y

todo lo necesario para crear la interfaz de usuario interactiva que se comunica con el backend.

La estructura también incluye archivos de configuración como angular.json, que define la configuración del proyecto Angular, y otros archivos como tsconfig.json que están relacionados con la configuración de TypeScript.

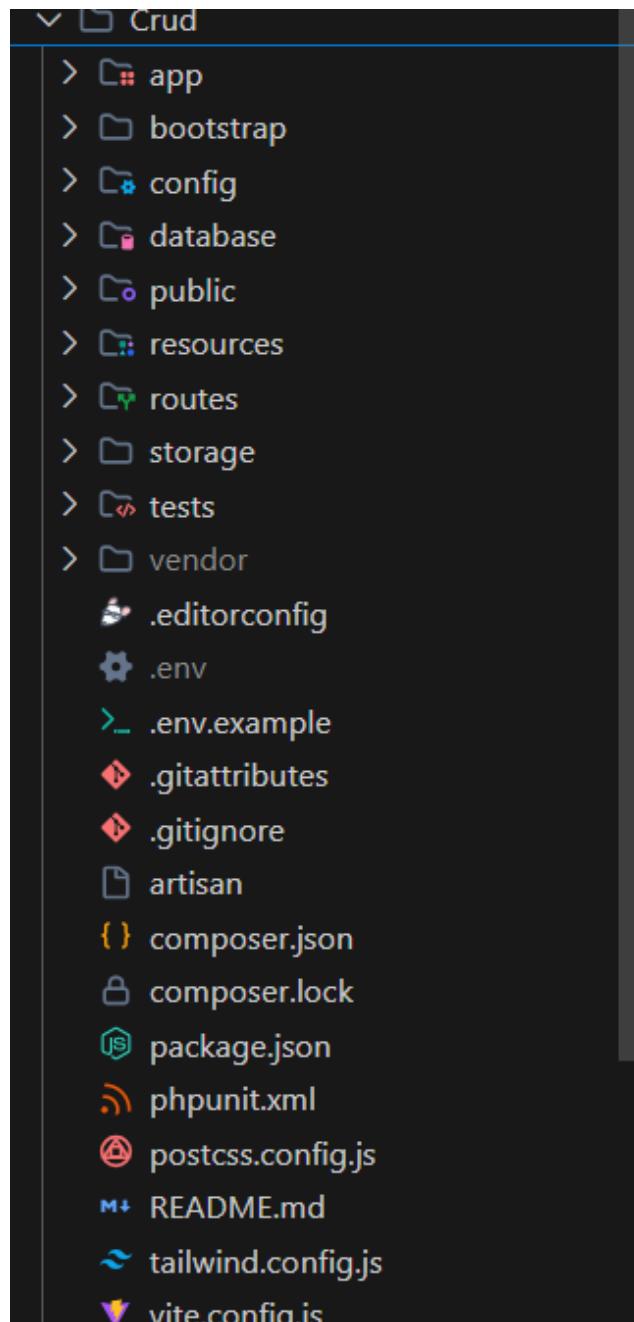


Imagen Anexo 2

En este proyecto tiene una estructura típica de una aplicación Laravel en el backend. A continuación, una descripción general de los principales directorios y archivos que aparecen:

1. Carpetas principales:

- app: Contiene la lógica principal de la aplicación, como controladores, modelos, middleware, y servicios. Es el núcleo del backend.
- bootstrap: Incluye los archivos necesarios para inicializar la aplicación, como el archivo app.php que configura el framework.
- config: Almacena archivos de configuración para diferentes aspectos de la aplicación, como base de datos, servicios externos, y más.
- database: Contiene migraciones, factories, y seeders para gestionar la base de datos.
- public: Carpeta accesible públicamente que sirve los activos como imágenes, archivos CSS y JavaScript. También incluye el punto de entrada de la aplicación (index.php).
- resources: Almacena vistas, archivos de idioma y otros recursos como plantillas Blade.
- routes: Define las rutas de la aplicación en archivos como web.php y api.php.
- storage: Almacena archivos generados por la aplicación, como logs, caché, y subidas temporales.
- tests: Contiene pruebas automatizadas para verificar la funcionalidad del sistema.
- vendor: Carpeta generada por Composer que contiene dependencias externas del proyecto.

2. Archivos importantes:

- .env: Archivo de configuración donde defines variables de entorno, como las credenciales de la base de datos y claves API.

- `composer.json` y `composer.lock`: Administran las dependencias PHP del proyecto mediante Composer.
- `artisan`: Herramienta de línea de comandos para tareas comunes como la ejecución de migraciones, la creación de controladores, y más.
- `package.json`: Archivo para gestionar dependencias de Node.js y herramientas de frontend (si se usan en el proyecto).

- **Backend**

Este archivo es el **.env** de un proyecto Laravel, utilizado para definir variables de entorno que configuran el comportamiento del proyecto como se ve en la siguiente imagen:

A screenshot of a code editor window showing a .env file. The editor has a dark background with light blue and green text. At the top left, there are three colored circles (red, yellow, green) representing window control buttons. The code is as follows:

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:8yMUAJ5bPa0lsmjke7urFPrF+qFGZYyEvFBvjMhpSnk=
4 APP_DEBUG=true
5 APP_TIMEZONE=UTC
6 APP_URL=http://localhost
7
8 APP_LOCALE=en
9 APP_FALLBACK_LOCALE=en
10 APP_FAKER_LOCALE=en_US
11
12 APP_MAINTENANCE_DRIVER=file
13 # APP_MAINTENANCE_STORE=database
14
15 PHP_CLI_SERVER_WORKERS=4
16
17 BCRYPT_ROUNDS=12
18
19 LOG_CHANNEL=stack
20 LOG_STACK=single
21 LOG_DEPRECATED_CHANNEL=null
22 LOG_LEVEL=debug
23
24 DB_CONNECTION=mysql
25 DB_HOST=127.0.0.1
26 DB_PORT=3306
27 DB_DATABASE=encuesta
28 DB_USERNAME=root
29 DB_PASSWORD=
```

Imagen Anexo 3

Teniendo en cuenta que la parte mas importante en este momento seria la configuracion de la base de datos, como se podría mencionar a continuación:

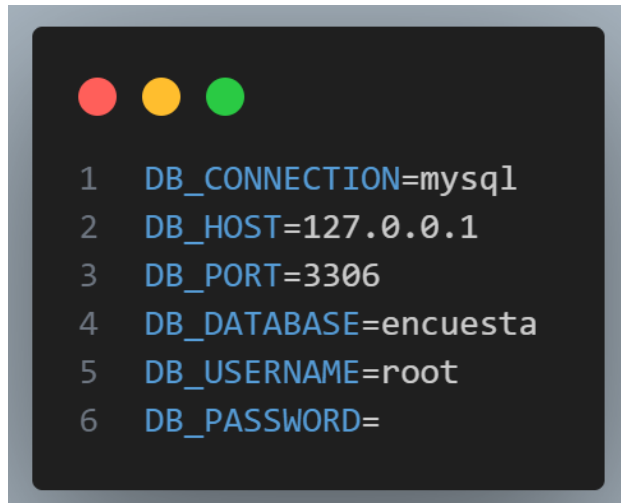


Imagen Anexo 4

- **Configuración de base de datos**

DB_CONNECTION=mysql:

Define el tipo de conexión con la base de datos (MySQL en este caso).

DB_HOST=127.0.0.1:

Especifica la dirección del servidor de base de datos (localhost).

DB_PORT=3306:

Puerto utilizado para conectarse al servidor de MySQL.

DB_DATABASE=encuesta:

Nombre de la base de datos a la que se conecta la aplicación.

DB_USERNAME=root:

Usuario para la conexión a la base de datos.

DB_PASSWORD=:

Contraseña para la conexión a la base de datos (vacía en este caso).

Ya que en este apartado es el que nos permitirá, tener la conexión con la base de datos, como se mostrara mas adelante.

Teniendo en la carpeta **database** del proyecto:

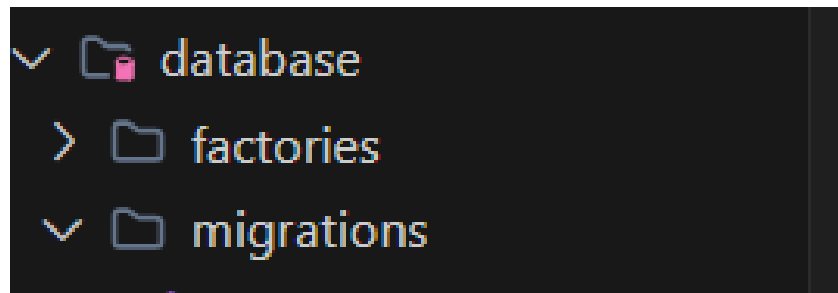


Imagen Anexo 5

Está organizada para gestionar la base de datos del proyecto, teniendo en cuenta que esta es una carpeta es la principal o la que nos permite tener el código de las tablas de la base de datos, que posteriormente se migran para crear las tablas con sus respectivas columnas, teniendo una breve descripción

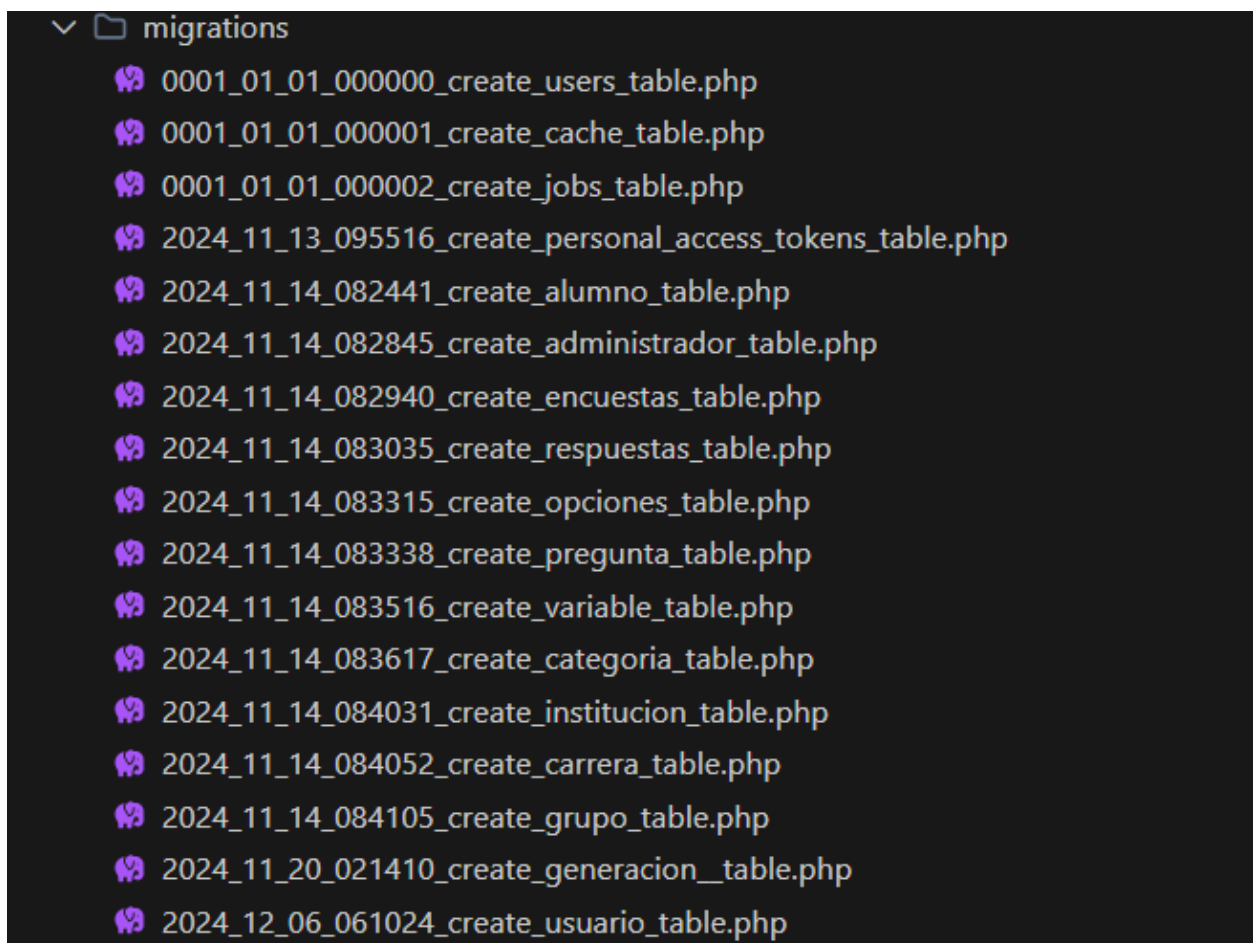


Imagen Anexo 6

migrations:

- Esta carpeta contiene los archivos de migración, que son scripts que definen la estructura de las tablas de la base de datos. Cada archivo incluye un nombre descriptivo (por ejemplo, `create_users_table`) y un timestamp que indica cuándo se creó la migración.
- En los nombres de los archivos se puede observar la organización cronológica, y cada migración define instrucciones para crear o modificar tablas de la base de datos. Ejemplo de tablas que aparecen:
 - `create_users_table`: Define la tabla de usuarios.
 - `create_permissions_table`: Crea una tabla para gestionar permisos.
 - `create_roles_table`: Define roles, probablemente para un sistema de autenticación.
 - `create_jobs_table`: Podría ser una tabla para tareas en segundo plano o gestión de empleos.
 - Otras tablas relacionadas con categorías, órdenes, productos, etc.

Propósito general:

- Migrations: Permiten la gestión controlada de cambios en la base de datos, asegurando que todas las instancias del proyecto tengan la misma estructura de datos. También permiten revertir cambios si es necesario.
- Factories: Son una herramienta para generar datos dinámicos y poblar la base de datos con información de prueba de manera rápida.

Teniendo en cuenta que se hará una descripción de una tabla, para tener en cuenta como es la relación entre este código y la base de datos por ejemplo:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('usuario', function (Blueprint $table) {
            $table->bigIncrements('idUsuario'); // Definir el campo como autoincrementable
            $table->integer('matricula')->unique();
            $table->string('nombres', 45);
            $table->string('primer_apellido', 45);
            $table->string('segundo_apellido', 45);
            $table->string('correo', 45)->unique();
            $table->string('password', 255);
            $table->enum('tipo', ['Alumno', 'Administrador']);
            $table->unsignedBigInteger('Alumno_Matricula')->nullable();
            $table->unsignedBigInteger('Administrador_idAdministrador')->nullable();
            $table->enum('genero', ['Masculino', 'Femenino', 'Otro']); // Opciones para género
            $table->integer('edad');
            $table->timestamps();
        });
    }
};
```

```

    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('usuario');
    }
};

```

Esta es una migración de Laravel que define la estructura de una tabla llamada **usuario** en la base de datos, tomando en cuenta algunos puntos como lo que podrían ser:

1. Propósito del archivo

Este archivo es una clase de migración que se encarga de crear la tabla usuario cuando se ejecuta la migración. La tabla incluye una variedad de columnas para almacenar datos relacionados con los usuarios.

2. Detalles de las columnas

- **idUsuario (bigIncrements):**
 - Clave primaria de la tabla, definida como auto-incremental.
- **matricula (integer):**
 - Número único que identifica al usuario, marcado como **único** (unique()).
- **nombres (string):**
 - Almacena el nombre del usuario con una longitud máxima de 45 caracteres.
- **primer_apellido y segundo_apellido (string):**
 - Guardan los apellidos del usuario, cada uno con un límite de 45 caracteres.

- **correo (string):**
 - Campo único para almacenar el correo electrónico del usuario con un límite de 45 caracteres.
- **password (string):**
 - Almacena la contraseña del usuario, con un tamaño máximo de 255 caracteres.
- **tipo (enum):**
 - Define el tipo de usuario, con dos posibles valores: **Alumno** o **Administrador**.
- **Alumno_Matricula (unsignedBigInteger):**
 - Relación opcional con otra tabla que utiliza este campo como clave foránea, ya que es nullable().
- **Administrador_idAdministrador (unsignedBigInteger):**
 - Relación opcional con otra tabla para identificar a los administradores, también nullable().
- **genero (enum):**
 - Almacena el género del usuario, con las opciones: **Masculino**, **Femenino**, y **Otro**.
- **edad (integer):**
 - Campo para guardar la edad del usuario.
- **timestamps():**
 - Agrega automáticamente dos columnas: **created_at** y **updated_at**, para registrar las fechas de creación y última actualización del registro.

3. Aplicación práctica

Cuando esta migración se ejecuta con el comando `php artisan migrate`, Laravel creará la tabla `usuario` con la estructura definida. Esto permite manejar información detallada y organizada de los usuarios, asegurando integridad y flexibilidad en los datos,

SELECT * FROM `usuario`

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

Opciones extra

	idUsuario	matricula	nombres	primer_apellido	segundo_apellido	correo
☐ Editar Copiar Borrar	4	1234445	Juan hh	Pérez gg	López gg	juan.peddrez@example.com
☐ Editar Copiar Borrar	6	1235	Judh	Pédz gg	López gg	juan.peddrez@hsj.com
☐ Editar Copiar Borrar	7	12435	Juffdh	Pédzff gg	Lóffpez gg	juan.peddrez@hffj.com
☐ Editar Copiar Borrar	8	13335	Juffdddh	Pddédzff gg	ddLóffpez gg	juan.pedddrez@hffj.com

☐ Seleccionar todo | Para los elementos que están marcados: [Editar](#) [Copiar](#) [Borrar](#) [Exportar](#)

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

Imagen anexo 7

En la imagen muestra los datos almacenados de nuestra tabla **usuario** de la base de datos a través de una interfaz de administración de MySQL, como **phpMyAdmin**.

Estructura y Datos

1. Columnas principales:

- **idUsuario:**
 - Es la clave primaria de la tabla. Identifica de forma única a cada registro.
- **matricula:**
 - Número único asignado a cada usuario.
- **nombres:**
 - Nombre(s) del usuario.

- **primer_apellido y segundo_apellido:**
 - Los apellidos del usuario.
- **correo:**
 - Dirección de correo electrónico asociada al usuario.
- **password:**
 - Contraseña del usuario, almacenada de forma encriptada (por ejemplo, usando bcrypt).
- **tipo:**
 - Tipo de usuario, que en este caso es Alumno para todos los registros.

Igual debemos tener en cuenta que el proceso de crear archivos para la creación de las tablas de deberá realizar por todos y cada una de las tablas que a continuación describiré, esto para que cada archivo, cada tabla, este bien configurada así como creada

Ahora teniendo en cuenta que esta parte se debe realizar por cada tabla por lo tanto se mostrara la imagen de las tablas creadas

Tabla alumnos

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('alumno', function (Blueprint $table) {
            $table->integer('matricula')->primary();
            $table->string('nombres', 45);
            $table->string('primer_apellido', 45);
            $table->string('segundo_apellido', 45);
            $table->unsignedBigInteger('Institucion_idInstitucion');
            $table->unsignedBigInteger('Carreras_idCarrera');
            $table->unsignedBigInteger('Generaciones_idGeneracion');
            $table->unsignedBigInteger('Grupo_idGrupo');
            $table->string('genero', 45);
            $table->integer('edad');
            $table->timestamps();
        });
    }
}
```

```
/**  
 * Reverse the migrations.  
 */  
public function down(): void  
{  
    Schema::dropIfExists('alumno');  
}  
};
```

Tabla del administrador

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```

```
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('administrador', function (Blueprint $table) {
            $table->id('idAdministrador');
            $table->string('nombre', 45);
            $table->unsignedBigInteger('Grupo_idGrupo');
            $table->integer('Alumno_Matricula');
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('administrador');
    }
};
```

Tabla encuesta

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```

```
{
```

```
    /**
```

```
     * Run the migrations.
```

```
    */
```

```
    public function up(): void
```

```
    {
```

```
        Schema::create('encuestas', function (Blueprint $table) {
```

```
            $table->id('idEncuestas');
```

```
            $table->string('fecha', 45);
```

```
            $table->integer('Alumno_Matricula');
```

```
            $table->unsignedBigInteger('Respuesta_idRespuesta');
```

```
            $table->timestamps();
```

```
        });
```

```
    }
```

```
    /**
```

```
     * Reverse the migrations.
```

```
    */
```

```
    public function down(): void
```

```
    {
```

```
        Schema::dropIfExists('encuestas');
```

```
    }
```

```
};
```

Tabla respuestas

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```

```
{
```

```
    /**
```

```
     * Run the migrations.
```

```
    */
```

```
    public function up(): void
```

```
    {
```

```
        Schema::create('respuestas', function (Blueprint $table) {
```

```
            $table->id('idRespuestas');
```

```
            $table->unsignedBigInteger('opciones_idOpciones');
```

```
            $table->unsignedBigInteger('opciones_pregunta_idPregunta');
```

```
            $table->unsignedBigInteger('opciones_pregunta_variable_idVariable');
```

```
            $table->timestamps();
```

```
        });
```

```
    }
```

```
    /**
```

```
     * Reverse the migrations.
```

```
    */
```

```
    public function down(): void
```

```
    {
```

```
        Schema::dropIfExists('respuestas');
```

```
    }
```

```
};
```

Tabla opciones

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```

```
{
```

```
    /**
```

```
     * Run the migrations.
```

```
    */
```

```
    public function up(): void
```

```
    {
```

```
        Schema::create('opciones', function (Blueprint $table) {
```

```
            $table->id('idOpciones');
```

```
            $table->string('texto', 45);
```

```
            $table->string('valor', 45);
```

```
            $table->unsignedBigInteger('pregunta_idPregunta');
```

```
            $table->timestamps();
```

```
        });
```

```
    }
```

```
    /**
```

```
     * Reverse the migrations.
```

```
    */
```

```
    public function down(): void
```

```
    {
```

```
        Schema::dropIfExists('opciones');
```

```
    }
```

```
};
```

Tabla Pregunta

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```

```
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('pregunta', function (Blueprint $table) {
            $table->id('idPregunta');
            $table->string('texto', 200);
            $table->unsignedBigInteger('variable_idVariable');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('pregunta');
    }
};
```

Table variable

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```

```
{
```

```
    /**
```

```
     * Run the migrations.
```

```
    */
```

```
    public function up(): void
```

```
    {
```

```
        Schema::create('variable', function (Blueprint $table) {
```

```
            $table->id('idVariable');
```

```
            $table->string('nombre', 45);
```

```
            $table->unsignedBigInteger('categoria_idCategoria');
```

```
            $table->timestamps();
```

```
        });
```

```
    }
```

```
    /**
```

```
     * Reverse the migrations.
```

```
    */
```

```
    public function down(): void
```

```
    {
```

```
        Schema::dropIfExists('variable');
```

```
    }
```

```
};
```

Tabla categoría

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```

```
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('categoria', function (Blueprint $table) {
            $table->id('idCategoria');
            $table->string('nombre', 45);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('categoria');
    }
};
```

Tabla Institución

<?php

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('institucion', function (Blueprint $table) {
            $table->id('idInstitucion');
            $table->string('nombre', 45);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('institucion');
    }
};
```

Tabla Carrera

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```

```
{
```

```
    /**
```

```
     * Run the migrations.
```

```
    */
```

```
    public function up(): void
```

```
    {
```

```
        Schema::create('carrera', function (Blueprint $table) {
```

```
            $table->id('idCarrera');
```

```
            $table->string('nombre', 45);
```

```
            $table->unsignedBigInteger('institucion_idInstitucion');
```

```
            $table->timestamps();
```

```
        });
```

```
    }
```

```
    /**
```

```
     * Reverse the migrations.
```

```
    */
```

```
    public function down(): void
```

```
    {
```

```
        Schema::dropIfExists('carrera');
```

```
    }
```

```
};
```

Tabla Grupo

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
```

```
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('grupo', function (Blueprint $table) {
            $table->id('idGrupo');
            $table->string('nombre', 45);
            $table->unsignedBigInteger('generaciones_idGeneracion');
            $table->unsignedBigInteger('idAdministrador');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('grupo');
    }
};
```

Tabla Generacion

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('generacion_', function (Blueprint $table) {
            $table->id('idGeneracion');
            $table->date('fecha');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('generacion_');
    }
};
```

Teniendo en cuenta todo esto ya deberíamos tener todas las tablas como podemos observar en la siguiente imagen:

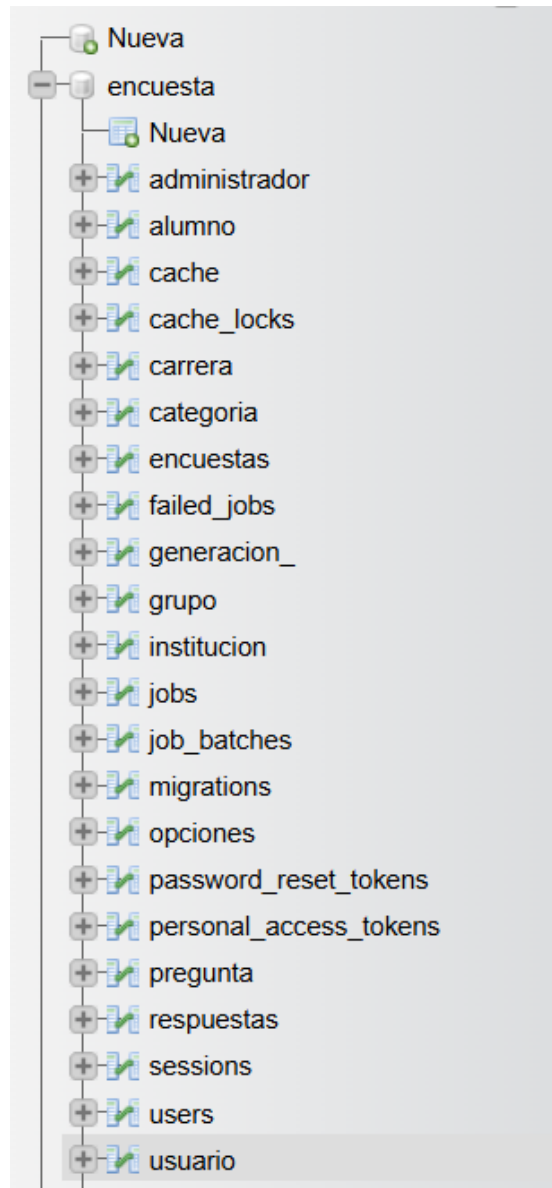


Imagen anexo 8

Como podemos observar que la base de datos es **Encuesta**, y al ver que esta bien configurado, podemos percatarnos que se han creado las tablas correspondientes.

Carpeta App, controladores y modelos

Controladores (Controllers)

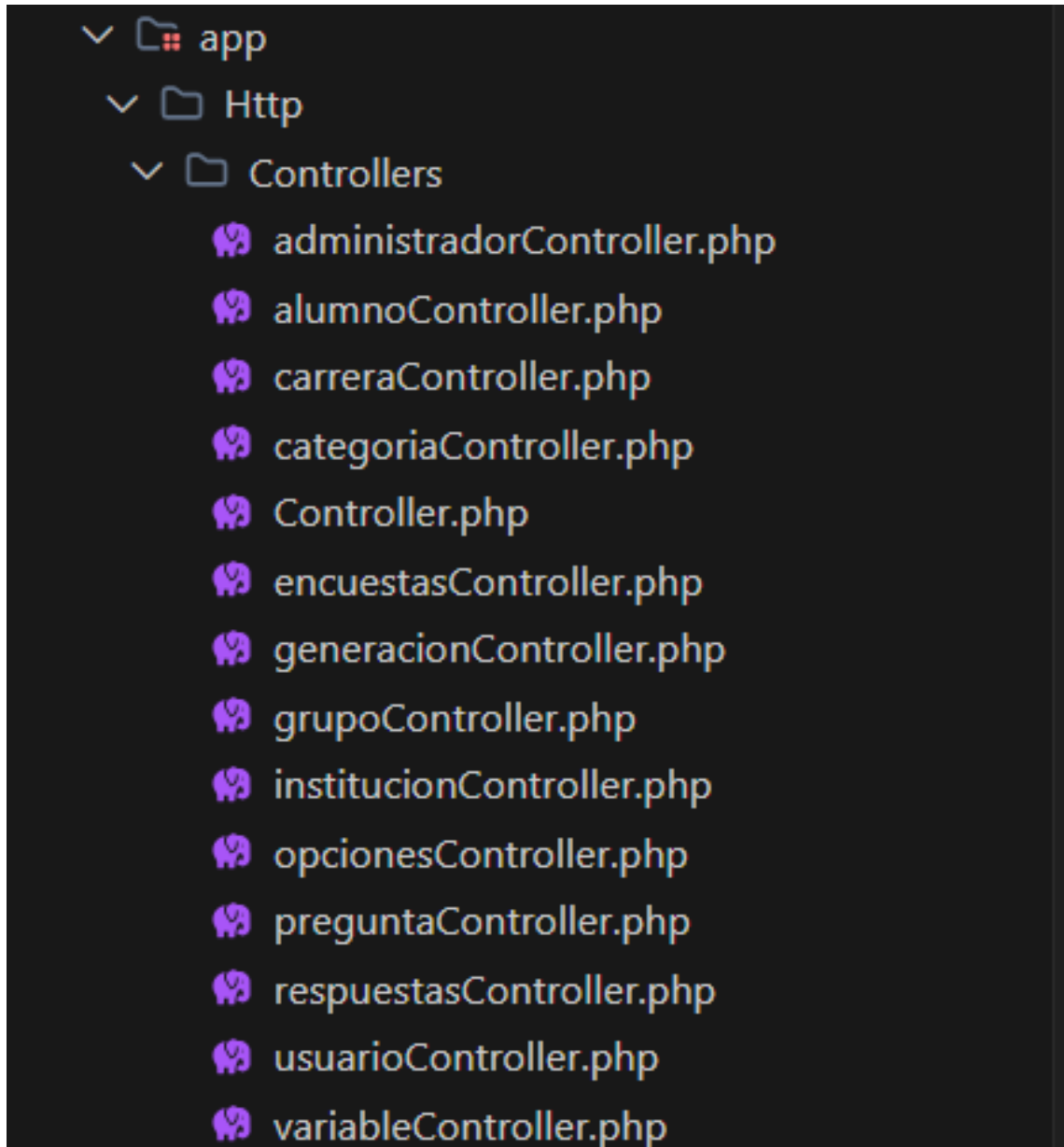


Imagen anexo 9

- Cada archivo en esta carpeta es un controlador que maneja las solicitudes HTTP y la lógica asociada para cada entidad del sistema (como alumno, usuario, pregunta, etc.).
- Ejemplos:

- **alumnoController.php:** Maneja las operaciones CRUD y otras interacciones relacionadas con la entidad Alumno.
- **encuestasController.php:** Gestiona las encuestas, como la creación, edición, y visualización.
- **preguntaController.php:** Se encarga de las operaciones relacionadas con las preguntas de las encuestas.

Estos controladores son responsables de procesar las solicitudes, interactuar con los modelos, y devolver respuestas, como vistas o datos JSON.

Controlador del usuario

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Usuario;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use function Laravel\Prompts\error;

class usuarioController extends Controller{
    public function indexUsuario()
    {
        $usuarios = Usuario::all();

        $data = [
            'usuarios' => $usuarios,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store (Request $request)
    {
        $validator = validator::make($request->all(),[
            'idUserio' => 'required',
            'matricula' => 'required',
            'nombres' => 'required',
```

```

        'primer_apellido' => 'required',
        'segundo_apellido' => 'required',
        'correo' => 'required|email',
        'password' => 'required',
        //'tipo' => 'required',
        //'Alumno_Matricula' => 'required',
        //'Administrador_idAdministrador' => 'required',
        'genero' => 'required',
        'edad' => 'required'
    ]);

    if ($validator -> fails()){
        $data = [
            'message' => 'Error en la validacion de datos',
            'errors' => $validator->errors(),
            'status' => 200
        ];
        return response()->json($data, 400);
    }

    $usuarios = Usuario::create([
        'idUserario' => $request -> idUsuario,
        'matricula' => $request -> matricula,
        'nombres' => $request -> nombres,
        'primer_apellido' => $request-> primer_apellido,
        'segundo_apellido' => $request -> segundo_apellido,
        'correo' => $request -> correo,
        'password' => $request -> password,
        'tipo' => $request -> tipo,
        'Alumno_Matricula' => $request -> Alumno_Matricula,
        'Administrador_idAdministrador' => $request -> Administrador_idAdministrador,
        'genero' => $request -> genero,
    ]);

```

```

        'edad' => $request -> edad
    });

    if (!$usuarios){
        $data = [
            'message'=> 'error al crear al usuario',
            'status' => 500
        ];
        return response()->json($data, 500);
    }

    $data =[
        'message' => $usuarios,
        'status' => 2001
    ];

    return response()->json($data, 201);
}

public function show($idUserario){

    $usuarios = Usuario::where('idUserario', $idUserario)->first();

    if (!$usuarios) {
        $data = [
            'message' => 'Estudiante no encontrado',
            'status' => 404
        ];
        return response()->json($data, 404);
    }
}

```

```

$data = [
    'usuario' => $usuarios,
    'status' => 200
];

return response()->json($data, 200);
}

public function destroy($idUserario)
{
    $usuarios = Usuario::where('idUserario', $idUserario)->first();

    if (!$usuarios) {
        $data = [
            'message' => 'Estudiante no encontrado',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $usuarios->delete();

    $data = [
        'message' => 'Estudiante eliminado',
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function update(Request $request, $idUserario){

```

```

$usuarios = Usuario::where('idUsuario', $idUsuario)->first();
if (!$usuarios) {
    $data = [
        'message' => 'Estudiante no encontrado',
        'status' => 404
    ];
    return response()->json($data, 404);
}

```

```

$validator = validator::make($request->all(),[
    'idUsuario' => 'required',
    'matricula' => 'required',
    'nombres' => 'required',
    'primer_apellido' => 'required',
    'segundo_apellido' => 'required',
    'correo' => 'required|email',
    'password' => 'required',
    'tipo' => 'required',
    'Alumno_Matricula' => 'required',
    'Administrador_idAdministrador' => 'required',
    'genero' => 'required',
    'edad' => 'required'
]);

```

```

if ($validator->fails()){
    $data = [
        'message' => 'Error de validacion',
        'errors' => $validator->errors(),
        'status' => 400
    ];
    return response()->json($data, 400);
}

```

```

    }
    $usuarios-> idUsuario = $request -> idUsuario;
    $usuarios-> matricula = $request -> matricula;
    $usuarios-> nombres = $request -> nombres;
    $usuarios-> primer_apellido = $request-> primer_apellido;
    $usuarios-> segundo_apellido = $request -> segundo_apellido;
    $usuarios-> correo = $request -> correo;
    $usuarios-> password = $request -> password;
    $usuarios-> tipo = $request -> tipo;
    $usuarios-> Alumno_Matricula = $request -> Alumno_Matricula;
    $usuarios-> Administrador_idAdministrador = $request ->
Administrador_idAdministrador;
    $usuarios-> genero = $request -> genero;
    $usuarios-> edad = $request -> edad;

    $usuarios ->save();

    $data = [
        'message'=> 'Estudiante actualizado',
        'usuarios' => $usuarios,
        'status' => 200
    ];

    return response()->json($data,200);
}
}

```

Controlador del administrador

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Administrador;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use function Laravel\Prompts\error;

class AdministradorController extends Controller
{
    public function indexAdministrador()
    {
        $administradores = Administrador::all();

        $data = [
            'administradores' => $administradores,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'idAdministrador' => 'required',
            'nombre' => 'required',
            'Grupo_idGrupo' => 'required',
            'Alumno_Matricula' => 'required'
        ]);

        if ($validator->fails()) {
            $data = [
                'message' => 'Error en la validación de datos',
                'errors' => $validator->errors(),
                'status' => 400
            ];
            return response()->json($data, 400);
        }

        $administrador = Administrador::create([
            'idAdministrador' => $request->idAdministrador,
```

```

        'nombre' => $request->nombre,
        'Grupo_idGrupo' => $request->Grupo_idGrupo,
        'Alumno_Matricula' => $request->Alumno_Matricula
    ]);

    if (!$administrador) {
        $data = [
            'message' => 'Error al crear el administrador',
            'status' => 500
        ];
        return response()->json($data, 500);
    }

    $data = [
        'message' => $administrador,
        'status' => 201
    ];

    return response()->json($data, 201);
}

public function show($idAdministrador)
{
    $administrador = Administrador::where('idAdministrador', $idAdministrador)->first();

    if (!$administrador) {
        $data = [
            'message' => 'Administrador no encontrado',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $data = [
        'administrador' => $administrador,
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function destroy($idAdministrador)
{
    $administrador = Administrador::where('idAdministrador', $idAdministrador)->first();

    if (!$administrador) {

```

```

    $data = [
        'message' => 'Administrador no encontrado',
        'status' => 404
    ];
    return response()->json($data, 404);
}

$administrador->delete();

$data = [
    'message' => 'Administrador eliminado',
    'status' => 200
];

return response()->json($data, 200);
}

public function update(Request $request, $idAdministrador)
{
    $administrador = Administrador::where('idAdministrador', $idAdministrador)->first();
    if (!$administrador) {
        $data = [
            'message' => 'Administrador no encontrado',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $validator = Validator::make($request->all(), [
        'idAdministrador' => 'required',
        'nombre' => 'required',
        'Grupo_idGrupo' => 'required',
        'Alumno_Matricula' => 'required'
    ]);

    if ($validator->fails()) {
        $data = [
            'message' => 'Error de validación',
            'errors' => $validator->errors(),
            'status' => 400
        ];
        return response()->json($data, 400);
    }

    $administrador->idAdministrador = $request->idAdministrador;
    $administrador->nombre = $request->nombre;

```

```
$administrador->Grupo_idGrupo = $request->Grupo_idGrupo;
$administrador->Alumno_Matricula = $request->Alumno_Matricula;

$administrador->save();

$data = [
    'message' => 'Administrador actualizado',
    'administrador' => $administrador,
    'status' => 200
];

return response()->json($data, 200);
}
}
```

Controlador alumno

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Alumno;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class AlumnoController extends Controller
{
    /**
     * Obtener todos los alumnos.
     */
    public function indexAlumno()
    {
        $alumnos = Alumno::with('usuario')->get(); // Incluir datos del usuario relacionado

        return response()->json([
            'alumnos' => $alumnos,
            'status' => 200,
        ], 200);
    }

    /**
     * Registrar un nuevo alumno.
     */
    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'usuario_id' => 'required|exists:usuario,idUsuario', // Relacionar con usuario
            'nombres' => 'required|string|max:45',
            'primer_apellido' => 'required|string|max:45',
            'segundo_apellido' => 'nullable|string|max:45',
            'Institucion_idInstitucion' => 'required|integer',
            'Carreras_idCarrera' => 'required|integer',
            'Generaciones_idGeneracion' => 'required|integer',
            'Grupo_idGrupo' => 'required|integer',
            'genero' => 'required|string|max:10',
            'edad' => 'required|integer|min:1',
        ]);

        if ($validator->fails()) {
            return response()->json([
```

```

        'message' => 'Error en la validación de datos.',
        'errors' => $validator->errors(),
        'status' => 422,
    ], 422);
}

$alumno = Alumno::create($request->all());

return response()->json([
    'message' => 'Alumno creado exitosamente.',
    'alumno' => $alumno,
    'status' => 201,
], 201);
}

/**
 * Obtener un alumno por su matrícula.
 */
public function show($matricula)
{
    $alumno = Alumno::with('usuario')->where('matricula', $matricula)->first();

    if (!$alumno) {
        return response()->json([
            'message' => 'Alumno no encontrado.',
            'status' => 404,
        ], 404);
    }

    return response()->json([
        'alumno' => $alumno,
        'status' => 200,
    ], 200);
}

/**
 * Eliminar un alumno.
 */
public function destroy($matricula)
{
    $alumno = Alumno::where('matricula', $matricula)->first();

    if (!$alumno) {
        return response()->json([
            'message' => 'Alumno no encontrado.',
            'status' => 404,

```

```

    ], 404);
}

$alumno->delete();

return response()->json([
    'message' => 'Alumno eliminado exitosamente.',
    'status' => 200,
], 200);
}

/**
 * Actualizar un alumno.
 */
public function update(Request $request, $matricula)
{
    $alumno = Alumno::where('matricula', $matricula)->first();

    if (!$alumno) {
        return response()->json([
            'message' => 'Alumno no encontrado.',
            'status' => 404,
        ], 404);
    }

    $validator = Validator::make($request->all(), [
        'nombres' => 'required|string|max:45',
        'primer_apellido' => 'required|string|max:45',
        'segundo_apellido' => 'nullable|string|max:45',
        'Institucion_idInstitucion' => 'required|integer',
        'Carreras_idCarrera' => 'required|integer',
        'Generaciones_idGeneracion' => 'required|integer',
        'Grupo_idGrupo' => 'required|integer',
        'genero' => 'required|string|max:10',
        'edad' => 'required|integer|min:1',
    ]);

    if ($validator->fails()) {
        return response()->json([
            'message' => 'Error en la validación de datos.',
            'errors' => $validator->errors(),
            'status' => 422,
        ], 422);
    }

    $alumno->update($request->all());
}

```

```
return response()->json([
    'message' => 'Alumno actualizado exitosamente.',
    'alumno' => $alumno,
    'status' => 200,
], 200);
}
}
```

Controlador de carrera

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Carrera;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use function Laravel\Prompts\error;

class CarreraController extends Controller{
    public function indexCarrera()
    {
        $carreras = Carrera::all();

        $data = [
            'carreras' => $carreras,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'idCarrera' => 'required',
            'nombre' => 'required',
            'institucion_idInstitucion' => 'required'
```

```
]);
```

```
if ($validator->fails()) {  
    $data = [  
        'message' => 'Error en la validación de datos',  
        'errors' => $validator->errors(),  
        'status' => 400  
    ];  
    return response()->json($data, 400);  
}
```

```
$carrera = Carrera::create([  
    'idCarrera' => $request->idCarrera,  
    'nombre' => $request->nombre,  
    'institucion_idInstitucion' => $request->institucion_idInstitucion  
]);
```

```
if (!$carrera) {  
    $data = [  
        'message' => 'Error al crear la carrera',  
        'status' => 500  
    ];  
    return response()->json($data, 500);  
}
```

```
$data = [  
    'message' => $carrera,  
    'status' => 201  
];
```

```
return response()->json($data, 201);
```

```
}
```

```
public function show($idCarrera)
{
    $carrera = Carrera::where('idCarrera', $idCarrera)->first();

    if (!$carrera) {
        $data = [
            'message' => 'Carrera no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }
```

```
    $data = [
        'carrera' => $carrera,
        'status' => 200
    ];

    return response()->json($data, 200);
}
```

```
public function destroy($idCarrera)
{
    $carrera = Carrera::where('idCarrera', $idCarrera)->first();

    if (!$carrera) {
        $data = [
            'message' => 'Carrera no encontrada',
            'status' => 404
        ];
```

```

        return response()->json($data, 404);
    }

    $carrera->delete();

    $data = [
        'message' => 'Carrera eliminada',
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function update(Request $request, $idCarrera)
{
    $carrera = Carrera::where('idCarrera', $idCarrera)->first();
    if (!$carrera) {
        $data = [
            'message' => 'Carrera no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $validator = Validator::make($request->all(), [
        'idCarrera' => 'required',
        'nombre' => 'required',
        'institucion_idInstitucion' => 'required'
    ]);

    if ($validator->fails()) {

```

```

    $data = [
        'message' => 'Error de validación',
        'errors' => $validator->errors(),
        'status' => 400
    ];
    return response()->json($data, 400);
}

$carrera->idCarrera = $request->idCarrera;
$carrera->nombre = $request->nombre;
$carrera->institucion_idInstitucion = $request->institucion_idInstitucion;

$carrera->save();

$data = [
    'message' => 'Carrera actualizada',
    'carrera' => $carrera,
    'status' => 200
];

return response()->json($data, 200);
}
}

```

Controlador de categoría

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Categoria;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class CategoriaController extends Controller{
    public function indexCategoria()
    {
        $categorias = Categoria::all();

        $data = [
            'categorias' => $categorias,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'idCategoria' => 'required',
            'nombre' => 'required'
        ]);

        if ($validator->fails()) {
            $data = [
                'message' => 'Error en la validación de datos',
                'errors' => $validator->errors(),
                'status' => 400
            ];
            return response()->json($data, 400);
        }

        $categoria = Categoria::create([
            'idCategoria' => $request->idCategoria,
            'nombre' => $request->nombre
        ]);

        if (!$categoria) {
```

```

    $data = [
      'message' => 'Error al crear la categoría',
      'status' => 500
    ];
    return response()->json($data, 500);
  }

  $data = [
    'message' => $categoria,
    'status' => 201
  ];

  return response()->json($data, 201);
}

public function show($idCategoria)
{
  $categoria = Categoria::where('idCategoria', $idCategoria)->first();

  if (!$categoria) {
    $data = [
      'message' => 'Categoría no encontrada',
      'status' => 404
    ];
    return response()->json($data, 404);
  }

  $data = [
    'categoria' => $categoria,
    'status' => 200
  ];

  return response()->json($data, 200);
}

public function destroy($idCategoria)
{
  $categoria = Categoria::where('idCategoria', $idCategoria)->first();

  if (!$categoria) {
    $data = [
      'message' => 'Categoría no encontrada',
      'status' => 404
    ];
    return response()->json($data, 404);
  }
}

```

```

$categoria->delete();

$data = [
    'message' => 'Categoría eliminada',
    'status' => 200
];

return response()->json($data, 200);
}

public function update(Request $request, $idCategoria)
{
    $categoria = Categoria::where('idCategoria', $idCategoria)->first();

    if (!$categoria) {
        $data = [
            'message' => 'Categoría no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $validator = Validator::make($request->all(), [
        'idCategoria' => 'required',
        'nombre' => 'required'
    ]);

    if ($validator->fails()) {
        $data = [
            'message' => 'Error de validación',
            'errors' => $validator->errors(),
            'status' => 400
        ];
        return response()->json($data, 400);
    }

    $categoria->idCategoria = $request->idCategoria;
    $categoria->nombre = $request->nombre;

    $categoria->save();

    $data = [
        'message' => 'Categoría actualizada',
        'categoria' => $categoria,
        'status' => 200
    ]

```

```
];  
  return response()->json($data, 200);  
}  
}
```

Controlador encuesta

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Encuestas;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class EncuestasController extends Controller{
    public function indexEncuestas()
    {
        $encuestas = Encuestas::all();

        $data = [
            'encuestas' => $encuestas,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'idEncuestas' => 'required',
            'fecha' => 'required|date',
            'Alumno_Matricula' => 'required',
            'Respuesta_idRespuesta' => 'required'
```

```

]);

if ($validator->fails()) {
    $data = [
        'message' => 'Error en la validación de datos',
        'errors' => $validator->errors(),
        'status' => 400
    ];
    return response()->json($data, 400);
}

$encuesta = Encuestas::create([
    'idEncuestas' => $request->idEncuestas,
    'fecha' => $request->fecha,
    'Alumno_Matricula' => $request->Alumno_Matricula,
    'Respuesta_idRespuesta' => $request->Respuesta_idRespuesta
]);

if (!$encuesta) {
    $data = [
        'message' => 'Error al crear la encuesta',
        'status' => 500
    ];
    return response()->json($data, 500);
}

$data = [
    'message' => $encuesta,
    'status' => 201
];

```

```

    return response()->json($data, 201);
}

public function show($idEncuestas)
{
    $encuesta = Encuestas::where('idEncuestas', $idEncuestas)->first();

    if (!$encuesta) {
        $data = [
            'message' => 'Encuesta no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $data = [
        'encuesta' => $encuesta,
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function destroy($idEncuestas)
{
    $encuesta = Encuestas::where('idEncuestas', $idEncuestas)->first();

    if (!$encuesta) {
        $data = [
            'message' => 'Encuesta no encontrada',
            'status' => 404
        ];
    }
}

```

```

];
return response()->json($data, 404);
}

$encuesta->delete();

$data = [
    'message' => 'Encuesta eliminada',
    'status' => 200
];

return response()->json($data, 200);
}

public function update(Request $request, $idEncuestas)
{
    $encuesta = Encuestas::where('idEncuestas', $idEncuestas)->first();

    if (!$encuesta) {
        $data = [
            'message' => 'Encuesta no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $validator = Validator::make($request->all(), [
        'idEncuestas' => 'required',
        'fecha' => 'required|date',
        'Alumno_Matricula' => 'required',
        'Respuesta_idRespuesta' => 'required'
    ]);

```

```

]);

if ($validator->fails()) {
    $data = [
        'message' => 'Error de validación',
        'errors' => $validator->errors(),
        'status' => 400
    ];
    return response()->json($data, 400);
}

$encuesta->idEncuestas = $request->idEncuestas;
$encuesta->fecha = $request->fecha;
$encuesta->Alumno_Matricula = $request->Alumno_Matricula;
$encuesta->Respuesta_idRespuesta = $request->Respuesta_idRespuesta;

$encuesta->save();

$data = [
    'message' => 'Encuesta actualizada',
    'encuesta' => $encuesta,
    'status' => 200
];

return response()->json($data, 200);
}
}

```

Controlador de generación

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Generacion;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class GeneracionController extends Controller
{
    public function indexGeneraciones()
    {
        $generaciones = Generacion::all();

        $data = [
            'generaciones' => $generaciones,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'idGeneracion' => 'required',
            'nombre' => 'required'
        ]);

        if ($validator->fails()) {
            $data = [
                'message' => 'Error en la validación de datos',
                'errors' => $validator->errors(),
                'status' => 400
            ];
            return response()->json($data, 400);
        }

        $generacion = Generacion::create([
            'idGeneracion' => $request->idGeneracion,
            'nombre' => $request->nombre
        ]);
    }
}
```

```

    if (!$generacion) {
        $data = [
            'message' => 'Error al crear la generación',
            'status' => 500
        ];
        return response()->json($data, 500);
    }

    $data = [
        'message' => $generacion,
        'status' => 201
    ];

    return response()->json($data, 201);
}

public function show($idGeneracion)
{
    $generacion = Generacion::where('idGeneracion', $idGeneracion)->first();

    if (!$generacion) {
        $data = [
            'message' => 'Generación no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $data = [
        'generacion' => $generacion,
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function destroy($idGeneracion)
{
    $generacion = Generacion::where('idGeneracion', $idGeneracion)->first();

    if (!$generacion) {
        $data = [
            'message' => 'Generación no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }
}

```

```

    }

    $generacion->delete();

    $data = [
        'message' => 'Generación eliminada',
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function update(Request $request, $idGeneracion)
{
    $generacion = Generacion::where('idGeneracion', $idGeneracion)->first();

    if (!$generacion) {
        $data = [
            'message' => 'Generación no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $validator = Validator::make($request->all(), [
        'idGeneracion' => 'required',
        'nombre' => 'required'
    ]);

    if ($validator->fails()) {
        $data = [
            'message' => 'Error de validación',
            'errors' => $validator->errors(),
            'status' => 400
        ];
        return response()->json($data, 400);
    }

    $generacion->idGeneracion = $request->idGeneracion;
    $generacion->nombre = $request->nombre;

    $generacion->save();

    $data = [
        'message' => 'Generación actualizada',
        'generacion' => $generacion,
    ];
}

```

```
        'status' => 200
    ];
    return response()->json($data, 200);
}
}
```

Controlador de grupo

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Grupo;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class GrupoController extends Controller{
    public function indexGrupos()
    {
        $grupos = Grupo::all();

        $data = [
            'grupos' => $grupos,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'idGrupo' => 'required',
            'nombre' => 'required',
            'generaciones_idGeneracion' => 'required',
            'idAdministrador' => 'required'
        ]);

        if ($validator->fails()) {
            $data = [
                'message' => 'Error en la validación de datos',
                'errors' => $validator->errors(),
                'status' => 400
            ];
            return response()->json($data, 400);
        }

        $grupo = Grupo::create([
            'idGrupo' => $request->idGrupo,
            'nombre' => $request->nombre,
            'generaciones_idGeneracion' => $request->generaciones_idGeneracion,
```

```

        'idAdministrador' => $request->idAdministrador
    ]);

    if (!$grupo) {
        $data = [
            'message' => 'Error al crear el grupo',
            'status' => 500
        ];
        return response()->json($data, 500);
    }

    $data = [
        'message' => $grupo,
        'status' => 201
    ];

    return response()->json($data, 201);
}

public function show($idGrupo)
{
    $grupo = Grupo::where('idGrupo', $idGrupo)->first();

    if (!$grupo) {
        $data = [
            'message' => 'Grupo no encontrado',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $data = [
        'grupo' => $grupo,
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function destroy($idGrupo)
{
    $grupo = Grupo::where('idGrupo', $idGrupo)->first();

    if (!$grupo) {
        $data = [
            'message' => 'Grupo no encontrado',

```

```

        'status' => 404
    ];
    return response()->json($data, 404);
}

$grupo->delete();

$data = [
    'message' => 'Grupo eliminado',
    'status' => 200
];

return response()->json($data, 200);
}

public function update(Request $request, $idGrupo)
{
    $grupo = Grupo::where('idGrupo', $idGrupo)->first();

    if (!$grupo) {
        $data = [
            'message' => 'Grupo no encontrado',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $validator = Validator::make($request->all(), [
        'idGrupo' => 'required',
        'nombre' => 'required',
        'generaciones_idGeneracion' => 'required',
        'idAdministrador' => 'required'
    ]);

    if ($validator->fails()) {
        $data = [
            'message' => 'Error de validación',
            'errors' => $validator->errors(),
            'status' => 400
        ];
        return response()->json($data, 400);
    }

    $grupo->idGrupo = $request->idGrupo;
    $grupo->nombre = $request->nombre;
    $grupo->generaciones_idGeneracion = $request->generaciones_idGeneracion;

```

```
$grupo->idAdministrador = $request->idAdministrador;

$grupo->save();

$data = [
    'message' => 'Grupo actualizado',
    'grupo' => $grupo,
    'status' => 200
];

return response()->json($data, 200);
}
}
```

Controlador de la institución

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Institucion;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class InstitucionController extends Controller
{
    public function indexInstituciones()
    {
        $instituciones = Institucion::all();

        $data = [
            'instituciones' => $instituciones,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'idInstitucion' => 'required',
            'nombre' => 'required'
        ]);
```

```

if ($validator->fails()) {
    $data = [
        'message' => 'Error en la validación de datos',
        'errors' => $validator->errors(),
        'status' => 400
    ];
    return response()->json($data, 400);
}

$institucion = Institucion::create([
    'idInstitucion' => $request->idInstitucion,
    'nombre' => $request->nombre
]);

if (!$institucion) {
    $data = [
        'message' => 'Error al crear la institución',
        'status' => 500
    ];
    return response()->json($data, 500);
}

$data = [
    'message' => $institucion,
    'status' => 201
];

return response()->json($data, 201);
}

```

```

public function show($idInstitucion)
{
    $institucion = Institucion::where('idInstitucion', $idInstitucion)->first();

    if (!$institucion) {
        $data = [
            'message' => 'Institución no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $data = [
        'institucion' => $institucion,
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function destroy($idInstitucion)
{
    $institucion = Institucion::where('idInstitucion', $idInstitucion)->first();

    if (!$institucion) {
        $data = [
            'message' => 'Institución no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }
}

```

```

    $institucion->delete();

    $data = [
        'message' => 'Institución eliminada',
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function update(Request $request, $idInstitucion)
{
    $institucion = Institucion::where('idInstitucion', $idInstitucion)->first();

    if (!$institucion) {
        $data = [
            'message' => 'Institución no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $validator = Validator::make($request->all(), [
        'idInstitucion' => 'required',
        'nombre' => 'required'
    ]);

    if ($validator->fails()) {
        $data = [
            'message' => 'Error de validación',

```

```

        'errors' => $validator->errors(),
        'status' => 400
    ];
    return response()->json($data, 400);
}

$institucion->idInstitucion = $request->idInstitucion;
$institucion->nombre = $request->nombre;

$institucion->save();

$data = [
    'message' => 'Institución actualizada',
    'institucion' => $institucion,
    'status' => 200
];

return response()->json($data, 200);
}
}

```

Controlador de opciones

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Opciones;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class OpcionesController extends Controller
{
    public function indexOpciones()
    {
        $opciones = Opciones::all();

        $data = [
            'opciones' => $opciones,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'idOpciones' => 'required',
            'texto' => 'required',
            'valor' => 'required|numeric',
        ]);
    }
}
```

```

        'pregunta_idPregunta' => 'required'
    });

    if ($validator->fails()) {
        $data = [
            'message' => 'Error en la validación de datos',
            'errors' => $validator->errors(),
            'status' => 400
        ];
        return response()->json($data, 400);
    }

    $opciones = Opciones::create([
        'idOpciones' => $request->idOpciones,
        'texto' => $request->texto,
        'valor' => $request->valor,
        'pregunta_idPregunta' => $request->pregunta_idPregunta
    ]);

    if (!$opciones) {
        $data = [
            'message' => 'Error al crear la opción',
            'status' => 500
        ];
        return response()->json($data, 500);
    }

    $data = [
        'message' => $opciones,
        'status' => 201
    ];

```

```

    return response()->json($data, 201);
}

public function show($idOpciones)
{
    $opcion = Opciones::where('idOpciones', $idOpciones)->first();

    if (!$opcion) {
        $data = [
            'message' => 'Opción no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $data = [
        'opcion' => $opcion,
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function destroy($idOpciones)
{
    $opcion = Opciones::where('idOpciones', $idOpciones)->first();

    if (!$opcion) {
        $data = [
            'message' => 'Opción no encontrada',

```

```

        'status' => 404
    ];
    return response()->json($data, 404);
}

$opcion->delete();

$data = [
    'message' => 'Opción eliminada',
    'status' => 200
];

return response()->json($data, 200);
}

public function update(Request $request, $idOpciones)
{
    $opcion = Opciones::where('idOpciones', $idOpciones)->first();

    if (!$opcion) {
        $data = [
            'message' => 'Opción no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $validator = Validator::make($request->all(), [
        'idOpciones' => 'required',
        'texto' => 'required',
        'valor' => 'required|numeric',
    ]);

```

```

        'pregunta_idPregunta' => 'required'
    ]);

    if ($validator->fails()) {
        $data = [
            'message' => 'Error de validación',
            'errors' => $validator->errors(),
            'status' => 400
        ];
        return response()->json($data, 400);
    }

    $opcion->idOpciones = $request->idOpciones;
    $opcion->texto = $request->texto;
    $opcion->valor = $request->valor;
    $opcion->pregunta_idPregunta = $request->pregunta_idPregunta;

    $opcion->save();

    $data = [
        'message' => 'Opción actualizada',
        'opcion' => $opcion,
        'status' => 200
    ];

    return response()->json($data, 200);
}
}

```

Controlador de preguntas

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Pregunta;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class PreguntaController extends Controller
{
    public function indexPregunta()
    {
        $preguntas = Pregunta::all();

        $data = [
            'preguntas' => $preguntas,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'idPregunta' => 'required',
            'texto' => 'required',
            'variable_idVariable' => 'required'
        ]);

        if ($validator->fails()) {
            $data = [
                'message' => 'Error en la validación de datos',
                'errors' => $validator->errors(),
                'status' => 400
            ];
            return response()->json($data, 400);
        }

        $pregunta = Pregunta::create([
            'idPregunta' => $request->idPregunta,
            'texto' => $request->texto,
            'variable_idVariable' => $request->variable_idVariable
        ]);
    }
}
```

```

]);

if (!$pregunta) {
    $data = [
        'message' => 'Error al crear la pregunta',
        'status' => 500
    ];
    return response()->json($data, 500);
}

$data = [
    'message' => $pregunta,
    'status' => 201
];

return response()->json($data, 201);
}

public function show($idPregunta)
{
    $pregunta = Pregunta::where('idPregunta', $idPregunta)->first();

    if (!$pregunta) {
        $data = [
            'message' => 'Pregunta no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $data = [
        'pregunta' => $pregunta,
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function destroy($idPregunta)
{
    $pregunta = Pregunta::where('idPregunta', $idPregunta)->first();

    if (!$pregunta) {
        $data = [
            'message' => 'Pregunta no encontrada',
            'status' => 404
        ];
    }
}

```

```

    ];
    return response()->json($data, 404);
}

$pregunta->delete();

$data = [
    'message' => 'Pregunta eliminada',
    'status' => 200
];

return response()->json($data, 200);
}

public function update(Request $request, $idPregunta)
{
    $pregunta = Pregunta::where('idPregunta', $idPregunta)->first();

    if (!$pregunta) {
        $data = [
            'message' => 'Pregunta no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $validator = Validator::make($request->all(), [
        'idPregunta' => 'required',
        'texto' => 'required',
        'variable_idVariable' => 'required'
    ]);

    if ($validator->fails()) {
        $data = [
            'message' => 'Error de validación',
            'errors' => $validator->errors(),
            'status' => 400
        ];
        return response()->json($data, 400);
    }

    $pregunta->idPregunta = $request->idPregunta;
    $pregunta->texto = $request->texto;
    $pregunta->variable_idVariable = $request->variable_idVariable;

    $pregunta->save();
}

```

```
$data = [  
  'message' => 'Pregunta actualizada',  
  'pregunta' => $pregunta,  
  'status' => 200  
];  
  
return response()->json($data, 200);  
}  
}
```

Controlador de respuestas

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Respuestas;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class RespuestasController extends Controller
{
    public function indexRespuestas()
    {
        $respuestas = Respuestas::all();

        $data = [
            'respuestas' => $respuestas,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'idRespuestas' => 'required',
            'opciones_idOpciones' => 'required',
            'opciones_pregunta_idPregunta' => 'required',
        ]);
    }
}
```

```

        'opciones_pregunta_variable_idVariable' => 'required'
    ]);

    if ($validator->fails()) {
        $data = [
            'message' => 'Error en la validación de datos',
            'errors' => $validator->errors(),
            'status' => 400
        ];
        return response()->json($data, 400);
    }

    $respuesta = Respuestas::create([
        'idRespuestas' => $request->idRespuestas,
        'opciones_idOpciones' => $request->opciones_idOpciones,
        'opciones_pregunta_idPregunta' => $request->opciones_pregunta_idPregunta,
        'opciones_pregunta_variable_idVariable' => $request->opciones_pregunta_variable_idVariable
    ]);

    if (!$respuesta) {
        $data = [
            'message' => 'Error al crear la respuesta',
            'status' => 500
        ];
        return response()->json($data, 500);
    }

    $data = [
        'message' => $respuesta,
        'status' => 201
    ]

```

```

];

return response()->json($data, 201);
}

public function show($idRespuestas)
{
    $respuesta = Respuestas::where('idRespuestas', $idRespuestas)->first();

    if (!$respuesta) {
        $data = [
            'message' => 'Respuesta no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $data = [
        'respuesta' => $respuesta,
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function destroy($idRespuestas)
{
    $respuesta = Respuestas::where('idRespuestas', $idRespuestas)->first();

    if (!$respuesta) {
        $data = [

```

```

        'message' => 'Respuesta no encontrada',
        'status' => 404
    ];
    return response()->json($data, 404);
}

$respuesta->delete();

$data = [
    'message' => 'Respuesta eliminada',
    'status' => 200
];

return response()->json($data, 200);
}

public function update(Request $request, $idRespuestas)
{
    $respuesta = Respuestas::where('idRespuestas', $idRespuestas)->first();

    if (!$respuesta) {
        $data = [
            'message' => 'Respuesta no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $validator = Validator::make($request->all(), [
        'idRespuestas' => 'required',
        'opciones_idOpciones' => 'required',
    ]);

```

```

        'opciones_pregunta_idPregunta' => 'required',
        'opciones_pregunta_variable_idVariable' => 'required'
    ]);

    if ($validator->fails()) {
        $data = [
            'message' => 'Error de validación',
            'errors' => $validator->errors(),
            'status' => 400
        ];
        return response()->json($data, 400);
    }

    $respuesta->idRespuestas = $request->idRespuestas;
    $respuesta->opciones_idOpciones = $request->opciones_idOpciones;
    $respuesta->opciones_pregunta_idPregunta = $request->opciones_pregunta_idPregunta;
    $respuesta->opciones_pregunta_variable_idVariable = $request-
>opciones_pregunta_variable_idVariable;

    $respuesta->save();

    $data = [
        'message' => 'Respuesta actualizada',
        'respuesta' => $respuesta,
        'status' => 200
    ];

    return response()->json($data, 200);
}
}

```

Controlador de variables

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\Variable;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class VariableController extends Controller
{
    public function indexVariable()
    {
        $variables = Variable::all();

        $data = [
            'variables' => $variables,
            'status' => 200
        ];

        return response()->json($data, 200);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'idVariable' => 'required',
            'nombre' => 'required',
            'categoria_idCategoria' => 'required'
```

```
]);
```

```
if ($validator->fails()) {  
    $data = [  
        'message' => 'Error en la validación de datos',  
        'errors' => $validator->errors(),  
        'status' => 400  
    ];  
    return response()->json($data, 400);  
}
```

```
$variable = Variable::create([  
    'idVariable' => $request->idVariable,  
    'nombre' => $request->nombre,  
    'categoria_idCategoria' => $request->categoria_idCategoria  
]);
```

```
if (!$variable) {  
    $data = [  
        'message' => 'Error al crear la variable',  
        'status' => 500  
    ];  
    return response()->json($data, 500);  
}
```

```
$data = [  
    'message' => $variable,  
    'status' => 201  
];
```

```
return response()->json($data, 201);
```

```
}
```

```
public function show($idVariable)
```

```
{
```

```
    $variable = Variable::where('idVariable', $idVariable)->first();
```

```
    if (!$variable) {
```

```
        $data = [
```

```
            'message' => 'Variable no encontrada',
```

```
            'status' => 404
```

```
        ];
```

```
        return response()->json($data, 404);
```

```
    }
```

```
    $data = [
```

```
        'variable' => $variable,
```

```
        'status' => 200
```

```
    ];
```

```
    return response()->json($data, 200);
```

```
}
```

```
public function destroy($idVariable)
```

```
{
```

```
    $variable = Variable::where('idVariable', $idVariable)->first();
```

```
    if (!$variable) {
```

```
        $data = [
```

```
            'message' => 'Variable no encontrada',
```

```
            'status' => 404
```

```
        ];
```

```

        return response()->json($data, 404);
    }

    $variable->delete();

    $data = [
        'message' => 'Variable eliminada',
        'status' => 200
    ];

    return response()->json($data, 200);
}

public function update(Request $request, $idVariable)
{
    $variable = Variable::where('idVariable', $idVariable)->first();

    if (!$variable) {
        $data = [
            'message' => 'Variable no encontrada',
            'status' => 404
        ];
        return response()->json($data, 404);
    }

    $validator = Validator::make($request->all(), [
        'idVariable' => 'required',
        'nombre' => 'required',
        'categoria_idCategoria' => 'required'
    ]);

```

```

if ($validator->fails()) {
    $data = [
        'message' => 'Error de validación',
        'errors' => $validator->errors(),
        'status' => 400
    ];
    return response()->json($data, 400);
}

$variable->idVariable = $request->idVariable;
$variable->nombre = $request->nombre;
$variable->categoria_idCategoria = $request->categoria_idCategoria;

$variable->save();

$data = [
    'message' => 'Variable actualizada',
    'variable' => $variable,
    'status' => 200
];

return response()->json($data, 200);
}
}

```

Modelos (Models)

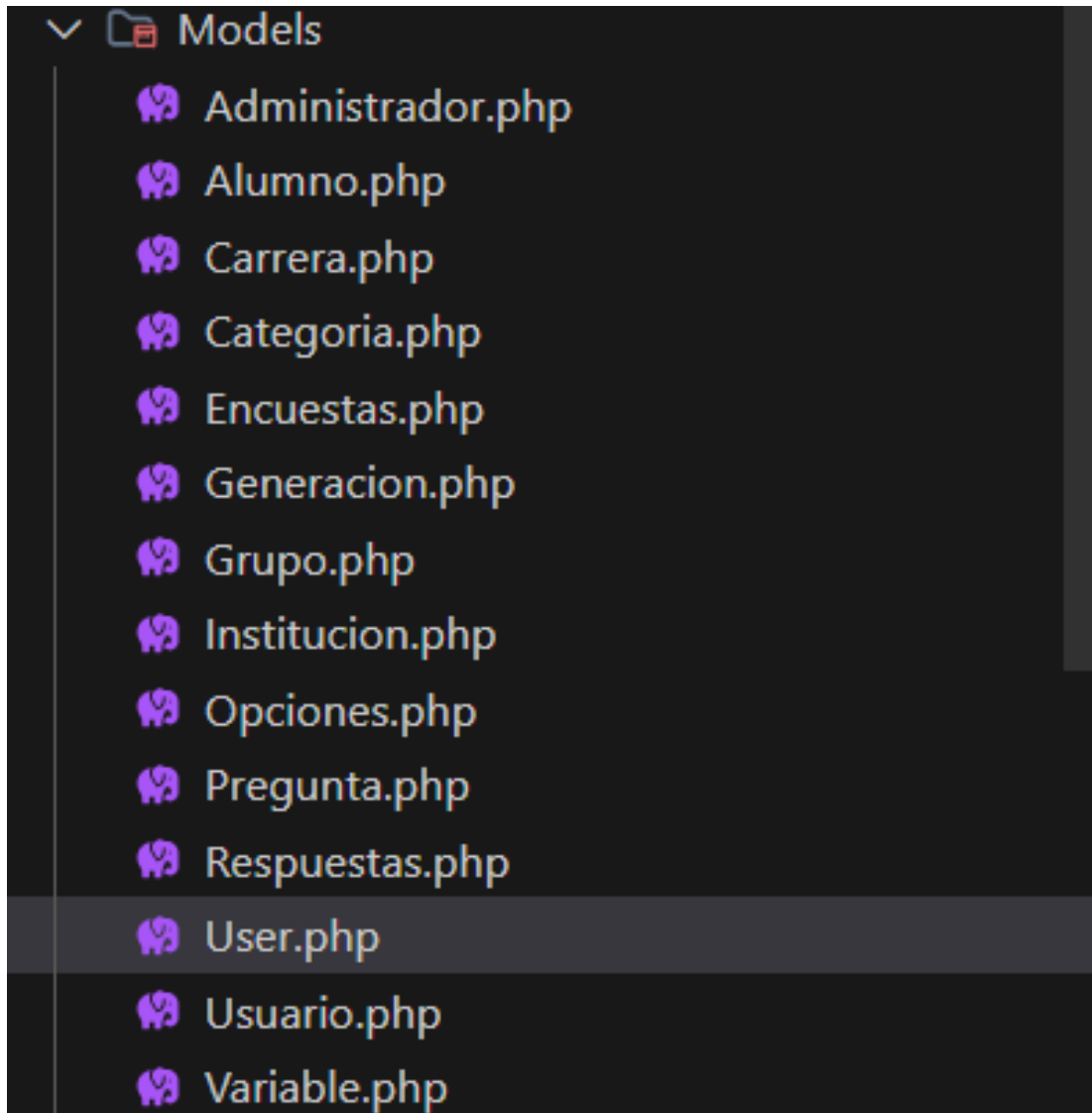


Imagen anexo 10

- Cada archivo representa un modelo que interactúa directamente con una tabla de la base de datos. Estos archivos definen las relaciones y la lógica asociada a cada entidad.
- Ejemplos:
 - **Alumno.php:** Representa la tabla alumno y sus relaciones (como grupos o encuestas).
 - **Categoria.php:** Gestiona la lógica relacionada con las categorías del sistema.

- **Encuestas.php:** Modela la estructura y operaciones relacionadas con la tabla encuestas.

Los modelos son fundamentales para la interacción con la base de datos mediante Eloquent ORM.

Modelo de usuario

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Usuario extends Model
{
    use HasFactory;

    protected $table = 'usuario'; // Nombre de la tabla
    protected $primaryKey = 'idUsuario'; // Clave primaria personalizada

    /**
     * Campos permitidos para asignación masiva.
     */
    protected $fillable = [
        'matricula',
        'nombres',
        'primer_apellido',
        'segundo_apellido',
        'correo',
        'password',
        //'tipo',
        //'Alumno_Matricula',
        //'Administrador_idAdministrador',
        'genero',
        'edad',
    ];
}
```

```

];

/**
 * Ocultar campos sensibles en las respuestas JSON.
 */
protected $hidden = ['password', 'created_at', 'updated_at'];

/**
 * Mutador para encriptar la contraseña automáticamente.
 */
public function setPasswordAttribute($value)
{
    if (!empty($value) && !password_get_info($value)['algo']) {
        $this->attributes['password'] = bcrypt($value);
    }
}

/**
 * Relación con el modelo Alumno.
 * Un usuario puede ser un alumno.
 */
public function alumno()
{
    return $this->hasOne(Alumno::class, 'usuario_id', 'idUserio');
}

/**
 * Relación con el modelo Administrador.
 * Un usuario puede ser un administrador.
 */
public function administrador()

```

```
{  
    return $this->hasOne(Administrador::class, 'usuario_id', 'idUsuario');  
}  
}
```

Modelo administrador

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Administrador extends Model{

    use HasFactory;

    protected $table = 'administrador';

    protected $primaryKey= 'idAdministrador';

    protected $fillable =[
        'idAdministrador',
        'nombre',
        'Grupo_idGrupo',
        'Alumno_Matricula'
    ];

    /**
     * Relación con el modelo Grupo.
     */
    public function grupo()
    {
        return $this->belongsTo(Grupo::class, 'Grupo_idGrupo', 'idGrupo');
```

```

}

/**
 * Relación con el modelo Alumno.
 */
public function alumno()
{
    return $this->belongsTo(Alumno::class, 'Alumno_Matricula', 'matricula');
}

/**
 * Relación con el modelo Usuario.
 */
public function usuarios()
{
    return $this->hasMany(Usuario::class, 'Administrador_idAdministrador',
'idAdministrador');
}
}

```

Modelo Alumno

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Alumno extends Model
{
    use HasFactory;

    // Nombre de la tabla asociada al modelo
    protected $table = 'alumno';

    // Clave primaria de la tabla
    protected $primaryKey = 'matricula';

    // Indicamos que la clave primaria no es autoincrementable
    public $incrementing = false;

    // Tipo de la clave primaria
    protected $keyType = 'string';

    // Campos que pueden ser llenados masivamente
    protected $fillable = [
        'matricula',
        'nombres',
        'primer_apellido',
        'segundo_apellido',
    ];
}
```

```

        'Institucion_idInstitucion',
        'Carreras_idCarrera',
        'Generaciones_idGeneracion',
        'Grupo_idGrupo',
        'genero',
        'edad',
    ];

    /**
     * Relación con el modelo Institución.
     * Un alumno pertenece a una institución.
     */
    public function institucion()
    {
        return $this->belongsTo(Institucion::class, 'Institucion_idInstitucion', 'idInstitucion');
    }

    /**
     * Relación con el modelo Carrera.
     * Un alumno pertenece a una carrera.
     */
    public function carrera()
    {
        return $this->belongsTo(Carrera::class, 'Carreras_idCarrera', 'idCarrera');
    }

    /**
     * Relación con el modelo Generación.
     * Un alumno pertenece a una generación.
     */
    public function generacion()

```

```

{
    return $this->belongsTo(Generacion::class, 'Generaciones_idGeneracion',
'idGeneracion');
}

/**
 * Relación con el modelo Grupo.
 * Un alumno pertenece a un grupo.
 */
public function grupo()
{
    return $this->belongsTo(Grupo::class, 'Grupo_idGrupo', 'idGrupo');
}

/**
 * Relación con el modelo Usuario.
 * Un alumno pertenece a un usuario.
 */
public function usuario()
{
    return $this->belongsTo(Usuario::class, 'matricula', 'Alumno_Matricula');
}
}

```

Modelo Carrera

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Carrera extends Model{

    use HasFactory;

    protected $table = 'carrera';

    protected $primaryKey= 'idCarrera';

    protected $fillable =[
        'idCarrera',
        'nombre',
        'institucion_idInstitucion'
    ];

    /**
     * Relación con el modelo Institucion.
     */
    public function institucion()
    {
        return $this->belongsTo(Institucion::class, 'institucion_idInstitucion', 'idInstitucion');
    }
}
```

```
/**  
 * Relación con el modelo Alumno.  
 */  
public function alumnos()  
{  
    return $this->hasMany(Alumno::class, 'Carreras_idCarrera', 'idCarrera');  
}  
}
```

Modelo Categoria

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Categoria extends Model{

    use HasFactory;

    protected $table = 'categoria';

    protected $primaryKey= 'idCategoria';

    protected $fillable =[
        'idCategoria',
        'nombre'
    ];

}
```

Modelo Encuesta

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Encuestas extends Model{

    use HasFactory;

    protected $table = 'encuestas';

    protected $primaryKey= 'idEncuestas';

    protected $fillable =[
        'idEncuestas',
        'fecha',
        'Alumno_Matricula',
        'Respuesta_idRespuesta'
    ];

    /**
     * Relación con el modelo Alumno.
     */
    public function alumno()
    {
        return $this->belongsTo(Alumno::class, 'Alumno_Matricula', 'matricula');
    }
}
```

```
/**  
 * Relación con el modelo Respuesta.  
 */  
public function respuesta()  
{  
    return $this->belongsTo(Respuestas::class, 'Respuesta_idRespuesta', 'idRespuesta');  
}  
}
```

Modelo generación

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Generacion extends Model{
    use HasFactory;

    protected $table = 'generaciones';

    protected $primaryKey= 'idGeneracion';

    protected $fillable = [
        'idGeneracion',
        'nombre',
    ];

    /**
     * Relación con el modelo Grupo.
     */
    public function grupos()
    {
        return $this->hasMany(Grupo::class, 'generaciones_idGeneracion', 'idGeneracion');
    }

    /**
     * Relación con el modelo Alumno.
```

```
*/  
public function alumnos()  
{  
    return $this->hasMany(Alumno::class, 'Generaciones_idGeneracion', 'idGeneracion');  
}  
}
```

Modelo grupo

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Grupo extends Model{

    use HasFactory;

    protected $table = 'grupo';

    protected $primaryKey= 'idGrupo';

    protected $fillable =[
        'idGrupo',
        'nombre',
        'generaciones_idGeneracion',
        'idAdministrador'
    ];

    public function alumno()
    {
        return $this->belongsTo(Alumno::class, 'Alumno_Matricula', 'matricula');
    }

    public function respuesta()
    {

```

```
        return $this->belongsTo(Respuestas::class, 'Respuesta_idRespuesta', 'idRespuesta');  
    }  
}
```

Modelo Institución

```
<?php
```

```
namespace App\Models;
```

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
```

```
use Illuminate\Database\Eloquent\Model;
```

```
class Institucion extends Model{
```

```
    use HasFactory;
```

```
    protected $table = 'institucion';
```

```
    protected $primaryKey= 'idInstitucion';
```

```
    protected $fillable =[
```

```
        'idInstitucion',
```

```
        'nombre'
```

```
    ];
```

```
    /**
```

```
     * Relación con el modelo Carrera.
```

```
     * Una institución puede tener muchas carreras asociadas.
```

```
     */
```

```
    public function carreras()
```

```
    {
```

```
        return $this->hasMany(Carrera::class, 'institucion_idInstitucion', 'idInstitucion');
```

```
    }
```

```
/**
 * Relación con el modelo Generacion.
 * Una institución puede tener muchas generaciones asociadas.
 */
public function generaciones()
{
    return $this->hasMany(Generacion::class, 'institucion_idInstitucion', 'idInstitucion');
}
}
```

Modelo opciones

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Opciones extends Model{

    use HasFactory;

    protected $table = 'opciones';

    protected $primaryKey= 'idOpciones';

    protected $fillable =[
        'idOpciones',
        'texto',
        'valor',
        'pregunta_idPregunta'
    ];

    /**
     * Relación con el modelo Pregunta.
     * Una opción pertenece a una única pregunta.
     */
    public function pregunta()
    {
        return $this->belongsTo(Pregunta::class, 'pregunta_idPregunta', 'idPregunta');
```

}
}

Modelo pregunta

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Pregunta extends Model{

    use HasFactory;

    protected $table = 'pregunta';

    protected $primaryKey= 'idPregunta';

    protected $fillable =[
        'idPregunta',
        'texto',
        'variable_idVariable'
    ];
    /**
     * Relación con el modelo Variable.
     * Una pregunta pertenece a una única variable.
     */
    public function variable()
    {
        return $this->belongsTo(Variable::class, 'variable_idVariable', 'idVariable');
    }
}
```

```
/**
 * Relación con el modelo Opciones.
 * Una pregunta tiene muchas opciones.
 */
public function opciones()
{
    return $this->hasMany(Opciones::class, 'pregunta_idPregunta', 'idPregunta');
}
}
```

Modelo respuestas

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Respuestas extends Model{

    use HasFactory;

    protected $table = 'respuestas';

    protected $primaryKey= 'idRespuestas';

    protected $fillable =[
        'idRespuestas',
        'opciones_idOpciones',
        'opciones_pregunta_idPregunta',
        'opciones_pregunta_variable_idVariable'
    ];

    /**
     * Relación con el modelo Opciones.
     * Una respuesta pertenece a una única opción.
     */
    public function opciones()
    {
        return $this->belongsTo(Opciones::class, 'opciones_idOpciones', 'idOpciones');
```

```

}

/**
 * Relación con el modelo Pregunta.
 * Una respuesta también se puede asociar con una pregunta a través de la opción.
 */
public function pregunta()
{
    return $this->hasOneThrough(Pregunta::class, Opciones::class, 'idOpciones', 'idPregunta',
'opciones_idOpciones', 'pregunta_idPregunta');
}

/**
 * Relación con el modelo Variable.
 * Una respuesta también se puede asociar con una variable a través de la opción.
 */
public function variable()
{
    return $this->hasOneThrough(Variable::class, Opciones::class, 'idOpciones', 'idVariable',
'opciones_idOpciones', 'variable_idVariable');
}
}

```

Modelo variable

```
<?php

namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Variable extends Model{

    use HasFactory;

    protected $table = 'variable';

    protected $primaryKey= 'idVariable';

    protected $fillable =[
        'idVariable',
        'nombre',
        'categoria_idCategoria'
    ];

    /**
     * Relación con el modelo Categoria.
     * Una variable pertenece a una categoría.
     */
    public function categoria()
    {
        return $this->belongsTo(Categoria::class, 'categoria_idCategoria', 'idCategoria');
    }
}
```

```
/**
 * Relación con el modelo Pregunta.
 * Una variable puede tener muchas preguntas asociadas.
 */
public function preguntas()
{
    return $this->hasMany(Pregunta::class, 'variable_idVariable', 'idVariable');
}
}
```

Ahora en la carpeta **routes** que contiene los archivos donde se definen las rutas del proyecto Laravel. En este caso, el archivo **api.php** está diseñado específicamente para manejar las rutas relacionadas con APIs.

Propósito del archivo **api.php**:

- Este archivo define las rutas que serán utilizadas por la API de la aplicación.
- Las rutas definidas aquí están prefijadas automáticamente con **/api**. Por ejemplo:
 - Una ruta definida como `Route::get('/usuarios', ...)` en `api.php` será accesible como **/api/usuarios**.

Uso común:

- **Definición de rutas RESTful:**
 - Se utiliza para asignar métodos HTTP (GET, POST, PUT, DELETE) a controladores específicos, facilitando la creación de APIs.

Como podemos observar en la siguiente imagen

```
<?php
```

```
use Illuminate\Http\Request;
```

```
use Illuminate\Support\Facades\Route;
```

```
//Controlador Usuario
```

```
use App\Http\Controllers\usuarioController;
```

```
Route::get('/Usuarios', [usuarioController::class, 'indexUsuario']);
```

```
Route::get('/Usuarios/{idUsuario}', [usuarioController::class, 'show']);
```

```
Route::post('/Usuarios', [usuarioController::class, 'store']);
```

```
Route::put('/Usuarios/{idUsuario}', [usuarioController::class, 'update']);
```

```

Route::delete('/Usuarios/{idUserario}', [usuarioController::class, 'destroy']);

//Controlador Alumno
use App\Http\Controllers\alumnoController;

Route::get('/Alumnos', [alumnoController::class, 'indexAlumno']);

Route::get('/Alumnos/{matricula}', [alumnoController::class, 'show']);

Route::post('/Alumnos', [alumnoController::class, 'store']);

Route::put('/Alumnos/{matricula}', [alumnoController::class, 'update']);

Route::delete('/Alumnos/{matricula}', [alumnoController::class, 'destroy']);

// Controlador Administrador
use App\Http\Controllers\administradorController;

Route::get('/Administrador', [administradorController::class, 'indexAdministrador']);

Route::get('/Administrador/{idAdministrador}', [administradorController::class, 'show']);

Route::post('/Administrador', [administradorController::class, 'store']);

Route::put('/Administrador/{idAdministrador}', [administradorController::class, 'update']);

Route::delete('/Administrador/{idAdministrador}', [administradorController::class, 'destroy']);

// Controlador Carrera
use App\Http\Controllers\carreraController;

```

```
Route::get('/Carrera', [carreraController::class, 'indexCarrera']);

Route::get('/Carrera/{idCarrera}', [carreraController::class, 'show']);

Route::post('/Carrera', [carreraController::class, 'store']);

Route::put('/Carrera/{idCarrera}', [carreraController::class, 'update']);

Route::delete('/Carrera/{idCarrera}', [carreraController::class, 'destroy']);

// Controlador Categoria
use App\Http\Controllers\categoriaController;

Route::get('/Categoria', [categoriaController::class, 'indexCategoria']);

Route::get('/Categoria/{idCategoria}', [categoriaController::class, 'show']);

Route::post('/Categoria', [categoriaController::class, 'store']);

Route::put('/Categoria/{idCategoria}', [categoriaController::class, 'update']);

Route::delete('/Categoria/{idCategoria}', [categoriaController::class, 'destroy']);

// Controlador Encuestas
use App\Http\Controllers\encuestasController;

Route::get('/Encuestas', [encuestasController::class, 'indexEncuestas']);

Route::get('/Encuestas/{idEncuestas}', [encuestasController::class, 'show']);
```

```

Route::post('/Encuestas', [encuestasController::class, 'store']);

Route::put('/Encuestas/{idEncuestas}', [encuestasController::class, 'update']);

Route::delete('/Encuestas/{idEncuestas}', [encuestasController::class, 'destroy']);

// Controlador Generacion
use App\Http\Controllers\generacionController;

Route::get('/Generaciones', [generacionController::class, 'indexGeneraciones']);

Route::get('/Generaciones/{idGeneracion}', [generacionController::class, 'show']);

Route::post('/Generaciones', [generacionController::class, 'store']);

Route::put('/Generaciones/{idGeneracion}', [generacionController::class, 'update']);

Route::delete('/Generaciones/{idGeneracion}', [generacionController::class, 'destroy']);

// Controlador Grupo
use App\Http\Controllers\grupoController;

Route::get('/Grupos', [grupoController::class, 'indexGrupos']);

Route::get('/Grupos/{idGrupo}', [grupoController::class, 'show']);

Route::post('/Grupos', [grupoController::class, 'store']);

Route::put('/Grupos/{idGrupo}', [grupoController::class, 'update']);

Route::delete('/Grupos/{idGrupo}', [grupoController::class, 'destroy']);

```

```
// Controlador Institucion
use App\Http\Controllers\institucionController;

Route::get('/Instituciones', [institucionController::class, 'indexInstituciones']);

Route::get('/Instituciones/{idInstitucion}', [institucionController::class, 'show']);

Route::post('/Instituciones', [institucionController::class, 'store']);

Route::put('/Instituciones/{idInstitucion}', [institucionController::class, 'update']);

Route::delete('/Instituciones/{idInstitucion}', [institucionController::class, 'destroy']);

// Controlador Opciones
use App\Http\Controllers\opcionesController;

Route::get('/Opciones', [opcionesController::class, 'indexOpciones']);

Route::get('/Opciones/{idOpciones}', [opcionesController::class, 'show']);

Route::post('/Opciones', [opcionesController::class, 'store']);

Route::put('/Opciones/{idOpciones}', [opcionesController::class, 'update']);

Route::delete('/Opciones/{idOpciones}', [opcionesController::class, 'destroy']);

// Controlador Pregunta
use App\Http\Controllers\preguntaController;

Route::get('/Pregunta', [preguntaController::class, 'indexPregunta']);
```

```

Route::get('/Pregunta/{idPregunta}', [preguntaController::class, 'show']);

Route::post('/Pregunta', [preguntaController::class, 'store']);

Route::put('/Pregunta/{idPregunta}', [preguntaController::class, 'update']);

Route::delete('/Pregunta/{idPregunta}', [preguntaController::class, 'destroy']);

// Controlador Respuestas
use App\Http\Controllers\respuestasController;

Route::get('/Respuestas', [respuestasController::class, 'indexRespuestas']);

Route::get('/Respuestas/{idRespuestas}', [respuestasController::class, 'show']);

Route::post('/Respuestas', [respuestasController::class, 'store']);

Route::put('/Respuestas/{idRespuestas}', [respuestasController::class, 'update']);

Route::delete('/Respuestas/{idRespuestas}', [respuestasController::class, 'destroy']);

// Controlador Variable
use App\Http\Controllers\variableController;

Route::get('/Variable', [variableController::class, 'indexVariable']);

Route::get('/Variable/{idVariable}', [variableController::class, 'show']);

Route::post('/Variable', [variableController::class, 'store']);

```

```
Route::put('/Variable/{idVariable}', [variableController::class, 'update']);
```

```
Route::delete('/Variable/{idVariable}', [variableController::class, 'destroy']);
```

Frontend

Después de terminar de hablar del backend ahora será con el frontend comenzando con las cabeceras, siendo estos el header, donde podemos ver títulos, imágenes como logos etc, etc así como el footer, donde en otros casos podrían ser información o logos

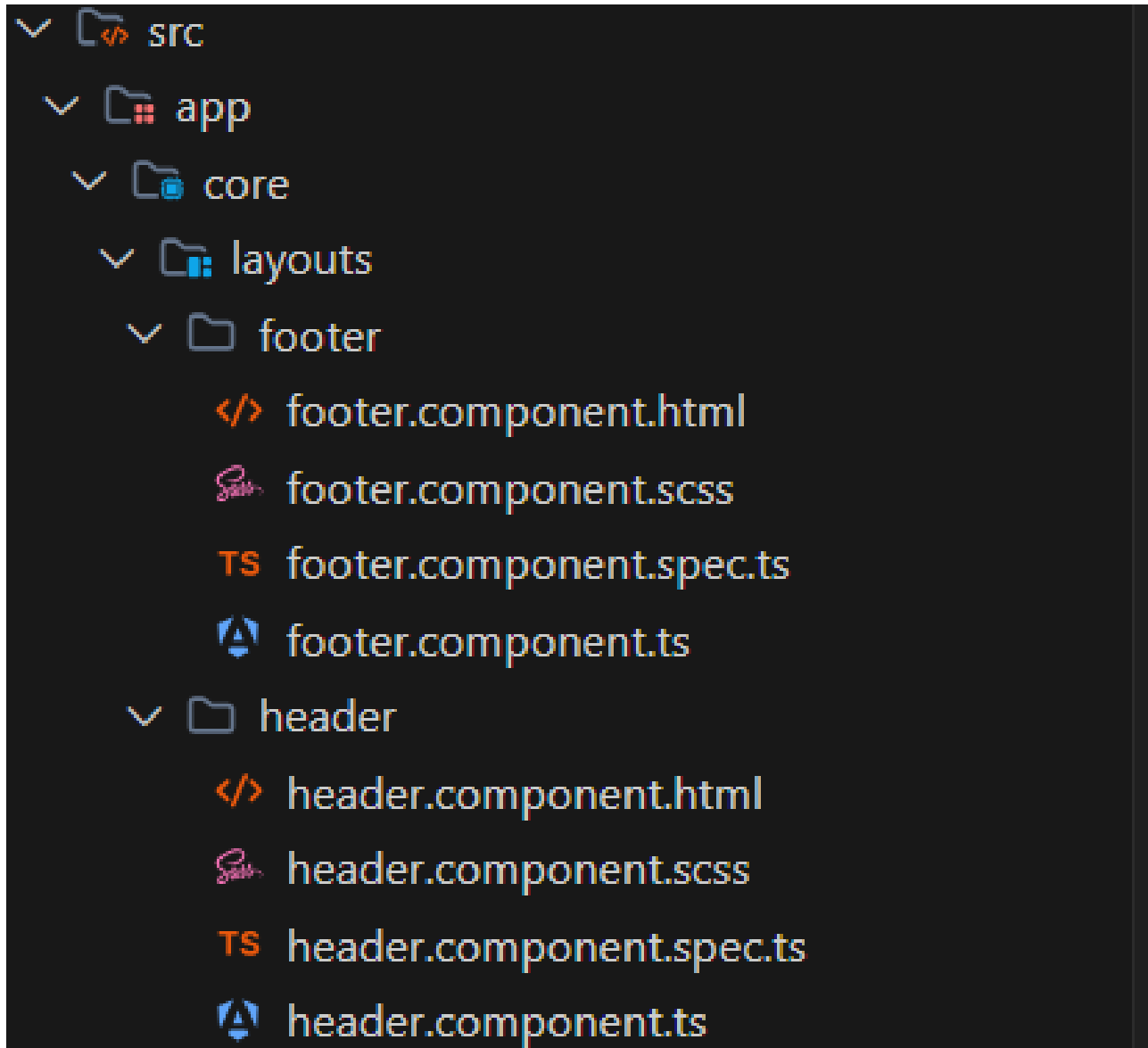


Imagen anexo 11

Header (cabecera)

Header.component.html

```
<!--El titulo de encabezado-->
<nav class="navbar navbar-expand-lg navbar-dark custom-navbar">
  <div class="container">
    <a class="navbar-brand" href="#">
      <!-- Logo -->
      

      <!-- Texto del encabezado -->
      <span class="custom-navbar-text d-inline-block align-middle">
        <h2>Cuestionario de Actitudes hacia el Servicio Comunitario</h2></span>

    </a>
  </div>
</nav>
```

Header.component.scss

```
// Importaciones de las tipografías usadas
@import
url('https://fonts.googleapis.com/css2?family=Merriweather:ital,wght@0,300;0,400;0,700;0,90
0;1,300;1,400;1,700;1,900&display=swap');
@import
url('https://fonts.googleapis.com/css2?family=Merriweather+Sans:ital,wght@0,300..800;1,300
..800&display=swap');
@import
url('https://fonts.googleapis.com/css2?family=Lato:ital,wght@0,100;0,300;0,400;0,700;0,900;
1,100;1,300;1,400;1,700;1,900&display=swap');
```

```

// Estilos globales
.body {
  background-color: #FFFFFF; /* Fondo Principal Blanco */
  color: #2C3E50; /* Texto en Azul Marino */
  font-family: 'Lato', sans-serif; /* Fuente Principal */
}

.h1, .h2, .h3, .h4, .h5, .h6 {
  font-family: 'Merriweather', serif; /* Fuente para Encabezados */
  color: #2C3E50; /* Color Azul Marino */
}

// Estilo para la parte del navbar
.custom-navbar {
  background-color: #2C3E50; /* Fondo en Azul Marino */
  padding: 15px 20px; /* Espaciado alrededor */
  display: flex;
  align-items: center;
}

.custom-navbar .navbar-brand {
  display: flex;
  align-items: center;
}

.custom-navbar .navbar-brand img {
  width: 65px; /* Tamaño del logo */
  height: 65px;
  margin-right: 15px; /* Espaciado entre logo y texto */
}

```

```
.custom-navbar-text {  
  color: #FFFFFF; /* Texto en Blanco */  
  font-family: "Merriweather Sans", sans-serif;  
  font-optical-sizing: auto;  
  font-style: normal;  
  font-size: 1.25rem; /* Tamaño de fuente del texto */  
}
```

Header.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';  
  
import { HeaderComponent } from './header.component';  
  
describe('HeaderComponent', () => {  
  let component: HeaderComponent;  
  let fixture: ComponentFixture<HeaderComponent>;  
  
  beforeEach(async () => {  
    await TestBed.configureTestingModule({  
      imports: [HeaderComponent]  
    })  
    .compileComponents();  
  
    fixture = TestBed.createComponent(HeaderComponent);  
    component = fixture.componentInstance;  
    fixture.detectChanges();  
  });  
  
  it('should create', () => {  
    expect(component).toBeTruthy();  
  });  
});
```

```
});
```

Header.component.ts

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'app-header',  
  standalone: true,  
  imports: [],  
  templateUrl: './header.component.html',  
  styleUrls: ['./header.component.scss']  
})  
export class HeaderComponent {  
  
}
```

Footer (pie de pagina)

Footer.component.html

```
<p>footer works!</p>
```

Footer.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
```

```
import { FooterComponent } from './footer.component';
```

```
describe('FooterComponent', () => {
```

```
  let component: FooterComponent;
```

```
  let fixture: ComponentFixture<FooterComponent>;
```

```
  beforeEach(async () => {
```

```
    await TestBed.configureTestingModule({
```

```
      imports: [FooterComponent]
```

```
    })
```

```
    .compileComponents();
```

```
    fixture = TestBed.createComponent(FooterComponent);
```

```
    component = fixture.componentInstance;
```

```
    fixture.detectChanges();
```

```
  });
```

```
  it('should create', () => {
```

```
    expect(component).toBeTruthy();
```

```
  });
```

```
});
```

Footer.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-footer',
  standalone: true,
  imports: [],
  templateUrl: './footer.component.html',
  styleUrls: ['./footer.component.scss']
})
export class FooterComponent {

}
```

Models

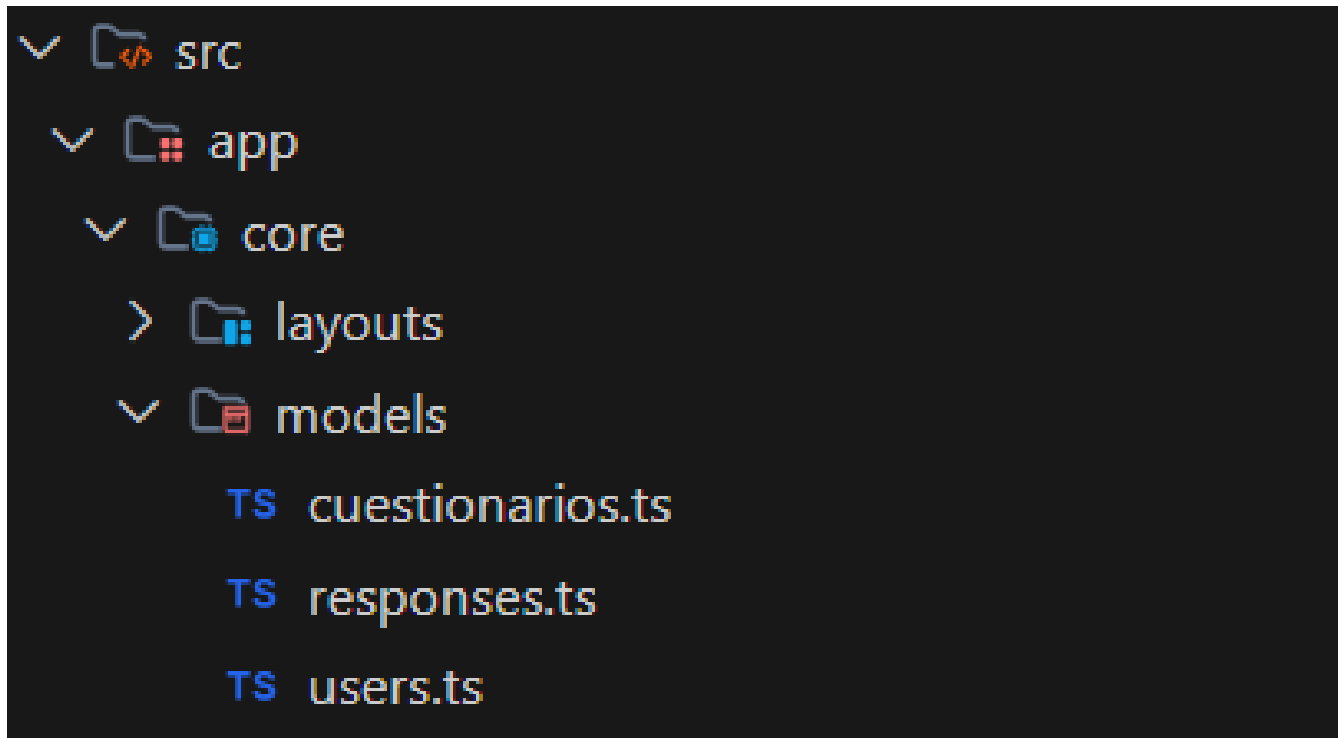


Imagen anexo 12

Responses.ts

```
export interface ResponseGet<T> {  
  data: T;      // Contiene los datos devueltos por el backend  
  status: number; // Código de estado de la respuesta (200, 404, etc.)  
  message: string; // Mensaje asociado a la respuesta  
}  
  
export interface ResponsePPD {  
  status: number; // Código de estado de la respuesta  
  message: string; // Mensaje asociado a la operación (éxito, error, etc.)  
}
```

```
export interface ErrorExcep {  
  status: number; // Código de estado de error (400, 500, etc.)  
  message: string; // Mensaje descriptivo del error  
}
```

Users.ts

```
export interface CreateUserBase {  
  idUsuario: number; // Clave primaria  
  nombres: string; // Nombres del usuario  
  primer_apellido: string; // Primer apellido  
  segundo_apellido: string; // Segundo apellido  
  correo: string; // Correo electrónico  
  password: string; // Contraseña  
  confirmPassword: string;  
  tipo: string; // Tipo de usuario ('Alumno' o 'Administrador')  
  genero: string; // Género (opcional)  
}
```

//Extensiones Específicas para Estudiantes y Administradores

```
export interface CreateStudent extends CreateUserBase {  
  edad: number;  
  matricula:string;  
  Alumno_Matricula: string; // Relación con el modelo Alumno  
  Administrador_idAdministrador?: null; // Siempre nulo para estudiantes  
}
```

```
export interface CreateAdmin extends CreateUserBase {  
  edad?:null;
```

```
matricula?:null;
Alumno_Matricula?: null; // Siempre nulo para administradores
Administrador_idAdministrador: number; // Relación con el modelo Administrador
}
```

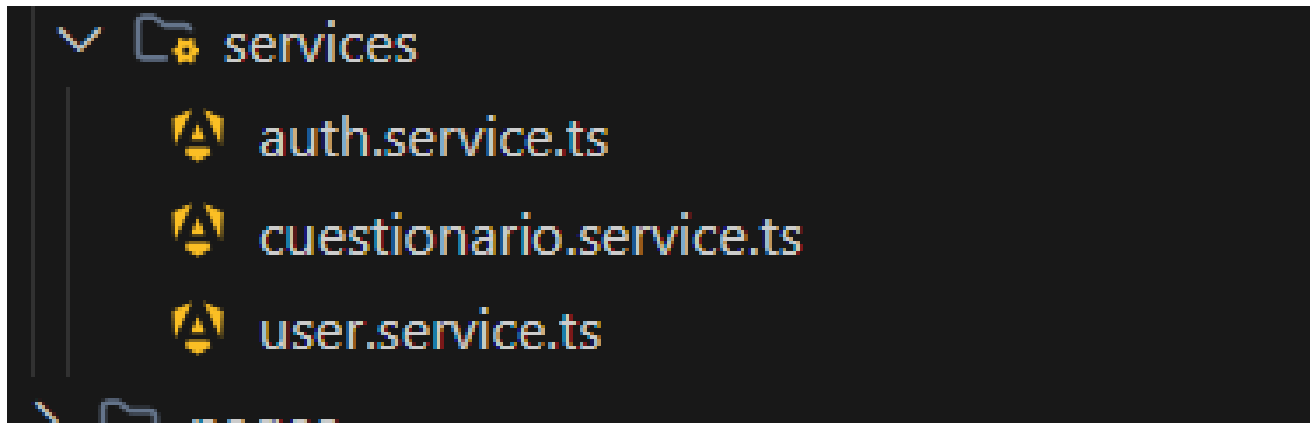
```
export interface getAdmis {
  idUsuario: number;
  nombres: string;
  primer_apellido: string;
  segundo_apellido: string;
  correo: string;
  tipo: string; // Debe ser 'Administrador'
  Administrador_idAdministrador: number;
}
```

```
export interface getStudents {
  idUsuario: number;
  matricula: string;
  nombres: string;
  primer_apellido: string;
  segundo_apellido: string;
  correo: string;
  tipo: string; // Debe ser 'Alumno'
  Alumno_Matricula: string;
}
```

```
export interface UpdateAdmi {
  idUsuario: number;
  nombres?: string;
  primer_apellido?: string;
```

```
segundo_apellido?: string;  
correo?: string;  
Administrador_idAdministrador?: number;  
}
```

Services



Auth.services.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable, of } from 'rxjs';
import { tap, catchError } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  private apiUrl = 'http://localhost:8000/api'; // URL base de tu API
  private currentUser: any = null; // Usuario autenticado

  constructor(private http: HttpClient) { }

  /**
   * Método para iniciar sesión.
   * @param credentials Credenciales del usuario (correo y contraseña).
   * @returns Observable con los datos del usuario autenticado.
   */
}
```

```

login(credentials: { correo: string; password: string }): Observable<any> {
  return this.http.post<any>(`${this.apiUrl}/login`, credentials).pipe(
    tap((user) => {
      this.setCurrentUser(user); // Almacenar el usuario autenticado
    }),
    catchError((error) => {
      console.error('Error en el inicio de sesión:', error);
      throw error;
    })
  );
}

/**
 * Establece el usuario autenticado en memoria.
 * @param user Datos del usuario autenticado.
 */
setCurrentUser(user: any): void {
  this.currentUser = user;
  localStorage.setItem('currentUser', JSON.stringify(user)); // Almacenar en localStorage
}

/**
 * Obtiene el usuario autenticado.
 * @returns Datos del usuario autenticado.
 */
getCurrentUser(): any {
  if (!this.currentUser) {
    const storedUser = localStorage.getItem('currentUser');
    if (storedUser) {
      this.currentUser = JSON.parse(storedUser);
    }
  }
}

```

```

    }
    return this.currentUser;
}

/**
 * Método para cerrar sesión.
 */
logout(): void {
    this.currentUser = null;
    localStorage.removeItem('currentUser'); // Eliminar del almacenamiento local
}

/**
 * Verifica si el usuario es administrador.
 * @returns True si el usuario es administrador.
 */
isAdmin(): boolean {
    const user = this.getCurrentUser();
    return user && user.tipo === 'Administrador';
}

/**
 * Verifica si el usuario es estudiante.
 * @returns True si el usuario es estudiante.
 */
isStudent(): boolean {
    const user = this.getCurrentUser();
    return user && user.tipo === 'Alumno';
}

/**

```

```
* Verifica si hay un usuario autenticado.  
* @returns True si hay un usuario autenticado.  
*/  
isAuthenticated(): boolean {  
    return !!this.getCurrentUser();  
}  
  
}
```

Cuestionario.service.ts

```
import { Injectable } from '@angular/core';
```

```
@Injectable({  
  providedIn: 'root'  
})
```

```
export class CuestionarioService {
```

```
  constructor() { }  
}
```

User.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { getAdmis, getStudents, UpdateAdmi } from '../models/users';
import { CreateUserBase, CreateStudent, CreateAdmin } from '../models/users';
import { ResponseGet, ResponsePPD } from '../models/responses';
```

```
@Injectable({
```

```
  providedIn: 'root',
```

```
})
```

```
export class UserService {
```

```
  // URL base de la API
```

```
  private apiUrl = 'http://localhost:8000/api';
```

```
  constructor(private http: HttpClient) {}
```

```
  /**
```

```
   * Registrar un estudiante
```

```
   * @param student Datos del estudiante
```

```
   */
```

```
  registerStudent(student: CreateStudent): Observable<ResponsePPD> {
    return this.http.post<ResponsePPD>(`${this.apiUrl}/Usuarios`, student);
  }
```

```
  /**
```

```
   * Registrar un administrador
```

```
   * @param admin Datos del administrador
```

```
   */
```

```
  registerAdmin(admin: CreateAdmin): Observable<ResponsePPD> {
```

```

        return this.http.post<ResponsePPD>(`${this.apiUrl}/Usuarios`, admin);
    }

    /**
     * Obtener todos los administradores
     */
    getAdmins(): Observable<ResponseGet<CreateUserBase[]>> {
        return this.http.get<ResponseGet<CreateUserBase[]>>(
            `${this.apiUrl}/Usuarios?tipo=Administrador`
        );
    }

    /**
     * Obtener todos los estudiantes
     */
    getStudents(): Observable<ResponseGet<CreateUserBase[]>> {
        return this.http.get<ResponseGet<CreateUserBase[]>>(
            `${this.apiUrl}/Usuarios?tipo=Alumno`
        );
    }

    /**
     * Actualizar un administrador
     * @param id ID del administrador a actualizar
     * @param admin Datos a actualizar
     */
    updateAdmin(id: number, admin: UpdateAdmi): Observable<ResponsePPD> {
        return this.http.put<ResponsePPD>(`${this.apiUrl}/Usuarios/${id}`, admin);
    }

    /**

```

```
* Obtener todos los usuarios (sin filtro)
*/
getAllUsers(): Observable<ResponseGet<CreateUserBase[]>> {
    return this.http.get<ResponseGet<CreateUserBase[]>>(`${this.apiUrl}/Usuarios`);
}
}
```

Pages (paginas)

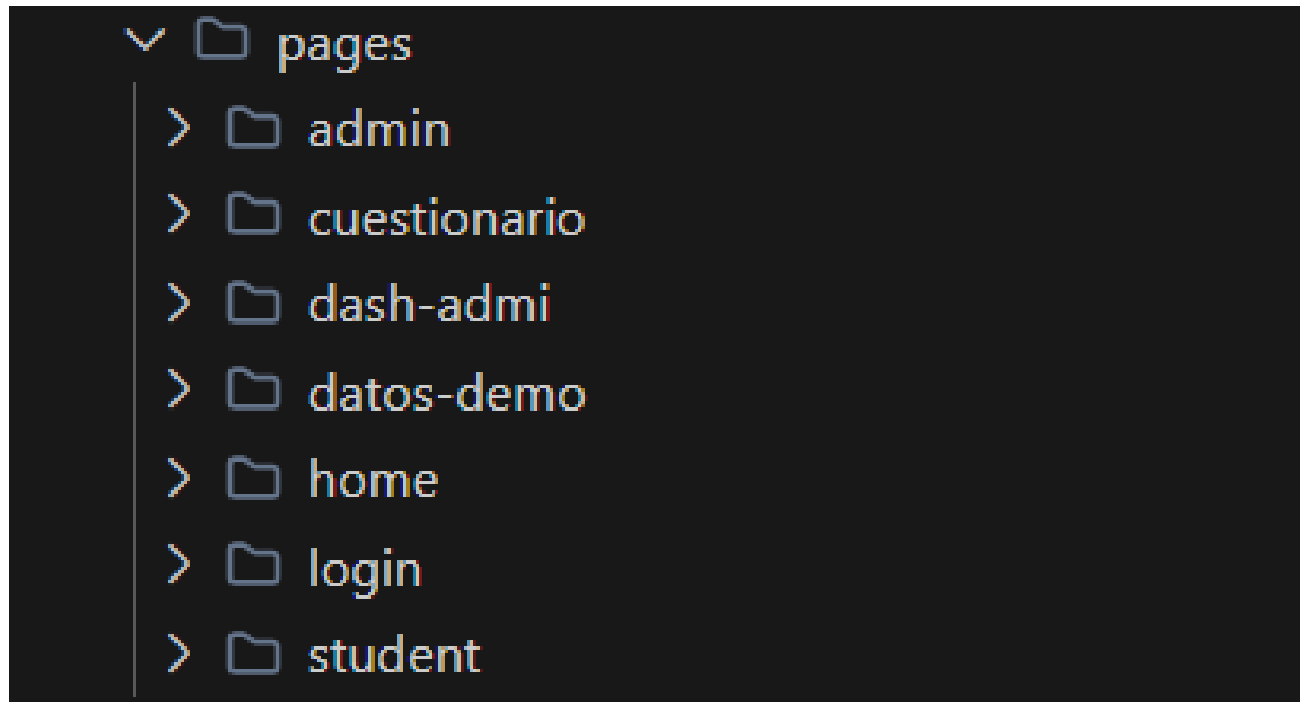


Imagen anexo 14

Admin-register

Admin-register.component.html

```
<section class="h-100 gradient-form" style="background-color: #eee;">
  <div class="container py-5 h-100">
    <div class="row d-flex justify-content-center align-items-center h-100">
      <div class="col-xl-10">
        <div class="card rounded-3 text-black">
          <div class="row g-0">
            <!-- Columna del formulario -->
            <div class="col-lg-6">
              <div class="card-body p-md-5 mx-md-4">
                <div class="text-center">
                  
                  <h4 class="mt-1 mb-5 pb-1">Registrar Administrador</h4>
                </div>

                <form (ngSubmit)="onSubmit()" #registerForm="ngForm">
                  <p>Por favor, ingresa los datos del administrador</p>

                  <!-- Nombres -->
                  <div class="form-outline mb-4">
                    <label class="form-label" for="nombres">Nombres</label>
                    <input
                      type="text"
                      id="nombres"
                      class="form-control"
                      name="nombres"
                      placeholder="Nombres"
                      [(ngModel)]="admin.nombres"
                    >
                  </div>
                </form>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
```

```

        required
        #nombresInput="ngModel"
    />
    <div *ngIf="nombresInput.invalid && nombresInput.touched" class="text-
danger">
        El nombre es obligatorio.
    </div>
</div>

<!-- Primer Apellido -->
<div class="form-outline mb-4">
    <label class="form-label" for="primer_apellido">Primer Apellido</label>
    <input
        type="text"
        id="primer_apellido"
        class="form-control"
        name="primer_apellido"
        placeholder="Primer Apellido"
        [(ngModel)]="admin.primer_apellido"
        required
        #primerApellidoInput="ngModel"
    />
    <div *ngIf="primerApellidoInput.invalid && primerApellidoInput.touched"
class="text-danger">
        El primer apellido es obligatorio.
    </div>
</div>

<!-- Segundo Apellido -->
<div class="form-outline mb-4">
    <label class="form-label" for="segundo_apellido">Segundo Apellido</label>

```

```

<input
  type="text"
  id="segundo_apellido"
  class="form-control"
  name="segundo_apellido"
  placeholder="Segundo Apellido"
  [(ngModel)]="admin.segundo_apellido"
  required
  #segundoApellidoInput="ngModel"
/>
<div *ngIf="segundoApellidoInput.invalid && segundoApellidoInput.touched"
class="text-danger">
  El segundo apellido es obligatorio.
</div>
</div>

<!-- Correo -->
<div class="form-outline mb-4">
  <label class="form-label" for="correo">Correo Electrónico</label>
  <input
    type="email"
    id="correo"
    class="form-control"
    name="correo"
    placeholder="Correo"
    [(ngModel)]="admin.correo"
    required
    #correoInput="ngModel"
  />
  <div *ngIf="correoInput.invalid && correoInput.touched" class="text-danger">
    Ingresa un correo válido.
  </div>
</div>

```

```

    </div>
</div>

<!-- Contraseña -->
<div class="form-outline mb-4">
  <label class="form-label" for="password">Contraseña</label>
  <input
    type="password"
    id="password"
    class="form-control"
    name="password"
    placeholder="Contraseña"
    [(ngModel)]="admin.password"
    required
    #passwordInput="ngModel"
  />
  <div *ngIf="passwordInput.invalid && passwordInput.touched" class="text-
danger">
    La contraseña es obligatoria.
  </div>
</div>

<!-- Confirmar Contraseña -->
<div class="form-outline mb-4">
  <label class="form-label" for="confirmarContraseña">Confirmar
Contraseña</label>
  <input
    type="password"
    id="confirmarContraseña"
    class="form-control"
    name="confirmarContraseña"

```

```

placeholder="Confirmar Contraseña"
[(ngModel)]="admin.confirmPassword"
required
#confirmPasswordInput="ngModel"
/>
<div *ngIf="confirmPasswordInput.invalid && confirmPasswordInput.touched"
class="text-danger">
    Confirma tu contraseña.
</div>
<div *ngIf="!validatePasswordsMatch() && confirmPasswordInput.touched"
class="text-danger">
    Las contraseñas no coinciden.
</div>
</div>

<!-- Tipo -->
<div class="form-outline mb-4">
    <label class="form-label" for="tipo">Tipo de Usuario</label>
    <select
        id="tipo"
        class="form-control"
        name="tipo"
        [(ngModel)]="admin.tipo"
        required
    >
        <option value="Administrador">Administrador</option>
    </select>
</div>

<!-- Género -->
<div class="form-outline mb-4">

```

```

<label class="form-label" for="genero">Género</label>
<input
  type="text"
  id="genero"
  class="form-control"
  name="genero"
  placeholder="Género"
  [(ngModel)]="admin.genero"
  required
/>
</div>

```

```

<!-- Botón de registro -->
<div class="text-center pt-1 mb-5 pb-1">
  <button
    class="btn btn-primary btn-block fa-lg gradient-custom-2 mb-3"
    type="submit"
    [disabled]="registerForm.invalid"
  >
    Registrar
  </button>
</div>
</form>

```

```

<!-- Botón para ir al Login -->
<div class="d-flex align-items-center justify-content-center pb-4">
  <p class="mb-0 me-2">¿Ya tienes cuenta?</p>
  <a routerLink="/login" class="btn btn-outline-danger">Login</a>
</div>
</div>
</div>

```

```
<!-- Columna de bienvenida -->  
<div class="col-lg-6 d-flex align-items-center gradient-custom-2">  
  <div class="text-white px-3 py-4 p-md-5 mx-md-4">  
    <h4 class="mb-4">Bienvenido</h4>  
    <p class="small mb-0">  
      Registra nuevos administradores para gestionar el sistema de encuestas de manera  
eficiente.  
    </p>  
  </div>  
</div>  
</div>  
</div>  
</div>  
</div>  
</div>
```

Admin-register.component.scss

```
.gradient-form {  
  background: linear-gradient(to right, #2c3e50, #f8fcfc);  
}  
  
.gradient-custom-2 {  
  background: linear-gradient(to right, #53c1b4, #2c3e50);  
  
  h4 {  
    font-size: 2rem; // Tamaño del título  
    font-weight: bold; // Negrita para el título  
  }  
  
  p {  
    font-size: 1.2rem; // Tamaño del párrafo  
    line-height: 1.6; // Espaciado entre líneas  
  }  
  
  @media (max-width: 768px) {  
    h4 {  
      font-size: 1.5rem; // Tamaño ajustado para pantallas pequeñas  
    }  
  
    p {  
      font-size: 1rem; // Tamaño ajustado para pantallas pequeñas  
    }  
  }  
}
```

```
.card {  
  border-radius: 1rem;  
  box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);  
}
```

```
.form-label {  
  font-weight: bold;  
}
```

```
.btn-primary {  
  background-color: #53c1b4;  
  border: none;  
}
```

```
.btn-outline-danger {  
  border: 1px solid #dc3545;  
  color: #dc3545;  
}
```

```
.btn-outline-danger:hover {  
  background-color: #dc3545;  
  color: #fff;  
}
```

Admin-register.component.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { ReactiveFormsModule, FormBuilder, FormGroup, Validators } from
'@angular/forms';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { CreateAdmin } from '../core/models/users';
import { UserService } from '../core/services/user.service';
import Swal from 'sweetalert2';
```

```
@Component({
  selector: 'app-admin-register',
  standalone: true,
  imports: [CommonModule, ReactiveFormsModule, FormsModule],
  templateUrl: './admin-register.component.html',
  styleUrls: ['./admin-register.component.scss']
})
```

```
export class AdminRegisterComponent {
  admin: CreateAdmin = {
    idUsuario: 0,
    nombres: "",
    primer_apellido: "",
    segundo_apellido: "",
    correo: "",
    password: "", // Contraseña principal
    confirmPassword: "", // Campo para confirmar la contraseña
    tipo: 'Administrador',
    Administrador_idAdministrador: 0,
    genero: "",
```

```
};
```

```
isSubmitting = false;
```

```
constructor(private userService: UserService, private router: Router) {}
```

```
// Función para validar si las contraseñas coinciden
```

```
validatePasswordsMatch(): boolean {
```

```
    return this.admin.password === this.admin.confirmPassword;
```

```
}
```

```
// Método que se ejecuta al enviar el formulario
```

```
onSubmit(): void {
```

```
    if (!this.validatePasswordsMatch()) {
```

```
        Swal.fire({
```

```
            icon: 'error',
```

```
            title: 'Error',
```

```
            text: 'Las contraseñas no coinciden. Por favor, verifica e inténtalo nuevamente.',
```

```
        });
```

```
        return;
```

```
    }
```

```
this.isSubmitting = true; // Desactivar el botón mientras se envían los datos
```

```
// Lógica para enviar datos al backend
```

```
this.userService.registerAdmin(this.admin).subscribe(
```

```
    (response) => {
```

```
        Swal.fire({
```

```
            icon: 'success',
```

```
            title: 'Registro Exitoso',
```

```
            text: 'El administrador ha sido registrado correctamente.',
```

```
}).then() => {  
  this.router.navigate(['/login']); // Redirige al login tras el registro  
});  
this.isSubmitting = false;  
},  
(error) => {  
  Swal.fire({  
    icon: 'error',  
    title: 'Error',  
    text: 'Hubo un problema al registrar al administrador. Por favor, inténtalo de nuevo.',  
  });  
  console.error('Error al registrar:', error);  
  this.isSubmitting = false;  
}  
);  
}  
  
}
```

Dash-admin.component.html

```
<div class="dashboard-wrapper" [class.sidebar-hidden]="isSidebarHidden">
  <!-- Sidebar -->
  <aside class="sidebar" [class.hidden]="isSidebarHidden">
    <div class="profile-section">
      <img class="profile-avatar" [src]="userAvatar" alt="User Avatar" />
      <h2 class="profile-name">{{ userName }}</h2>
      <p class="profile-email">{{ userEmail }}</p>
    </div>
    <nav class="menu">
      <ul>
        <li routerLink="/dashboard" routerLinkActive="active">
          <a><i class="fas fa-home"></i> Dashboard</a>
        </li>
        <li routerLink="/usuarios" routerLinkActive="active">
          <a><i class="fas fa-users"></i> Usuarios</a>
        </li>
        <li routerLink="/crear-encuestas" routerLinkActive="active">
          <a><i class="fas fa-edit"></i> Crear Encuestas</a>
        </li>
        <li routerLink="/reportes" routerLinkActive="active">
          <a><i class="fas fa-chart-line"></i> Reportes</a>
        </li>
        <li (click)="logout()">
          <a><i class="fas fa-sign-out-alt"></i> Cerrar Sesión</a>
        </li>
      </ul>
    </nav>
  </aside>
```

```

<!-- Main Content -->
<div class="main-content">
  <!-- Fondo con curva -->
<div class="wave-background"></div>

  <!-- Top Bar -->
  <header class="topbar">
    <button class="hamburger" (click)="toggleSidebar()">
      <i class="fas fa-bars"></i>
    </button>
    <a routerLink="/dashboard" class="home-link"><i class="fas fa-home"></i> Home</a>
  </header>

  <!-- Dashboard Content -->
  <section class="dashboard-content">
    <div class="stats-cards">
      <div class="card">
        <h3>Usuarios</h3>
        <p>10</p>
      </div>
      <div class="card">
        <h3>Encuestas</h3>
        <p>3</p>
      </div>
    </div>

  </section>
</div>

```

Dash-admin.component.scss

```
.dashboard-wrapper {  
  display: grid;  
  grid-template-columns: 250px 1fr; /* Sidebar fijo y contenido dinámico */  
  height: 100vh; /* Altura completa de la pantalla */  
  font-family: 'Lato', sans-serif;  
  transition: grid-template-columns 0.3s ease; /* Suaviza el ajuste del grid */  
  
  .sidebar {  
    background-color: #1d3557;  
    color: white;  
    padding: 20px;  
    transition: width 0.3s ease, padding 0.3s ease;  
  
    .profile-section {  
      display: flex; /* Activa flexbox para manejar la alineación */  
      flex-direction: column; /* Asegura que los elementos estén en columna */  
      align-items: center; /* Centra horizontalmente los elementos */  
      justify-content: center; /* Centra verticalmente los elementos */  
      text-align: center; /* Asegura que el texto esté centrado */  
      margin-bottom: 20px; /* Espaciado inferior para separar del menú */  
  
      .profile-avatar {  
        width: 80px;  
        height: 80px;  
        border-radius: 50%;  
        margin-bottom: 10px; /* Espaciado entre la imagen y el nombre */  
      }  
  
      .profile-name {
```

```

    font-size: 1.2em;
    margin: 5px 0; /* Espaciado entre el nombre y el correo */
}

.profile-email {
    font-size: 0.9em;
    color: #a8dadac;
}
}

/* Ajuste específico para los íconos */
.menu {
    ul {
        list-style: none; /* Elimina los puntos de la lista */
        padding: 0;

        li {
            margin: 15px 0;

            a {
                display: flex; /* Asegura que los íconos y el texto estén alineados */
                align-items: center; /* Centra verticalmente los elementos */
                text-decoration: none; /* Elimina subrayados */
                color: white; /* Color de texto */
                padding: 10px 15px;
                border-radius: 5px;
                transition: background-color 0.3s ease, color 0.3s ease;

                i {
                    margin-right: 10px; /* Espacio entre el ícono y el texto */

```

```

    font-size: 1.2em; /* Ajusta el tamaño del ícono */
}

&:hover {
    background-color: #457b9d; /* Color al pasar el ratón */
}
}

/* Estilo para la opción activa */
&.active a {
    background-color: #f1faee;
    color: #53c1b4;

    i {
        color: #53c1b4; /* Color del ícono en la opción activa */
    }
}
}
}
}

/* Cuando la barra lateral está oculta */
&.hidden {
    width: 0; /* La barra lateral ocupa 0 ancho */
    padding: 0; /* Elimina el padding para evitar espacio vacío */
    overflow: hidden; /* Oculta contenido sobrante */
}
}

```

```
.main-content {  
  background-color: #f8f9fa;  
  transition: all 0.3s ease; /* Suaviza la expansión del contenido */
```

```
.topbar {  
  display: flex;  
  align-items: center;  
  justify-content: space-between;  
  padding: 10px 20px;  
  background-color: white;
```

```
.hamburger {  
  font-size: 1.2em;  
  cursor: pointer;  
  background: none;  
  border: none;  
  color: #1d3557;  
}
```

```
.home-link {  
  display: flex;  
  align-items: center;  
  text-decoration: none;  
  font-size: 1em;  
  color: #1d3557;
```

```
i {  
  margin-right: 5px;  
}
```

```
&:hover {
```

```
    color: #53c1b4;
  }
}
}
```

```
.dashboard-content {
  padding: 20px;
```

```
.stats-cards {
  display: flex;
  justify-content: space-between;
  margin-bottom: 20px;
```

```
.card {
  flex: 1;
  margin: 0 10px;
  padding: 20px;
  background-color: white;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  text-align: center;
```

```
h3 {
  color: #457b9d;
}
```

```
p {
  font-size: 1.5em;
  font-weight: bold;
}
}
```

```

}

.charts {
  display: flex;

  .chart-card {
    flex: 1;
    margin: 0 10px;
    padding: 20px;
    background-color: white;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    text-align: center;
  }
}
}

/* Ajuste dinámico del grid */
&.sidebar-hidden {
  grid-template-columns: 0 1fr; /* La barra lateral desaparece, y el contenido ocupa todo el
espacio */
}
}

```

Dash-admin.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-dash-admin',
  standalone: true,
  imports: [],
  templateUrl: './dash-admin.component.html',
  styleUrls: ['./dash-admin.component.scss']
})
export class DashAdminComponent {
  isSidebarHidden = false;
  userName = "";
  userEmail = "";
  userAvatar = this.getAvatar('other'); // Cambiar a 'female' o 'other'

  toggleSidebar() {
    this.isSidebarHidden = !this.isSidebarHidden;
  }

  getAvatar(gender: string): string {
    switch (gender) {
      case 'male':
        return '../assets/avatars/male-avatar.png';
      case 'female':
        return '../assets/avatars/female-avatar.png';
      default:
        return '../assets/avatars/other-two.png';
    }
  }
}
```

```
logout() {  
  console.log('Cerrando sesión...');  
  // Lógica para cerrar sesión  
}  
  
}
```

Cuestionario

Cuestionario.component.html

```
<!-- Aquí van las preguntas del cuestionario -->
<div class="container">
  <div class="row justify-content-center">
    <div class="col-sm-12">
      <div class="container mt-4">
        <h1 class="text-title">Cuestionario de actitudes de servicio comunitario</h1>
        <hr class="custom-hr">
        <h5 class="text-title2 py-2">
          Lea cuidadosamente cada oración y seleccione la opción que mejor describa su reacción
          a cada afirmación.
        </h5>

        <form>
          <!-- Mostramos solo las preguntas de la página actual -->
          <div *ngFor="let question of currentQuestions; let i = index" class="question-
container">
            <div class="question-header">
              <p><strong>{{ question }}</strong></p>
            </div>
            <div class="options-container">
              <ng-container *ngFor="let option of options; let j = index">
                <label class="form-check-label">
                  <span>{{ option }}</span>
                  <input
                    class="form-check-input"
                    type="radio"
                    [(ngModel)]="selections[(currentPage - 1) * questionsPerPage + i]"
```

```

        [value]="j+1"
        [name]="question' + ((currentPage - 1) * questionsPerPage + i)">
    </label>
</ng-container>
</div>
</div>
</form>

<!-- Paginación visual -->
<nav aria-label="Page navigation" class="mt-4">
    <ul class="pagination justify-content-center">
        <li class="page-item" [class.disabled]="currentPage === 1">
            <a class="page-link" (click)="goToPage(currentPage - 1)">Anterior</a>
        </li>
        <li
            class="page-item"
            *ngFor="let page of [].constructor(totalPages); let i = index"
            [ngClass]="{'active-page': currentPage === i + 1}"
        >
            <a class="page-link" (click)="goToPage(i + 1)">
                {{ i + 1 }}
            </a>
        </li>
        <li class="page-item" [class.disabled]="currentPage === totalPages">
            <a class="page-link" (click)="goToPage(currentPage + 1)">Siguiete</a>
        </li>
    </ul>
</nav>
<br>
</div>

```

</div>

</div>

</div>

Cuestionario.component.scss

```
@import
url('https://fonts.googleapis.com/css2?family=Merriweather:ital,wght@0,300;0,400;0,700;0,900&display=swap');
@import
url('https://fonts.googleapis.com/css2?family=Merriweather+Sans:ital,wght@0,300..800;1,300..800&display=swap');
@import
url('https://fonts.googleapis.com/css2?family=Lato:ital,wght@0,100;0,300;0,400;0,700;0,900&display=swap');

// Configuración global de colores y fuentes
body {
  background-color: #FFFFFF;
  color: #2C3E50;
  font-family: 'Lato', sans-serif;
}

h1, h2, h3, h4, h5, h6 {
  font-family: 'Merriweather', serif;
  color: #2C3E50;
}

// Encabezados de texto
.text-title, .text-title2 {
  font-family: 'Merriweather Sans', sans-serif;
  font-optical-sizing: auto;
}

// Personalización de las opciones
.question-container {
  margin-bottom: 1.5rem;
}

.question-header {
  background-color: #2C3E50;
  color: #f8f9fa;
  padding: 0.5rem;
  border-radius: 0.25rem 0.25rem 0 0;
  font-weight: bold;
}

.options-container {
  display: flex;
  justify-content: space-between;
```

```

background-color: #1ABC9C;
padding: 1rem;
border-radius: 0 0 0.25rem 0.25rem;
}

.form-check-label {
display: flex;
flex-direction: column;
align-items: center;
text-align: center;
margin: 0 0.5rem;
}

.form-check-input {
margin-top: 0.5rem;
}

// Estilos de la paginación
.page-link {
color: #17a2b8;
background-color: white;
border: 1px solid #17a2b8;
transition: background-color 0.3s ease, color 0.3s ease;
}

.page-link:hover {
background-color: #138496;
color: white;
}

.page-link:active {
background-color: #3eaca0;
color: white;
transform: scale(0.98);
}

.page-item.disabled .page-link {
background-color: #e0e0e0;
color: #6c757d;
border-color: #e0e0e0;
}

/* Clase para resaltar la página activa */
.active-page .page-link {
background-color: #53c1b4; /* Color verde para la página activa */
color: white; /* Texto blanco para la página activa */
}

```

```
    font-weight: bold;  
}
```

```
.page-item .page-link {  
    color: black; /* Color por defecto para las páginas no activas */  
}
```

```
.page-item .page-link:hover {  
    background-color: #f0f0f0; /* Color de fondo cuando el usuario pasa el cursor por encima */  
}
```

Cuestionario.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';

import Swal from 'sweetalert2';

@Component({
  selector: 'app-cuestionario',
  standalone: true,
  imports: [ CommonModule, FormsModule, ReactiveFormsModule ],
  templateUrl: './cuestionario.component.html',
  styleUrls: ['./cuestionario.component.scss']
})
export class CuestionarioComponent {

  questions: string[] = [ /* Lista de tus preguntas */
    // Preguntas...
    '10. Los grupos comunitarios necesitan nuestra ayuda.',
    '11. Hay muchas personas en la comunidad que necesitan ayuda.',
    '12. Hay necesidades en la comunidad.',
    '13. Hay personas que tienen necesidades que no están satisfechas.',
    '14. El trabajo voluntario en agencias comunitarias ayuda a resolver problemas sociales.',
    '15. Los voluntarios en las agencias comunitarias marcan la diferencia, aunque sea una pequeña diferencia.',

    '16. Los estudiantes universitarios voluntarios pueden ayudar a mejorar la comunidad local.',
    '17. El voluntariado en proyectos comunitarios puede mejorar enormemente los recursos de la comunidad.',
    '18. Cuanta más gente ayude, mejor será la situación.',
    '19. Contribuir con mis habilidades hará de la comunidad un lugar mejor.',
    '20. Mi contribución a la comunidad marcará una diferencia real.',
    '21. Puedo marcar la diferencia en la comunidad.',

    '22. Es importante ayudar a las personas en general.',
    '23. Mejorar las comunidades es importante para mantener una sociedad de calidad.',
    '24. Nuestra comunidad necesita buenos voluntarios.',
    '25. Todas las comunidades necesitan buenos voluntarios.',
    '26. Es importante brindar un servicio útil a la comunidad a través de la comunidad.',
    '27. Cuando conozco personas que están pasando por momentos difíciles, me pregunto si estaría en su lugar.',
    '28. Me siento mal porque algunos miembros de la comunidad sufren de falta de recursos.',
```

'29. Me siento mal por la disparidad entre los miembros de la comunidad.'

'30. Si me ofreciera como voluntario, tendría menos tiempo para mis tareas escolares.'

'31. Si me ofreciera como voluntario, habría perdido la oportunidad de ganar dinero en un puesto remunerado.'

'32. Si me ofreciera como voluntario, tendría menos energía.'

'33. Si me ofreciera como voluntario, tendría menos tiempo para trabajar.'

'34. Si me ofreciera como voluntario, tendría menos tiempo libre.'

'35. Si fuera voluntario, tendría menos tiempo para pasar con mi familia.'

'36. Si me ofreciera como voluntario, estaría contribuyendo al mejoramiento de la comunidad.'

'37. Si me ofreciera como voluntario, experimentaría una satisfacción personal al saber que estoy ayudando a otros.'

'38. Si me ofreciera como voluntario, conocería a otras personas que disfrutan del servicio comunitario.'

'39. Si me ofreciera como voluntario, estaría desarrollando nuevas habilidades.'

'40. Si me ofreciera como voluntario, haría contactos valiosos para mi carrera profesional.'

'41. Si me ofreciera como voluntario, obtendría una experiencia valiosa para mi currículum.'

'42. La falta de participación en el servicio comunitario causará graves daños a nuestra sociedad.'

'43. Sin servicio comunitario, los ciudadanos desfavorecidos de hoy no tienen esperanza.'

'44. El servicio comunitario es necesario para mejorar nuestras comunidades.'

'45. Es fundamental que los ciudadanos se involucren para ayudar a sus comunidades.'

'46. El servicio comunitario es un componente crucial de la solución a los problemas comunitarios.'

'47. Participaré en un proyecto de servicio comunitario el próximo año.'

'48. Buscaré una oportunidad para hacer servicio comunitario el próximo año.'

];

```
options: string[] = [
    "Totalmente en desacuerdo",
    "En desacuerdo",
    "Algo en desacuerdo",
    "Ni estoy de acuerdo ni en desacuerdo",
    "Parcialmente de acuerdo",
    "De acuerdo",
    "Totalmente de acuerdo",
];
```

```
selections: number[] = Array(this.questions.length).fill(0); // Inicializa las selecciones a 0
```

```
currentPage: number = 1; // Página actual
```

```
questionsPerPage: number = 10; // Número de preguntas por página
```

```

// Cálculo del total de páginas
get totalPages(): number {
  return Math.ceil(this.questions.length / this.questionsPerPage);
}

// Obtiene las preguntas para la página actual
get currentQuestions(): string[] {
  const start = (this.currentPage - 1) * this.questionsPerPage;
  const end = start + this.questionsPerPage;
  return this.questions.slice(start, end);
}

// **Método que faltaba para validar las respuestas de la página actual**
validatePage(): boolean {
  const start = (this.currentPage - 1) * this.questionsPerPage;
  const end = start + this.questionsPerPage;
  // Verifica que todas las selecciones en la página actual sean diferentes de 0 (es decir,
  respondidas)
  return this.selections.slice(start, end).every(selection => selection !== 0);
}

// Cambiar de página solo si las preguntas de la página actual están respondidas
goToPage(page: number): void {
  if (page >= 1 && page <= this.totalPages) {
    if (page < this.currentPage || this.validatePage()) { // Permitir ir a la página anterior sin
    validar
      this.currentPage = page;
    } else {
      Swal.fire({
        icon: 'error',
        title: 'ERROR',
        text: 'Por favor complete todas las respuestas antes de continuar.',
        confirmButtonText: 'OK'
      });
    }
  }
}

calculateSum() {
  // Implementación de la suma
}
}

```

Datos-demo

Datos-demo.component.html

```
<!--PRIMERA PARTE-->
<!--La parte de los Datos demográficos de los usuarios -->
<div class="container py-4">
  <div class="row justify-content-center">
    <div class="col-sm-12">
      <div class="row container">
        <h1 class="demo-title">Datos Demográficos</h1>
        <hr class="custom-hr">

        <!-- Para ingresar datos o pedir los datos a los usuarios -->
        <form [formGroup]="cuestionarioForm" (ngSubmit)="onSubmit()">
          <div class="row mt-4">
            <div class="col-4">
              <label for="form-matricula" class="form-label title">1. Matrícula</label>
              <input id="form-matricula" type="text" class="form-control"
formControlName="matricula">
              <div *ngIf="matricula?.invalid && (matricula?.dirty || matricula?.touched)"
class="text-danger">
                <div *ngIf="matricula?.errors?.['required']">
                  La matrícula es obligatoria.
                </div>
                <div *ngIf="matricula?.errors?.['pattern']">
                  La matrícula debe ser un número de 4 o 6 dígitos.
                </div>
              </div>
            </div>
          </div>
        </div>

        <!-- Esta parte es para lo del genero -->
        <div class="row mt-4">
          <div class="col-4">
            <label class="form-label title">2. Género</label>
            <div class="form-check">
              <label class="form-check-label"><input class="form-check-input" type="radio"
formControlName="genero" value="masculino"> Masculino</label>
            </div>
            <div class="form-check">
              <label class="form-check-label"><input class="form-check-input" type="radio"
formControlName="genero" value="femenino"> Femenino</label>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

```

        <label class="form-check-label"><input class="form-check-input" type="radio"
formControlName="genero" value="otro"> Otro</label>
    </div>
    <div *ngIf="genero?.invalid && (genero?.dirty || genero?.touched)" class="text-
danger">
        El género es obligatorio.
    </div>
    <div *ngIf="genero?.value === 'otro'">
        <input type="text" class="form-control mt-2" formControlName="otroGenero"
placeholder="Especifique su género">
        <div *ngIf="otroGenero?.invalid && (otroGenero?.dirty || otroGenero?.touched)"
class="text-danger">
            Especifique su género es obligatorio.
        </div>
    </div>
</div>

<!-- Aquí va la parte de Edad -->
<div class="row mt-4">
    <div class="col-4">
        <label for="form-edad" class="form-label title">3. Edad</label>
        <div class="input-group">
            <input id="form-edad" type="number" class="form-control"
formControlName="edad" min="0" max="999" placeholder="Introduce tu edad">
            <span class="input-group-text">años</span>
        </div>
        <div *ngIf="edad?.invalid && (edad?.dirty || edad?.touched)" class="text-danger">
            <div *ngIf="edad?.errors?.['required']">
                La edad es obligatoria.
            </div>
            <div *ngIf="edad?.errors?.['min']">
                La edad no puede ser menor de 0.
            </div>
            <div *ngIf="edad?.errors?.['max']">
                La edad no puede ser mayor de 999.
            </div>
        </div>
    </div>
</div>

<!-- Esta es la primera pregunta de opción múltiple -->
<div class="row mt-4">
    <div class="col-4">
        <label class="form-label title">4. Lo que te describe mejor:</label>
        <div class="form-check">

```

```

        <input class="form-check-input" type="radio" id="opcion1"
formControlName="descripcion" value="Estudiante actual de secundaria">
        <label class="form-check-label" for="opcion1">Estudiante actual de
secundaria</label>
    </div>
    <div class="form-check">
        <input class="form-check-input" type="radio" id="opcion2"
formControlName="descripcion" value="Estudiante universitario actual">
        <label class="form-check-label" for="opcion2">Estudiante universitario
actual</label>
    </div>
    <div class="form-check">
        <input class="form-check-input" type="radio" id="opcion3"
formControlName="descripcion" value="Universidad graduada en el último año">
        <label class="form-check-label" for="opcion3">Universidad graduada en el último
año</label>
    </div>
    <div class="form-check">
        <input class="form-check-input" type="radio" id="opcion4"
formControlName="descripcion" value="Estudiante graduado">
        <label class="form-check-label" for="opcion4">Estudiante graduado</label>
    </div>
    <div class="form-check">
        <input class="form-check-input" type="radio" id="opcion5"
formControlName="descripcion" value="Otro">
        <label class="form-check-label" for="opcion5">Otro</label>
    </div>
    <div *ngIf="descripcion?.invalid && (descripcion?.dirty || descripcion?.touched)"
class="text-danger">
        La descripción es obligatoria.
    </div>
    <div *ngIf="descripcion?.value === 'Otro'">
        <input type="text" class="form-control mt-2" formControlName="otroDescripcion"
placeholder="Especifique su descripción">
        <div *ngIf="otroDescripcion?.invalid && (otroDescripcion?.dirty ||
otroDescripcion?.touched)" class="text-danger">
            Especificar su descripción es obligatorio.
        </div>
    </div>
</div>
</div>
</div>

<!--Aquí van las preguntas de opciones SI y NO-->
<div *ngFor="let question of questions; let i = index" class="question-container mt-4">
    <div class="question-header title">
        <p><strong>{{ question }}</strong></p>

```

```

</div>
<div class="options-container">
  <label class="form-check-label me-3" *ngFor="let option of options; let j = index">
    <input
      class="form-check-input me-2"
      type="radio"
      [formControlName]='question' + (i + 1)"
      [value]="option">
    <span>{{ option }}</span>
  </label>
</div>
</div>

<!-- Pregunta abierta -->
<div class="question-container mt-4">
  <div class="question-header title">
    <p><strong>8. Si ha trabajado como voluntario en los últimos doce meses, ¿Cuántas
horas de trabajo voluntario por semana en promedio?</strong></p>
  </div>
  <div class="options-container">
    <input
      type="text"
      class="form-control question8-input"
      formControlName="question8"
      placeholder="Escribe el número de horas">
    <div *ngIf="question8?.invalid && (question8?.dirty || question8?.touched)"
class="text-danger">
      Esta pregunta es obligatoria.
    </div>
  </div>
</div>

<!-- Segunda Pregunta de opción múltiple -->
<div class="question-container mt-4">
  <div class="question-header title">
    <p><strong>9. ¿Has participado en un Descanso Alternativo?</strong></p>
  </div>
  <div class="options-container">
    <div class="form-check">
      <input
        class="form-check-input"
        type="radio"
        id="option1"
        formControlName="participacion"
        value="Si, solo uno">
      <label class="form-check-label" for="option1">

```

```

        Si, solo uno.
    </label>
</div>

<div class="form-check">
    <input
        class="form-check-input"
        type="radio"
        id="option2"
        formControlName="participacion"
        value="Si, dos">
    <label class="form-check-label" for="option2">
        Si, dos.
    </label>
</div>

<div class="form-check">
    <input
        class="form-check-input"
        type="radio"
        id="option3"
        formControlName="participacion"
        value="Sí, tres o más">
    <label class="form-check-label" for="option3">
        Sí, tres o más.
    </label>
</div>

<div class="form-check">
    <input
        class="form-check-input"
        type="radio"
        id="option4"
        formControlName="participacion"
        value="No">
    <label class="form-check-label" for="option4">
        No
    </label>
</div>

    <div *ngIf="participacion?.invalid && (participacion?.dirty ||
participacion?.touched)" class="text-danger">
        Debe seleccionar una opción.
    </div>
</div>
</div>

```

```
<br>
<hr class="custom-hr">

<!-- La parte del boton -->
<div class="row mt-4">
  <div class="col-12 text-end">
    <!-- Botón Siguiente -->
    <button type="button" (click)="nextPage()" class="btn">Siguiente</button>
  </div>
</div>

</form>
</div>
</div>
</div>
</div>
```

Datos-demo.component.scss

```
//Importaciones de las tipografias usadas
@import
url('https://fonts.googleapis.com/css2?family=Merriweather:ital,wght@0,300;0,400;0,700;0,900;1,300;1,400;1,700;1,900&display=swap');
@import
url('https://fonts.googleapis.com/css2?family=Merriweather+Sans:ital,wght@0,300..800;1,300..800&display=swap');
@import
url('https://fonts.googleapis.com/css2?family=Lato:ital,wght@0,100;0,300;0,400;0,700;0,900;1,100;1,300;1,400;1,700;1,900&display=swap');

// Estilos globales
.body {
  background-color: #FFFFFF; /* Fondo Principal Blanco */
  color: #2C3E50; /* Texto en Azul Marino */
  font-family: 'Lato', sans-serif; /* Fuente Principal */
}

.h1, .h2, .h3, .h4, .h5, .h6 {
  font-family: 'Merriweather', serif; /* Fuente para Encabezados */
  color: #2C3E50; /* Color Azul Marino */
}

//Esta parte es el estilo del titulo
.demo-title {
  color: #2C3E50; /* Azul Marino */
  font-family: "Merriweather Sans", sans-serif;
  font-optical-sizing: auto;
  font-style: normal;
}

.custom-hr {
  border: 3px solid #7F8C8D; /* Gris Oscuro y más gruesa */
  width: 100%; /* Ajusta el ancho si es necesario */
  margin-top: 0.5rem; /* Espacio opcional entre el texto y la línea */
}

.title {
  font-weight: 800; /* Extra-bold, muy grueso */
  color: #1ABC9C; /* Cambia el color del texto, si lo deseas */
  font-family: 'Lato', sans-serif;
  font-size: 17px;
  font-optical-sizing: auto;
  font-style: normal;
}
```

```

}

/* styles.css o archivo CSS específico del componente */
.input-group-text {
  background-color: #e9ecef; /* Fondo gris claro para el texto "años" */
}

/* Ajusta el tamaño del campo de texto para la pregunta abierta */
.question8-input {
  max-width: 250px; /* Ajusta el ancho máximo como desees */
  width: 100%; /* Asegura que el campo use el 100% del contenedor */
  box-sizing: border-box; /* Incluye el padding y el border en el ancho total */
}

/* Si necesitas más ajustes de tamaño para que coincida con el de "edad" */
.form-control {
  height: auto; /* Ajusta la altura del campo de texto */
  padding: 0.375rem 0.75rem; /* Ajusta el padding si es necesario */
}

/* Ajusta el tamaño del campo de texto para la pregunta abierta */
.question8-input {
  max-width: 250px; /* Ajusta el ancho máximo como desees */
  width: 100%; /* Asegura que el campo use el 100% del contenedor */
  box-sizing: border-box; /* Incluye el padding y el border en el ancho total */
}

/* Si necesitas más ajustes de tamaño para que coincida con el de "edad" */
.form-control {
  height: auto; /* Ajusta la altura del campo de texto */
  padding: 0.375rem 0.75rem; /* Ajusta el padding si es necesario */
}

.btn {
  padding: 10px;
  font-size: 16px;
  border-radius: 8px;
  border: 2px solid #2C3E50;
  box-shadow: 0 0 6px #2C3E50;
  transition: box-shadow .4s ease, background-color .4s ease, color .4s ease;
  cursor: pointer;
  background-color: #2C3E50;
  color: #FFFFFFF;
}

&:hover {
  box-shadow: 0 0 15px #2980B9;
}

```

```
background-color: #1ABC9C;  
color: #FFFFFF;  
font-weight: bold;  
}  
}
```

Datos-demo.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ReactiveFormsModule, FormBuilder, FormGroup, Validators } from
'@angular/forms';
import { CommonModule } from '@angular/common';
import { Router } from '@angular/router'; // Importa Router

@Component({
  selector: 'app-datos-demo',
  standalone: true,
  imports: [CommonModule, ReactiveFormsModule],
  templateUrl: './datos-demo.component.html',
  styleUrls: ['./datos-demo.component.scss'] // Corrige 'styleUrl' a 'styleUrls'
})
export class DatosDemoComponent implements OnInit {
  cuestionarioForm: FormGroup;
  questions: string[] = [
    '5. ¿Es usted miembro actual o alumno de una institucion?',
    '6. ¿Es usted miembro de alguna organizacion referente al servicio?',
    '7. ¿Ha sido voluntario durante los últimos doce meses?',
  ];
  options: string[] = ['Sí', 'No'];

  currentPage: number = 0;
  itemsPerPage: number = 5;

  constructor(private fb: FormBuilder, private router: Router) { // Inyecta Router
    this.cuestionarioForm = this.fb.group({
      matricula: ['', [
        Validators.required,
        Validators.pattern(/^\d{4,6}$/)
      ]],
      genero: ['', Validators.required],
      otroGenero: [''],
      edad: ['', [
        Validators.required,
        Validators.min(0),
        Validators.max(999)
      ]],
      descripcion: ['', Validators.required],
      otroDescripcion: [''],
      question1: ['', Validators.required],
      question2: ['', Validators.required],
      question3: ['', Validators.required],
      participacion: ['', Validators.required],
    });
  }
}
```

```

        question8: ['', Validators.required]
    });
}

ngOnInit() {
    this.cuestionarioForm.get('genero')?.valueChanges.subscribe(value => {
        const otroGeneroControl = this.cuestionarioForm.get('otroGenero');
        if (value === 'otro') {
            otroGeneroControl?.setValidators(Validators.required);
        } else {
            otroGeneroControl?.clearValidators();
        }
        otroGeneroControl?.updateValueAndValidity();
    });

    this.cuestionarioForm.get('descripcion')?.valueChanges.subscribe(value => {
        const otroDescripcionControl = this.cuestionarioForm.get('otroDescripcion');
        if (value === 'Otro') {
            otroDescripcionControl?.setValidators(Validators.required);
        } else {
            otroDescripcionControl?.clearValidators();
        }
        otroDescripcionControl?.updateValueAndValidity();
    });
}

get matricula() {
    return this.cuestionarioForm.get('matricula');
}

get genero() {
    return this.cuestionarioForm.get('genero');
}

get otroGenero() {
    return this.cuestionarioForm.get('otroGenero');
}

get edad() {
    return this.cuestionarioForm.get('edad');
}

get descripcion() {
    return this.cuestionarioForm.get('descripcion');
}

```

```

get otroDescripcion() {
    return this.cuestionarioForm.get('otroDescripcion');
}

get question1() {
    return this.cuestionarioForm.get('question1');
}

get question2() {
    return this.cuestionarioForm.get('question2');
}

get question3() {
    return this.cuestionarioForm.get('question3');
}

get participacion() {
    return this.cuestionarioForm.get('participacion');
}

get question8() {
    return this.cuestionarioForm.get('question8');
}

// Método para navegar a la página siguiente
nextPage() {
    if (this.cuestionarioForm.valid) {
        this.router.navigate(['/cuestionario']); // Cambia la ruta a tu página de encuesta
    } else {
        this.cuestionarioForm.markAllAsTouched(); // Marca todos los campos como tocados
        alert('Por favor, completa todos los campos antes de continuar.');
```

```
if (this.cuestionarioForm.valid) {  
  const formData = this.cuestionarioForm.value;  
  console.log('Formulario válido:', formData);  
} else {  
  this.cuestionarioForm.markAllAsTouched();  
}  
}  
}
```

Home

Home.component.html

```
<div class="wave-background"></div> <!-- Fondo con curva -->

<div class="container d-flex justify-content-center align-items-center alto">
  <!-- Contenedor principal con texto e imagen alineados -->
  <div class="content">
    <div class="text-content">
      <h1 class="title">CSAS</h1>
      <p class="subtitle">"Sé parte del cambio. ¡Expresa tus actitudes hacia el
servicio!"</p>
      <button class="login-button">Login</button>
    </div>
    <div class="image-content">
      
    </div>
  </div>
</div>
```

Home.component.scss

```
.wave-background {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 115vh; /* Alinea la altura al 100% de la pantalla */
  background-image: url('../assets/Imagen/Wave-vector.png'); /* Ruta de la imagen */
  background-color: #f8fcfc;
  background-repeat: no-repeat;
  background-size: cover; /* Cubre todo el fondo */
  background-position: center; /* Centra la imagen */
  z-index: -1; /* Coloca el fondo detrás del contenido */
}

.container {
  position: relative;
  width: 100%;
  height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
  overflow: hidden;
  background-color: transparent;
}

.content {
  display: flex;
  align-items: center;
  justify-content: space-between;
  max-width: 80%;
  padding: 20px; /* Agrega espacio interno para separación */
  z-index: 1; /* Asegura que el contenido esté sobre el fondo */
}

.text-content {
  text-align: center;
  margin-right: 90px; /* Espacio entre el texto y la imagen */
  color: #091b1b;
}

.title {
  font-size: 3em;
  font-weight: bold;
  margin: 0;
}

.subtitle {
```

```

    font-size: 1.5em;
    color: #cc71c6;
    margin: 10px 0;
    font-weight: 600; /* Semi-bold */
}

.login-button {
    margin-top: 20px;
    padding: 10px 20px;
    background-color: #53c1b4;
    color: #091b1b;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 1em;
    font-weight: 650; /* Semi-bold */

    &:hover {
        background-color: #45a494;
    }
}

.image-content {
    max-width: 80%; /* Ajusta el ancho máximo de la imagen */
    display: flex;
    justify-content: center;
    align-items: center;

    .decorative-image {
        width: 110%; /* Hace que la imagen ocupe todo el ancho del contenedor */
        max-width: 600px; /* Tamaño máximo de la imagen */
    }
}

```

Home.component.ts

```
import { Component } from '@angular/core';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { DatosDemoComponent } from '../datos-demo/datos-demo.component';
import { CuestionarioComponent } from '../cuestionario/cuestionario.component';

import { LoginComponent } from '../login/login.component';
import { AdminComponent } from '../admin/admin.component';
import { AdminRegisterComponent } from '../admin/admin-register/admin-
register.component';
import { StudentComponent } from '../student/student.component';
import { StudentRegisterComponent } from '../student/student-register/student-
register.component';

@Component({
  selector: 'app-home',
  standalone: true,
  imports: [
    ReactiveFormsModule,
    FormsModule,
    DatosDemoComponent,
    CuestionarioComponent,
    LoginComponent,
    AdminComponent,
    AdminRegisterComponent,
    StudentComponent,
    StudentRegisterComponent
  ],

  templateUrl: './home.component.html',
  styleUrls: ['./home.component.scss']
})
export class HomeComponent {

}
```

Login

Login.component.html

```
<section class="h-100 gradient-form" style="background-color: #eee;">
  <div class="container py-5 h-100">
    <div class="row d-flex justify-content-center align-items-center h-100">
      <div class="col-xl-10">
        <div class="card rounded-3 text-black">
          <div class="row g-0">
            <!-- Columna del formulario -->
            <div class="col-lg-6">
              <div class="card-body p-md-5 mx-md-4">
                <div class="text-center">
                  
                  <h4 class="mt-1 mb-5 pb-1">Iniciar Sesión</h4>
                </div>

                <!-- Formulario -->
                <form (ngSubmit)="login()" #loginForm="ngForm">
                  <p>Por favor, ingresa tu usuario</p>

                  <!-- Campo de correo -->
                  <div class="form-outline mb-4">
                    <label class="form-label" for="correo">Correo</label>
                    <input
                      type="email"
                      id="correo"
                      class="form-control"
                      name="correo"
                      placeholder="Usuario"
                      [(ngModel)]="correo"
                      required
                      #correoInput="ngModel"
                    />
                    <div *ngIf="correoInput.invalid && correoInput.touched" class="text-danger">
                      Este campo es obligatorio.
                    </div>
                  </div>

                  <!-- Campo de contraseña -->
                  <div class="form-outline mb-4">
```

```

<label class="form-label" for="password">Contraseña</label>
<input
  type="password"
  id="password"
  class="form-control"
  name="password"
  placeholder="Contraseña"
  [(ngModel)]="password"
  required
  #passwordInput="ngModel"
/>
<div *ngIf="passwordInput.invalid && passwordInput.touched" class="text-
danger">
  Este campo es obligatorio.
</div>
</div>

```

```

<!-- Botón de login -->
<div class="text-center pt-1 mb-5 pb-1">
  <button
    class="btn btn-primary btn-block fa-lg gradient-custom-2 mb-3"
    type="submit"
    [disabled]="loginForm.invalid"
  >
    Login
  </button>
</div>

```

```

<!-- Registro -->
<div class="d-flex align-items-center justify-content-center pb-4">
  <p class="mb-0 me-2">¿No tienes cuenta?</p>
  <a routerLink="/register" class="btn btn-outline-danger">
    Crear cuenta nueva
  </a>
</div>
</form>
</div>
</div>

```

```

<!-- Columna de bienvenida -->
<div class="col-lg-6 d-flex align-items-center gradient-custom-2">
  <div class="text-white px-3 py-4 p-md-5 mx-md-4">
    <h4 class="mb-4">Encuestas CSAS</h4>
    <p class="small mb-0">
      Una plataforma para administrar encuestas y recopilar información del servicio
      social.
    </p>
  </div>
</div>

```

```

</p>
</div>
</div>
</div>
</div>
</div>
</div>
</section>

```

Login.component.scss

```
body {
  font-family: 'Nunito Sans', sans-serif;
}

.gradient-form {
  background: linear-gradient(to right, #53c1b4, #f8fcfc);
}

.gradient-custom-2 {
  background: linear-gradient(to right, #53c1b4, #2C3E50);
}

h4 {
  font-size: 2rem; // Tamaño del título
  font-weight: bold; // Negrita para el título
}

p {
  font-size: 1.2rem; // Tamaño del párrafo
  line-height: 1.6; // Espaciado entre líneas
}

@media (max-width: 768px) {
  h4 {
    font-size: 1.5rem; // Tamaño ajustado para pantallas pequeñas
  }

  p {
    font-size: 1rem; // Tamaño ajustado para pantallas pequeñas
  }
}

.card {
  border-radius: 1rem;
  box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);
}

.form-label {
  font-weight: bold;
  font-family: 'Nunito Sans', sans-serif;; // Aplica la fuente personalizada
}

.btn {
```

```
    transition: all 0.3s ease;
  }

.btn-primary {
  background-color: #53c1b4;
  border: none;
  color: white; // Color para el Texto
}

.btn-primary:hover {
  background-color: #45b2a6; // Un color más oscuro en hover
  color: white; // Cambia a blanco en hover
}

.btn-outline-danger {
  border: 1px solid #dc3545;
  color: #dc3545;
}

.btn-outline-danger:hover {
  background-color: #dc3545;
  color: #fff;
}
```

Login.component.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { ReactiveFormsModule, FormBuilder, FormGroup, Validators } from
'@angular/forms';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { AuthService } from '../core/services/auth.service';
import Swal from 'sweetalert2';
```

```
@Component({
  selector: 'app-login',
  standalone: true,
  imports: [CommonModule, ReactiveFormsModule, FormsModule],
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})
```

```
export class LoginComponent {
  correo: string = "";
  password: string = "";
```

```
  constructor(private authService: AuthService, private router: Router) {}
```

```
  // Método para manejar el inicio de sesión
```

```
  login(): void {
    if (!this.correo || !this.password) {
      Swal.fire({
        icon: 'error',
        title: 'Campos obligatorios',
        text: 'Por favor, ingresa tu correo y contraseña.',
      });
      return;
    }
  }
```

```
  const credentials = { correo: this.correo, password: this.password };
```

```
  this.authService.login(credentials).subscribe(
    (user) => {
      this.authService.setCurrentUser(user);
```

```
      Swal.fire({
        icon: 'success',
        title: 'Inicio de sesión exitoso',
        text: `Bienvenido, ${user.nombres || 'Usuario'}`,
      }).then(() => {
```

```

        this.redirectUserByRole();
    });
},
(error) => {
    Swal.fire({
        icon: 'error',
        title: 'Error al iniciar sesión',
        text: 'Correo o contraseña incorrectos. Por favor, intenta nuevamente.',
    });
    console.error('Error al iniciar sesión:', error);
}
);
}

// Redirigir al usuario según su rol
private redirectUserByRole(): void {
    const user = this.authService.getCurrentUser();

    if (!user) {
        this.router.navigate(['/login']);
        return;
    }

    if (this.authService.isAdmin()) {
        this.router.navigate(['/dash-admin']);
    } else if (this.authService.isStudent()) {
        this.router.navigate(['/dash-student']);
    } else {
        this.router.navigate(['/login']);
    }
}
}
}

```

Student

Student-register.component.html

```
<section class="h-100 gradient-form" style="background-color: #eee;">
  <div class="container py-5 h-100">
    <div class="row d-flex justify-content-center align-items-center h-100">
      <div class="col-xl-10">
        <div class="card rounded-3 text-black">
          <div class="row g-0">
            <!-- Columna del formulario -->
            <div class="col-lg-6">
              <div class="card-body p-md-5 mx-md-4">
                <div class="text-center">
                  
                  <h4 class="mt-1 mb-5 pb-1">Registrar Estudiante</h4>
                </div>

                <form (ngSubmit)="onSubmit()" #registerForm="ngForm">
                  <p>Por favor, ingresa los datos del estudiante</p>

                  <!-- Matrícula -->
                  <div class="form-outline mb-4">
                    <label class="form-label" for="matricula">Matrícula</label>
                    <input
                      type="text"
                      id="matricula"
                    >
```

```

        class="form-control"
        name="matricula"
        placeholder="Matrícula"
        [(ngModel)]="student.matricula"
        required
        #matriculaInput="ngModel"
    />
    <div *ngIf="matriculaInput.invalid && matriculaInput.touched" class="text-
danger">
        La matrícula es obligatoria.
    </div>
</div>

<!-- Nombres -->
<div class="form-outline mb-4">
    <label class="form-label" for="nombres">Nombres</label>
    <input
        type="text"
        id="nombres"
        class="form-control"
        name="nombres"
        placeholder="Nombres"
        [(ngModel)]="student.nombres"
        required
        #nombresInput="ngModel"
    />
    <div *ngIf="nombresInput.invalid && nombresInput.touched" class="text-
danger">
        El nombre es obligatorio.
    </div>
</div>

```

```

<!-- Primer Apellido -->
<div class="form-outline mb-4">
  <label class="form-label" for="primer_apellido">Primer Apellido</label>
  <input
    type="text"
    id="primer_apellido"
    class="form-control"
    name="primer_apellido"
    placeholder="Primer Apellido"
    [(ngModel)]="student.primer_apellido"
    required
    #primerApellidoInput="ngModel"
  />
  <div *ngIf="primerApellidoInput.invalid && primerApellidoInput.touched"
class="text-danger">
    El primer apellido es obligatorio.
  </div>
</div>

<!-- Segundo Apellido -->
<div class="form-outline mb-4">
  <label class="form-label" for="segundo_apellido">Segundo Apellido</label>
  <input
    type="text"
    id="segundo_apellido"
    class="form-control"
    name="segundo_apellido"
    placeholder="Segundo Apellido"
    [(ngModel)]="student.segundo_apellido"
    required

```

```

        #segundoApellidoInput="ngModel"
    />
    <div *ngIf="segundoApellidoInput.invalid && segundoApellidoInput.touched"
class="text-danger">
        El segundo apellido es obligatorio.
    </div>
</div>

```

```

<!-- Correo Electrónico -->

```

```

<div class="form-outline mb-4">
    <label class="form-label" for="correo">Correo Electrónico</label>
    <input
        type="email"
        id="correo"
        class="form-control"
        name="correo"
        placeholder="Correo"
        [(ngModel)]="student.correo"
        required
        #correoInput="ngModel"
    />
    <div *ngIf="correoInput.invalid && correoInput.touched" class="text-danger">
        Ingresa un correo válido.
    </div>
</div>

```

```

<!-- Contraseña -->

```

```

<div class="form-outline mb-4">
    <label class="form-label" for="password">Contraseña</label>
    <input
        type="password"

```

```

        id="password"
        class="form-control"
        name="password"
        placeholder="Contraseña"
        [(ngModel)]="student.password"
        required
        #passwordInput="ngModel"
    />

    <div *ngIf="passwordInput.invalid && passwordInput.touched" class="text-
danger">

        La contraseña es obligatoria.

    </div>
</div>

<!-- Confirmar Contraseña -->
<div class="form-outline mb-4">
    <label class="form-label" for="confirmPassword">Confirmar Contraseña</label>
    <input
        type="password"
        id="confirmPassword"
        class="form-control"
        name="confirmPassword"
        placeholder="Confirmar Contraseña"
        [(ngModel)]="student.confirmPassword"
        required
        #confirmPasswordInput="ngModel"
    />

    <div *ngIf="confirmPasswordInput.invalid && confirmPasswordInput.touched"
class="text-danger">

        Confirma tu contraseña.

    </div>

```

```
</div>
```

```
<!-- Tipo de Usuario -->
```

```
<div class="form-outline mb-4">
```

```
  <label class="form-label" for="tipo">Tipo de Usuario</label>
```

```
  <select
```

```
    id="tipo"
```

```
    class="form-control"
```

```
    name="tipo"
```

```
    [(ngModel)]="student.tipo"
```

```
    required
```

```
  >
```

```
    <option value="Estudiante" selected>Estudiante</option>
```

```
  </select>
```

```
</div>
```

```
<!-- Género -->
```

```
<div class="form-outline mb-4">
```

```
  <label class="form-label" for="genero">Género</label>
```

```
  <input
```

```
    type="text"
```

```
    id="genero"
```

```
    class="form-control"
```

```
    name="genero"
```

```
    placeholder="Género"
```

```
    [(ngModel)]="student.genero"
```

```
    required
```

```
  />
```

```
</div>
```

```
<!-- Edad -->
```

```

<div class="form-outline mb-4">
  <label class="form-label" for="edad">Edad</label>
  <input
    type="number"
    id="edad"
    class="form-control"
    name="edad"
    placeholder="Edad"
    [(ngModel)]="student.edad"
    required
  />
</div>

<!-- Botón de registro -->
<div class="text-center pt-1 mb-5 pb-1">
  <button
    class="btn btn-primary btn-block fa-lg gradient-custom-2 mb-3"
    type="submit"
    [disabled]="registerForm.invalid"
  >
    Registrar
  </button>
</div>

<!-- Ir a iniciar sesión -->
<div class="d-flex align-items-center justify-content-center pb-4">
  <p class="mb-0 me-2">¿Ya tienes cuenta?</p>
  <a routerLink="/login" class="btn btn-outline-danger">Login</a>
</div>
</form>
</div>

```

```
</div>
```

```
<!-- Columna de bienvenida -->
```

```
<div class="col-lg-6 d-flex align-items-center gradient-custom-2">
```

```
<div class="text-white px-3 py-4 p-md-5 mx-md-4">
```

```
<h4 class="mb-4">Bienvenido</h4>
```

```
<p class="small mb-0">
```

Registra nuevos estudiantes para gestionar el sistema de encuestas de manera eficiente.

```
</p>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</section>
```

Student-register.component.scss

```
.gradient-form {  
  background: linear-gradient(to right, #2c3e50, #53c1b4);  
}  
  
.gradient-custom-2 {  
  background: linear-gradient(to right, #53c1b4, #2c3e50);  
  
  h4 {  
    font-size: 2rem; // Tamaño del título  
    font-weight: bold; // Negrita para el título  
  }  
  
  p {  
    font-size: 1.2rem; // Tamaño del párrafo  
    line-height: 1.6; // Espaciado entre líneas  
  }  
  
  @media (max-width: 768px) {  
    h4 {  
      font-size: 1.5rem; // Tamaño ajustado para pantallas pequeñas  
    }  
  
    p {  
      font-size: 1rem; // Tamaño ajustado para pantallas pequeñas  
    }  
  }  
}
```

```
.card {  
  border-radius: 1rem;  
  box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);  
}
```

```
.form-label {  
  font-weight: bold;  
}
```

```
.btn-primary {  
  background-color: #53c1b4;  
  border: none;  
}
```

```
.btn-outline-danger {  
  border: 1px solid #dc3545;  
  color: #dc3545;  
}
```

```
.btn-outline-danger:hover {  
  background-color: #dc3545;  
  color: #fff;  
}
```

Student-register.component.ts

```
import { Component } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { Router } from '@angular/router';
import { CreateStudent } from '../../core/models/users';
import { UserService } from '../../core/services/user.service';
import Swal from 'sweetalert2';
```

```
@Component({
  selector: 'app-student-register',
  standalone: true,
  imports: [CommonModule, ReactiveFormsModule, FormsModule],
  templateUrl: './student-register.component.html',
  styleUrls: ['./student-register.component.scss']
})
```

```
export class StudentRegisterComponent {
  student: CreateStudent = {
    idUsuario: 0,
    matricula: "",
    nombres: "",
    primer_apellido: "",
    segundo_apellido: "",
    correo: "",
    password: "",
    confirmPassword: "",
    tipo: 'Alumno',
    Alumno_Matricula: "",
    genero: "",
```

```

    edad: 0,
};

isSubmitting = false; // Estado para evitar múltiples envíos

constructor(private userService: UserService, private router: Router) {}

// Método para manejar el envío del formulario
onSubmit(): void {
    if (!this.validatePasswordsMatch()) {
        Swal.fire({
            icon: 'error',
            title: 'Error',
            text: 'Las contraseñas no coinciden. Por favor, verifica e inténtalo nuevamente.',
        });
        return;
    }

    // Convertir edad a número (si está como string)
    this.student.edad = Number(this.student.edad);

    if (isNaN(this.student.edad)) {
        Swal.fire({
            icon: 'error',
            title: 'Error',
            text: 'La edad debe ser un número válido.',
        });
        return;
    }

    this.isSubmitting = true;

```

```

this.userService.registerStudent(this.student).subscribe(
  (response) => {
    Swal.fire({
      icon: 'success',
      title: 'Registro Exitoso',
      text: 'El estudiante ha sido registrado correctamente.',
    }).then(() => {
      // Redirigir al login tras el registro (opcional)
      this.router.navigate(['/login']);
    });
    this.isSubmitting = false;
  },
  (error) => {
    // Manejo de errores desde el backend
    if (error.status === 400) {
      Swal.fire({
        icon: 'error',
        title: 'Error',
        text: 'La matrícula ya está registrada. Por favor, utiliza una diferente.',
      });
    } else {
      Swal.fire({
        icon: 'error',
        title: 'Error',
        text: 'Hubo un problema al registrar al estudiante. Por favor, inténtalo de nuevo.',
      });
    }
    console.error('Error al registrar estudiante:', error);
    this.isSubmitting = false;
  }
}

```

```
);  
}  
  
// Validar que las contraseñas coincidan  
validatePasswordsMatch(): boolean {  
    return this.student.password === this.student.confirmPassword;  
}  
  
}
```

App.component.ts

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { ReactiveFormsModule } from '@angular/forms';
import { FormsModule } from '@angular/forms';
import { HomeComponent } from '../pages/home/home.component';
import { HeaderComponent } from '../core/layouts/header/header.component';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet, ReactiveFormsModule, FormsModule, HomeComponent,
HeaderComponent],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'Proyecto-CSAS';
}
```

App.routes.ts

```
import { Routes } from '@angular/router';
import { HomeComponent } from './pages/home/home.component';
import { DatosDemoComponent } from './pages/datos-demo/datos-demo.component';
import { CuestionarioComponent } from './pages/cuestionario/cuestionario.component';
import { LoginComponent } from './pages/login/login.component';
import { AdminComponent } from './pages/admin/admin.component';
import { AdminRegisterComponent } from './pages/admin/admin-register/admin-
register.component';
import { StudentComponent } from './pages/student/student.component';
import { StudentRegisterComponent } from './pages/student/student-register/student-
register.component';
import { DashAdmiComponent } from './pages/dash-admi/dash-admi.component'; // Importa el
nuevo componente
import { DashStudentComponent } from './pages/student/dash-student/dash-
student.component';

export const routes: Routes = [
  { path: '', redirectTo: 'home', pathMatch: 'full' },
  //{ path: '**', redirectTo: 'home' }, // Para manejar rutas no encontradas

  { path: 'home', component: HomeComponent },

  { path: 'login', component: LoginComponent },

  { path: 'datos', component: DatosDemoComponent },
  { path: 'cuestionario', component: CuestionarioComponent },

  { path: 'admin', component: AdminComponent },
  { path: 'register-admin', component: AdminRegisterComponent },
```

```
{ path: 'dash-admin', component: DashAdmiComponent }, // Nueva ruta para  
DashAdmiComponent
```

```
{ path: 'student', component: StudentComponent },  
{ path: 'register-student', component: StudentRegisterComponent },  
{ path: 'dash-student', component: DashStudentComponent },
```

```
];
```

Assets

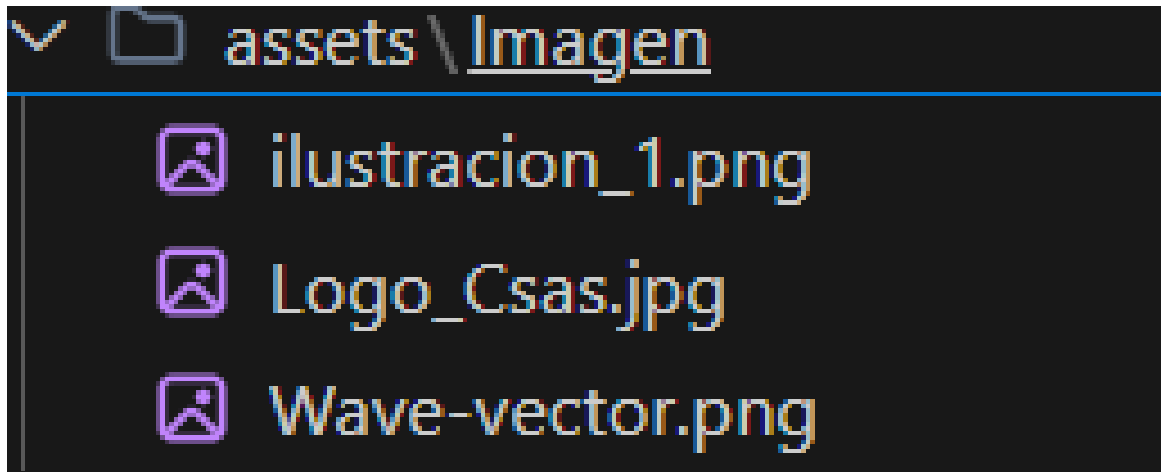


Imagen anexo 15

Esta es la carpeta en la cual se ponen las imágenes que posteriormente se llaman en las pantallas