“DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA WEB PARA LA EVALUACIÓN Y SEGUIMIENTO LONGITUDINAL DE ACTITUDES DE SERVICIO COMUNITARIO UTILIZANDO CSAS”

TESIS PARA TITULACIÓN INTEGRAL

INGENIERÍA EN SISTEMAS COMPUTACIONALES

EGRESADO:
NATIVIDAD ARRAZOLA CAN

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

Calkiní, Campeche, México 2025

Educación
Secretaría de Educación Pública

TECNOLÓGICO NACIONAL DE MÉXICO

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/03/25/651
INGENIERÍA EN SISTEMAS COMPUTACIONALES

CONSTANCIA DE NO INCONVENIENCIA

MODALIDAD: **TESIS**

C. NATIVIDAD ARRAZOLA CAN 7491
PASANTE DE LA CARRERA DE
INGENIERÍA EN SISTEMAS COMPUTACIONALES.
PRESENTE

Con relación a lo establecido en el Reglamento de Estudios de Licenciatura, capítulo VII, fracción I, vigente en este Instituto, y una vez emitido el fallo favorable por el director DRA. MARLENE MÉNDEZ MORENO, y el co-director DR. GONZALO MIGUEL QUETZ AGUIRRE, respectivamente, este Instituto consta de no tener inconveniencia para la realización del acto protocolario de titulación integral de su trabajo profesional titulado: **""DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA WEB PARA LA EVALUACIÓN Y SEGUIMIENTO LONGITUDINAL DE SERVICIO COMUNITARIO UTILIZANDO CSAS""**.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Sabiduría, Fortaleza de Nuestra Cultura®

ITESCAM
INSTITUTO TECNOLÓGICO SUPERIOR DE CALKINÍ
"DEPARTAMENTO DE ADMINISTRACIÓN ESCOLAR Y APOYO A ESTUDIANTES"
CLAVE: 04MSU0013P
CALKINÍ, CAMPECHE, MEXICO.

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

C.c.p. Archivo

2025 Año de La Mujer Indígena

Dedicatoria.

A mi madre Elania Minelia Can Zi, quien, con su amor incondicional, dedicación y palabras de aliento ha sido mi mayor ejemplo de fortaleza y ternura. Gracias por acompañarme en cada paso, por creer en mí incluso en los momentos en los que dudé, y por hacerme sentir que todo sueño es alcanzable con esfuerzo y perseverancia.

A mi padre Fidencio Arrazola Can, mi guía y mi inspiración, quien con su trabajo incansable y su sabiduría ha sido el pilar de mis metas. Gracias por tus enseñanzas, por motivarme a superar cada obstáculo y por mostrarme siempre el valor del esfuerzo y la honestidad.

A mi hermanita Mariana Arrazola Can, mi compañera de aventuras, quien con su sonrisa y alegría ha iluminado mi camino. Gracias por recordarme la importancia de compartir y disfrutar cada pequeño momento.

A ustedes, mi familia, dedico este proyecto con todo mi amor y gratitud. Sus enseñanzas, apoyo y cariño son la base de cada logro que he alcanzado.

Agradecimientos.

En primer lugar, quiero expresar mi más sincero agradecimiento a mis padres, cuyo amor, apoyo y confianza han sido el fundamento de todos mis logros. Gracias por su constante orientación, esfuerzo y sacrificios, y por impulsarme a seguir adelante en los momentos difíciles.

A mi hermana, por su compañía, sus palabras de ánimo y por ser una fuente constante de alegría y motivación en mi vida. Gracias por creer en mí y por recordarme siempre la importancia de soñar en grande.

Agradezco a mis asesores el Dr. Gonzalo Miguel Quetz Aguirre y la Dra. Marlene Méndez Moreno por guiarme en este proyecto mediante sus invaluables consejos y observaciones.

Finalmente, extiendo mi gratitud a todos los docentes que, con su esfuerzo y compromiso, contribuyeron a mi formación a lo largo de este camino académico. Sus lecciones van más allá del aula y han dejado una huella importante en mi desarrollo profesional y personal.

Índice

Índice Figuras.

Índice Tablas.

Resumen.

En el presente proyecto tiene como objetivo de diseñar e implementar una plataforma web que permita evaluar y dar seguimiento a lo largo del tiempo a las actitudes de estudiantes y egresados hacia el servicio comunitario, utilizando el Cuestionario de Actitudes hacia el Servicio Comunitario (CSAS). La plataforma automatiza la recolección, almacenamiento y análisis de datos, ofreciendo una solución más eficiente y precisa que los métodos tradicionales.

La necesidad de herramientas tecnológicas que permitan una evaluación continua y detallada motiva este proyecto, superando las limitaciones de los métodos tradicionales. Además de automatizar el proceso, la plataforma proporciona visualizaciones gráficas intuitivas para identificar cambios en las actitudes a lo largo del tiempo.

La metodología incluye el desarrollo del sistema con tecnologías como Angular, Laravel y MySQL. Los administradores pueden gestionar encuestas y analizar resultados a través de herramientas gráficas dinámicas. Las pruebas realizadas validaron su funcionalidad, reduciendo errores en la captura de datos y mejorando la interpretación de resultados.

Esta solución tecnológica representa un avance en la evaluación educativa, permitiendo una toma de decisiones basada en datos reales y un seguimiento más detallado de las actitudes hacia el servicio comunitario.

Capítulo 1

INTRODUCCIÓN.

El servicio comunitario es fundamental para el desarrollo integral de los estudiantes universitarios, ya que fomenta competencias sociales, empatía y compromiso con la comunidad. Estas actividades, promovidas en numerosos programas educativos, buscan tanto el aprendizaje práctico como el fortalecimiento de valores cívicos. Sin embargo, evaluar y realizar un seguimiento de las actitudes de los estudiantes hacia el servicio comunitario a lo largo del tiempo continúa siendo un desafío debido a las limitaciones de los métodos tradicionales de recolección de datos.

El Cuestionario de Actitudes hacia el Servicio Comunitario (CSAS), basado en una escala Likert, es una herramienta clave para medir estas actitudes. Sin embargo, su implementación mediante encuestas en papel o formularios electrónicos no integrados dificulta el análisis longitudinal eficiente, generando problemas como errores en la captura de datos y un procesamiento manual que consume tiempo.

Este proyecto propone el diseño e implementación de una plataforma web que automatice la evaluación y el análisis de los resultados del CSAS. La plataforma permitirá a las instituciones educativas gestionar encuestas, almacenar datos de manera segura y realizar análisis longitudinales para identificar tendencias y cambios en las actitudes de los estudiantes hacia el servicio comunitario.

El desarrollo se centrará en implementar una interfaz intuitiva y herramientas gráficas dinámicas para facilitar el análisis de datos. Inicialmente, la plataforma será probada en un grupo piloto de estudiantes y se espera que, tras esta fase, esté lista para una implementación a mayor escala. Se utilizarán tecnologías modernas como Angular, Laravel y MySQL, siguiendo una metodología ágil para garantizar funcionalidad, seguridad y facilidad de uso.

1.1. ANTECEDENTES.

La importancia del servicio social en la educación superior ha sido ampliamente reconocida por diversas instituciones que buscan optimizar su seguimiento y evaluación mediante herramientas tecnológicas. García-Ancira, Castillo-Elizondo y Salinas-Reyna (2016) realizaron un estudio descriptivo sobre el uso de la plataforma Nexus para monitorear el servicio social de los estudiantes de ingeniería en la Universidad Autónoma de Nuevo León. Los autores encontraron que la plataforma ofrecía ventajas como la flexibilidad en horarios, la evaluación continua de competencias y la tutoría personalizada, facilitando así el seguimiento de los proyectos comunitarios.

En 2020, Vargas-González y Morales-Martínez propusieron un cuestionario para la autoevaluación de experiencias en aprendizaje-servicio, permitiendo que los estudiantes reflexionaran sobre su desempeño durante el servicio social. Esta herramienta se convirtió en una valiosa fuente de información para las instituciones educativas, al proporcionar datos relevantes sobre el impacto del servicio social en el desarrollo personal y profesional de los estudiantes (Vargas-González & Morales-Martínez, 2020).

Otro estudio clave fue el de Lizcano, P. & Pérez, S. (2016), quienes abordaron el impacto de las plataformas digitales en el seguimiento de las actitudes de los estudiantes hacia el servicio comunitario. Su investigación mostró que el uso de estas herramientas no solo facilita la recolección de datos, sino que permite realizar un análisis detallado de las actitudes de los estudiantes y, en consecuencia, desarrollar estrategias más efectivas para fomentar su participación en proyectos comunitarios.

Por su parte, García, Gómez y Sánchez (2023) exploraron el uso de tecnologías avanzadas, como la inteligencia artificial, para la evaluación de competencias en entornos educativos. Proponen un sistema de recomendaciones que permite ofrecer retroalimentación personalizada a los estudiantes, ayudando a mejorar su desempeño en el servicio social y alineando sus esfuerzos con las necesidades de la comunidad.

Morillo-Flores, Vargas, Fuster-Guillén y Tamashiro-Tamashiro (2023) investigaron el impacto del aprendizaje-servicio en la formación de estudiantes universitarios. Su estudio concluye que la participación en actividades de servicio social fomenta la adquisición de

competencias actitudinales y reflexivas en los estudiantes, promoviendo un desarrollo integral. Traver-Martí y Ferrández-Berrueco (2016), en una investigación relacionada, construyeron y validaron un cuestionario de actitudes hacia la innovación educativa en la universidad, lo que subraya la relevancia de los cuestionarios como herramientas para evaluar el impacto de programas educativos.

En otro estudio, Prado, Higuera y Carvajal (2020) diseñaron y validaron un cuestionario para la autoevaluación de experiencias de aprendizaje-servicio universitario, proporcionando una herramienta estructurada para que los estudiantes reflexionen sobre su experiencia y desempeño en el servicio social. Esto evidencia la importancia de la autoevaluación en el contexto educativo, destacando cómo el cuestionario puede servir como un recurso valioso para el desarrollo personal y profesional.

Ramírez y Pizarro (2005) argumentan que una reflexión profunda sobre las experiencias prácticas, junto con los contenidos académicos, permite que el servicio social se convierta en una experiencia significativa de aprendizaje. Siguiendo esta línea, Tumino & Korniejczuk (2011) examinan cómo la acción y la reflexión intencionada en actividades de servicio social impactan en el desarrollo personal de los estudiantes. Este estudio utilizó una metodología cuasi-experimental mixta, evaluando las fases actitudinales mediante el Cuestionario de Actitudes hacia el Servicio Comunitario (CSAS) y los estadios reflexivos a través de la propuesta de Green. Los resultados mostraron diferencias significativas en los estadios de reflexión entre el grupo experimental y el grupo control, resaltando que los estudiantes con una reflexión guiada lograron cambios actitudinales más profundos. Un 53.85% del grupo experimental alcanzó el estadio reflexivo más alto, comparado con solo un 4.17% en el grupo control, lo que sugiere que la reflexión estructurada puede ser una herramienta clave para el desarrollo personal.

Por otro lado, Salamanca Molano (2016) evaluó el impacto del servicio social obligatorio en estudiantes de secundaria del Instituto Técnico Central, enfocado en el desarrollo de competencias ciudadanas. Utilizando un modelo de diferencias en diferencias, los resultados indicaron efectos negativos en algunas actitudes hacia la diversidad y en las competencias cognitivas de los estudiantes con puntajes bajos en la línea de base. No obstante, el análisis cualitativo demostró que los estudiantes valoraron positivamente la experiencia, destacando el

sentido de ayuda comunitaria. Este estudio subraya la importancia de evaluar el servicio social como una política pública educativa y su relevancia en la formación de competencias ciudadanas.

Estos estudios proporcionan una base teórica sólida para el presente proyecto, al señalar que las actividades de servicio social, cuando se acompañan de un proceso de reflexión y evaluación estructurada, pueden fomentar el desarrollo de competencias personales y ciudadanas en los estudiantes. A partir de estos antecedentes, este proyecto propone el diseño e implementación de una plataforma web que permita no solo el seguimiento longitudinal de las actitudes hacia el servicio comunitario, sino también la generación de recomendaciones personalizadas basadas en el desempeño individual de cada estudiante, con el objetivo de optimizar su formación integral.

1.2. PLANTEAMIENTO

En las instituciones educativas, el servicio comunitario juega un papel crucial en la formación integral de los estudiantes, fomentando valores como la solidaridad, el compromiso social y la empatía. Sin embargo, evaluar las actitudes hacia estas actividades y realizar un seguimiento continuo presenta diversos retos. Los métodos tradicionales utilizados para esta evaluación, en su mayoría estáticos, solo proporcionan una fotografía puntual de las actitudes de los estudiantes, sin permitir un análisis longitudinal que identifique cambios o tendencias a lo largo del tiempo.

La falta de herramientas tecnológicas adecuadas complica la labor de los administradores académicos para medir el impacto del servicio comunitario en los estudiantes. Esto conduce a decisiones basadas en información parcial o desactualizada, limitando la capacidad de diseñar estrategias efectivas para mejorar estos programas. Además, los métodos tradicionales carecen de opciones avanzadas para el análisis de datos, como visualizaciones gráficas interactivas o reportes dinámicos que faciliten la interpretación de resultados.

En este contexto, la Community Service Attitudes Scale (CSAS), basada en una escala Likert, surge como una solución prometedora. Sin embargo, su implementación efectiva y su seguimiento longitudinal requieren una plataforma tecnológica que administre, evalúe y analice los resultados de manera dinámica. Actualmente, no existen sistemas accesibles y eficientes que

integren esta funcionalidad para las instituciones educativas, lo que refuerza la necesidad de desarrollar este proyecto.

1.3. JUSTIFICACIÓN

El diseño e implementación de una plataforma web para la evaluación y seguimiento longitudinal de actitudes hacia el servicio comunitario responde a la necesidad de contar con herramientas tecnológicas eficaces para el análisis continuo de estas actitudes en los estudiantes. Los métodos tradicionales, como encuestas en papel o formularios electrónicos no integrados, presentan limitaciones importantes: no permiten realizar un seguimiento longitudinal preciso y dificultan el análisis detallado de los cambios en las actitudes a lo largo del tiempo, lo que reduce la efectividad de los programas educativos.

La plataforma propuesta automatizará la recolección de datos mediante el Cuestionario de Actitudes hacia el Servicio Comunitario (CSAS), almacenando los resultados de manera segura y generando visualizaciones gráficas que facilitarán el análisis. Al ofrecer un sistema centralizado y automatizado, esta solución reducirá los errores humanos, mejorará la precisión de los datos y aumentará la eficiencia en la gestión de los programas de servicio comunitario.

Los principales beneficiarios serán las instituciones educativas, que podrán gestionar las evaluaciones de forma más eficiente y obtener información más detallada y confiable para la toma de decisiones. Asimismo, los estudiantes recibirán una evaluación más precisa y objetiva, lo que contribuirá a un mejor seguimiento de su desarrollo en competencias sociales y compromiso comunitario.

Además, este proyecto tiene implicaciones futuras significativas. La plataforma podrá adaptarse para evaluar otros contextos académicos, como competencias transversales o programas extracurriculares. Gracias a su escalabilidad y el uso de tecnologías modernas como Angular, Laravel y MySQL, este sistema puede expandir su aplicación a nuevas áreas, consolidándose como una solución tecnológica integral que impulsa la innovación en la evaluación educativa.

1.4. OBJETIVOS GENERALES Y ESPECÍFICOS.

1.4.1. Objetivo general.

Desarrollar e implementar una plataforma web que permita la evaluación y el seguimiento longitudinal de las actitudes hacia el servicio comunitario en estudiantes de instituciones educativas, utilizando el Cuestionario de Actitudes hacia el Servicio Comunitario (CSAS) como herramienta de medición, con el fin de fomentar la comprensión del impacto del servicio comunitario en el desarrollo de habilidades y actitudes.

1.4.2. Objetivos específicos

- Diseñar una interfaz intuitiva y amigable que facilite la navegación y el uso tanto para los administradores como para los estudiantes participantes.
- Implementar un sistema de registro y seguimiento de los usuarios que permita el almacenamiento seguro de sus respuestas en una base de datos.
- Desarrollar un módulo de análisis de resultados, donde se presenten gráficos e informes que reflejen el cambio en las actitudes hacia el servicio comunitario.
- Configurar la plataforma para permitir la realización de múltiples encuestas en diferentes periodos, a fin de hacer un seguimiento longitudinal de las respuestas.
- Garantizar la seguridad de los datos a través de mecanismos de cifrado y resguardo en bases de datos protegidas.

1.5. ALCANCES Y LIMITACIONES.

1.5.1. Alcance del proyecto.

El presente proyecto tiene como objetivo desarrollar una plataforma web que permita la evaluación y el seguimiento longitudinal de las actitudes hacia el servicio comunitario de estudiantes y egresados de instituciones educativas.

A continuación, se detallan las características principales del proyecto:

- Interfaz amigable e intuitiva para facilitar el uso de los estudiantes y administradores.
- La plataforma permitirá la creación y administración de encuestas basadas en el Cuestionario de Actitudes hacia el Servicio Comunitario (CSAS).
- Los usuarios podrán registrar sus respuestas en diferentes momentos para permitir el seguimiento a lo largo del tiempo.

- Seguridad en el almacenamiento de datos, con la información resguardada en bases de datos protegidas.
- Acceso multiusuario, que permitirá que diferentes instituciones educativas utilicen el sistema para sus propios estudios.
- Posibilidad de agregar nuevas encuestas o modificar las existentes a medida que la plataforma crezca.
- Posibilidad de exportar los datos de respuestas para su análisis en otras aplicaciones de estadística, facilitando un mayor aprovechamiento de los resultados recopilados.

1.5.2. Limitaciones.

- La plataforma está diseñada exclusivamente para la evaluación de actitudes hacia el servicio comunitario, mediante el uso de la prueba CSAS.
- Requiere conexión a internet para que los usuarios puedan acceder y completar las encuestas.
- El sistema está diseñado para funcionar principalmente en navegadores web modernos, por lo que podría no ser compatible con navegadores obsoletos.
- No incluye análisis avanzados de datos más allá de las visualizaciones predefinidas.
- El proyecto se limitará a su uso en instituciones educativas que deseen realizar evaluaciones sobre actitudes de servicio comunitario.
- Cualquier personalización para otras instituciones o la adaptación a diferentes contextos requiere desarrollo adicional que no está contemplado en la fase inicial del proyecto.

1.6. PREGUNTA DE INVESTIGACIÓN.

Por lo tanto, con respecto al alcance y las limitaciones de la plataforma, el presente trabajo busca responder a la siguiente pregunta de investigación:

¿Cómo puede una plataforma web automatizar y mejorar la recolección y análisis de datos hacia el servicio comunitario en estudiantes y egresados, utilizando el Cuestionario de Actitudes hacia el Servicio Comunitario (CSAS)?

1.7. INFORMACIÓN DE LA EMPRESA.

Lugar donde se realizará el proyecto

- **Nombre del departamento:**

  División de Ingeniería en Sistemas Computacionales perteneciente a la Subdirección Académica de Investigación e Innovación.

  Edificio A - Planta Alta

  996-8134870 Ext. 1001

- **Estructura departamental:**

  Subdirección Académica de Investigación e Innovación

- **Croquis del departamento:**

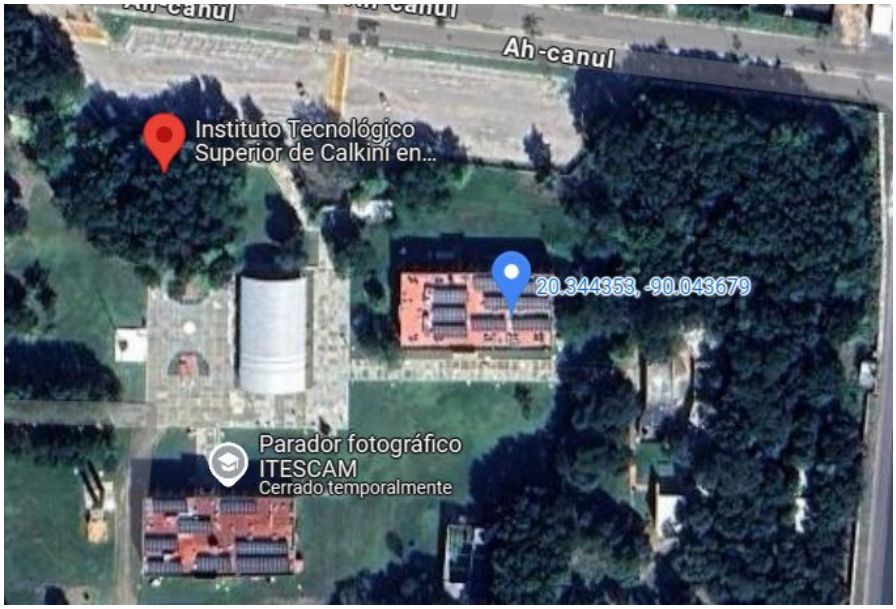


Figura 1.7.1. Croquis del departamento

Lugar de realización:

- **Nombre de la empresa y dirección.**

  Instituto Tecnológico Superior de Calkiní, en el Estado de Campeche

  Av. Ah Canul S/N por Carretera Federal.

  C.P. 24900 Calkiní, Campeche.

  Tel. 996-8134870

- **Giro.**

Educación, con un amplio sentido social y humano, al desarrollo sustentable de la región, del Estado y del país, atendiendo desde su ámbito de competencia, las necesidades de formación y actualización de profesionales competitivos, de investigación y desarrollo tecnológico y de conservación y extensión de la cultura.

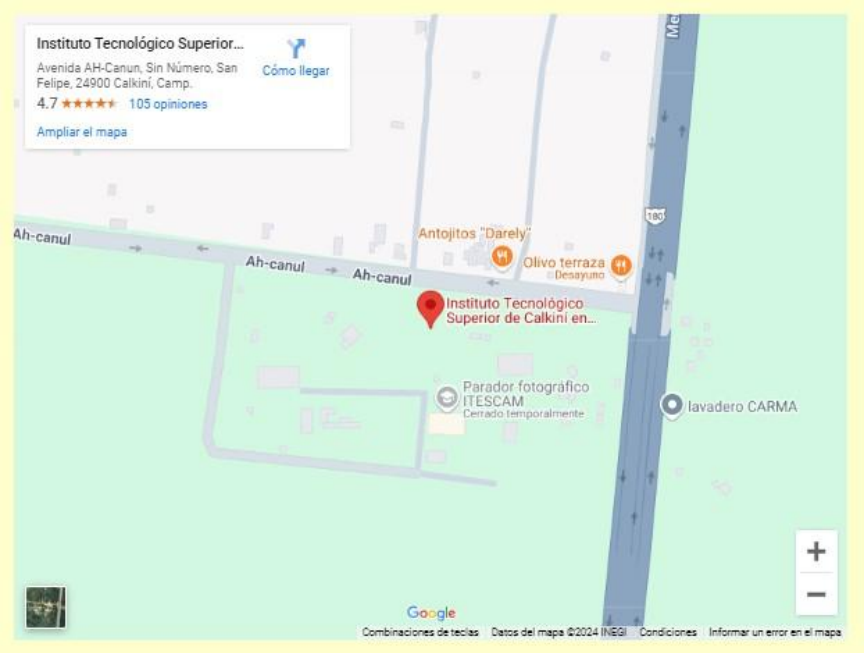


Figura 1.7.2. Croquis de ubicación

Estructura organizacional:

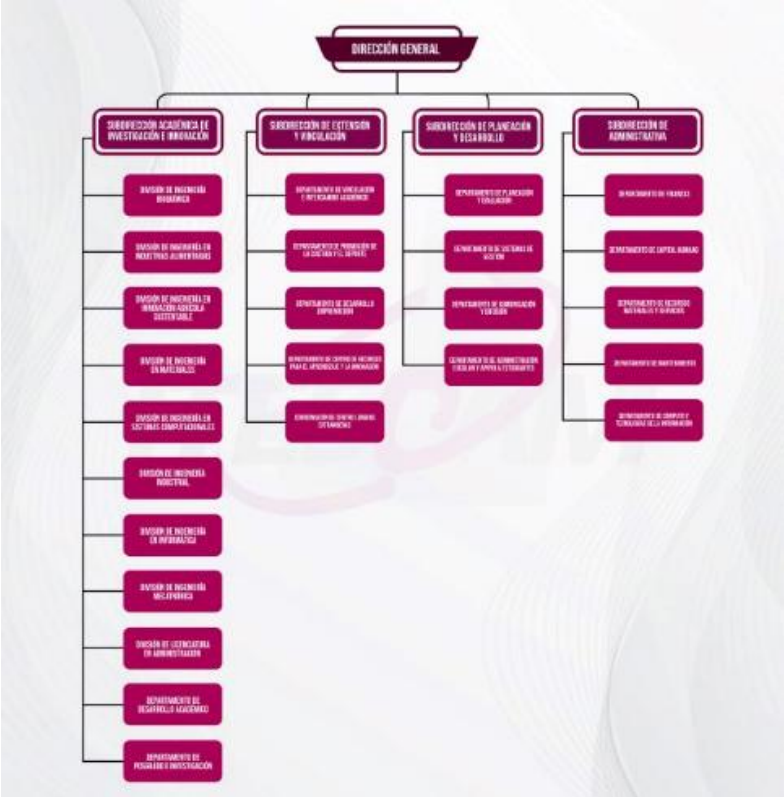


Figura 1.7.3. Estructura organizacional.

1.8. CRONOGRAMA

Tabla 1.8.1. Descripción de las actividades del cronograma.

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 la prueba 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 administración de grupos y encuestas.	15/10/2024	21/10/2024
	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

1.8.1. Descripción detallada de las actividades

En esta sección, se presenta una descripción detallada de cada una de las fases y actividades definidas en el cronograma.

Investigación inicial:

- **Definir objetivos y alcance del proyecto:** Se crearán los objetivos principales de esta plataforma, centrándose en la implementación de la prueba CSAS, así como en el seguimiento de los resultados. A continuación, se deben definir los tipos de usuarios: administradores, estudiantes, y lo que cada uno recibe.
- **Revisión de literatura y casos similares:** Se explorarán plataformas basadas en encuestas que utilizan la escala Likert para la evaluación, así como su implementación técnica. Esto servirá como un punto de referencia para la estructura del sistema y las mejores prácticas.
- **Reunión con asesores y definición de requerimientos técnicos:** Se establecerán los requerimientos funcionales y no funcionales del sistema, los cuales incluirán el diseño de la base de datos, las funcionalidades del módulo de administración, y la implementación de la encuesta CSAS.

Planificación del proyecto

- **Elaborar cronograma detallado y asignación de recursos:** Se asignar tiempo y recursos para las respectivas fases de desarrollo con el fin de crear un cronograma tangible para la implementación y prueba del sistema. Eso también incluye las responsabilidades de los desarrolladores de frontend y backend.

Diseño del frontend

- **Diseño de la interfaz de usuario:** Se crearán un esquema de la plataforma o lo que viene siendo la maquetación de diseño de la plataforma, y prototipos de la plataforma, asegurando una experiencia de usuario intuitiva tanto para los administradores como para los estudiantes. Se priorizará la claridad en la presentación de las preguntas y los resultados de las encuestas.
- **Formularios integrados CSAS implementados:** Se diseñarán los formularios interactivos que permitirán a los estudiantes completar la encuesta CSAS utilizando

una escala de Likert. Se asegurarán validaciones de datos y restricciones para evitar respuestas incompletas.

- **Visualizaciones gráficas:** Parte del trabajo consistirá en la presentación gráfica de los resultados para apoyar a los administradores en el análisis y seguimiento de actitudes a lo largo del tiempo.

Diseño del backend

- **Modelado de la base de datos:** Se definirá la estructura de la base de datos que almacenará las respuestas de las encuestas, los usuarios, y el historial de resultados. Se establecerán relaciones entre las tablas para garantizar la integridad de los datos y facilitar su análisis.
- **Arquitectura API y endpoints:** Se desarrollará una API RESTful para facilitar la comunicación entre el frontend y el backend, estableciendo endpoints para crear, actualizar y recuperar resultados y usuarios.
- **Creación del sistema de autenticación y gestión de usuarios:** Que permitirá a los usuarios acceder a la plataforma de manera segura entre administradores y estudiantes, y funciones disponibles.
- **Desarrollo de un módulo para gestionar grupos y encuestas:** Se desarrollarán las herramientas que permitirán a los administradores gestionar los grupos de estudiantes y configurar las encuestas. Esto incluirá la creación de encuestas nuevas y el seguimiento de la participación de los usuarios.

Integración

- **Integración del sistema de frontend con el backend:** El frontend o interfaz de usuario se integrará con el sistema backend. Ahora, el frontend debe comunicarse con los endpoints de la API, de modo que los datos de usuario, las respuestas de la prueba CSAS y los resultados representados gráficamente se envían y se presentan en tiempo real.

Pruebas y Validación

- **Pruebas de funcionalidad del sistema:** Se realizará la prueba e integración de todas las funcionalidades de la plataforma, desde la finalización de la encuesta hasta la visualización de los resultados.
- **Pruebas de usabilidad y retroalimentación de usuarios:** Se realizarán rondas de pruebas con los usuarios finales (es decir, administradores y estudiantes), para recopilar comentarios sobre la experiencia de uso del producto. Se evaluará la claridad de las interfaces y la funcionalidad del sistema.
- **Correcciones y ajustes:** Se identificarán y corregirán errores basados en pruebas, y se realizarán ajustes en el diseño o funcionamiento del sistema para alinearse con los objetivos del proyecto.

Documentación y entrega

- **Preparación de la documentación técnica del sistema:** Se documentarán todas las fases de desarrollo: arquitectura, APIs, estructura de base de datos y flujo de trabajo del sistema.

Capítulo 2

MARCO TEÓRICO

2.1. INTRODUCCIÓN AL MARCO TEÓRICO.

El presente capítulo establece las bases teóricas que sustentan esta investigación, proporcionando un marco de referencia basado en estudios previos y conceptos clave relacionados con la evaluación de actitudes hacia el servicio comunitario en entornos educativos. Para comprender la importancia de esta evaluación, se analiza el impacto del servicio comunitario en la formación estudiantil y su relación con el desarrollo de competencias sociales y valores cívicos. Además, se revisa el concepto de actitudes y la relevancia de su medición en el ámbito educativo. Se presenta la Escala de Actitudes hacia el Servicio Comunitario (CSAS) como una herramienta de evaluación válida y confiable.

Asimismo, se aborda el papel del análisis estadístico en la interpretación de datos y la necesidad de integrar herramientas tecnológicas que optimicen los procesos de recolección y análisis de información. Finalmente, se introduce la metodología ágil Scrum, utilizada para el desarrollo de la plataforma propuesta. El análisis de estos aspectos permitirá fundamentar el desarrollo de la plataforma propuesta y evaluar su impacto en la medición de actitudes hacia el servicio comunitario.

2.2. EL SERVICIO COMUNITARIO EN EL CONTEXTO EDUCATIVO.

El servicio comunitario es una estrategia educativa que combina la enseñanza teórica con experiencias prácticas en entornos reales. A través de estas actividades, los estudiantes desarrollan habilidades sociales y fortalecen su aprendizaje experiencial, promoviendo valores de responsabilidad y compromiso social (González & Araujo, 2016).

El servicio comunitario ha sido reconocido como un componente esencial en la educación universitaria, promoviendo valores de ciudadanía activa, solidaridad y responsabilidad social. Estas iniciativas generan un impacto positivo tanto en los estudiantes, quienes desarrollan competencias sociales y profesionales, como en las comunidades beneficiadas, fortaleciendo la integración entre la educación y la sociedad (Corredor & Romero, 2017).

El servicio comunitario no solo contribuye a la integración de la educación con la comunidad, sino que también impacta significativamente en la formación de los estudiantes, desarrollando su sentido de responsabilidad social y compromiso con el entorno. Estas iniciativas permiten generar un aprendizaje más significativo y fortalecer la relación entre la academia y la sociedad (Torres Gómez, Ramos González, & Villegas, 2023).

El seguimiento longitudinal de las actitudes de los estudiantes es clave para comprender cómo evolucionan sus valores y comportamientos a lo largo del tiempo. Esto exige herramientas tecnológicas que permitan recopilar, analizar y visualizar los datos de manera eficiente, facilitando la identificación de patrones y tendencias. Por tanto, el servicio comunitario no solo fomenta el aprendizaje experiencial, sino que también exige el desarrollo de sistemas que respalden su evaluación y optimización continua.

Sin embargo, para maximizar el impacto de estas actividades, es necesario contar con herramientas que permitan evaluar y analizar el progreso de los estudiantes en términos de actitudes y valores hacia el servicio comunitario.

2.3. ACTITUDES Y SU RELEVANCIA EN EL SERVICIO COMUNITARIO.

Las actitudes son predisposiciones aprendidas que determinan las respuestas de las personas frente a objetos, situaciones o individuos. Estas comprenden tres componentes esenciales: el afectivo (emociones), el cognitivo (creencias) y el conductual (comportamientos), constituyéndose en elementos clave del comportamiento humano (Briñol, Falces, & Becerra, 2007). En el contexto del servicio comunitario, las actitudes desempeñan un papel clave al influir en la motivación de los estudiantes y su nivel de compromiso con las actividades propuestas.

Analizar las actitudes hacia el servicio comunitario permite identificar factores que promueven o limitan la participación estudiantil en estas iniciativas. Elementos como la percepción de utilidad, las experiencias previas y los valores culturales influyen notablemente en la disposición de los estudiantes para involucrarse activamente en estas actividades (Blanco Cano & García Martín, 2021). Analizar estas dinámicas permite a las instituciones educativas diseñar

estrategias más efectivas para fomentar la participación y el compromiso en los programas de servicio comunitario.

Medir las actitudes hacia el servicio comunitario no solo proporciona información valiosa sobre la percepción de los estudiantes, sino que también permite adaptar los programas a sus necesidades y expectativas. Esto garantiza que las iniciativas sean más efectivas y enriquecedoras tanto para los estudiantes como para las comunidades beneficiadas.

Para maximizar el impacto de estas mediciones, es fundamental contar con herramientas tecnológicas avanzadas que faciliten la recolección y el análisis de datos de manera continua y precisa. La integración de tecnologías permite realizar un seguimiento longitudinal de las actitudes, identificando patrones y tendencias que informen mejor las decisiones educativas. Esto subraya la importancia de vincular la medición de actitudes con sistemas automatizados como la plataforma propuesta en este proyecto, lo que optimiza tanto la calidad como la eficiencia de los programas educativos.

2.4. HERRAMIENTAS PARA EVALUAR ACTITUDES: LA ESCALA CSAS.

La Escala de Actitudes hacia el Servicio Comunitario (CSAS) es una herramienta ampliamente utilizada en el ámbito educativo para medir actitudes relacionadas con el compromiso cívico, el interés en el servicio comunitario y el sentido de responsabilidad social de los estudiantes. Fue desarrollada inicialmente por Shiarella, Mccarthy y Tucker (2000) con el objetivo de medir las motivaciones y creencias de los estudiantes sobre el impacto del servicio comunitario en su desarrollo personal y profesional, aunque su relevancia persiste en la actualidad. Recientemente, Correa, Brussino y Reyna (2024) llevaron a cabo la construcción y validación de una escala para evaluar actitudes hacia personas de distinta clase social, lo que evidencia la continua necesidad y desarrollo de instrumentos que midan actitudes sociales en diversos contextos.

Estos esfuerzos destacan la importancia de contar con herramientas confiables y adaptables para evaluar actitudes en diferentes entornos culturales y académicos.

2.4.1. Limitaciones de la implementación tradicional de la CSAS.

El método tradicional de aplicación de la CSAS, mediante encuestas en papel o formularios electrónicos no integrados, presenta importantes limitaciones. Estos métodos requieren procesamiento manual, lo que incrementa el riesgo de errores y dificulta el análisis longitudinal de los datos (Delgado Rodríguez & Llorca Díaz, 2004). Además, el uso de encuestas en papel implica costos elevados, mayor tiempo de recolección de datos y una reducción en la precisión de los resultados debido a la posible pérdida o deterioro de los cuestionarios físicos.

Los estudios longitudinales en educación han demostrado que la recopilación de datos en múltiples momentos del tiempo es esencial para evaluar la evolución de las actitudes de los estudiantes (Pascual Arias, López-Pastor, & Hortigüela-Alcalá, 2023). Sin embargo, sin una plataforma automatizada, el manejo de estos datos puede ser complejo y propenso a errores, limitando su utilidad para la toma de decisiones en el ámbito educativo.

2.4.2. Automatización de la CSAS y su impacto en la educación.

La automatización de la CSAS a través de una plataforma tecnológica permite maximizar su utilidad al eliminar las barreras asociadas con los métodos tradicionales. Al integrar la recolección de datos, su almacenamiento seguro y el análisis dinámico, se obtienen resultados más precisos y accesibles para los educadores y administradores (Vargas Ruiz, 2003).

Beneficios clave de la automatización de la CSAS:

- **Reducción de errores:** Al minimizar la intervención manual en la recopilación de datos.
- **Mayor eficiencia en el análisis:** Facilita la identificación de patrones y tendencias en el tiempo.
- **Toma de decisiones basada en datos:** Permite que los administradores educativos adapten programas de servicio comunitario según resultados reales.
- **Accesibilidad y seguridad de la información:** Los datos se almacenan en plataformas digitales seguras y pueden ser recuperados fácilmente.

La implementación de plataformas digitales para encuestas educativas se ha convertido en una práctica estándar en estudios de actitudes y aprendizaje en educación superior. Estas

herramientas permiten generar reportes dinámicos y análisis en tiempo real, lo que facilita la retroalimentación de los resultados a estudiantes y docentes.

2.5. ANÁLISIS ESTADÍSTICO PARA LA INVESTIGACIÓN EDUCATIVA.

El análisis estadístico es un componente clave en la evaluación educativa, ya que permite interpretar los datos recopilados mediante herramientas como la Escala de Actitudes hacia el Servicio Comunitario (CSAS). Estas técnicas facilitan la identificación de patrones, tendencias y relaciones entre variables, proporcionando una base sólida para mejorar los programas educativos.

En el contexto de este proyecto, el análisis estadístico permitirá realizar un seguimiento longitudinal de las actitudes hacia el servicio comunitario, optimizando la toma de decisiones. Al identificar cambios y tendencias en los datos, los educadores y administradores podrán diseñar estrategias más efectivas y personalizadas para los estudiantes.

El análisis de datos longitudinales implica la recopilación de información en diferentes momentos temporales para identificar tendencias en la evolución de actitudes y comportamientos. Para ello, es crucial utilizar modelos de análisis que permitan comparar datos en distintos periodos, asegurando la trazabilidad de la información. Los estudios longitudinales han sido ampliamente utilizados en la investigación educativa para evaluar cambios en el tiempo, permitiendo detectar patrones en la evolución de actitudes (Arnau & Bono, 2008).

Uno de los métodos estadísticos más relevantes en este contexto es el Análisis de Varianza (ANOVA), que permite comparar diferencias significativas en la evolución de actitudes entre distintos grupos de estudiantes. Por ejemplo, Alfaro Ureña, Ortega A. y Lozano (2022) aplicaron el ANOVA para comparar el aprendizaje de estudiantes en tres modalidades educativas, demostrando su eficacia en la detección de cambios y variaciones significativas en el tiempo. En el caso de la CSAS, este método puede ser utilizado para analizar cómo evolucionan las actitudes hacia el servicio comunitario en diferentes periodos y contextos educativos, lo que permite realizar ajustes en los programas de formación.

La automatización del procesamiento de estos datos a través de una plataforma web permite una interpretación más rápida y precisa, facilitando la toma de decisiones basada en evidencia. A diferencia de los métodos manuales, la digitalización y el análisis automatizado de los datos de la

CSAS permiten aplicar modelos estadísticos como ANOVA y regresión con mayor eficiencia, reduciendo el tiempo de procesamiento y minimizando errores humanos. Esto asegura resultados más confiables y facilita la generación de informes dinámicos para los administradores educativos.

2.6. HERRAMIENTAS TECNOLÓGICAS EN LA GESTIÓN Y VISUALIZACIÓN DE DATOS.

Las herramientas tecnológicas desempeñan un papel crucial en la implementación de sistemas que automatizan y optimizan procesos clave en la evaluación educativa. En este proyecto, tecnologías como Angular, Laravel y MySQL fueron seleccionadas por su capacidad para soportar aplicaciones web escalables, dinámicas y seguras, adaptadas a las necesidades del usuario final.

Angular, como marco de desarrollo de frontend, permite la creación de interfaces de usuario dinámicas e interactivas, mejorando la experiencia del usuario. Laravel, por su parte, ofrece una estructura robusta para el backend, gestionando la lógica del servidor y la interacción con la base de datos. MySQL, un sistema de gestión de bases de datos relacionales, asegura un manejo eficiente de grandes volúmenes de información, fundamental para el análisis longitudinal de los datos recopilados.

La integración de estas tecnologías con herramientas complementarias como Bootstrap para diseño responsivo y JWT para autenticación segura garantiza un sistema eficiente y adaptable. Este conjunto tecnológico permite no solo la automatización de la recolección y análisis de datos, sino también la optimización de la experiencia del usuario, lo que contribuye directamente a los objetivos del proyecto.

La trazabilidad de datos es un factor esencial en el seguimiento longitudinal de actitudes. La implementación de plataformas digitales permite no solo almacenar información, sino también realizar análisis comparativos de respuestas a lo largo del tiempo.

Gracias a tecnologías como bases de datos relacionales y herramientas de visualización, se pueden generar informes detallados que facilitan la comprensión de los resultados y la identificación de patrones en la evolución de las actitudes de los estudiantes. Este enfoque garantiza que la evaluación del servicio comunitario no solo sea más precisa, sino que también permita a las instituciones educativas tomar decisiones basadas en datos reales y actualizados.

2.7. METODOLOGÍA SCRUM: UN ENFOQUE ÁGIL PARA EL DESARROLLO.

En general, la metodología Scrum es un marco ágil, que es ampliamente utilizado en la gestión de proyectos tecnológicos y se base en un enfoque iterativo e incremental. Scrum se centra en ciclos de trabajos cortos que son conocidos como sprints, que permiten la entrega de mejoras constantes del producto como también permitir que se produzcan cambios en los requerimientos. Este método también se basa, por ejemplo, en la comunicación continua, la autoorganización del equipo, y la mejora del sistema en su conjunto (Schwaber & Sutherland, 2020).

El desarrollo basado en Scrum permite dividir el trabajo para realizar entregas funcionales del mismo modo que se pueden ir realizando ajustes a medida que se avanza en la elaboración del proyecto, por lo que también permite tener una mayor flexibilidad y eficacia en la ejecución de las tareas. Este modelo de trabajo es uno de los más utilizados para el desarrollo del software por su capacidad para optimizar la gestión del tiempo y de los recursos, así como por reducir los riesgos debidos a la incertidumbre que presentan los proyectos de carácter tecnológico.

Capítulo 3.

METODOLOGÍA.

3.1. METODOLOGÍA DE DESARROLLO DEL SISTEMA.

3.1.1. Metodología ágil (Scrum).

Para el desarrollo de esta plataforma web, se ha seleccionado la metodología Scrum, el marco de trabajo ágil que proporciona en la gestión iterativa e incremental del desarrollo de software. Esta metodología fue seleccionada para tal efecto por su capacidad para adaptarse a cambios en los requerimientos, su orientación hacia la entrega continua de valor y como mejora de la eficiencia en lo que sería la parte de la planificación y la ejecución del proyecto.

Scrum como se basa en la planificación de sprints. Se trata de ciclos cortos de desarrollo durante las cuales se implementan y entregan funcionalidades específicas. Cada sprint tiene su duración concreta, durante la cual se llevan a cabo las tareas prioritarias; de este modo se contribuye a la optimización de los recursos y a la mejora continua del sistema mientras se está desarrollando.

A lo largo del desarrollo, se realizan también reuniones diarias de seguimiento (daily scrum) en las cuales cada uno de los miembros del equipo informa sobre su estado de avance, las restricciones y los planes a realizar. De esta forma se permite una rápida identificación de los problemas y una mejor coordinación del equipo, manteniendo los objetivos del proyecto alineados con los avances en la implementación.

3.1.1.1. Fases y procesos dentro del desarrollo de la plataforma.

La elaboración de la plataforma fue organizada siguiendo el enfoque Scrum, así dividiendo el proceso en Sprints. A continuación, se especifican las fases del desarrollo, en donde cada Sprint estaba orientado a la integración de funcionalidades para garantizar la progresiva entrega del sistema.

Tabla 3. 1. Fases del desarrollo de la plataforma.

Sprint	Objetivo	Actividades y entregables.
Sprint 1: Diseño inicial y configuración.	Establecer la base del sistema y estructura del proyecto.	• Definición de requerimientos y modelo de datos. • Configuración de entorno de desarrollo (Angular, Laravel, MySQL). • Creación de repositorio en GitHub. • Diseño preliminar de la interfaz de usuario (wireframes)
Sprint 2: Autenticación y control de acceso.	Implementar el sistema de inicio de sesión y gestión de usuarios.	• Implementación del registro y login con JWT. • Creación de roles (administrador, estudiante, evaluador). • Pruebas de autenticación y manejo de sesiones.
Sprint 3: Gestión de encuestas CSAS	Desarrollar el módulo para la creación y administración de encuestas.	• CRUD de encuestas (crear, actualizar, eliminar). • Implementación de preguntas en formato Likert. • Diseño y validación de la estructura del cuestionario.

Sprint 4: Aplicación del cuestionario y almacenamiento de repuestas.	Permitir que los estudiantes respondan encuestas y almacenar respuestas.	• Interfaz para responder encuestas. • Almacenamiento en base de datos (respuestas vinculadas a usuario y encuesta). • Validaciones para evitar respuestas vacías.
Sprint 5: Dashboard y reportes.	Implementar visualización de datos en tiempo real.	• Creación de panel de control para administradores. • Implementación de gráficos estadísticos. • Generación de reportes.
Sprint 6: Pruebas y optimización.	Realizar pruebas funcionales y optimizar la plataforma.	• Pruebas unitarias e integración. • Optimización de carga de datos. • Corrección de errores y mejoras en UI/UX.
Sprint 7: Implementación y documentación final.	Desplegar la plataforma y entregar la documentación.	• Implementación en servidor. • Pruebas finales de usuario. • Reducción de manual de usuario y documentación técnica.

3.1.1.2. **Justificación de la elección de scrum.**

Scrum fue seleccionado como la metodología de desarrollo para este proyecto debido a su capacidad para la adaptación a los cambios en los requerimientos, lo que hace mejorar la planificación en los entornos dinámicos. Debido a la flexibilidad y el enfoque iterativo de Scrum que permite una entrega continua de valor en cada fase del desarrollo, asegurando la mejora progresiva de la plataforma.

Además, uno de los principales beneficios en usar el enfoque Scrum es que nos facilita la parte de la gestión de equipos en proyectos tecnológicos y educativos, lo promueve a una comunicación constante entre los desarrolladores, evaluadores y administradores. En el contexto de cuestionario de actitudes hacia el servicio comunitario (CSAS), la metodología permite responder a necesidades específicas del sistema, optimizando los procesos de recolección y análisis de datos.

3.1.2. Roles y actividades en la plataforma.

En la metodología Scrum, cada integrante del equipo cumple un rol especifico que contribuye al desarrollo y ágil del sistema.

3.1.2.1. **Roles en Scrum.**

- **Scrum Máster:** Facilita las reuniones diarias (daily scrum), elimina obstáculos y guía al equipo para asegurar el cumplimiento de la metodología ágil.
- **Product Owner:** Define y prioriza los requerimientos del sistema, asegurando que las funcionalidades entregadas aporten valor al usuario final.
- **Development Team:** Se encargan de la implementación técnica de las funcionalidades planificadas en cada sprint, realizando pruebas y documentando el código.

A continuación, se presenta una tabla que se resume los roles y sus respectivas responsabilidades dentro del ciclo de desarrollo de la plataforma:

Tabla 3. 2. Roles en el equipo de desarrollo.

Tipo de usuarios	Descripción	Descripción
Scrum Máster	Responsable del proyecto	Dra. Marlene Méndez Moreno

Development Team	Desarrollador del proyecto	Ing. Natividad Arrazola Can

3.1.2.2. **Tipos de usuarios en la plataforma.**

En la plataforma desarrollada, los usuarios tienen diferentes roles y responsabilidades, cada uno cumpliendo funciones específicas para la gestión y análisis de los encuestadores CSAS. A continuación, se describen los principales tipos de usuarios del sistema:

La plataforma contempla tres tipos principales de usuarios, cada uno con permisos y responsabilidades específicas.

Tabla 3. 3. Tipos de usuarios del sistema.

Tipo de usuarios	Descripción
Administrativos	Gestionan la plataforma, administran usuarios, proyectos y cuestionarios, generan reportes personalizados.
Estudiantes	Responden a los cuestionarios y visualizan su progreso en el tiempo.
Evaluadores	Encargados de diseñar y configurar los cuestionarios CSAS, analizar los resultados y generar informes.

3.2. HERRAMIENTAS TECNOLGICAS.

3.2.1. Tecnologías empleadas en el desarrollo.

En esta sección se realizará la descripción breve pero detallada sobre los programas y herramientas empleados para el desarrollo del proyecto, que permitió un desarrollo eficiente, desarrollo escalable y colaboración continua a lo largo del desarrollo de la plataforma web.

Se emplearon diversas herramientas y lenguajes de programación para la creación de la plataforma, los cuales se detallan a continuación:

3.2.1.1. Angular

Es una plataforma de desarrollo y un framework de código abierto construida sobre HTML y TypeScript para crear aplicaciones web limpias y escalables. Permite la creación rápida de

aplicaciones dinámicas basadas en una sola página (SPA, por sus siglas en inglés), y proporciona una amplia gama de bibliotecas bien integradas que manejan el enrutamiento, formularios, contenido, comunicación API, y mucho más. En el proyecto, Angular fue utilizado para desarrollar la interfaz de usuario, permitiendo crear una experiencia interactiva y receptiva para los usuarios (Google, 2024).

Ventajas de usar Angular:

- **Desarrollo estructurado:** Permite usar TypeScript, lo que facilita el mantenimiento del código mientras permite a los usuarios definir su propia lógica y también sus propios componentes, llevando a la reutilización de componentes.
- **Enlace de datos:** La vista y el modelo de datos se actualizan entre si automáticamente.
- **Componentes reutilizables:** La aplicación se descompuso en pequeños módulos para facilitar el mantenimiento.
- **Compatibilidad:** Angular es multiplataforma, siendo compatible con muchos navegadores, permitiendo así que una aplicación Angular se ejecute en la mayoría de los navegadores y sistemas operativos.

3.2.1.2. Bootstrap

Un marco de aplicación ampliamente utilizado, Bootstrap fue usado como marco de diseño para asegurar que la plataforma tenga un diseño receptivo y accesible en móviles, tabletas y escritorios (Otto & Thornton, 2011).

Ventajas de Bootstrap:

- **Diseño adaptable:** Asegura que el diseño se adapte bien diferentes tamaños de pantalla.
- **Componentes predefinidos:** Haciendo que la implementación de botones, formularios y tablas sea más rápida.

3.2.1.3. Laravel

Laravel (framework) es un marco de aplicación web PHP con sintaxis expresiva y elegante. Este se utilizó para diseñar la lógica de backend de la plataforma, como la gestión de usuarios, autenticación e interacción con bases de datos (Otwell, 2011).

Ventajas de Laravel:

- **Rutas y controladores:** Las rutas se configuraron para facilitar la comunicación entre el frontend y las APIs.
- **Migrations y Eloquent ORM:** Las migraciones también permite al usuario manejar la estructura de la base de datos por si misma, mientras que Eloquent ORM ayuda al usuario a manejar las relaciones entre las tablas.
- **Middleware de autenticación:** Middleware para implementar autenticación de usuarios segura en Laravel.

3.2.1.4. MySQL

Una de las bases de datos relacionales de código abierto para almacenar, organizar y consultar datos eficazmente. Para este proyecto, se seleccionó MySQL porque puede manejar grandes cantidades de datos e integrase bien con Laravel (MySQL, 2024).

Ventajas de MySQL:

- **Gestión de tablas:** Se crearon varias tablas para almacenar la información de usuarios, grupos y cuestionarios.
- **Relaciones entre tablas:** Para asegurar la integridad entre los diferentes módulos del sistema, se establecieron relaciones con claves foráneas.

3.2.1.5. JSON Web Tokens (JWT)

JSON Web Tokens (JWT) se utilizó para la autenticación de usuarios. Este estándar proporciona transferencia segura de datos entre un cliente y un servidor, utilizando tokens firmados digitalmente (JSON Web Tokens., 2024).

Las ventajas de JWT en este proyecto incluyen:

- **Autenticación basada en tokens:** Cuando un usuario inicia sesión, el servidor genera tokens que el usuario envía con cada solicitud, por lo que no se necesita gestión de sesiones.
- **Seguridad:** Los tokens están firmados digitalmente, añadiendo seguridad extra a los datos que se transmiten.

3.2.1.6. GitHub

GitHub se utilizó como herramienta de control de versiones, facilitando la colaboración y gestión del código fuente (GitHub, Inc., 2024).

Los beneficios de usar GitHub en el proyecto:

- **Control de versiones:** Permitió llevar un registro detallado de los cambios realizados en el código.
- **Colaboración:** Las ramas facilitaron el trabajo en equipo y la integración continua entre el frontend y el backend.

3.2.2. Lenguaje de programación.

3.2.2.1. TypeScript

Angular, el framework principal utilizado para el frontend, está basado en TypeScript. TypeScript es un lenguaje de programación tipado que extiende JavaScript y permite un código más estructurado, mejorando la mantenibilidad y la detección temprana de errores (Microsoft., 2012).

Características más destacadas de TypeScript:

- **Tipado estático:** Facilita la detección de errores durante el desarrollo.
- **Compatibilidad con JavaScript:** Al ser una extensión, todo código JavaScript es válido en TypeScript.
- **Compatibilidad con herramientas modernas:** TypeScript se integra con sistemas de desarrollo como Angular y ofrece autocompletado y depuración avanzada.

3.2.2.2. JavaScript

Aunque TypeScript es el lenguaje principal, JavaScript sigue siendo esencial para las interacciones del lado del cliente. Es utilizado en el frontend principalmente para las interacciones de los componentes y la lógica de eventos (Ecma International., 1997).

Características clave de JavaScript:

- **Manipulación del DOM:** Actualiza la interfaz de usuario dinámicamente sin necesidad de recargar la página.
- **Manejo de eventos:** Facilita la interacción con el usuario mediante la respuesta a eventos en tiempo real.

3.2.2.3. HTML y CSS

Ambos son esenciales para la estructura y el diseño visual de la plataforma. HTML proporciona el esqueleto de las páginas web, mientras que CSS gestiona la presentación y el diseño. Además, Bootstrap se utiliza para crear un diseño responsive de forma rápida y sencilla (W3C., 2024).

3.2.2.4. PHP

Laravel, un framework de PHP, es el núcleo del backend en tu proyecto. Se utiliza para gestionar la lógica del servidor y la interacción con la base de datos (The PHP Group., 1995).

Características principales de PHP en Laravel:

- **Desarrollo basado en servidores:** PHP maneja las solicitudes del frontend, procesando los datos y respondiendo con la información necesaria.
- **Autenticación y autorización:** Laravel facilita la creación de sistemas seguros para la autenticación y autorización de usuarios.
- **Patrón MVC (Modelo-Vista-Controlador):** Laravel sigue este patrón para organizar el código y las funciones del sistema.

3.3. TÉCNICAS DE ANÁLISIS DE DATOS.

3.3.1. Métodos estadísticos aplicados.

El análisis de datos juega un papel crucial en la evaluación de actitudes y resultados del servicio comunitario. Existen diversas técnicas estadísticas aplicables en este ámbito:

3.3.1.1. Métodos paramétricos.

Métodos como el ANOVA y las pruebas t de Student son ideales para comparar grupos y analizar relaciones entre variables en condiciones de normalidad. Estas técnicas ofrecen una base robusta para extraer conclusiones significativas en estudios de mayor escala (Velázquez, 2023).

3.3.1.2. Métodos no paramétricos.

En casos donde los datos no cumplen con los supuestos de normalidad, los métodos no paramétricos, como la prueba de Kruskal-Wallis o Mann-Whitney, proporcionan soluciones confiables. Estos métodos son particularmente útiles para estudios con muestras pequeñas o datos categóricos (González, 2023).

3.3.1.3. Minería de datos.

La minería de datos complementa los métodos estadísticos tradicionales, permitiendo descubrir patrones complejos y segmentar grupos en función de sus actitudes. Herramientas como el análisis de componentes principales y el clustering han demostrado su eficacia en el análisis educativo (Han et al., 2012).

3.4. MODELO DE BASE DE DATOS.

3.4.1. Diseño de la base de datos.

Después del análisis de los requisitos del sistema, se diseñó el modelo relacional para la base de datos que se utilizará para almacenar toda la información del sistema. Esta base de datos servirá para gestionar los datos de los usuarios, encuestas, preguntas, respuestas, y los grupos a los que pertenecerán los usuarios.

Es importante mencionar que la base de datos está implementada en MySQL, debido a su robustez y facilidad de manejo. Asimismo, para la gestión eficiente de los datos y la interacción

con el sistema, se integrarán APIs de servicio para garantizar un acceso rápido y una administración sencilla de la información almacenada.

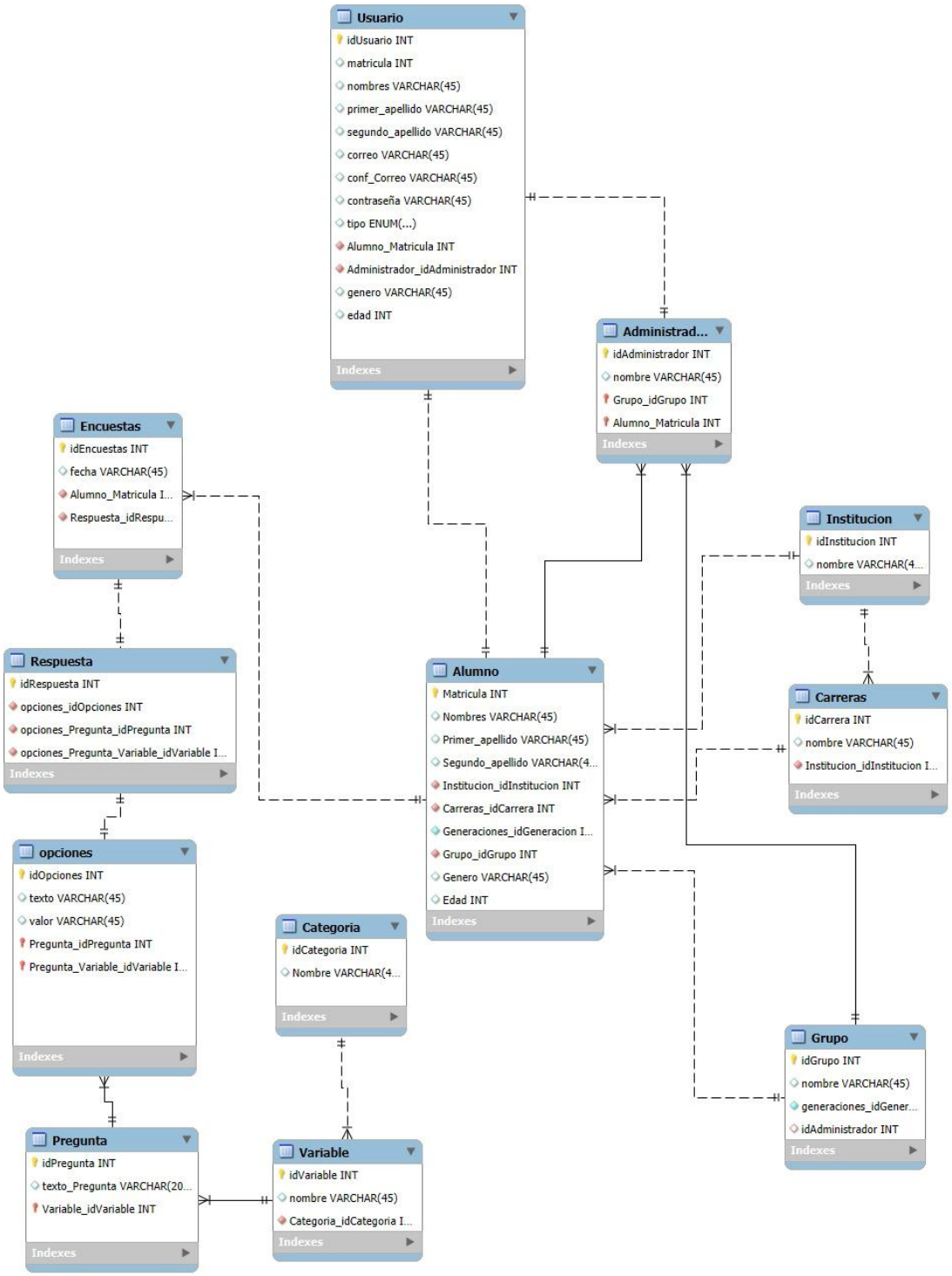


Figura 3. 1. Modelo Relacional

3.4.1.1. **Explicación del modelo relacional**

El diagrama muestra el modelo relacional de una base de datos diseñada para una plataforma de encuestas. Cada tabla representa una entidad con sus respectivos atributos y relaciones:

- **Usuario:** Contiene la información principal de los usuarios, como matrícula, nombres, apellidos, correo y contraseña. También se vincula con Alumno y Administrador para diferenciar los tipos de usuarios.
- **Alumno:** Almacena datos específicos de los estudiantes, como matrícula, nombres, apellidos, carrera, institución, generación y grupo al que pertenecen.
- **Administrador:** Guarda la información de los administradores, que están relacionados con un grupo específico.
- **Encuestas:** Registra las encuestas realizadas, vinculando cada encuesta con un alumno y las respuestas proporcionadas.
- **Respuesta:** Almacena las respuestas dadas por los usuarios, con una relación hacia las opciones seleccionadas y las preguntas correspondientes.
- **Pregunta y opciones:** Define las preguntas y sus posibles opciones de respuesta. Cada pregunta está relacionada con una variable.
- **Variable y categoría:** Las variables representan los temas de las preguntas y están categorizadas mediante la tabla Categoría.
- **Institución, carreras, Grupo:** Estas tablas representan datos complementarios que permiten clasificar a los alumnos por su institución, carrera, y grupo.

Este modelo permite gestionar encuestas, almacenar respuestas y organizar a los usuarios, especialmente alumnos y administradores, en un sistema.

3.5. CASOS DE USO Y ARQUITECTURA DEL SISTEMA.

3.5.1. Diagramas de casos de uso.

3.5.1.1. Diagrama para iniciar sesión.

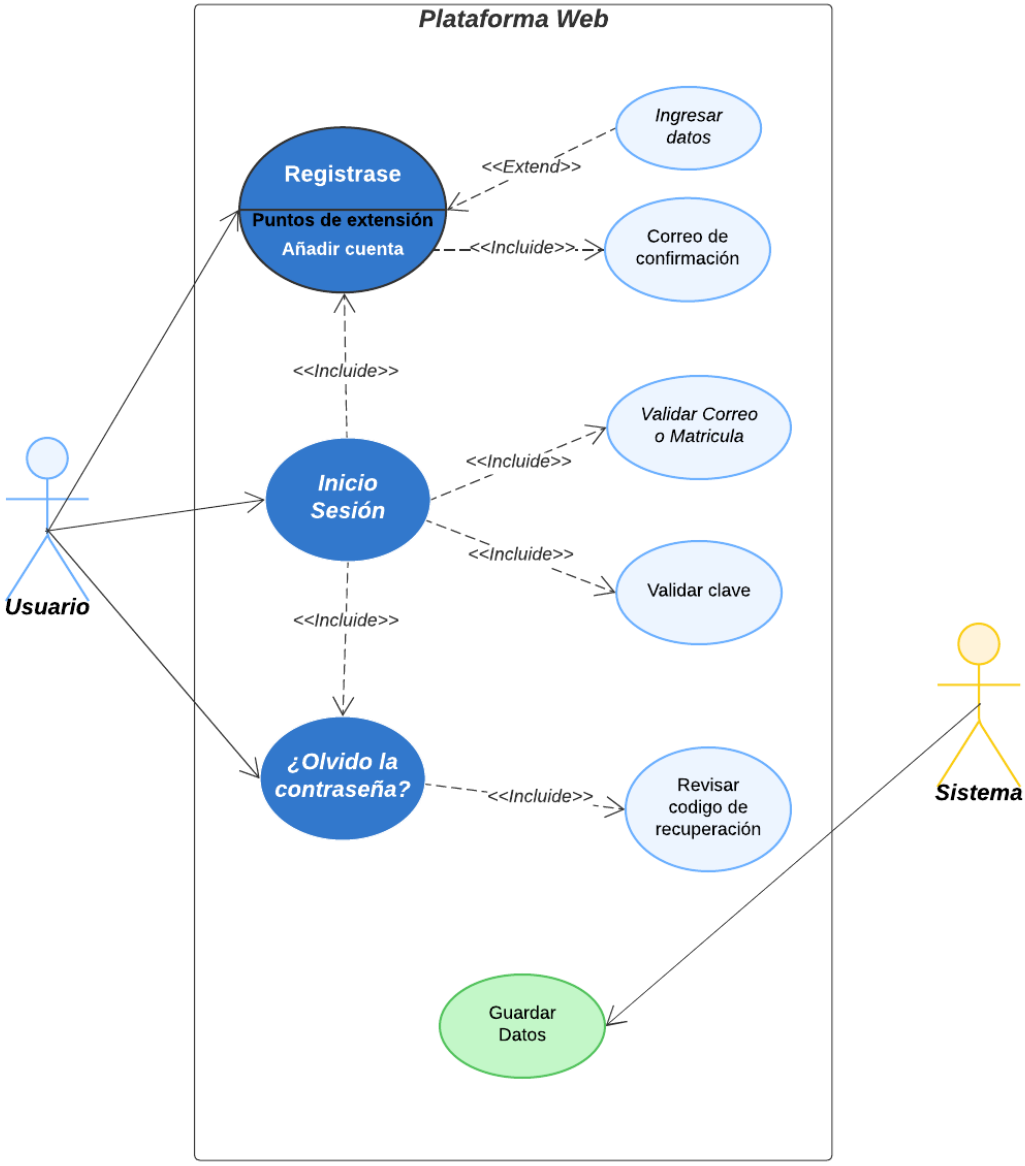


Figura 3. 2. Diagrama de uso para inicio de sesión.

Explicación del diagrama de casos de uso para iniciar sesión.

Este diagrama de casos de uso ilustra los principales procesos y funcionalidades relacionados con el inicio de sesión en la plataforma web. Representa los pasos necesarios para que un usuario pueda registrarse, iniciar sesión, o recuperar su contraseña en caso de haberla olvidado.

Actores:

- **Usuario:** Representa al usuario de la plataforma, quien interactúa con el sistema para registrarse, iniciar sesión o recuperar su contraseña.
- **Sistema:** El sistema de la plataforma que realiza acciones automáticas como validar datos y guardar la información de usuario.

Casos de uso principales:

Registrarse.

Este caso de uso permite que el usuario se registre en la plataforma. Incluye varias acciones que ayudan a completar el proceso de registro, como:

- **Ingresar datos:** El usuario ingresa sus datos personales requeridos para el registro.
- **Correo de confirmación:** Después de ingresar sus datos, el sistema envía un correo de confirmación al usuario para verificar la cuenta.
- **Validar Correo o Matrícula:** El sistema valida el correo electrónico o matrícula ingresada por el usuario para confirmar su identidad.

Iniciar sesión.

Este caso de uso permite que el usuario acceda a su cuenta en la plataforma. El sistema realiza los siguientes pasos para autenticar al usuario:

- **Validar clave:** El sistema verifica que la contraseña ingresada por el usuario sea correcta.
- Si el proceso de validación es exitoso, el usuario puede acceder a la plataforma.

Objetivo del diagrama.

El objetivo de este diagrama es mostrar de forma visual los procesos de autenticación y registro en la plataforma, resaltando cada acción requerida para que el usuario pueda acceder al sistema de manera segura y efectiva. Cada caso de uso refleja una función o proceso clave en el inicio de sesión, el registro y la recuperación de contraseñas, lo que facilita al lector comprender cómo el sistema maneja la autenticación del usuario.

3.5.1.2. **Diagrama para el administrador.**

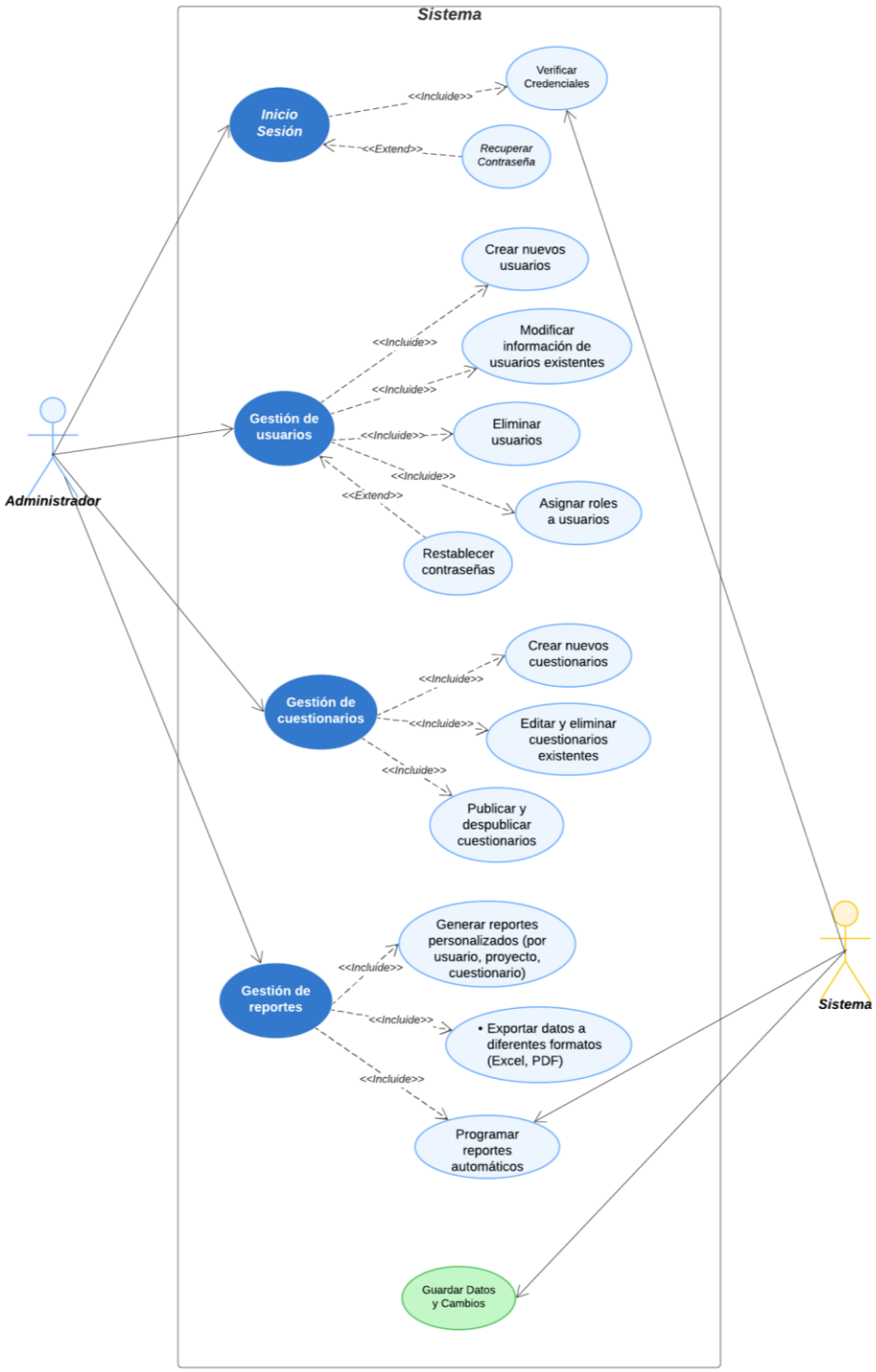


Figura 3. 3. Diagrama de uso para el administrador.

Explicación del diagrama de casos de uso del administrador

Este diagrama de casos de uso describe las funcionalidades disponibles para el administrador en la plataforma web. Su objetivo es ilustrar cómo el administrador puede gestionar usuarios, cuestionarios y reportes dentro del sistema. A continuación, se detallan los actores, casos de uso y sus interacciones.

Actores:

- **Administrador:** Es el usuario con permisos especiales en la plataforma. Este rol permite gestionar usuarios, cuestionarios y reportes, así como realizar tareas de mantenimiento y supervisión del sistema.
- **Sistema:** La plataforma que ejecuta tareas específicas como verificar credenciales, guardar datos y realizar acciones automáticas programadas.

Casos de uso principales:

Inicio de sesión.

El administrador inicia sesión en la plataforma. Este caso de uso incluye:

- **Verificar Credenciales:** El sistema valida las credenciales ingresadas por el administrador.
- **Recuperar Contraseña:** En caso de que el administrador haya olvidado su contraseña, puede seguir un proceso de recuperación de credenciales.

Gestión de usuarios.

Permite al administrador realizar varias acciones relacionadas con los usuarios del sistema:

- **Crear nuevos usuarios:** Agrega nuevos usuarios a la plataforma.
- **Modificar información de usuarios existentes:** Permite actualizar los datos de los usuarios ya registrados.
- **Eliminar usuarios:** Remueve usuarios de la plataforma.
- **Asignar roles a usuarios:** Asigna roles específicos, como usuario regular o administrador, a los distintos usuarios.

- **Restablecer contraseñas:** Permite al administrador cambiar o restablecer las contraseñas de los usuarios.

Gestión de cuestionarios.

Facilita la administración de cuestionarios que estarán disponibles para los usuarios:

- **Crear nuevos cuestionarios:** Crea cuestionarios que los usuarios podrán responder en la plataforma.
- **Editar y eliminar cuestionarios existentes:** Permite modificar o borrar cuestionarios ya creados.
- **Publicar y despublicar cuestionarios:** Da control sobre la disponibilidad de los cuestionarios en la plataforma.

Gestión de reportes.

Proporciona al administrador opciones avanzadas para la generación y gestión de reportes:

- **Generar reportes personalizados:** El administrador puede crear reportes específicos basados en parámetros como usuario, proyecto o cuestionario.
- **Exportar datos a diferentes formatos (Excel, PDF):** Permite al administrador descargar los reportes en distintos formatos para su análisis o documentación.
- **Programar reportes automáticos:** Configura el sistema para generar reportes periódicamente de forma automática.

Objetivo del diagrama.

Este diagrama muestra cómo el administrador interactúa con el sistema para realizar tareas de gestión, supervisión y mantenimiento. La representación visual permite ver de forma clara y organizada los distintos procesos administrativos, así como las dependencias entre ellos. Facilita la comprensión del alcance y las responsabilidades del rol del administrador en la plataforma.

3.5.1.3. **Diagrama para el estudiante.**

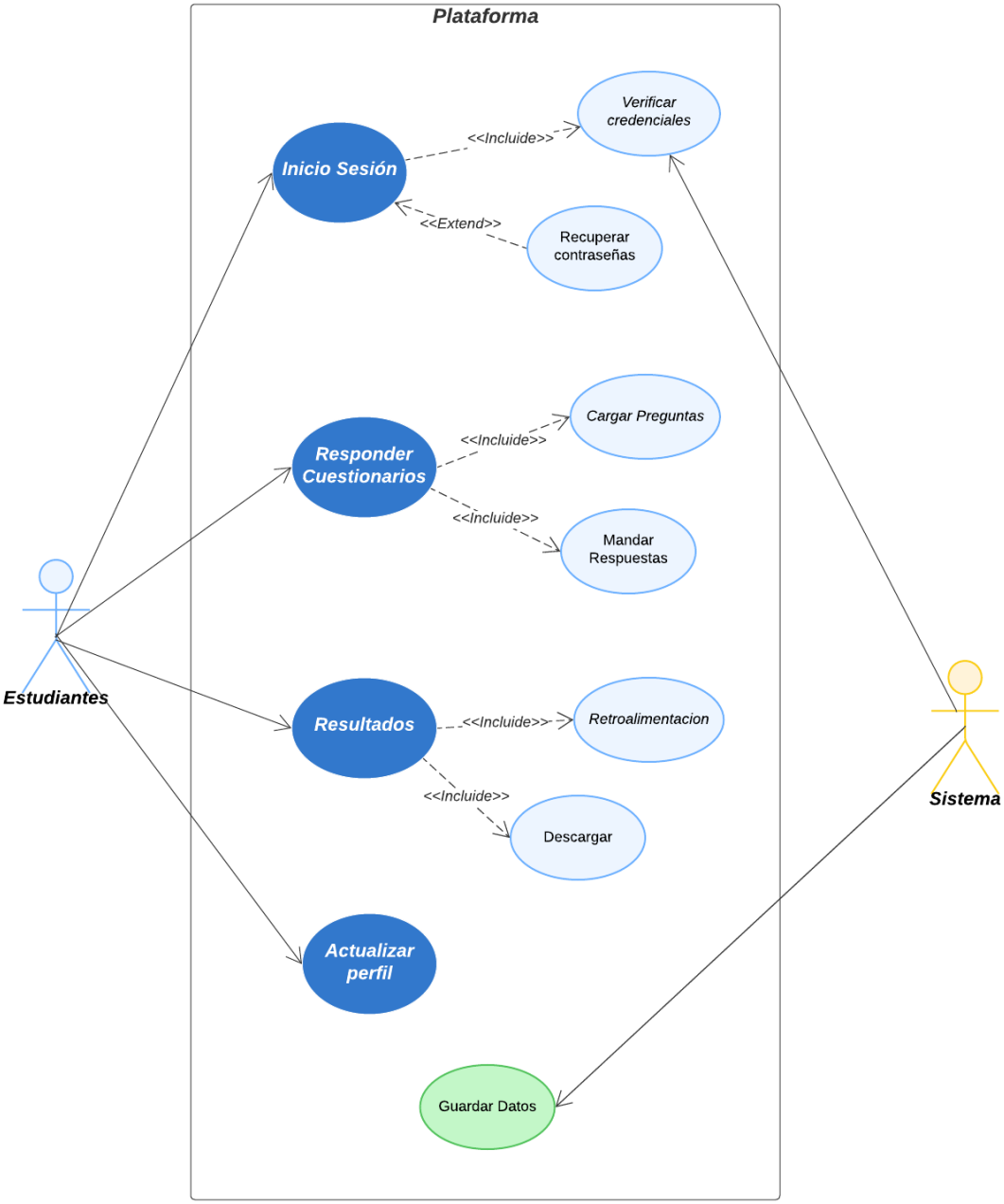


Figura 3. 4. Diagrama de uso para el estudiante.

Explicación del diagrama de casos de uso para el estudiante

Este diagrama de casos de uso representa las funcionalidades que tiene un estudiante en la plataforma web. En él, se muestran los distintos casos de uso que permiten al estudiante interactuar

con la plataforma, tales como iniciar sesión, responder cuestionarios, ver resultados y actualizar su perfil.

Actores:

- **Estudiante:** Es el usuario principal de la plataforma que realiza actividades como responder cuestionarios, revisar resultados y actualizar su perfil.
- **Sistema:** Representa a la plataforma que facilita las funciones y almacena los datos.

Casos de uso principales:

Inicio de sesión.

Este caso de uso permite al estudiante acceder a su cuenta en la plataforma. Incluye las siguientes acciones:

- **Verificar credenciales:** El sistema comprueba que las credenciales proporcionadas por el estudiante sean correctas para permitir el acceso.
- **Recuperar contraseñas:** Si el estudiante ha olvidado su contraseña, puede iniciar un proceso de recuperación para restablecerla.
- **Responder cuestionarios:** Permite al estudiante completar los cuestionarios asignados en la plataforma. Este caso de uso incluye:
- **Cargar preguntas:** El sistema carga las preguntas del cuestionario para que el estudiante pueda responderlas.
- **Resultados:** El estudiante puede revisar los resultados de los cuestionarios que ha completado. Este caso de uso incluye:
- **Retroalimentación:** El sistema proporciona al estudiante una retroalimentación basada en sus respuestas.
- **Guardar datos:** En cada uno de los casos de uso anteriores, el sistema guarda los cambios o datos proporcionados para garantizar que la información esté actualizada y sea accesible en el futuro.

Objetivo del diagrama.

Este diagrama está diseñado para ilustrar cómo los estudiantes interactúan con la plataforma para llevar a cabo sus actividades académicas y de autoevaluación. Cada caso de uso

muestra los pasos necesarios para completar una tarea específica, y el uso de relaciones ayuda a desglosar y organizar las funciones de una manera estructurada. Esto facilita la comprensión de los procesos que el estudiante debe seguir y cómo la plataforma responde a sus acciones.

3.5.1.4. **Diagrama para el evaluador**

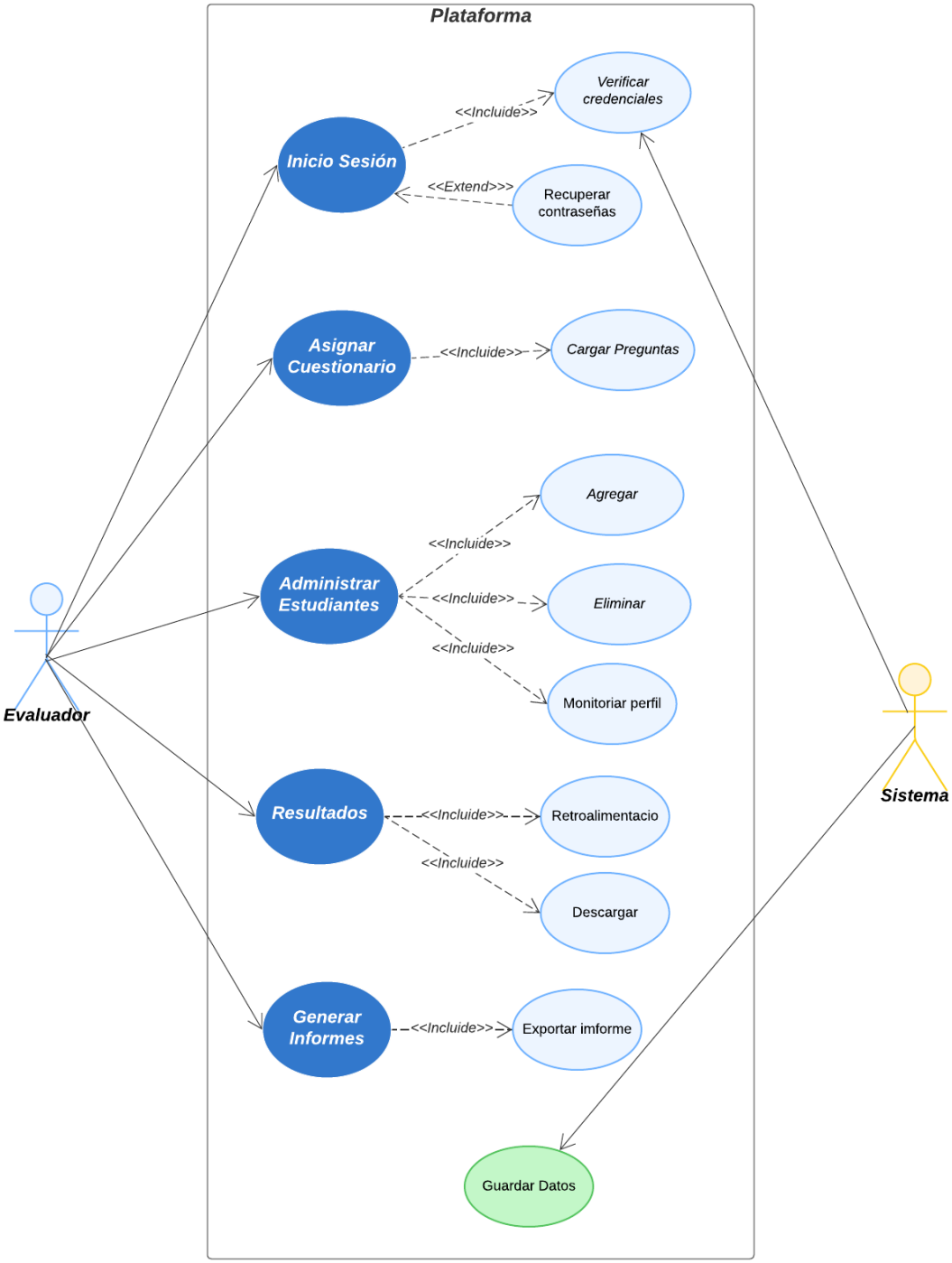


Figura 3. 5. Diagrama de uso para el evaluador.

Explicación del diagrama de casos de uso para el evaluador

Este diagrama de casos de uso representa las interacciones del evaluador con la plataforma, detallando las tareas relacionadas con la gestión de estudiantes y cuestionarios, así como la generación de informes.

Actores:

- **Evaluador:** Usuario con privilegios para administrar estudiantes, asignar cuestionarios y generar informes sobre los resultados.
- **Sistema:** La plataforma que facilita las funciones de administración y almacenamiento de datos.

Casos de uso principales:

Inicio de sesión.

Permite al evaluador acceder a la plataforma con sus credenciales. Incluye:

- **Verificar credenciales:** Comprobación de credenciales para autorizar el acceso.
- **Recuperar contraseñas:** Funcionalidad de recuperación de contraseña en caso de olvido.

Asignar cuestionario.

Permite al evaluador asignar cuestionarios a los estudiantes. Este caso de uso incluye:

- **Cargar preguntas:** El evaluador puede cargar las preguntas necesarias para el cuestionario asignado.

Administrar estudiantes.

Facilita la administración de los estudiantes inscritos en la plataforma, con los siguientes subprocesos:

- **Agregar:** Agregar un nuevo estudiante a la plataforma.
- **Eliminar:** Quitar a un estudiante de la plataforma.

Resultados.

El evaluador puede revisar los resultados de los cuestionarios completados por los estudiantes. Este caso de uso incluye:

- **Retroalimentación:** Proveer retroalimentación a los estudiantes con base en sus respuestas.
- **Descargar:** Descargar los resultados en un formato accesible.

Generar informes.

Facilita la creación de informes de los resultados y actividades de los estudiantes. Incluye:

- **Exportar informe:** Permite al evaluador exportar un informe en un formato adecuado para su análisis o almacenamiento externo.
- **Guardar datos:** En todos los casos de uso, el sistema almacena los datos relevantes, garantizando la persistencia de la información y la disponibilidad para futuras consultas.

Objetivo del Diagrama.

El objetivo del diagrama es visualizar cómo el evaluador utiliza la plataforma para administrar a los estudiantes y cuestionarios, revisar los resultados, y generar informes. La estructuración de los casos de uso con relaciones ayuda a comprender las acciones necesarias para cada tarea y cómo el sistema responde en consecuencia.

Capítulo 4.

RESULTADOS Y DISCUSIÓN.

4.1. INTRODUCCIÓN A LOS RESULTADOS.

En este capítulo se presentan los resultados a partir de la evaluación de Actitudes hacia el servicio comunitario (CSAS), donde se llevó a cabo el funcionamiento del interfaz y la lógica de su funcionamiento. Los resultados obtenidos en el desarrollo y pruebas iniciales de la plataforma web para automatizar la recolección y análisis de datos del CSAS.

Se muestran capturas de pantalla, estadísticas recopiladas y ejemplos del análisis automatizado que facilita el sistema.

4.2. FUNCIONAMIENTO Y PRUEBAS DE LA PLATAFORMA.

La plataforma permite recopilar respuestas en tiempo real y de igual manera almacenarlas en una base de datos estructurada.

4.2.1. Registro e inicio de sesión.

La plataforma cuenta con un módulo de autenticación que permite a los usuarios registrarse e iniciar sesión según el rol que pertenece, haciendo esto más eficaz porque si un usuario no está registrado no podrá acceder al contenido de la plataforma.

4.2.1.1. Interfaz de inicio de sesión.

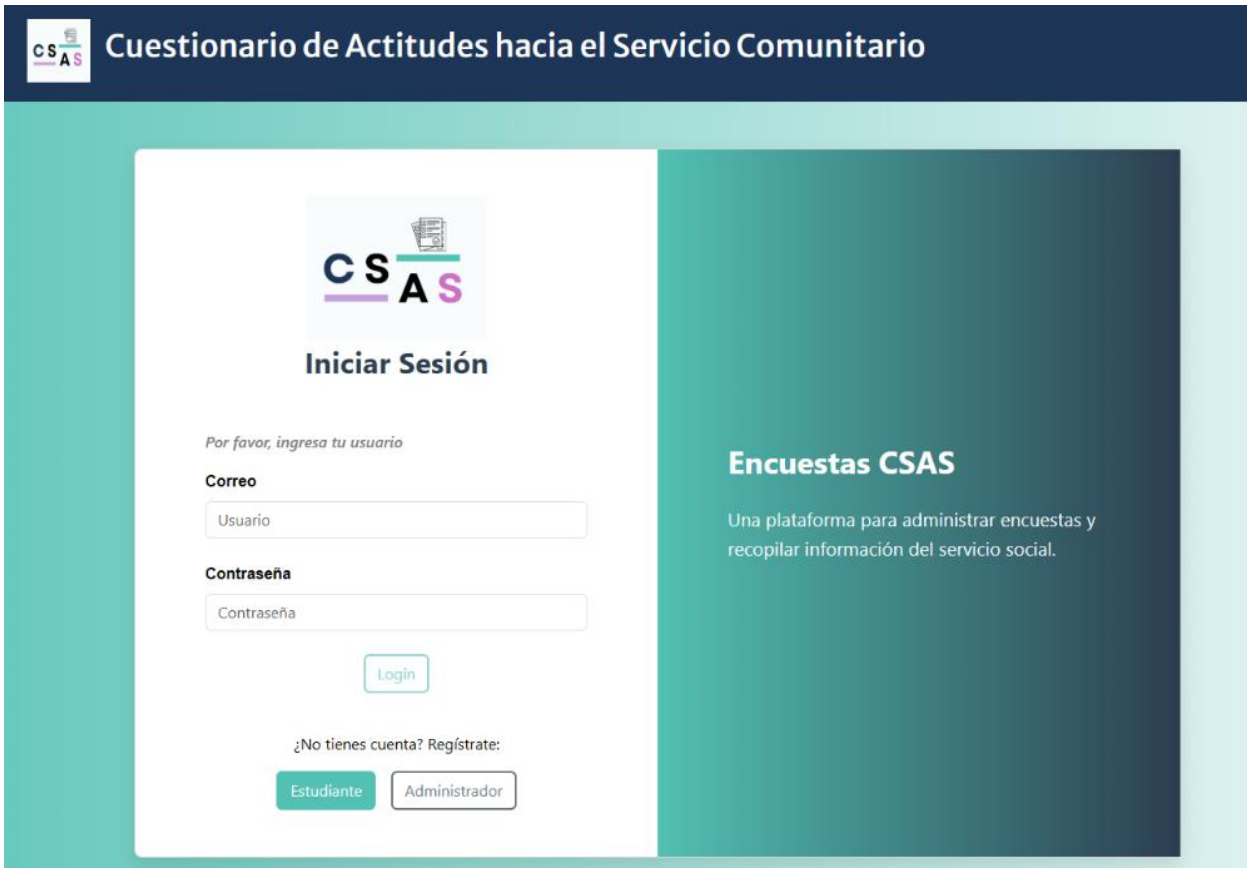


Figura 4. 1. Interfaz de inicio de sesión

En la figura 4.1 se muestra la pantalla de sesión, en donde los usuarios ingresan sus credenciales.

En la figura 4.2 y 4.3 respetivamente de administradores y estudiantes, dependiendo de su rol, se redirigen a diferentes paneles o tableros: los administradores gestionan encuestas y usuarios, mientras que los estudiantes solo pueden responder cuestionarios.

Figura 4. 2. Panel de usuario rol estudiante.

Figura 4. 3. Panel de usuario rol administrador.

4.2.1.2. Interfaz de registro.

En las figuras 4.4 y 4.5 se muestra el formulario de registro, donde se solicitan datos básicos, como nombre, correo electrónico, contraseña.

Dependiendo igual del tipo de usuario si es estudiante se solicitan datos como matricula, institución, carrera, semestre o grado y grupo, y para lo de administrador un ID e institución, género y edad.

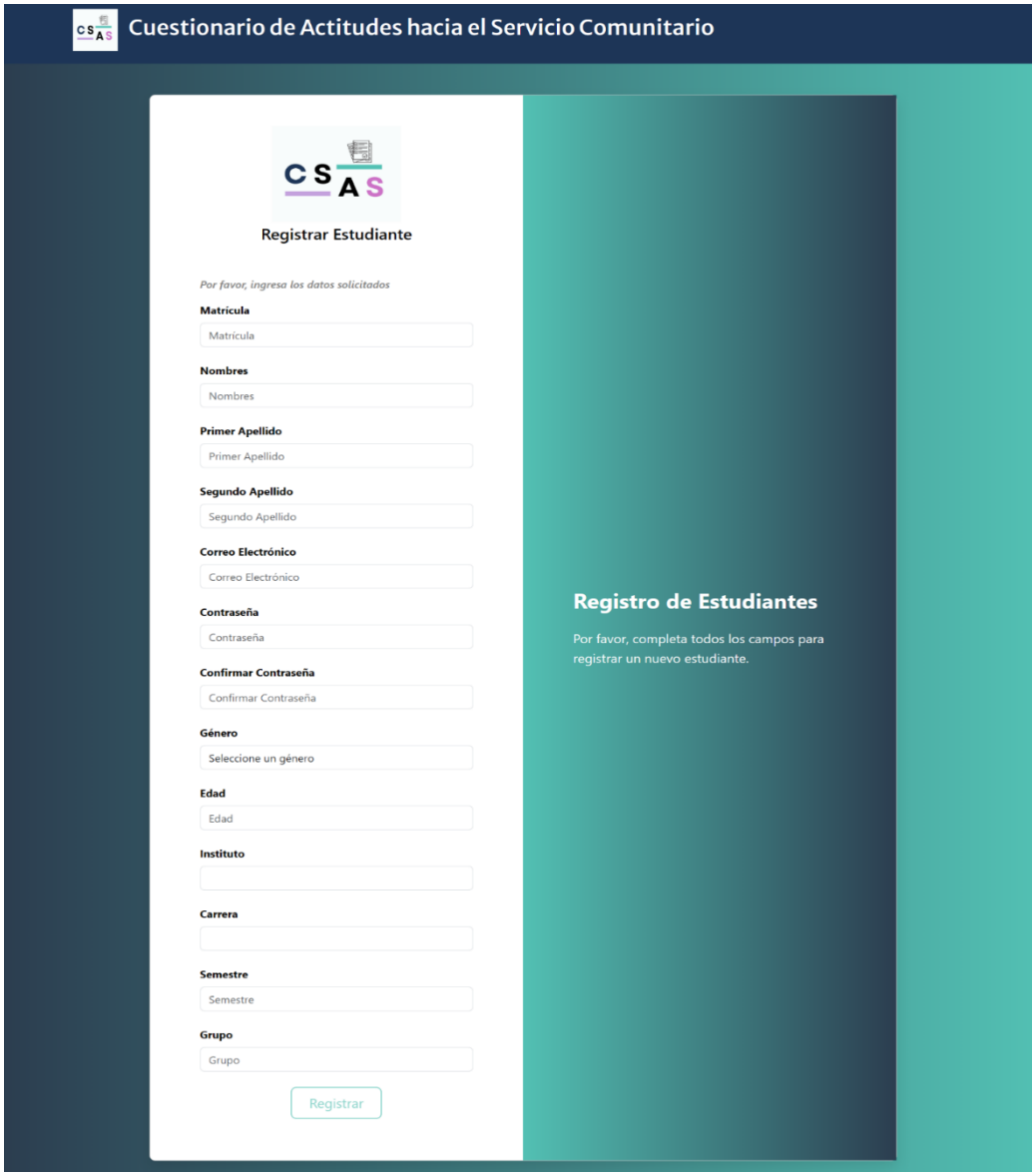


Figura 4. 4. Interfaz de registro para usuarios estudiantes.

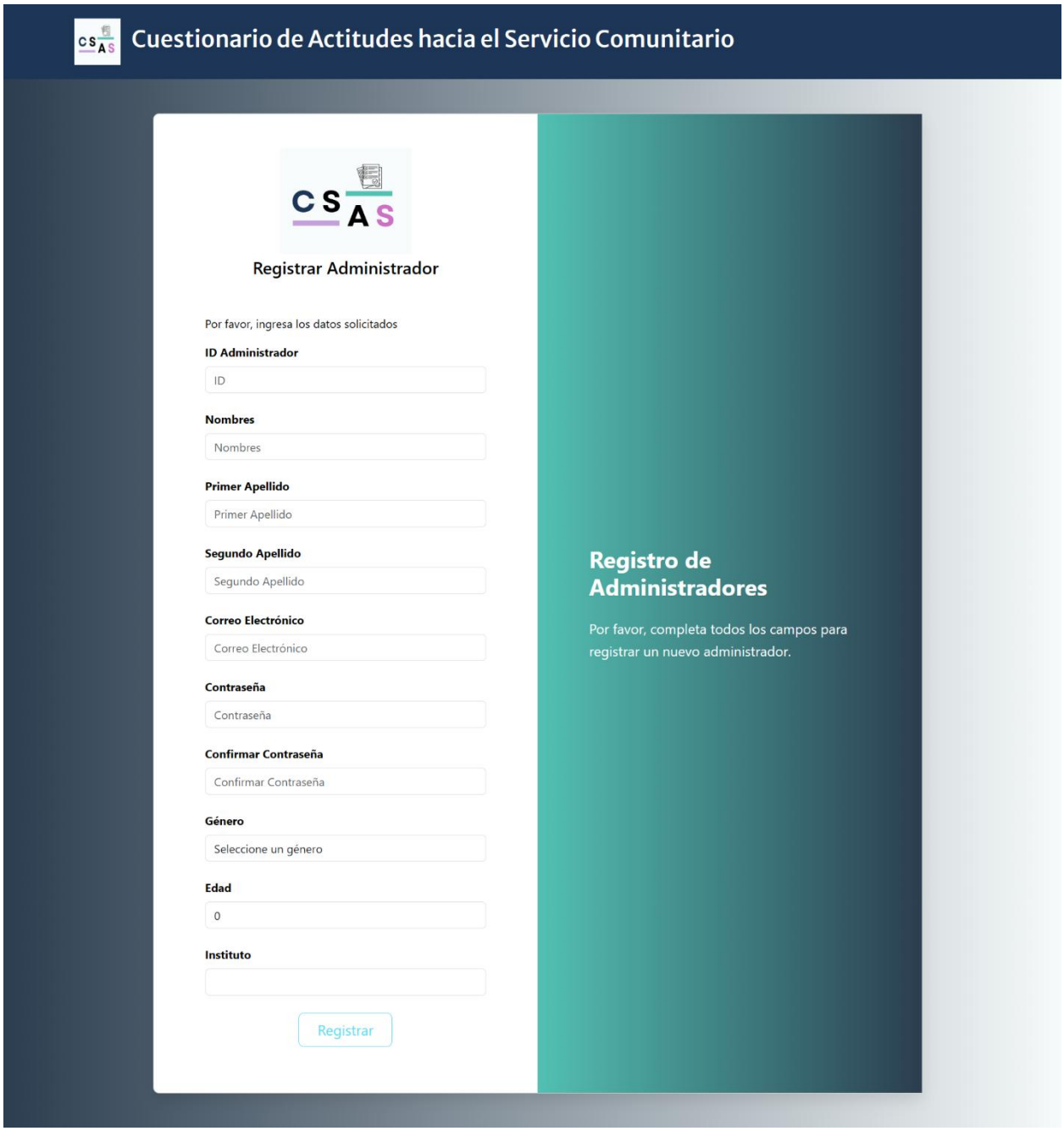


Figura 4. 5. Interfaz de registro para usuarios administrador.

Figura 4. 6. Validación de los campos.

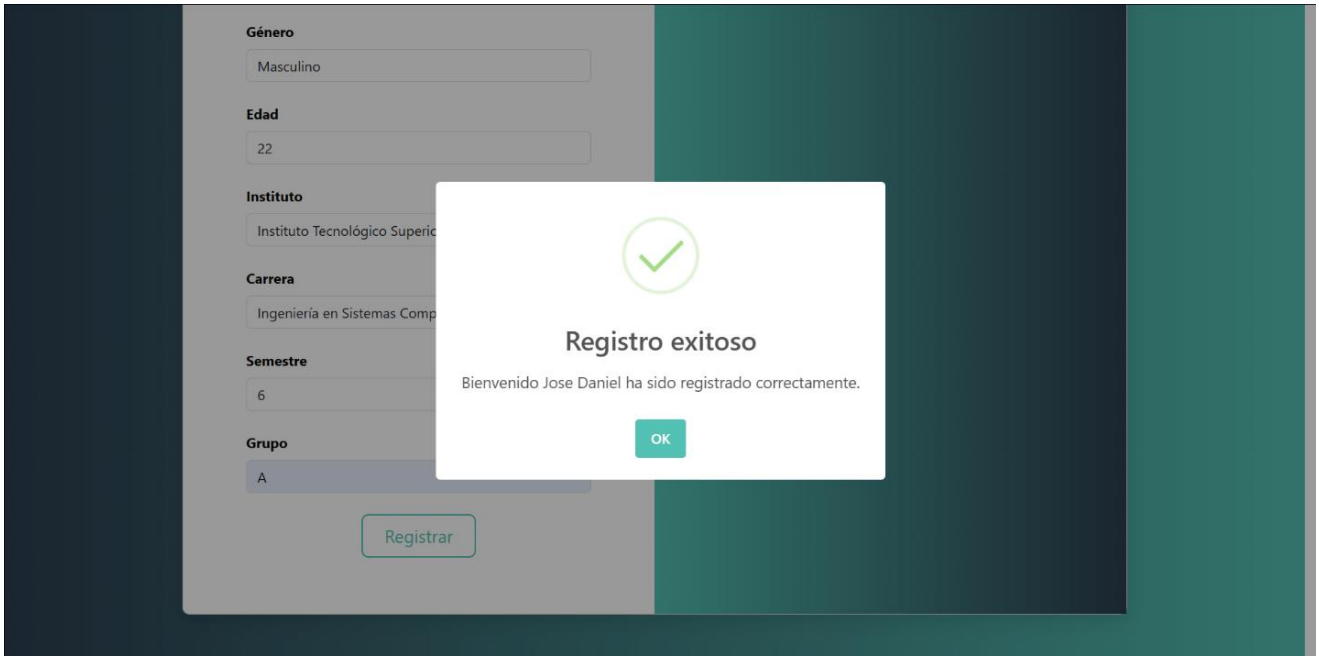


Figura 4. 7. Notificación de registro ha sido exitoso.

4.2.1.3. Interfaz de gestión de usuarios.

La plataforma cuenta con un apartado para la gestión de usuarios o listado de usuario, solo que dependiendo si el usuario es de tipo administrador se muestra el apartado dentro de su papel, como se muestra en la figura 4.8 en donde se ve la lista de los usuarios con los campos correspondiente a sus datos, en donde se puede agregar nuevos usuarios, modificaciones sobre los datos de los usuarios como se muestra en la figura 4.9 y de igual manera poder eliminar cuentas.

Figura 4. 8. Modulo para la gestión de usuarios.

Figura 4. 9. Ventana de modificación de usuario.

4.2.2. Aplicación del cuestionario CSAS.

El cuestionario de actitudes hacia el servicio comunitario (CSAS) es una herramienta clave en la evaluación de las actitudes de estudiantes. En la plataforma desarrollada, los estudiantes pueden acceder al cuestionario de manera digital, responder las preguntas en formato Likert y enviar sus respuestas, los cuales son almacenadas en la base de datos para posterior análisis.

4.2.2.1. Acceso y compleción del cuestionario.

Para acceder al cuestionario, los estudiantes deben de iniciar sesión en la plataforma con sus credenciales. Una vez autenticados, son dirigidos al panel de usuario para estudiantes, en donde pueden visualizar las encuestas disponibles o asignados previamente por los administradores.

En la figura 4.10 se muestra como un usuario ingresa sus credenciales y accede al panel principal que corresponde al de estudiantes.

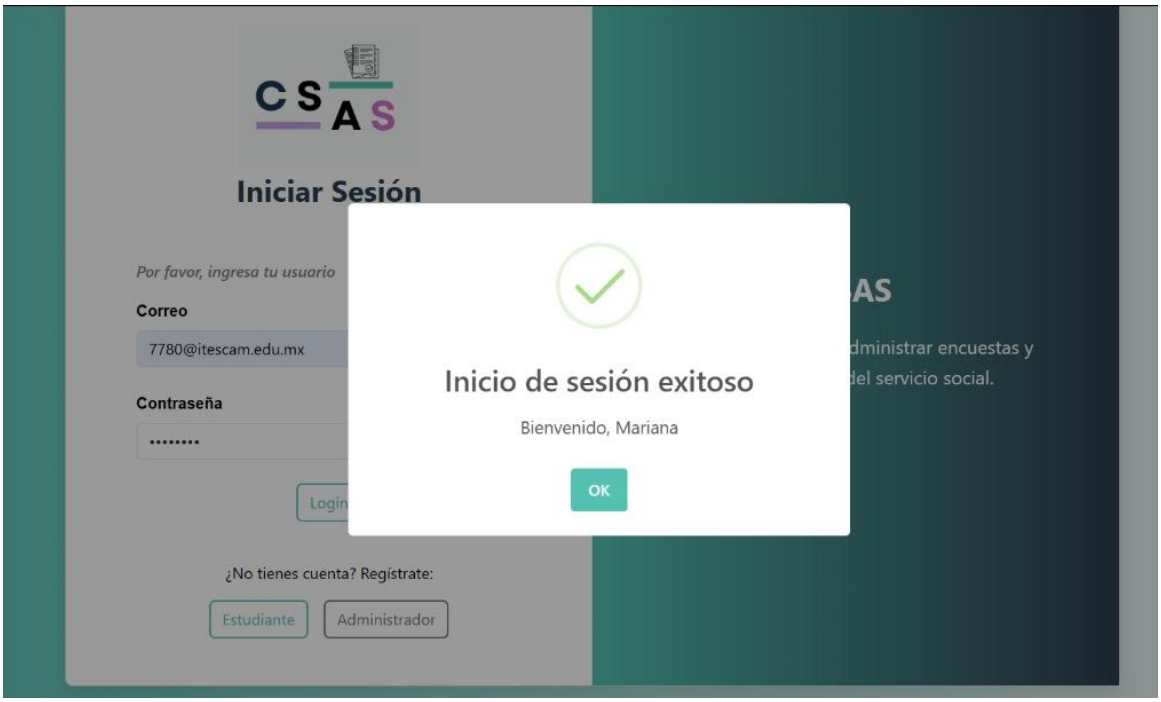


Figura 4. 11. Inicio de sesión ha sido exitoso.

Figura 4. 10. Usuario accediendo a la plataforma para contestar el cuestionario CSAS.

En la figura 4.12 se muestra el panel para el usuario de tipo estudiante, en donde se encuentran las listas de las encuestas activas y pendientes asignadas a cada usuario.

Figura 4. 12. Panel de usuario tipo estudiante.

En la figura 4.13 se muestran la interfaz del cuestionario tal como la visualizan los estudiantes.

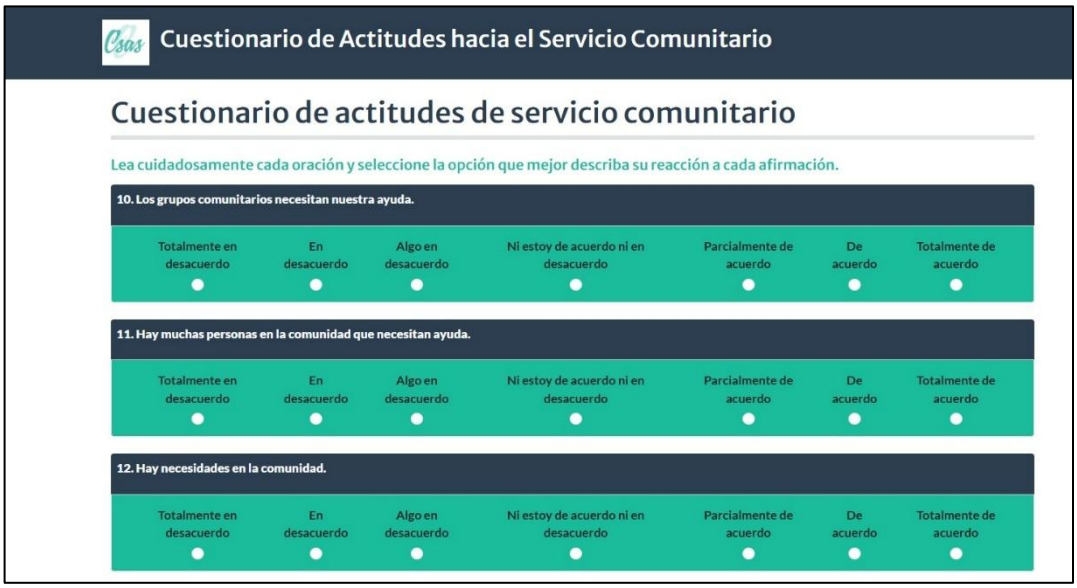


Figura 4. 13. Visualización del cuestionario.

Después de seleccionar una encuesta activa, el sistema muestra las instrucciones generales y permite al estudiante comenzar a responder como se muestra en la siguiente figura 4.14.

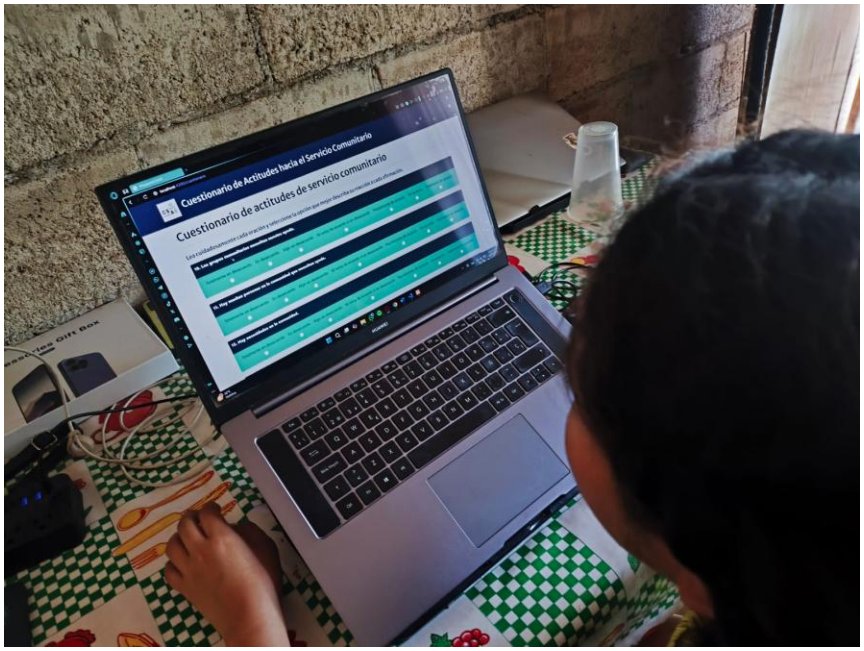


Figura 4. 14. Usuario contestando el cuestionario en la plataforma web.

4.2.2.2. Interfaz del cuestionario.

La interfaz del cuestionario está diseñada para ser intuitiva. Cada pregunta se presenta con su respectiva escala Likert de repuestas, así asegurando que los estudiantes seleccionen su nivel de acuerdo con facilidad.

En la figura 4.15 se muestra una alerta que indica que faltan preguntas por responder. Solo cuando todas preguntas sean contestadas se podrán pasar a la siguiente página y así respectivamente hasta que se llegue a la página final y terminar de contestar cada pregunta se mostrara la notificación que ha finalizado la encuesta con éxito, como se muestra en la figura 4.17.

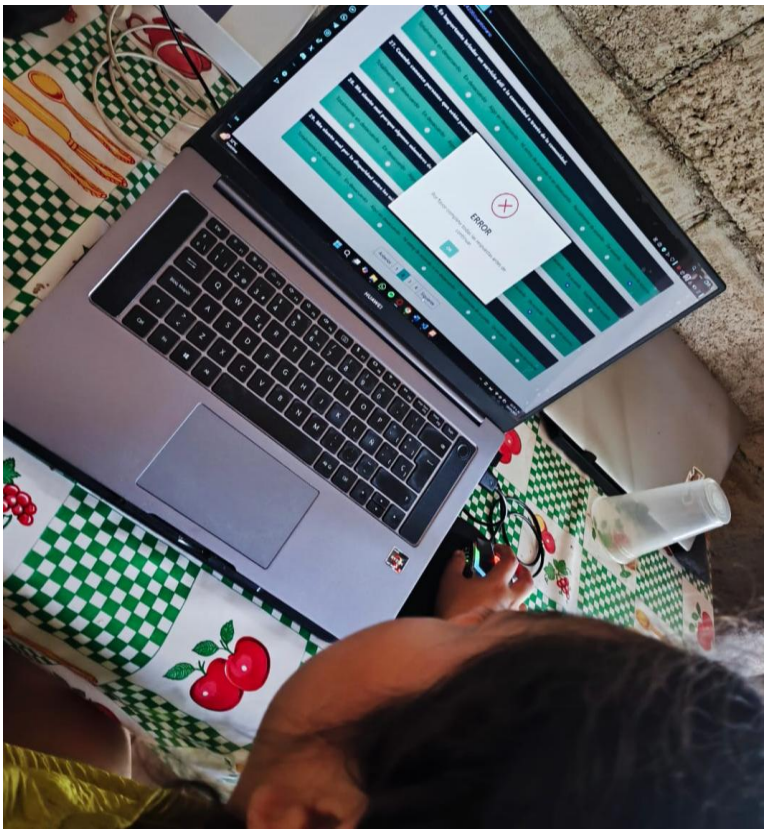


Figura 4. 16. Notificación de error al intentar avanzar sin completar todas las preguntas.

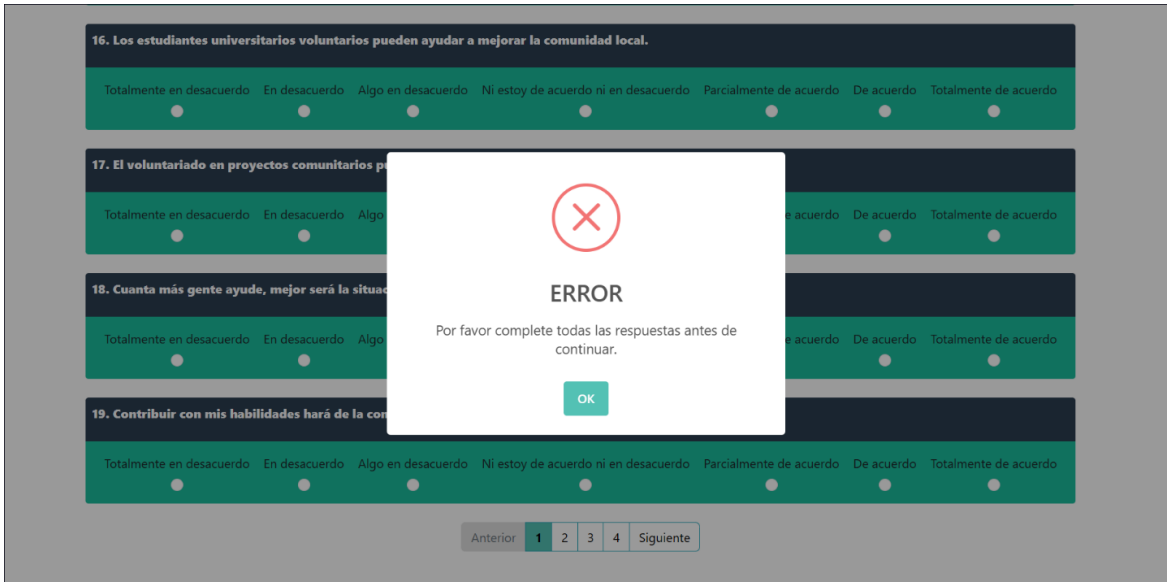


Figura 4. 15. Visualización de alerta de error por falta de repuesta en alguna pregunta.

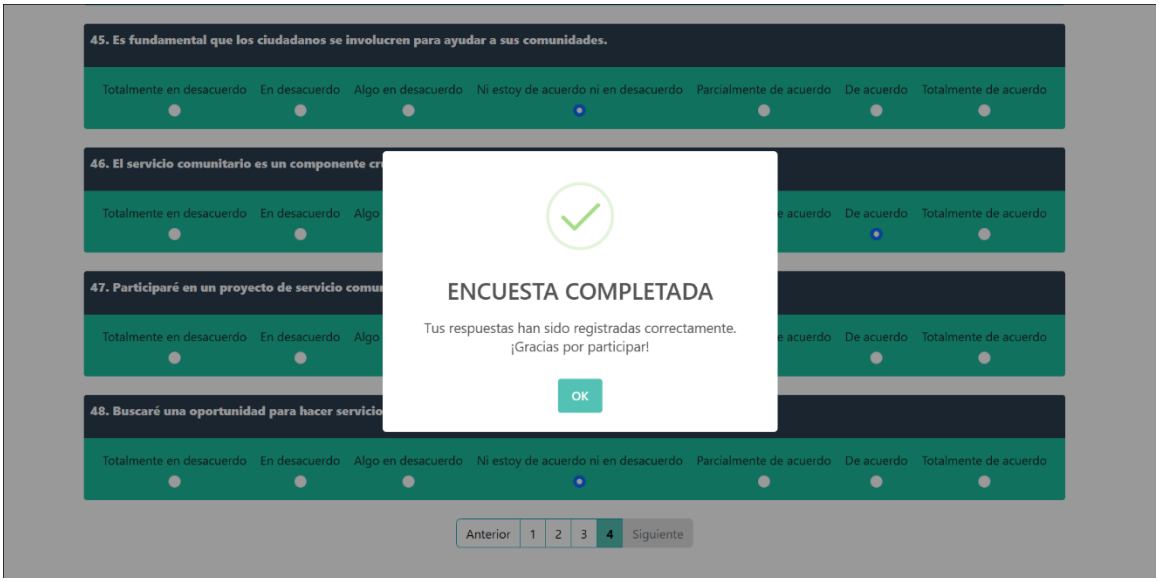


Figura 4. 17. Notificación de finalización del cuestionario en la plataforma.

Figura 4. 18. Visualización de la notificación de encuesta a completada y agradecimiento de participación.

4.2.2.3. Almacenamiento de Datos en la base de datos.

Una vez que el estudiante envía sus repuestas, los datos son registrados en la base de datos de la plataforma.

En la figura 4.19 se muestra que cada repuesta se almacena de la siguiente manera, ID de la repuesta para el orden dentro de la tabla, ID de los estudiantes que es la matricula, el ID de encuesta el numero de la encuesta lo que sirve para identificarlos, ID de la pregunta para identificar la pregunta especifica, en la columna de repuesta es la opción seleccionada (valor de la escala Likert), y por último esta la fecha de registro que indica cuando se llevó a cabo las respuestas.

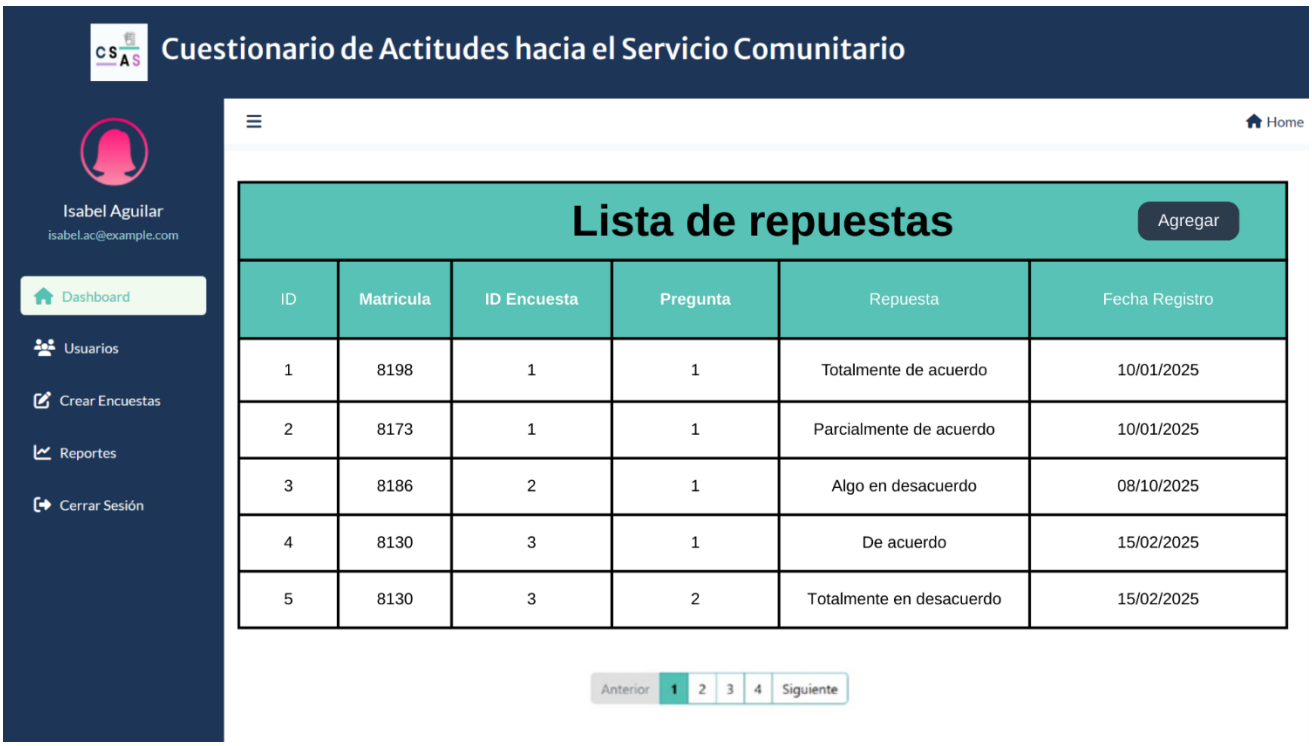


Figura 4. 19. Se visualiza la lista de repuestas de las encuestas.

4.2.3. Análisis automatizado de datos.

La plataforma permite recopilar respuestas en tiempo real y almacenarlas en una base de datos estructurada. A través de las herramientas estadísticas integradas, el sistema genera reportes sobre la evolución de las actitudes de los estudiantes hacia el servicio comunitario.

Si bien para filtrar los resultados por categorización de preguntas por dimensiones aún está en desarrollo, los gráficos generados hasta el momento muestran la distribución de repuestas

individuales en el cuestionario CSAS. Por ejemplo, las respuestas a preguntas específicas sobre la percepción del servicio comunitario se presentan en diagramas circulares, permitiendo una rápida interpretación de los resultados obtenidos.

Ejemplo de visualización de los datos en la plataforma.

A continuación, se presentan capturas de pantallas que muestran los reportes generados en la plataforma:

En la figura 4.20 se muestra la distribución de repuestas por género, se muestra el porcentaje de participantes masculinos y femeninos, En la misma figura se aprecia la gráfica de participación en instituciones de servicio comunitario, lo que permite conocer cuántos estudiantes han participado activamente en iniciativas sociales.

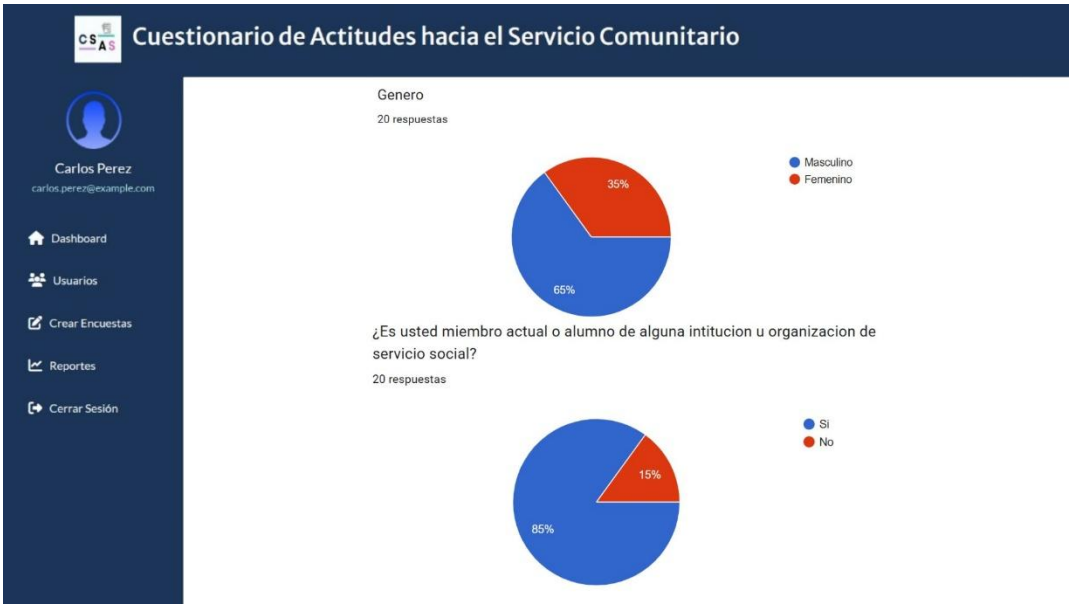


Figura 4. 20. Distribución de repuestas.

En la siguiente figura 4.21 se visualiza la gráfica sobre la opinión sobre la necesidad de apoyo comunitario, lo que refleja el nivel de acuerdo de los participantes en relación con la importancia del servicio comunitario.

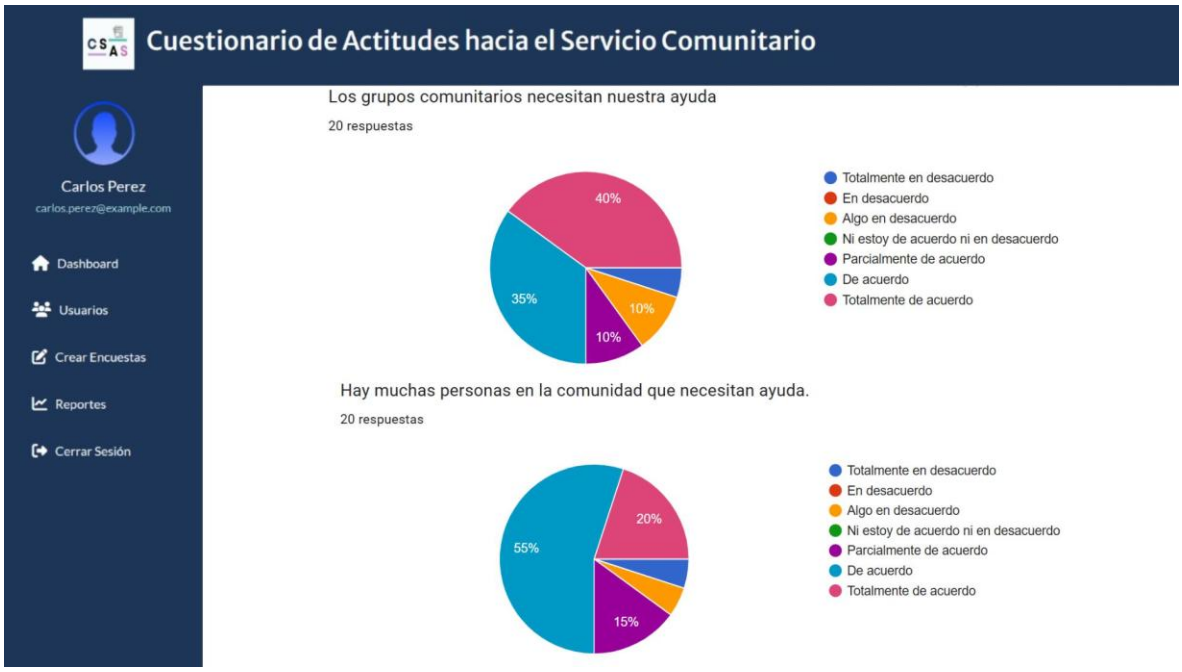


Figura 4. 21. Graficas del nivel de necesidad para el apoyo comunitario.

4.3. COMPARACIÓN CON MÉTODOS TRADICIONALES.

Para evaluar las diferencias entre ambos métodos de recolección y análisis de datos, se ha elaborado la siguiente tabla comparativa. Esta permite visualizar de manera clara las ventajas y desventajas de cada enfoque, resaltando aspectos clave como el tiempo de procesamiento, la precisión de los datos, la accesibilidad y la trazabilidad de la información.

A continuación, se presentan los principales análisis a partir de esta comparación:

Tabla 4. 1. Tabla comparativa entre método tradicional y plataforma web.

Criterios	Método tradicional (Encuestas en papel).	Método automatizado (CSAS – Plataforma Web).
Tiempo de recolección	Lento, depende de la distribución y recolección física de encuestas.	Instantáneo, las respuestas se registran en tiempo real.

Errores de transcripción	Alto riesgo de errores humanos al digitalizar respuestas manualmente.	Mínimo, las respuestas se almacenan directamente en la base de datos.
Análisis de datos	Requiere procesamiento manual en hojas de cálculo.	Generación automática de gráficos y reportes.
Accesibilidad de la información	Limitada, requiere acceso físico a los formularios en papel.	Accesible desde cualquier dispositivo con conexión a internet.
Costos operativos	Elevados (impresión, distribución, almacenamiento de encuestas).	Reducidos (uso de almacenamiento digital)
Seguimiento longitudinal	Difícil, requiere seguimiento manual de datos históricos.	Sencillo, permite cambios a lo largo del tiempo con datos estructurados.

En comparación con los métodos tradicionales de recolección en papel, la automatización de la CSAS ha reducido el tiempo, eliminando errores de transcripción y facilitando el acceso inmediato a estadísticas descriptivas.

Adicionalmente, la automatización permite generar reportes de manera inmediata, evitando la demora del procesamiento manual de datos en hojas de cálculo la accesibilidad digital también permite a los administradores y evaluadores consultar los resultados en tiempo real, sin necesidad de revisar formularios físicos.

4.4. DISCUSIÓN DE RESULTADOS.

En esta sección el propósito principal es analizar los resultados obtenidos hasta el momento y como estos responden a la pregunta de investigación:

¿Cómo puede una plataforma web automatizar y mejorar la recolección y análisis de datos longitudinales de actitudes hacia el servicio comunitario en estudiantes y egresados, utilizando el Cuestionario de Actitudes hacia el Servicio Comunitario (CSAS)?

Dado que la plataforma aun esta en desarrollo y no se ha completado la integración de todas sus funcionalidades, el análisis se basará en los avances alcanzados hasta el momento y simplemente reconocerá las características que todavía están en desarrollo como no evaluadas aún.

4.4.1. Automatización de la recolección de datos.

Actualmente esto permite comenzar ya que la plataforma ahora admite registro de usuarios, autenticación y almacenamiento básico de datos. Pero la interfaz grafica para crear encuestas para los administradores aún está en desarrollo. Esto también significa que, aunque el backend ya soporta la gestión de encuestas, los usuarios no pueden visualizarlas ni administrarlas de manera intuitiva desde el frontend.

Punto clave:

Dado que los estudiantes no puedan ver que encuesta le pertenece o cual encuesta contestar en la plataforma, la recopilación de datos aun no opera de manera completamente autónoma. Necesita completar la conexión entre el frontend y el backend para que los administradores puedan formular, manipular y realizar encuestas.

4.4.2. Mejoras en el análisis de datos.

La plataforma ya posee una base de datos estructurada que puede usarse para almacenar respuestas de encuestas y generar informes básicos. Se ha creado algunos gráficos en función de los datos de respuesta de los estudiantes; Sin embargo, estas son respuestas generales porcentuales y no de una categoría especifica.

Punto clave:

Aunque ahora se pueden generar estadísticas básicas, las herramientas de análisis avanzados (por ejemplo, seguimiento longitudinal y análisis de tenencias) aún deben implementarse. Se sugiere más desarrollos, en términos de filtraos avanzados o agrupación o clasificación por carrera, para permitir un análisis mas preciso de los resultados.

4.4.3. Desafíos identificados en la implementación.

La plataforma ha encontrado múltiples desafíos técnicos y operativos durante su desarrollo, tales como:

- El frontend y backend no están integrados al 100% entre sí, lo que lleva a que el CRUD de encuestas y usuarios no funcione correctamente.
- La capacidad de ver que encuestas tienen disponibles los estudiantes y almacenar las respuestas de los estudiantes desde el frontend al backend.
- El tiempo de desarrollo, porque implementar funcionalidades básicas lleva mucho mas tiempo del esperado, ya que el sistema es complejo y hay una necesidad constante de pruebas.

4.4.4. Efectividad de la plataforma para evaluar actitudes a lo largo del tiempo.

A pesar de las limitaciones mencionadas, el diseño de la plataforma apunta a cumplir con los objetivos plateados en la investigación. Sus mejoras en las encuestas CSAS desde la automatización del almacenamiento y análisis representa un gran avance respecto a lo que solíamos hacer. Aunque esto bueno para las etapas iniciales de desarrollo, hasta que tenga un producto completo con todas las funciones listas y frente a usuarios reales, no se puede medir con precisión.

Conclusión parcial de los resultados:

Esta plataforma tiene el potencial de optimizar la recopilación de datos longitudinales, pero se necesita hacer más trabajo para permitir algunas funcionales básicas, para probar plenamente su efectividad. Sugerimos proceder con el CRUD de encuestas y respuestas, y permitir una mejor visualización de los resultados para un análisis más detallado.

4.5. LIMITACIONES Y TRABAJO FUTURO.

En esta sección se detallan las principales limitaciones encontradas en el desarrollo de la plataforma, así como las oportunidades de mejora para futuras versiones.

4.5.1. Limitaciones técnicas identificadas

- **Integración incompleta entre frontend y backend:** Aunque la lógica en el backend ya está implementada, la interfaz gráfica no está del todo a completada, por lo que no permite gestionar encuestas de manera visual.

- **Análisis de datos limitado:** Actualmente, solo se pueden visualizar gráficos básicos, sin opciones avanzadas de segmentación o comparación a lo largo del tiempo.
- **Dependencia de conexión a internet:** La plataforma requiere acceso a internet para todas sus funciones, lo que podría limitar su uso en entornos con conectividad inestable.
- **Interfaz de usuario en desarrollo:** Algunas funcionalidades, como la gestión de encuestas y usuarios, aún no están disponibles en el frontend.

4.5.2. Mejoras y trabajo futuro

Para optimizar la plataforma y garantizar su efectividad, se sugieren las siguientes mejoras:

1. **Finalización de la gestión de encuestas:** Implementar en el frontend las opciones de creación, adición y eliminación de encuestas para los administradores.
2. **Optimización del almacenamiento de respuestas:** Permitir que los alumnos puedan visualizar que encuestas tienen disponibles y almacenar correctamente sus respuestas.
3. **Mejora en los reportes y análisis de datos:** Incluir herramientas avanzadas para el seguimiento longitudinal, permitiendo identificar tendencias en el tiempo.
4. **Implementación de un sistema de notificaciones:** Notificar a los estudiantes cuando una nueva encuesta este disponible para su participación.
5. **Pruebas con usuarios finales:** Una vez completadas las funcionalidades básicas, realizar pruebas con estudiantes y administradores para validar la usabilidad y efectividad de la plataforma.

Capítulo 5.

CONCLUSIONES Y RECOMENDACIONES.

5.1. CONCLUSIONES.

Este proyecto tuvo como objetivo desarrollar e implementar una plataforma web para la automatización del cuestionario de actitudes hacia del servicio comunitario (CSAS), con el propósito de mejorar la recolección y el análisis de datos longitudinales en estudiantes y egresados de instituciones educativas.

Además, a lo largo de la investigación y el desarrollo realizado, se ha mostraron que la digitalización de encuestas a través de una plataforma web mejoraría la recolección y el almacenamiento de datos, ya que aumenta la seguridad y la velocidad con la que se puede recopilar la información. También reduce los errores humanos, mejora la organización de los datos y proporciona un análisis en tiempo real. Gracias a la integración de tecnologías como Angular, Laravel y MySQL se ha logrado construir una plataforma lo suficientemente usable y segura que es capaz de controlar tantas encuestas y de almacenar tantos datos estructurados respecto de ellas para generar reportes que se puedan usar en los análisis educativos.

A través de aplicaciones de diferentes métodos, fue posible el realizar un desarrollo ágil y organizado mediante la implementación de Scrum. Las iteraciones progresivas con las se realizó la construcción del sistema coincidir con las practicas esenciales para la arquitectura del software cuya particularidad es ir de lo más simple a lo más complejo, por tal motivo, es común el uso de Scrum al principio de cada día. Así como las herramientas para los análisis estadísticos mediante ANOVA y los modelos de regresión han sido de mucho valor para la evolución de las respuestas a lo largo del tiempo.

Sin embargo, la plataforma no se ha desarrollado completamente desde la recolección de datos hasta la etapa de análisis, razón por la cual falta una automatización completa del análisis de datos longitudinales y actividades clave, incluida la producción automatizada de informes por categoría de cuestionario, que aún no están integradas en la plataforma.

Por lo tanto, podemos concluir parcialmente respecto a la pregunta de investigación:

Si, la plataforma web ha sido una buena manera de digitalizar la recopilación de datos, reduciendo errores y asegurando un almacenamiento estructurado. Aún no se ha logrado una automatización completa del análisis de datos longitudinales, ya que aun falta implementar la generación de reportes detallados y la integración de algunas herramientas estadísticas avanzadas.

A pesar de estas limitaciones, los hallazgos muestran que la digitalización del CSAS representa un avance significativo en la evaluación de actitudes estudiantiles, proporcionando una herramienta mas eficiente, confiable y accesible. Con futuras mejoras, la plataforma tiene el potencial de convertirse en una herramienta robusta para la evaluación y seguimiento longitudinal de actitudes hacia el servicio comunitario en estudiantes y egresados.

5.2. RECOMENDACIONES.

Para mejorar la efectividad de la plataforma y alcanzar una automatización completa en la recolección y análisis de datos longitudinales, se sugieren las siguientes recomendaciones:

- Finalizar el desarrollo del módulo de generación de informes; Esto permite que la plataforma procese automáticamente los datos recopilados y proporcione inteligencia bajo demanda.
- Prueba piloto con usuarios reales (estudiantes y administradores) en diferentes instituciones para evaluar la usabilidad de la plataforma y su efectividad en diferentes entornos educativos.
- Implementar modelos predictivos más preferidos y análisis longitudinal, como métodos de aprendizaje automático o minería de datos, para identificar patrones en el desarrollo de actitudes hacia el servicio a la comunidad.
- Mejorar la experiencia del usuario de la plataforma y facilitar la navegación. Esto cubriría aspectos como:
    - Asegurar y escalar el sistema-unificar garantías de que los datos se almacenan de manera segura sin preocuparse por la seguridad de sus datos y también se puede escalar según sea necesario ya que diferentes institutos educativos tienen requisitos variados.

En este punto, a pesar de que la plataforma ya ha demostrado su utilidad en digitalizar encuestas y almacenar datos, la plataforma aún necesita mejoras en los aspectos de automatización de análisis longitudinal. Dadas las optimizaciones recomendadas, la plataforma tiene la capacidad de ser una herramienta necesaria en la educación basada en datos y la medición de la propensión hacia el servicio comunitario a través de programas educativos.

Referencias Bibliográficas.

Alfaro Ureña, D., Ortega A., B., & Lozano, B. (2022). Aplicación del análisis de varianza para comparar el aprendizaje de los estudiantes en tres modalidades: : virtual sincrónica, virtual asincrónica y presencial. *7*(1). Obtenido de https://revistas.up.ac.pa/index.php/guacamaya/article/view/3182

Arnau, J., & Bono, R. (2008). Estudios longitudinales. Modelos de diseño y análisis. *Escritos De Psicología - Psychological Writings*. Obtenido de https://doi.org/10.24310/espsiescpsi.v2i1.13356

Blanco Cano, E., & García Martín, J. (7 de Julio de 2021). El impacto del aprendizaje-servicio (ApS) en diversas variables psicoeducativas del alumnado universitario las actitudes cívicas, el pensamiento crítico, las habilidades de trabajo en grupo, la empatía y el autoconcepto. Una revisión sistemática. (U. C. Madrid, Ed.) *Revista Complutense de Educación*. Recuperado el 29 de Enero de 2025, de http://hdl.handle.net/10366/155872

Briñol, P., Falces, C., & Becerra, A. (2007). Actitudes. *Psicología social*. Recuperado el 2025 de Enero de 29, de https://www.researchgate.net/publication/271838160_Actitudes

Correa, P., Brussino, S., & Reyna, C. (4 de Julio de 2024). Construcción y validación de una escala para evaluar actitudes hacia personas de distinta clase social. *Revista de Psicología (PUCP), 42*(2). Obtenido de http://dx.doi.org/10.18800/psico.202402.015

Corredor, Z., & Romero, R. (2017). Impacto del Servicio Comunitario en Educación Universitaria. *Revista Scientific, 7*(24), 45-60. Recuperado el 2025 de 01 de 10, de https://www.researchgate.net/publication/361283063_Impacto_del_Servicio_Comunitario_en_Educacion_Universitaria

DATAtab. (s.f.). *Pruebas paramétricas y no paramétricas*. Obtenido de https://datatab.es/tutorial/parametric-and-non-parametric-tests

Delgado Rodríguez, M., & Llorca Díaz, J. (2004). Estudios longitudinales: concepto y particularidades. Recuperado el 30 de 01 de 2025, de

http://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S1135-57272004000200002&lng=es.

Ecma International. (1997). *ECMAScript Language Specification.* . Obtenido de https://ecma-international.org/publications-and-standards/standards/ecma-262/

Educación algor. (s.f.). *Concepto y naturaleza de las actitudes.* . Obtenido de https://cards.algoreducation.com/es/content/jot3iM5n/naturaleza-actitudes-comportamiento

FasterCapital. (23 de Junio de 2024). *Analisis de datos longitudinales a traves del tiempo y las tendencias dominar el analisis de datos longitudinales*. Obtenido de https://fastercapital.com/es/contenido/Analisis-de-datos-longitudinales--a-traves-del-tiempo-y-las-tendencias--dominar-el-analisis-de-datos-longitudinales.html

García, J., Gómez, M., & Sánchez, R. (2023). *Inteligencia artificial y educación: nuevas fronteras en la evaluación de competencias.* Revista Internacional de Tecnología Educativa.

García-Ancira, Castillo-Elizondo, & Salinas-Reyna. (2016). *Seguimiento al servicio social del estudiantado de ingeniería a través de la plataforma Nexus.* Revista académica.

GitHub, Inc. (2024). *GitHub Documentation*. Obtenido de https://docs.github.com/es

Gonçalves, M. J. (13 de 10 de 2021). *hiberus.com*. Obtenido de https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/

González, C., & Araujo, W. (2016). El Servicio Comunitario, Una Mirada Teórica. *Revista Scientific, 1*(2), 54-74. Recuperado el 30 de 12 de 2024, de https://www.redalyc.org/journal/5636/563660227005/563660227005.pdf

Google. (2024). *Angular*. Recuperado el 12 de 08 de 2024, de https://angular.dev

JSON Web Tokens. (2024). *Introduction to JSON Web Tokens.* Obtenido de https://jwt.io/introduction

Lizcano, P., & Pérez, S. (2016). *Impacto de las plataformas digitales en el seguimiento del servicio comunitario.* Revista Tecnología y Sociedad.

Microsoft. (2012). *TypeScript: JavaScript with syntax for types.* Obtenido de https://www.typescriptlang.org

Mora Mera, M. M., Ochoa Gonzalez, C. R., Cango Zhinín, M. Á., & Gutiérrez Bastidas, J. O. (2024). *Innovación Educativa enlaUniversidad:Uso deTic eInteligencia Artificial para Mejorar laEnseñanza y Evaluación.* Reincisol. Obtenido de https://doi.org/10.59282/reincisol.V3(6)6409-6427

MySQL. (2024). *MySQL Documentation. Oracle Corporation.* Obtenido de https://dev.mysql.com

Otto, M., & Thornton, J. (2011). *Bootstrap*. Obtenido de https://getbootstrap.com

Otwell, T. (2011). *Laravel: The PHP Framework for Web Artisans*. Obtenido de https://laravel.com

Pascual Arias, C., López-Pastor, V., & Hortigüela-Alcalá. (2023). Estudio longitudinal sobre los efectos del desarrollo de la Evaluación Formativa y Compartida en la Formación Inicial del Profesorado. *Revista Iberoamericana de Evaluación Educativa*. Obtenido de https://www.researchgate.net/publication/367043927

Salamanca Molano, D. (2016). *Evaluación del Servicio Social Estudiantil Obligatorio en la Formación de Competencias Ciudadanas.* Caso de la escuela tecnológica - Instituto Técnico Central . Obtenido de https://repositorio.uniandes.edu.co/entities/publication/f4369507-5864-4f63-8745-716672ca0699

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: La guía definitiva de Scrum: Las reglas del juego.* . Obtenido de Scrum.org: https://www.scrumguides.org/

Shiarella, A., Mccarthy, A., & Tucker, M. (2000). Development and Construct Validity of Scores on the Community Service Attitudes Scale. *Educational and Psychological Measurement, 60*(2), 286-300. Obtenido de https://doi.org/10.1177/00131640021970510

Silvia, K. (03 de 2021). *¿Qué es metodología Scrum? (todo lo que debes saber).* Obtenido de https://blog.kueski.com/finanzas-personales/emprender/metodologia-scrum/?gad_source=1&gclid=Cj0KCQiAire5BhCNARIsAM53K1iLNXgBh6nytSMQ4EhXY03AvMNm3V01QzV5Y7QHGXyFWnLLE0chKR8aAi06EALw_wcB

The PHP Group. (1995). *Hypertext Preprocessor.* Obtenido de https://www.php.net

Torres Gómez, M., Ramos González, B., & Villegas, V. (2023). Servicio Social Comunitario y su Impacto en la Práctica Educativa para la Formación. *Ciencia Latina Revista Científica Multidisciplinar, 7*(4). Recuperado el 2025 de 01 de 15, de https://doi.org/10.37811/cl_rcm.v7i4.7343

Tumino, M., & Korniejczuk, V. (2011). *Conexión entre la Acción, la Pregunta y la Reflexión de los Estudiantes.* Anuario Digital de Investigación Educativa. Recuperado el 02 de 10 de 2024, de https://revistas.bibdigital.uccor.edu.ar/index.php/adiv/article/view/3729

Tumino, M., & Korniejczuk, V. (2012). *Acción y reflexión: actitud del estudiante hacia el aprendizaje*. Obtenido de https://www.scielo.org.ar/scielo.php?pid=S1669-27212012000100005&script=sci_arttext

Vargas Ruiz, R. (2003). Escala de actitudes hacia la tecnología en el aprendizaje escolar aplicada a niños y niñas de primaria pública en Costa Rica. Análisis de validez y confiabilidad. *Actualidades en psicología*. Obtenido de http://pepsic.bvsalud.org/scielo.php?script=sci_arttext&pid=S0258-64442003000100002&lng=pt&tlng=es.

Vargas-González, A., & Morales-Martínez, J. (2020). *Diseño y validación de un cuestionario para la autoevaluación de experiencias de aprendizaje-servicio universitario.* Revista de Educación Superior.

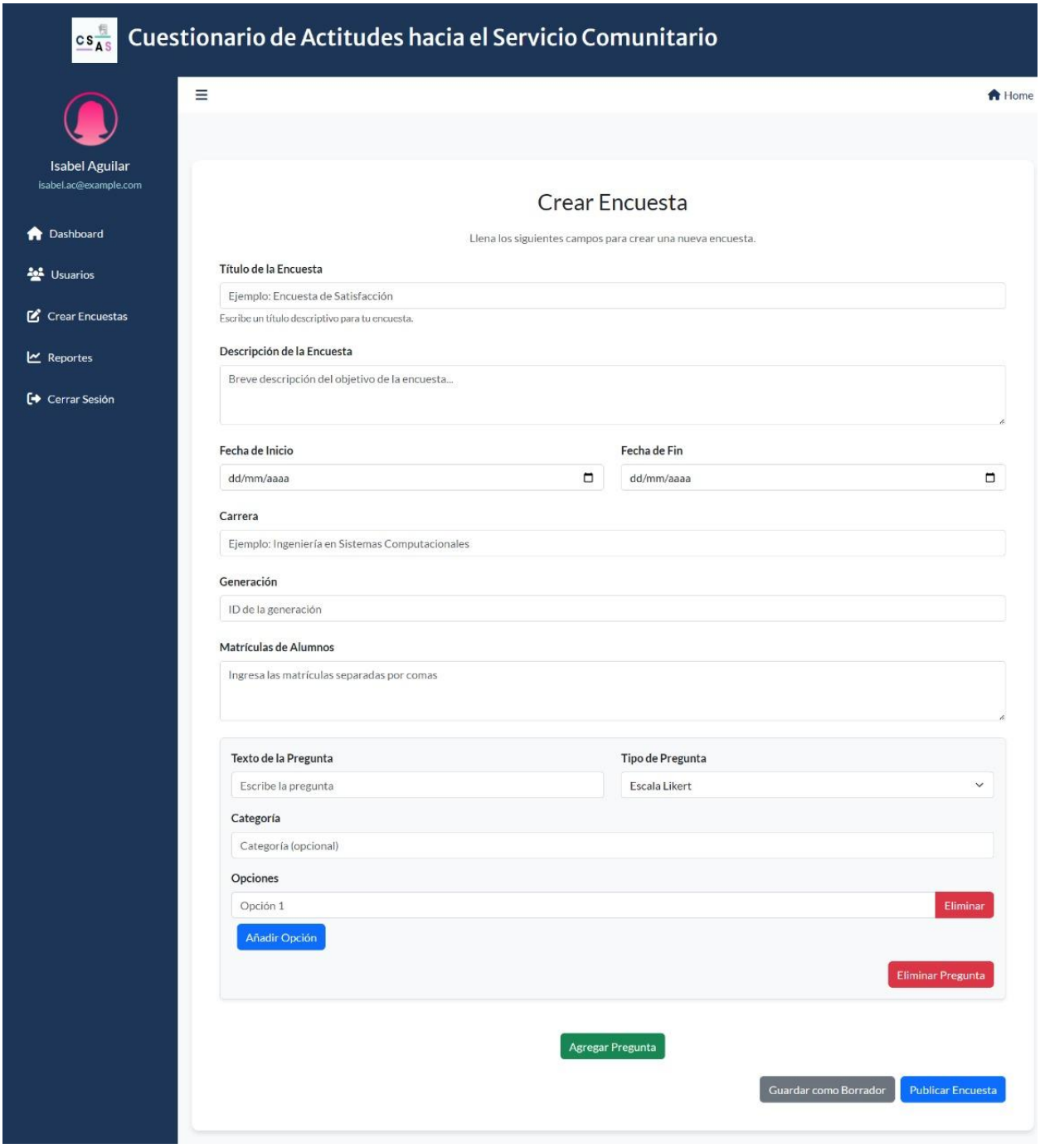
Vargas-Murillo, G. (2019). *Competencias digitales y su integración con herramientas tecnológicas en educación superior.* Cuadernos Hospital de clínicas.

W3C. (2024). *HTML & CSS Standards.* Obtenido de https://www.w3.org/standards/

Anexos.

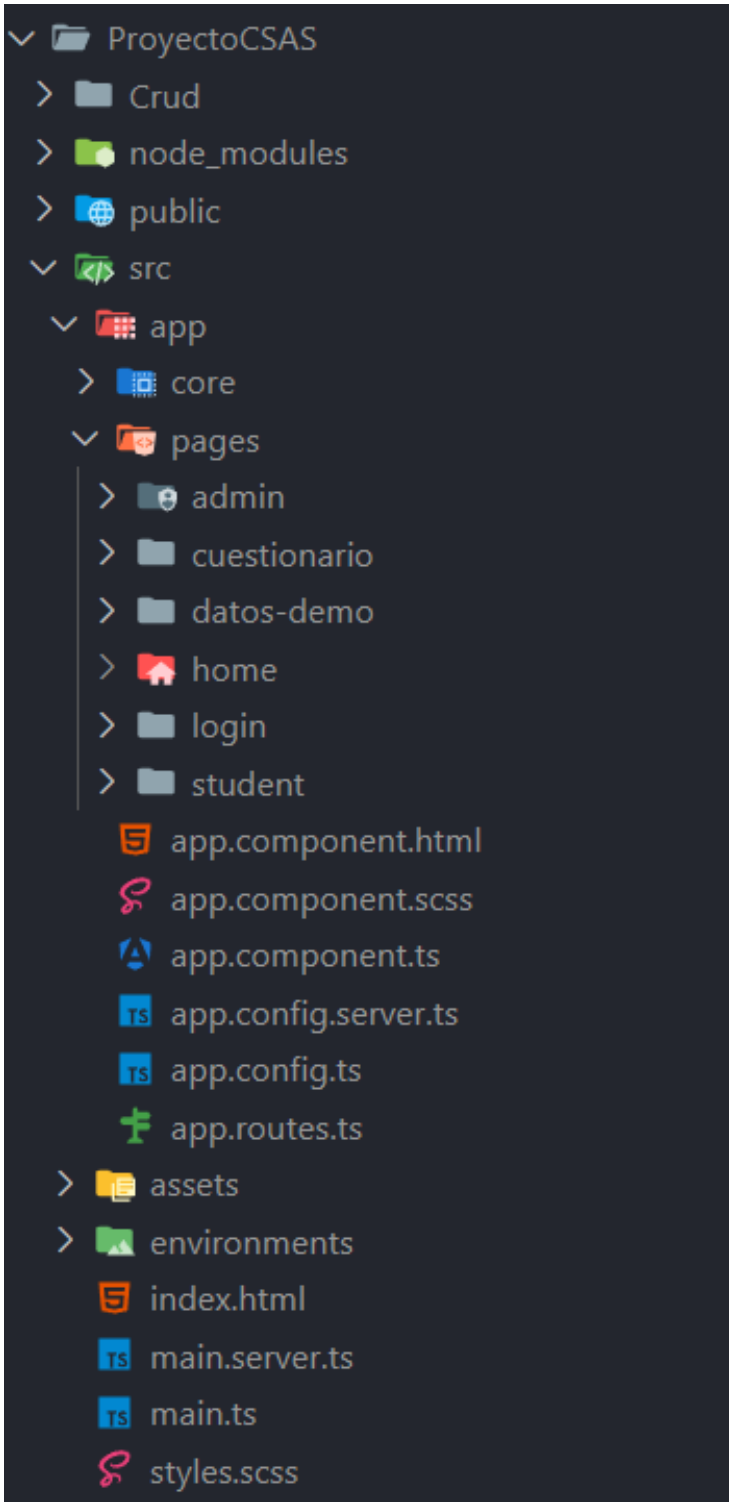
Anexo 1. 1. Pantalla de inicio de la plataforma CSAS.

Anexo 1. 2. Modulo para la gestión de encuestas.



Anexo 1. 3. Modulo para crear encuestas.

Anexo 1. 4. Modulo para cerrar sesión.



Anexo 1. 5. Estructura de las carpetas del proyecto CSAS.

FRONTEND (Diseño, Botones, Páginas).

1. **Código de la página principal**

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">
        <img src="assets/Imagen/ilustracion_1.png" alt="Imagen decorativa" class="decorative-image">
    </div>
  </div>
</div>
```

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 */
    }
   }
  }
}
```

2. **Código de la página de inicio de sesión**

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">
                  <img
                    src="assets/Imagen/Logo_Csas.jpg"
                    style="width: 185px;"
                    alt="logo"
```

```
  />
  <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>
        </div>
      </div>
    </div>
  </div>
</section>
```

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;
}
```

Typescript

```
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',
  styleUrl: './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']);
    }
  }
}
```

3. **Código de la página de registro para el usuario del administrador**

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">
                  <img src="assets/Imagen/Logo_Csas.jpg" style="width: 185px;" alt="logo" />
                  <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"
                      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>

      <!-- 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>
</section>
```

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;
}
```

Typescript

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { ReactiveFormsModule} 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',
  styleUrl: './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;
      }
    );
  }

}
```

4. **Código de la página de registro para el usuario del alumno**

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">
                  <img
                    src="assets/Imagen/Logo_Csas.jpg"
                    style="width: 185px;"
                    alt="logo"
                  />
                  <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>
</section>
```

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;
}
```

Typescript

```
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',
  styleUrl: './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;
 }

}
```

5. **Código de la página para ingresar datos**

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>

  <!-- 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 class="form-check">
              <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>
        </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>

      <!--Aqui 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>
```

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: #FFFFFF;

  &:hover {
    box-shadow: 0 0 15px #2980B9;
    background-color: #1ABC9C;
    color: #FFFFFF;
    font-weight: bold;
  }
}
```

Typescript

```
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.');
  }
}
```

```
  // Método para navegar a la página anterior
  previousPage() {
    if (this.currentPage > 0) {
      this.currentPage--;
    }
  }

  getTotalPages() {
    return Math.ceil((this.questions.length + 1) / this.itemsPerPage);
  }

  onSubmit() {
    if (this.cuestionarioForm.valid) {
      const formData = this.cuestionarioForm.value;
      console.log('Formulario válido:', formData);
    } else {
      this.cuestionarioForm.markAllAsTouched();
    }
  }
}
```

6. Código de la página para visual de la encuesta

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)">Siguiente</a>
          </li>
        </ul>
      </nav>
      <br>
    </div>
  </div>
</div>
```

SCSS

```
// Estilos de tipografía y colores
@import
url('https://fonts.googleapis.com/css2?family=Merriweather:ital,wght@0,300;0,400;0,70
0;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
*/
}
```

Typescript

```
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
  }
}
```

7. **Código de la página principal para Administrador (Dashboard)**

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>

</div>
```

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: #a8dadc;
   }
  }

  /* 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 */
 }
}
```

Typescript

```
import { Component } from '@angular/core';

@Component({
 selector: 'app-dash-admin',
 standalone: true,
 imports: [],
 templateUrl: './dash-admin.component.html',
 styleUrl: './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
  }

}
```

8. **Códigos extra sobre mas diseños (Menús, Servicies y modelos (lógicos))**

HTML (Diseño del encabezado de la página)

```
<!--El titulo de encabezado-->
<nav class="navbar navbar-expand-lg navbar-dark custom-navbar">
  <div class="container">
    <a class="navbar-brand" href="#">
      <!-- Logo -->
      <img src="../../../../assets/Imagen/Logo_Csas.png" alt="Logo" width="65" height="65" class="d-inline-block align-top">

      <!-- 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>
```

SCSS

```
// Importaciones de las tipografías 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 */

}


// 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 */
}
```

Modelos

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.service.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();
  }
}
```

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 estudiante
  registerStudent(student: CreateStudent): Observable<ResponsePPD> {
    return this.http.post<ResponsePPD>(`${this.apiUrl}/Alumnos`, student);
  }

  // Registrar administrador
  registerAdmin(admin: CreateAdmin): Observable<ResponsePPD> {
    return this.http.post<ResponsePPD>(`${this.apiUrl}/Administrador`, admin);
  }

  // Obtener todos los administradores
  getAdmins(): Observable<ResponseGet<getAdmis[]>> {
    return this.http.get<ResponseGet<getAdmis[]>>(`${this.apiUrl}/Administrador`);
  }

  // Obtener todos los estudiantes
```

```
  getStudents(): Observable<ResponseGet<getStudents[]>> {
    return this.http.get<ResponseGet<getStudents[]>>(`${this.apiUrl}/Alumnos`);
  }

  // Actualizar administrador
  updateAdmin(id: number, admin: UpdateAdmi): Observable<ResponsePPD> {
    return this.http.put<ResponsePPD>(`${this.apiUrl}/Administrador/${id}`, admin);
  }

  // Obtener todos los usuarios (general)
  getAllUsers(): Observable<ResponseGet<CreateUserBase[]>> {
    return this.http.get<ResponseGet<CreateUserBase[]>>(`${this.apiUrl}/Usuarios`);
  }
}
```

BACKEND (Servidor, Base De Datos, Lógica).

1. **Código de los controladores para el servidor**

AdministradorController.php

```
<?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);
    }
}
```

AlumnoController.php

```
<?php

namespace App\Http\Controllers;

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

class alumnoController extends Controller{
    public function indexAlumno()
    {
        $alumnos = Alumno::all();

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

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

    public function store (Request $request)
    {
```

```
$validator = validator::make($request->all(),[
    'matricula'=>'required',
    'nombres'=>'required',
    'primer_apellido'=>'required',
    'segundo_apellido'=>'required',
    'Institucion_idInstitucion'=>'required',
    'Carreras_idCarrera'=>'required',
    'Generaciones_idGeneracion'=>'required',
    'Grupo_idGrupo'=>'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);
}
$alumnos = Alumno::create([
    'matricula'=> $request -> matricula,
    'nombres'=> $request -> nombres,
    'primer_apellido'=> $request -> primer_apellido,
    'segundo_apellido'=> $request -> segundo_apellido,
    'Institucion_idInstitucion'=> $request -> Institucion_idInstitucion,
    'Carreras_idCarrera'=> $request -> Carreras_idCarrera,
    'Generaciones_idGeneracion'=> $request -> Generaciones_idGeneracio,
    'Grupo_idGrupo'=> $request -> Grupo_idGrupo,
    'genero'=> $request -> genero,
```

```
        'edad'=> $request -> edad
    ]);

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

    $data =[
        'message' => $alumnos,
        'status' => 2001
    ];
    return response()->json($data, 201);
}
public function show($matricula){

    $alumnos = Alumno::where('matricula', $matricula)->first();

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

    $data = [
        'alumno' => $alumnos,
```

```
        'status' => 200
    ];

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

    public function destroy($matricula)
    {
        $alumnos = Alumno::where('matricula', $matricula)->first();

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

        $alumnos->delete();

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

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

    }

public function update(Request $request, $matricula){
    $alumnos = Alumno::where('matricula', $matricula)->first();
```

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

$validator = validator::make($request->all(),[
    'matricula'=>'required',
    'nombres'=>'required',
    'primer_apellido'=>'required',
    'segundo_apellido'=>'required',
    'Institucion_idInstitucion'=>'required',
    'Carreras_idCarrera'=>'required',
    'Generaciones_idGeneracion'=>'required',
    'Grupo_idGrupo'=>'required',
    'genero'=>'required',
    'edad'=>'required'
]);

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

$alumnos-> matricula = $request -> matricula;
```

```
        $alumnos-> nombres =  $request -> nombres;
        $alumnos-> primer_apellido = $request-> primer_apellido;
        $alumnos-> segundo_apellido = $request -> segundo_apellido;
        $alumnos-> Institucion_idInstitucion = $request -> Institucion_idInstitucion;
        $alumnos-> Carreras_idCarrera = $request -> Carreras_idCarrera;
        $alumnos-> Generaciones_idGeneracion = $request -> Generaciones_idGeneracio;
        $alumnos-> Grupo_idGrupo = $request -> Grupo_idGrupo;
        $alumnos-> genero = $request -> genero;
        $alumnos-> edad = $request -> edad;

        $alumnos ->save();

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

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

CarreraController.php

```
<?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);
    }
}
```

CategoriaController.php

```
<?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);
```

```
    }
}
```

EncuestasController.php

```
<?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);
    }
}
```

GeneracionController.php

```
<?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);
  }
}
```

GrupoController.php

```
<?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);
    }
}
```

InstitucionController.php

```
<?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);
    }
}
```

OpcionesController.php

```
<?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);
}
```

```
}
```

PreguntaController.php

```
<?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);
    }
}
```

RespuestasController.php

```
<?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);
    }
}
```

UsuarioController.php

```
<?php

namespace App\Http\Controllers;

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

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(), [
            'matricula' => 'required|integer|unique:usuario,matricula',
            'nombres' => 'required|string|max:45',
            'primer_apellido' => 'required|string|max:45',
            'segundo_apellido' => 'nullable|string|max:45',
```

```
            'correo' => 'required|email|unique:usuario,correo',
            'password' => 'required|string|min:8', // Validación de contraseña más segura
            'tipo' => 'required|in:Alumno,Administrador',
            'Alumno_Matricula' => 'nullable|integer',
            'Administrador_idAdministrador' => 'nullable|integer',
            'genero' => 'nullable|string|max:45',
            'edad' => 'required|integer|min:1'
        ]);

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

        $usuarios = Usuario::create($request->all());

        return response()->json([
            'message' => 'Usuario creado exitosamente',
            'usuario' => $usuarios,
            'status' => 201
        ], 201);
    }

    public function show($idUsuario)
    {
        $usuarios = Usuario::find($idUsuario);

        if (!$usuarios) {
```

```
        return response()->json([
            'message' => 'Usuario no encontrado',
            'status' => 404
        ], 404);
    }

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

public function destroy($idUsuario)
{
    $usuarios = Usuario::find($idUsuario);

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

    $usuarios->delete();

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

```
public function update(Request $request, $idUsuario)
{
    $usuarios = Usuario::find($idUsuario);

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

    $validator = Validator::make($request->all(), [
       'matricula' => 'required|integer|unique:usuario,matricula,' . $idUsuario . ',idUsuario',
        'nombres' => 'required|string|max:45',
        'primer_apellido' => 'required|string|max:45',
        'segundo_apellido' => 'nullable|string|max:45',
        'correo' => 'required|email|unique:usuario,correo,' . $idUsuario . ',idUsuario',
        'password' => 'nullable|string|min:8', // Solo actualizar si se envía
        'tipo' => 'required|in:Alumno,Administrador',
        'Alumno_Matricula' => 'nullable|integer',
        'Administrador_idAdministrador' => 'nullable|integer',
        'genero' => 'nullable|string|max:45',
        'edad' => 'required|integer|min:1'
    ]);

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

```
        }

        // Actualizar solo los campos necesarios
        $usuarios->update($request->all());

        return response()->json([
            'message' => 'Usuario actualizado exitosamente',
            'usuario' => $usuarios,
            'status' => 200
        ], 200);
    }
}
```

VariableController.php

```
<?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);
    }
}
```

2. **Código de los modelos para la lógica de la Base de datos.**

Administrador.php

```
<?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');
    }
}
```

Alumno.php

```
<?php

namespace App\Models;

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

class Alumno extends Model{

    use HasFactory;

    protected $table = 'alumno';

    protected $primaryKey= 'matricula';

    protected $fillable =[
        'matricula',
        'nombres',
        'primer_apellido',
        'segundo_apellido',
        'Institucion_idInstitucion',
        'Carreras_idCarrera',
        'Generaciones_idGeneracion',
        'Grupo_idGrupo',
        'genero',
        'edad'
    ];

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

```
     */
    public function institucion()
    {
        return $this->belongsTo(Institucion::class, 'Institucion_idInstitucion', 'idInstitucion');
    }

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

    /**
     * Relación con el modelo Generacion.
     */
    public function generacion()
    {
                return $this->belongsTo(Generacion::class, 'Generaciones_idGeneracion',
'idGeneracion');
    }

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

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

Carrera.php

```
<?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');
    }
}
```

Categoria.php

```
<?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'
    ];

}
```

Encuestas.php

```
<?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');
    }
}
```

Generacion.php

```
<?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');
    }
}
```

Grupo.php

```
<?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');
        }
    }
```

Institucion.php

```
<?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');
    }
}
```

Opciones.php

```
<?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');
    }
}
```

Pregunta.php

```
<?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');
    }
}
```

Respuestas.php

```
<?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');
    }
}
```

Usuario.php

```
<?php

namespace App\Models;

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

class Usuario extends Model
{
    use HasFactory;

    protected $table = 'usuario';

    protected $primaryKey = 'idUsuario';

    protected $fillable = [
        'idUsuario',
        'matricula',
        'nombres',
```

```
        'primer_apellido',
        'segundo_apellido',
        'correo',
        'password', // Se usará en el código, pero mapeará a "contraseña"
        'tipo',
        'Alumno_Matricula',
        'Administrador_idAdministrador',
        'genero',
        'edad'
    ];

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

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

    /**
     * Accessor para obtener el valor de la columna "contraseña" usando "password".
     */
```

```
    public function getPasswordAttribute()
    {
        return $this->attributes['contraseña'];
    }

    /**
     * Mutator para establecer el valor en la columna "contraseña" usando "password".
     */
    public function setPasswordAttribute($value)
    {
        // Encriptar la contraseña antes de almacenarla
        $this->attributes['contraseña'] = bcrypt($value);
    }
}
```

Variable.php

```
<?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');
    }
}
```

3. **Código de creación de tablas y lógica de la base de datos.**

Usuario_table.php

```
<?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'); // Clave primaria y 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); // Cambiado a 'password' para consistencia con el
modelo
            $table->enum('tipo', ['Alumno', 'Administrador']);
            $table->unsignedBigInteger('Alumno_Matricula')->nullable();
            $table->unsignedBigInteger('Administrador_idAdministrador')->nullable();
            $table->string('genero', 45);
            $table->integer('edad');
            $table->timestamps();
        });
    }

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

Alumno_table.php

```
<?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');
    }
};
```

Administrador_table.php

```
<?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');
    }
};
```

Encuestas_table.php

```
<?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');
    }
};
```

Respuestas_table.php

```
<?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');
    }
};
```

Opciones_table.php

```
<?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');
    }
};
```

Pregunta_table.php

```
<?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');
    }
};
```

Variable_table.php

```
<?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');
    }
};
```

Categoria_table.php

```
<?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');
    }
};
```

Institucion_table.php

```
<?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');
    }
};
```

Carrera_table.php

```
<?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');
    }
};
```

Grupo_table.php

```
<?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');
    }
};
```

Generacion_table.php

```
<?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_');
    }
};
```