PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC

DESARROLLO DE UN SISTEMA DE GESTIÓN ESCOLAR PARA LA UNIDAD EDUCATIVA JESÚS DE NAZARETH BASADO EN MONITOREO DE ASISTENCIAS Y COMPORTAMIENTO ESTUDIANTIL

Proyecto de titulación previo a la obtención del título de: Tecnólogo Superior en Desarrollo de Software

Autores:

Umatambo Llumiquinga Victor Ariel

Vinueza Chiluisa Cristian Stalin

Acuña Estrada Adrian Samuel

Tutor: Ing. Chiliquinga Tutachá Andrés Mauricio

Quito, Ecuador

2025

Índice

Índice de Tablas

Índice de Figuras

DECLARACIÓN y AUTORIZACIÓN

Yo, **UMATAMBO LLUMIQUINGA VICTOR ARIEL** con C.I. 1726727546 autor(a) del trabajo de Integración Curricular intitulado: **"SISTEMA DE GESTIÓN ESCOLAR PARA LA SUPERVISIÓN DE ASISTENCIAS, SEGURIDAD Y COMPORTAMIENTO EN UNIDAD EDUCATIVA JESÚS DE NAZARETH"**, previa a la obtención del título de **Tecnólogo Superior en Desarrollo de Software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, agosto 2025

UMATAMBO LLUMIQUINGA VICTOR ARIEL

C.I. 1726727546

DECLARACIÓN y AUTORIZACIÓN

Yo, **VINUEZA CHILUISA CRISTIAN STALIN** con C.I. 1723551352 autor del trabajo de Integración Curricular intitulado: **"SISTEMA DE GESTIÓN ESCOLAR PARA LA SUPERVISIÓN DE ASISTENCIAS, SEGURIDAD Y COMPORTAMIENTO EN UNIDAD EDUCATIVA JESÚS DE NAZARETH"**, previa a la obtención del título de **Tecnólogo Superior en Desarrollo de Software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, agosto 2025

VINUEZA CHILUISA CRISTIAN STALIN

C.I. 1723551352

DECLARACIÓN y AUTORIZACIÓN

Yo, **ACUÑA ESTRADA ADRIAN SAMUEL** con C.I. 1726621871 autor del trabajo de Integración Curricular intitulado: **"SISTEMA DE GESTIÓN ESCOLAR PARA LA SUPERVISIÓN DE ASISTENCIAS, SEGURIDAD Y COMPORTAMIENTO EN UNIDAD EDUCATIVA JESÚS DE NAZARETH"**, previa a la obtención del título de **Tecnólogo Superior en Desarrollo de Software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, agosto 2025

ACUÑA ESTRADA ADRIAN SAMUEL

C.I. 1726621871

Agradecimientos

Agradezco a Dios, por la vida y la salud. A mi familia por ser un pilar fundamental, por su apoyo en cada etapa de mi formación.

Cristian Stalin Vinueza

Agradecimientos

Agradezco a Dios por la vida, por darme la fuerza y salud para enfrentar cualquier obstáculo día a día.

A mi familia, especialmente a mis padres, Victor Umatambo y Viviana Llumiquinga, por su amor y apoyo incondicional que siempre me han brindado, a mis hermanos por su compañía y sus palabras de aliento, a mis abuelos por su cariño y presencia, les agradezco por estar conmigo en esta parte tan importante de mi vida.

A mis amigos y compañeros de tesis, por su arduo trabajo y compromiso dedicados a este trabajo, por los momentos de alegría y también de estrés vividos y compartidos durante la carrera.

A mi pareja, quien estuvo conmigo en los momentos difíciles con palabras de aliento. Gracias por tu total apoyo, tu serenidad y por creer en mí incluso cuando yo no lo hacía.

A nuestro tutor de tesis, el Ing. Andrés Mauricio Chiliquinga Tutachá, por compartir su conocimiento y ser una voz guía para culminar de la mejor manera el desarrollo de nuestro trabajo.

Finalmente, quiero agradecer a la Mgtr. Diana Esmeralda Molina Calvopiña, coordinadora de la tecnología en Desarrollo de Software, por ser más que una figura de autoridad, una consejera y amiga.

Victor Ariel Umatambo Llumiquinga

Agradecimientos

Agradezco a mi familia, quienes han sido un pilar esencial a lo largo de mi formación. En especial, a mi madre, Magali Estrada, por su amor incondicional, su apoyo constante y por ser mi mayor fuente de motivación en cada paso de este camino.

A mis amigos, por estar presentes en los momentos más exigentes, brindándome su compañía, ánimo y comprensión en las jornadas de esfuerzo, estrés y cansancio. Su apoyo ha sido invaluable.

A mi pareja, Andrea Aguilar, por su paciencia y por acompañarme con cariño, aliento y dedicación a lo largo de esta etapa de mi vida.

Adrian Samuel Acuña Estrada

Introducción

La implementación de un sistema integral para el control de asistencia y la seguridad de estudiantes responde a la necesidad de garantizar una comunicación eficaz entre las instituciones educativas y los padres de familia, así como de fortalecer los protocolos de seguridad escolar. Este proyecto está orientado a optimizar los procesos de registro de asistencia, notificación de inasistencias y autorización de retiros, contribuyendo a la tranquilidad de los padres y al control administrativo de las instituciones.

La importancia de este sistema radica en que aborda problemáticas comunes como el desconocimiento oportuno de inasistencias, el riesgo de retiros no autorizados y la falta de trazabilidad en la gestión escolar. En el campo técnico y tecnológico, este proyecto destaca por integrar herramientas de desarrollo de software, gestión de datos y automatización, mejorando la eficiencia de los procesos escolares mediante el uso de tecnologías accesibles y modernas.

El sistema utiliza técnicas como la generación y lectura de códigos QR para garantizar un control seguro y ágil. Además, se apoya en bases de datos para el registro y la trazabilidad de la información, interfaces web y móviles para una experiencia de usuario intuitiva, y notificaciones automáticas para mantener a los padres informados en tiempo real. Estas tecnologías aseguran la fiabilidad, accesibilidad y escalabilidad del sistema, ofreciendo una solución innovadora a las demandas actuales de las instituciones educativas.

Antecedentes

La necesidad de mejorar los procesos de control escolar ha impulsado el desarrollo de soluciones tecnológicas que permitan automatizar tareas y centralizar la información en las instituciones educativas. En muchas unidades educativas, los registros de asistencia, comportamiento y autorización de retiros continúan realizándose de manera manual, lo que genera ineficiencia, errores y falta de control. La Unidad Educativa Jesús de Nazareth no cuenta actualmente con un sistema digital que permita gestionar de forma integrada estos aspectos críticos, lo que la coloca en una situación similar a la de muchas instituciones que aún dependen de procesos tradicionales. Esta realidad evidencia la urgencia de

implementar herramientas tecnológicas modernas que respondan a las necesidades reales del entorno escolar.

Planteamiento del Problema

En la Unidad Educativa Jesús de Nazareth se ha identificado un problema recurrente relacionado con la seguridad en el retiro de estudiantes y la gestión de asistencias. En múltiples ocasiones, familiares no autorizados por el representante legal intentan retirar a los estudiantes, generando situaciones de riesgo y conflictos administrativos. Este problema afecta la confianza de los padres hacia la institución y pone en peligro la integridad de los estudiantes.

Además, el proceso de notificación de inasistencias presenta deficiencias que complican la comunicación oportuna entre la institución y los padres. Cuando un estudiante falta reiteradamente, los padres son informados de manera tardía y se ven obligados a realizar un extenso papeleo para justificar las ausencias, lo que resulta tedioso y poco práctico tanto para ellos como para el personal administrativo.

Este problema técnico-tecnológico se inserta en un contexto sociocultural donde los padres buscan mayor seguridad y control en la educación de sus hijos. Asimismo, la situación socioeconómica de muchas familias exige soluciones que sean accesibles y eficientes, evitando gastos y trámites innecesarios.

El proyecto plantea una solución innovadora basada en tecnología que permitirá implementar un sistema automatizado para la autorización de retiros y la notificación inmediata de inasistencias, conflictos u observaciones relevantes. Esto no solo optimizará los procesos administrativos, sino que también fortalecerá la confianza de los padres hacia la institución, garantizando un entorno educativo más seguro y organizado.

Justificación

El desarrollo de un sistema integral que gestione asistencias, retiros y observaciones estudiantiles se justifica por la necesidad urgente de mejorar la eficiencia administrativa, la seguridad institucional y la comunicación entre la institución y los representantes legales.

Actualmente, las instituciones educativas enfrentan diversos desafíos para garantizar la

trazabilidad de las asistencias y los permisos de salida de los estudiantes. La falta de herramientas tecnológicas impide realizar una supervisión efectiva y en tiempo real.

Este sistema brindará una solución concreta a esas problemáticas, permitiendo el registro centralizado de eventos relevantes, la generación de códigos QR únicos para retiros autorizados y el envío automático de notificaciones a los padres. Con ello, se fortalecerá la confianza hacia la institución y se fomentará una cultura de transparencia y prevención, sin añadir complicaciones para los usuarios.

Objetivo General

Desarrollar un sistema que garantice la seguridad y optimice la gestión de recurrencias del Estudiante en la Unidad Educativa Jesús de Nazareth mediante el uso de herramientas tecnológicas, promoviendo la confianza de los padres y mejorando los procesos administrativos escolares.

Objetivos Específicos

Identificar y analizar los requerimientos técnicos y administrativos necesarios para desarrollar un sistema eficiente de autorización de retiros y control de recurrencias, adaptado a las necesidades específicas de la institución educativa.

Desarrollar un módulo de autorización de retiros estudiantiles que incorpore códigos QR, garantizando la seguridad y confiabilidad de las autorizaciones.

Desarrollar un módulo para el registro e informe de asistencias, conflictos u observaciones de estudiantes, con notificaciones y formularios de justificación para padres.

Metodología Desarrollo de Software Scrum

El presente proyecto adopta un enfoque aplicado, orientado a la resolución práctica de problemas técnicos y tecnológicos detectados en la Unidad Educativa Jesús de Nazareth. Este tipo de estudio combina elementos exploratorios para comprender el entorno y las necesidades específicas de la institución, y un enfoque aplicado para desarrollar una solución tecnológica que cumpla con los requerimientos identificados.

Se emplearán entrevistas y encuestas estructuradas con directivos, docentes y representantes legales para recopilar datos cualitativos y cuantitativos sobre los desafíos relacionados con el control de

asistencias, autorizaciones de retiro y el seguimiento del comportamiento estudiantil con el objetivo de levantar los requisitos del sistema.

El desarrollo del sistema se llevará a cabo siguiendo las mejores prácticas de la ingeniería de software y metodologías ágiles como SCRUM. Este enfoque permitirá una colaboración activa entre el equipo técnico y los usuarios finales, asegurando que el sistema cumpla con las expectativas funcionales y de usabilidad.

Marco Conceptual

Diseño Front-End: Utilizando React, se implementará una interfaz de usuario dinámica y modular, garantizando una experiencia intuitiva y centrada en las necesidades de los usuarios. Los prototipos iniciales serán desarrollados en Figma para obtener retroalimentación temprana.

Desarrollo Back-End: Node.js será utilizado para gestionar la lógica del servidor, permitiendo la integración en tiempo real de módulos clave como el control de asistencias, notificaciones automáticas y la autorización de retiros.

Gestión de Datos: Se combinarán PostgreSQL y MongoDB para el manejo de datos. PostgreSQL será utilizado para datos relacionales críticos, como registros de estudiantes y asistencias, mientras que MongoDB permitirá almacenar datos no estructurados y semiestructurados, como historiales de observaciones, reportes de conflictos y eventos. Este enfoque híbrido garantizará flexibilidad y robustez en el sistema.

Seguridad del Sistema: Para reforzar la seguridad, se implementarán códigos QR únicos en los procesos de autorización de retiros, complementados con autenticación de usuarios y registros de actividad.

Notificaciones Automatizadas: A través de servicios de notificaciones, se configurarán alertas automáticas que notifiquen a los padres sobre asistencias, conflictos y observaciones relevantes. El sistema será sometido a pruebas funcionales y de usuario final en un entorno controlado, ajustando los módulos según los comentarios recibidos. Esta metodología garantizará que el sistema desarrollado no

solo resuelva el problema identificado, sino que también sea una herramienta tecnológica sostenible y escalable, adaptada a las necesidades futuras de la institución.

Capítulo I

Marco Teórico

En la actualidad, la transformación digital en el ámbito educativo se ha convertido en un componente clave para mejorar la gestión institucional, la seguridad estudiantil y la comunicación con los representantes legales. Diversos estudios demuestran que la incorporación de tecnologías digitales en instituciones educativas permite automatizar procesos críticos, como el control de asistencia, la supervisión del comportamiento y la autorización de retiros, generando beneficios tanto para el personal administrativo como para las familias.

En el caso ecuatoriano, el Ministerio de Educación ha impulsado iniciativas como el *Sistema de Gestión Escolar* (SGE), una plataforma nacional destinada a registrar información académica, asistencia, matrícula y otros datos relevantes de los estudiantes de instituciones fiscales. Sin embargo, su alcance está orientado principalmente a tareas administrativas generales, sin integrar funcionalidades específicas para el control en tiempo real de asistencias o mecanismos de seguridad como el uso de códigos QR para autorizar retiros escolares (Ministerio de Educación, 2023).

Por su parte, algunas instituciones particulares y fiscomisionales han adoptado soluciones tecnológicas propias. Un ejemplo destacado es la **Unidad Educativa Rincón Del Saber (UERS)**, ubicada en Chillogallo, la cual utiliza el sistema **IDUKAY**, una plataforma de gestión escolar que permite registrar notas, asistencias y realizar comunicaciones entre docentes y padres de familia a través de una aplicación móvil. No obstante, si bien IDUKAY ofrece herramientas académicas completas, su funcionalidad en términos de control de seguridad presencial o autorización digital de retiros aún es limitada, lo que evidencia que existe margen para propuestas más integrales y personalizadas.

En este contexto, el desarrollo de un sistema integral que permita supervisar asistencias, autorizar retiros mediante códigos QR y registrar reportes de comportamiento representa una solución alineada con las necesidades actuales del entorno educativo. Dicho sistema no solo responde a una demanda de

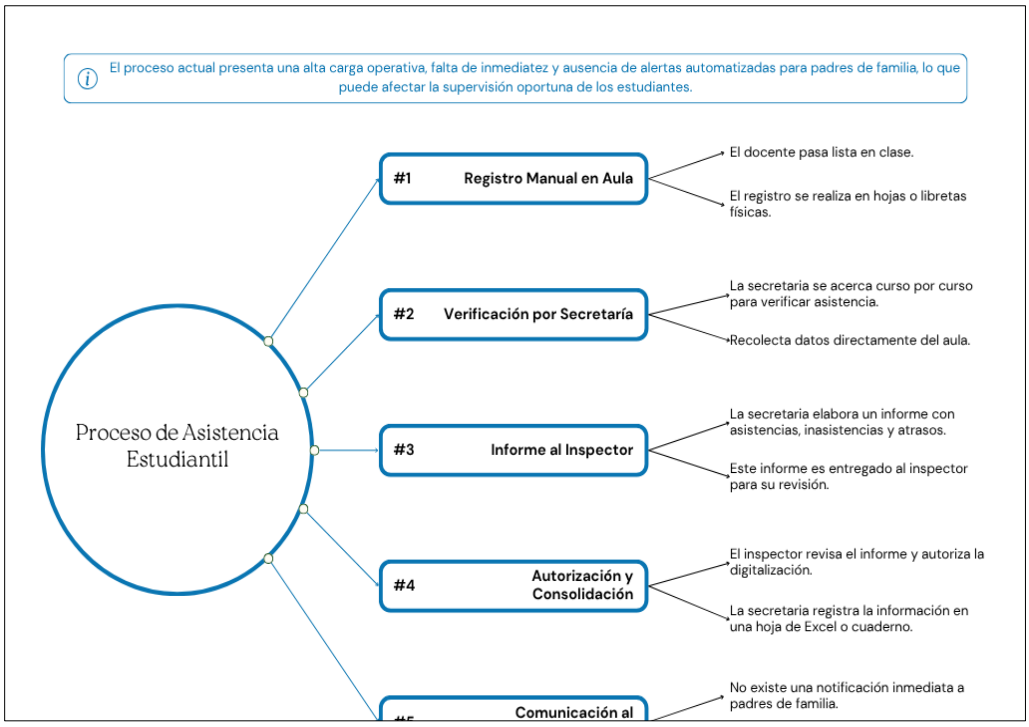
modernización tecnológica, sino que también contribuye a fortalecer la seguridad institucional y la comunicación efectiva con los padres de familia.

Procesos actuales.

La siguiente sección expone los procesos manuales actuales en la Unidad Educativa Jesús de Nazareth, los cuales carecen de automatización, trazabilidad y comunicación inmediata. Se evidencia la necesidad de modernizarlos para mejorar la eficiencia, la seguridad y el vínculo con las familias.

Figura 1

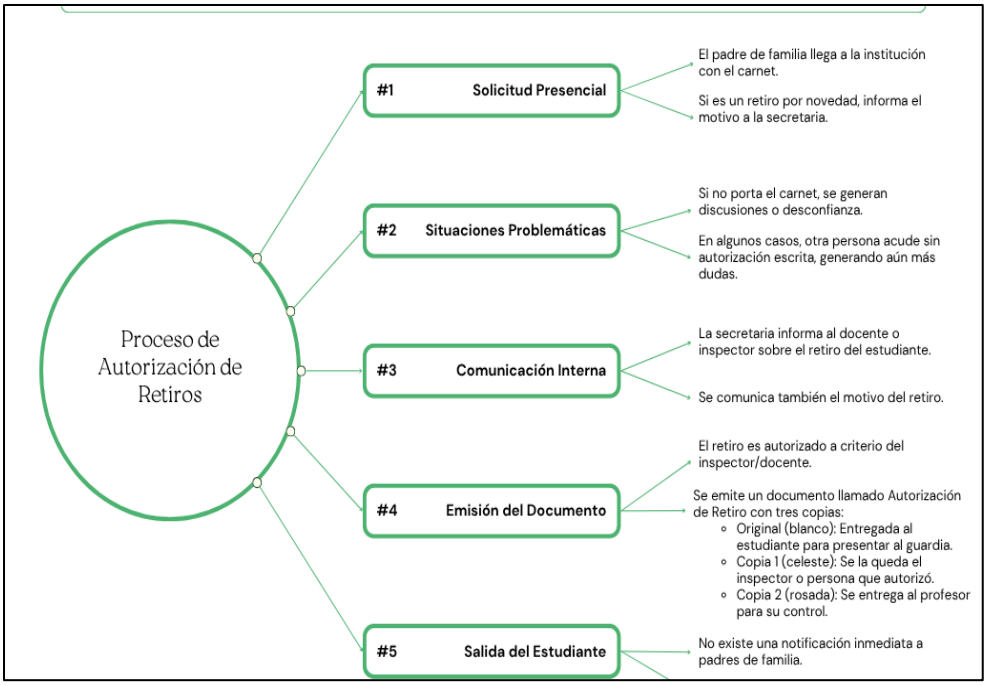
Proceso manual de asistencia estudiantil en la U.E. Jesús de Nazareth



Nota: Procesos actuales de la toma de asistencia de los estudiantes.

Figura 2

Proceso manual de autorización de retiros en la U.E. Jesús de Nazareth.



Nota: Procesos actuales de los retiros de los estudiantes.

Figura 3

Proceso manual de reportes de comportamiento e incidentes en la U.E. Jesús de Nazareth.

Nota: Procesos actuales de los registros comportamentales de los estudiantes.

Infraestructura actual.

A continuación, se detalla los recursos tecnológicos disponibles en la Unidad Educativa Jesús de Nazareth que serán utilizados para la implementación del sistema propuesto. Se presentan los equipos informáticos existentes en cada área institucional, así como el acceso a dispositivos móviles por parte de usuarios clave como docentes, personal administrativo, seguridad y representantes legales.

Tabla 1

Infraestructura Tecnológica Actual

Área	Cantidad	Dispositivo	Características
Inspección General	1	PC HP (Escritorio)	Intel Core i9, 32 GB RAM, SSD 1 TB, Windows 10 Home
Secretaría	3	PC HP (Escritorio)	Intel Core i5, 8 GB RAM, HDD 1 TB, Windows 10 Home
Aulas (Profesores)	1 por aula	PC HP (Escritorio)	Intel Core i5, 8 GB RAM, HDD 1 TB, Windows 10 Home
Seguridad	1	PC HP (Escritorio)	Intel Core i3, 4 GB RAM, HDD 500 GB, Windows 10 Home
Todos (incl. padres de familia, profesores, seguridad, etc.)	Variable	Dispositivo móvil Android	Diferentes modelos y gamas. Permiten uso de apps móviles y recepción de notificaciones.

Nota: Esta tabla contiene la infraestructura tecnológica actual de la U.E. Jesús de Nazareth

Marco de trabajo Scrum

Para el desarrollo del Sistema de Gestión Escolar para la Unidad Educativa "Jesús de Nazareth", se ha seleccionado el marco de trabajo Scrum. Esta es una metodología ágil diseñada para gestionar el desarrollo de productos complejos de manera adaptativa. Ken Rubin (2013) lo describe como un marco de trabajo que "utiliza un enfoque iterativo e incremental para optimizar la previsibilidad y controlar el riesgo". Su flexibilidad es ideal para un proyecto de software educativo donde la retroalimentación de los usuarios finales (docentes y administrativos) es clave.

Puntos Clave del Marco Scrum

El éxito de Scrum en este proyecto se fundamentará en sus principios y características operativas:

Enfoque Empírico de Inspección y Adaptación: Scrum no se basa en un plan detallado e inamovible, sino en la experiencia y en la evidencia. El proceso se basa en frecuentes "ciclos de inspección y adaptación que permiten al equipo aprender y responder al cambio" (Rubin, 2013). Esto permitirá que el desarrollo del sistema escolar se ajuste continuamente a las necesidades reales de la institución.

Equipos Autoorganizados: En lugar de ser gestionados por una figura externa, los equipos Scrum tienen la autonomía para tomar decisiones sobre su propio trabajo. Según Mike Cohn (2010), "los equipos auto-organizados son más productivos y motivados porque se les da la autoridad para tomar decisiones locales sobre su trabajo". Esta autonomía fomenta la responsabilidad y la innovación en el equipo que desarrollará el sistema.

Entrega de Valor Incremental: El objetivo principal es entregar software funcional de manera regular. En lugar de esperar al final del proyecto, Scrum se centra en producir un "incremento de producto potencialmente entregable al final de cada Sprint" (Pressman & Maxim, 2020), asegurando que la Unidad Educativa "Jesús de Nazareth" reciba valor de forma temprana y continua.

Actores del Proceso (El Equipo Scrum)

Product Owner (Dueño del Producto): Actúa como la voz del cliente y de los interesados. Es la "única persona responsable de las decisiones sobre el producto y de gestionar el Product Backlog" (Rubin, 2013). Para este proyecto, definirá y priorizará las funcionalidades del sistema, como el módulo de asistencias, el registro de comportamiento y la generación de reportes para los docentes.

Scrum Master: Es un líder servicial y un coach ágil para el equipo. Su principal responsabilidad es "ayudar a todos a entender y aplicar Scrum de manera correcta y eliminar los impedimentos" (Cohn, 2010) que puedan obstaculizar el progreso del equipo de desarrollo.

Developers (El Equipo de Desarrollo): Son los profesionales que construyen el sistema. Este equipo posee todas las habilidades necesarias, como análisis, diseño, programación y pruebas, para "transformar los elementos del Product Backlog en un incremento de producto funcional en cada Sprint" (Rubin, 2013).

Ceremonias de Scrum

Scrum emplea una serie de reuniones planificadas o “ceremonias” que se repiten de forma regular. Estas reuniones ayudan a mantener un ritmo constante de trabajo y reducen la necesidad de encuentros improvisados. Además, permiten revisar el progreso y realizar los ajustes necesarios durante el desarrollo del proyecto.

Sprint: Es el ciclo central de trabajo en Scrum, con una duración corta y fija (por ejemplo, dos o tres semanas). Durante este tiempo, el equipo se enfoca en avanzar en la construcción de una parte funcional del producto.

Sprint Planning (Planificación del Sprint): Es una reunión que se realiza al inicio de cada ciclo de trabajo. En ella, todo el equipo colabora para definir un objetivo claro y decidir qué tareas se pueden completar durante ese período.

Daily Scrum (Scrum Diario): Una reunión diaria de 15 minutos para el Equipo de Desarrollo. Su propósito es "sincronizar el trabajo y planificar las próximas 24 horas" (Cohn, 2010), no está pensada para informar a la gerencia.

Sprint Review (Revisión del Sprint): Se realiza al final del Sprint para revisar el avance del producto con los interesados. Es una sesión práctica donde se "muestra lo que el equipo ha logrado y se recoge retroalimentación directa" (Cohn, 2010).

Sprint Retrospective (Retrospectiva del Sprint): Es la última reunión del ciclo. Es una oportunidad para que el equipo reflexione sobre su forma de trabajar e identifique "al menos una mejora procesal para implementar en el siguiente Sprint" (Rubin, 2013).

Sprints

Para el desarrollo del sistema se ha establecido un plan de trabajo estructurado en iteraciones (sprints), siguiendo la metodología ágil SCRUM. Cada sprint contempla tareas específicas

orientadas a cumplir con los objetivos del proyecto y dar solución progresiva a los problemas identificados durante el análisis de requisitos.

Tabla 2

Actividades Sprint I

Iteración I: Levantamiento de Requerimientos		
Duración 2 Semanas		
Tarea	**Responsable**	**Duración (tiempo)**
Planificación de la toma de requerimientos	Adrian Acuña, Victor Umatambo, Cristian Vinueza	2 días
Reunión con el equipo de desarrollo para definir el alcance	Adrian Acuña, Victor Umatambo, Cristian Vinueza	1 día
Visita a la Unidad Educativa Jesús de Nazareth	Victor Umatambo	1 día
Entrevista con autoridades de la Unidad Educativa	Victor Umatambo	2 días
Redacción de Historias de Usuario	Adrian Acuña, Victor Umatambo, Cristian Vinueza	3 días
Diseño de Base de datos	Adrian Acuña, Victor Umatambo, Cristian Vinueza	2 días
Creación de la Base de Datos (Modelo Conceptual y Físico)	Adrian Acuña, Victor Umatambo, Cristian Vinueza	2 días

Nota: La tabla muestra las tareas planificadas para la iteración I, correspondiente al levantamiento de requerimientos.

Tabla 3

Actividades Sprint II

Iteración II: Desarrollo del Sistema Web Backend		
Duración 2 Semanas		
Tarea	**Responsable**	**Duración (tiempo)**
Creación de Diccionario de Datos	Adrian Acuña, Victor Umatambo, Cristian Vinueza	1 día
Establecer el entorno de ejecución, repositorio en la nube y definir la arquitectura de desarrollo (MVC)	Adrian Acuña, Victor Umatambo, Cristian Vinueza	1 día
Implementar la conexión a la Base de Datos con Sequelize	Adrian Acuña	1 día

Desarrollo de Modelos para migración de Base de Datos	Adrian Acuña, Victor Umatambo, Cristian Vinueza	2 días
Desarrollo de Controladores para manejo de lógica de negocio	Adrian Acuña, Victor Umatambo, Cristian Vinueza	5 días
Desarrollo de Rutas para definir Endpoints	Adrian Acuña, Victor Umatambo, Cristian Vinueza	2 días
Levantamiento del servidor local y corrección de errores	Adrian Acuña, Victor Umatambo, Cristian Vinueza	1 día
Pruebas con Postman	Cristian Vinueza	1 día

Nota: La tabla muestra las tareas planificadas para la iteración II, correspondiente al desarrollo del backend del sistema web.

Tabla 4

Actividades Sprint III

Iteración III: Diseño y Desarrollo del Sistema Frontend		
Duración 2 Semanas		
Tarea	**Responsable**	**Duración (tiempo)**
Definición de Módulos del Sistema	Adrian Acuña, Victor Umatambo, Cristian Vinueza	1 día
Diseño de Vistas Web del Sistema	Adrian Acuña, Victor Umatambo, Cristian Vinueza	4 días
Establecer el entorno de ejecución, repositorio en la nube y arquitectura del Frontend	Adrian Acuña, Victor Umatambo, Cristian Vinueza	1 día
Creación de Login del Sistema con API de autenticación	Victor Umatambo	4 días
Construcción de navegación a partir de roles	Adrian Acuña, Victor Umatambo, Cristian Vinueza	3 días

Nota: La tabla muestra las tareas planificadas para la iteración III, correspondiente al diseño y desarrollo del sistema frontend.

Tabla 5

Actividades Sprint IV

Iteración IV: Desarrollo de Módulos del Sistema Frontend		
Duración 2 Semanas		
Tarea	**Responsable**	**Duración (tiempo)**
Creación Vista Administrativos	Victor Umatambo	3 días
Creación Vista Profesores	Cristian Vinueza	3 días
Creación Vista Representantes Legales	Adrian Acuña	2 días

Creación Vista Personal de Seguridad	Adrian Acuña	2 días
Protección de Rutas	Adrian Acuña, Victor Umatambo, Cristian Vinueza	2 días

Nota: La tabla muestra las tareas planificadas para la iteración IV, correspondiente al desarrollo de los módulos específicos del sistema frontend para cada tipo de usuario.

Tabla 6

Actividades Sprint V

Iteración V: Conexión a APIs Frontend		
Duración 2 Semanas		
Tarea	**Responsable**	**Duración (tiempo)**
Gestión de Usuarios, Cursos y Asignación de Usuarios a Cursos en Módulo Administrativo	Victor Umatambo	3 días
Gestión de Asistencias e Incidentes por Estudiante y Curso Módulo Profesores	Cristian Vinueza	3 días
Ver Asistencias e Incidentes de Representados y Generar QRs para autorización de salida de Estudiantes Módulo Representantes Legales	Adrian Acuña	3 días
Escaneo de QR y redirección hacia página de estudiante para autorización de salida Módulo Personal de Seguridad	Adrian Acuña	2 días
Integración de Módulos y Corrección de Conflictos	Adrian Acuña, Victor Umatambo, Cristian Vinueza	2 días

Nota: La tabla muestra las tareas planificadas para la iteración V, correspondiente a la conexión del frontend con las APIs del sistema y la integración de los distintos módulos funcionales.

Tabla 7

Actividades Sprint VI

Iteración VI: Pruebas y Features		
Duración 2 Semanas		
Tarea	**Responsable**	**Duración (tiempo)**
Implementación y pruebas de envío de credenciales por correo electrónico	Victor Umatambo	2 días
Pruebas de funcionamiento por módulos	Victor Umatambo	1 día

Pruebas de funcionamiento entre módulos	Cristian Vinueza	1 día
Pruebas de interfaz y experiencia de usuario	Cristian Vinueza	2 días
Verificación de Persistencia de Datos en la Base de Datos	Adrian Acuña	1 día
Pruebas de Login con Credenciales Válidas e Inválidas	Adrian Acuña	2 días
Validación de Navegación y Visibilidad por roles	Adrian Acuña, Victor Umatambo, Cristian Vinueza	1 día
Implementación y pruebas de Rutas Protegidas	Adrian Acuña, Victor Umatambo, Cristian Vinueza	3 días

Nota: La tabla muestra las tareas planificadas para la iteración VI, correspondiente a la validación funcional del sistema mediante pruebas por módulos, pruebas de seguridad, experiencia de usuario e implementación de características finales.

Capítulo II

Análisis y Diseño

La Unidad Educativa Jesús de Nazareth en la actualidad no posee un sistema que facilite el registro de atrasos, inasistencias y un registro de comportamiento estudiantil. Además, en la actualidad los retiros de los estudiantes a la hora de la salida no tienen un método de verificación por parte de las autoridades o personal administrativo si dicha persona está autorizada a retirar al estudiante.

Después de un análisis exhaustivo de estos procesos institucionales se identificaron varias oportunidades de mejora para facilitar los procesos administrativos, toma de asistencia, de registro de novedades comportamentales y de retiro del estudiante. Se identificó un retraso en la notificación al padre de familia sobre los atrasos, faltas, novedades y su posterior seguimiento ya que dichos procesos se realizan utilizando documentos físicos, lo cual genera demoras en el flujo de información para el representante legal.

Ante esta realidad se propone el desarrollo de un sistema de gestión escolar que optimice los procesos actuales, mediante el uso de herramientas tecnológicas que permitan el proceso de registro y visualización de información generada diariamente en el aula y al momento de retirar al estudiante.

Generando la transformación digital de los procesos que se realizan en formato físico, y permitiendo la trazabilidad de la información, mediante la asignación de roles, usuarios con los cuales se podrá identificar a los diferentes actores del sistema. Todos estos registros se almacenarán en una base de datos en PostgreSQL. Además, en el análisis de requerimientos se identificó la necesidad de integrar múltiples roles dentro del sistema, cada uno con funcionalidades específicas ayudaran al desempeño de sus responsabilidades institucionales. Por tal motivo el sistema se diseñará en módulos independientes, mismos que se podrán acceder mediante un inicio de sesión seguro usando JWT.

A continuación, se describen los roles y sus respectivos módulos:

Administrador: Tendrá el acceso completo a un módulo de gestión, desde el cual podrá crear, actualizar y eliminar registros de usuarios, profesores, cursos, estudiantes, representantes legales. Además, podrá consultar reportes generales sobre asistencias, incidentes y actividad del sistema.

Profesor: Tendrá su propio modulo en el cual podrá visualizar sus cursos y estudiantes asignados y podrá registrar asistencias, atrasos e incidentes disciplinarios. Este módulo estará desarrollado para funcionar desde equipos de escritorio o móviles, para facilitar el registro en tiempo real la información de la jornada escolar.

Representante Legal: Recibirán notificaciones de su representado y podrán generar códigos QR válidos para el retiro del estudiante. Cada código será de único uso y tendrá una validez máxima de 30 min. Este módulo será responsive ya que está planificado para ser usado en un dispositivo móvil.

Personal de seguridad: En este módulo se incluirá el escaneo del código QR para el retiro del estudiante. Este verificará y validará en tiempo real la autorización de retiro

Análisis de requisitos

La necesidad de implementar un sistema digital en la Unidad Educativa Jesús de Nazareth surge a partir de dos problemáticas evidentes en el funcionamiento diario de la institución.

La primera problemática se relaciona con el proceso de retiro de estudiantes. Actualmente, este procedimiento se realiza mediante un carné físico que debe portar el padre de familia o la persona autorizada. Sin embargo, en la práctica, se han presentado situaciones frecuentes donde el carné es olvidado, deteriorado o incluso no está en manos de quien realiza el retiro. Además, en muchas ocasiones, el padre de familia envía a un tercero sin previo aviso, lo que genera desconfianza por parte del personal institucional. Esta situación ha derivado en malentendidos y molestias, ya que algunos representantes asumen que el personal educativo conoce a todos los familiares del estudiante, lo cual no siempre es posible. Para solventar esta necesidad, se propone el uso de un código QR personalizado, que se genera desde la aplicación móvil. Este código puede ser escaneado fácilmente por el personal y compartido digitalmente con la persona autorizada, aprovechando que el dispositivo móvil es un objeto de uso constante.

La segunda problemática está relacionada con la gestión manual de información académica y administrativa, como el registro de asistencias, inasistencias, incidentes y comunicados oficiales. Actualmente, estos procesos se realizan en formato físico, lo que no solo ralentiza la gestión interna, sino

que además limita la comunicación con los padres de familia. Estos solo pueden acceder a dicha información acudiendo personalmente a la institución, lo cual resulta incómodo y poco práctico. Con la implementación de la aplicación móvil, los padres podrán realizar un seguimiento en tiempo real del comportamiento de sus hijos, recibir notificaciones instantáneas sobre ausencias o incidentes, y estar al tanto de los comunicados institucionales, mejorando la comunicación y la participación.

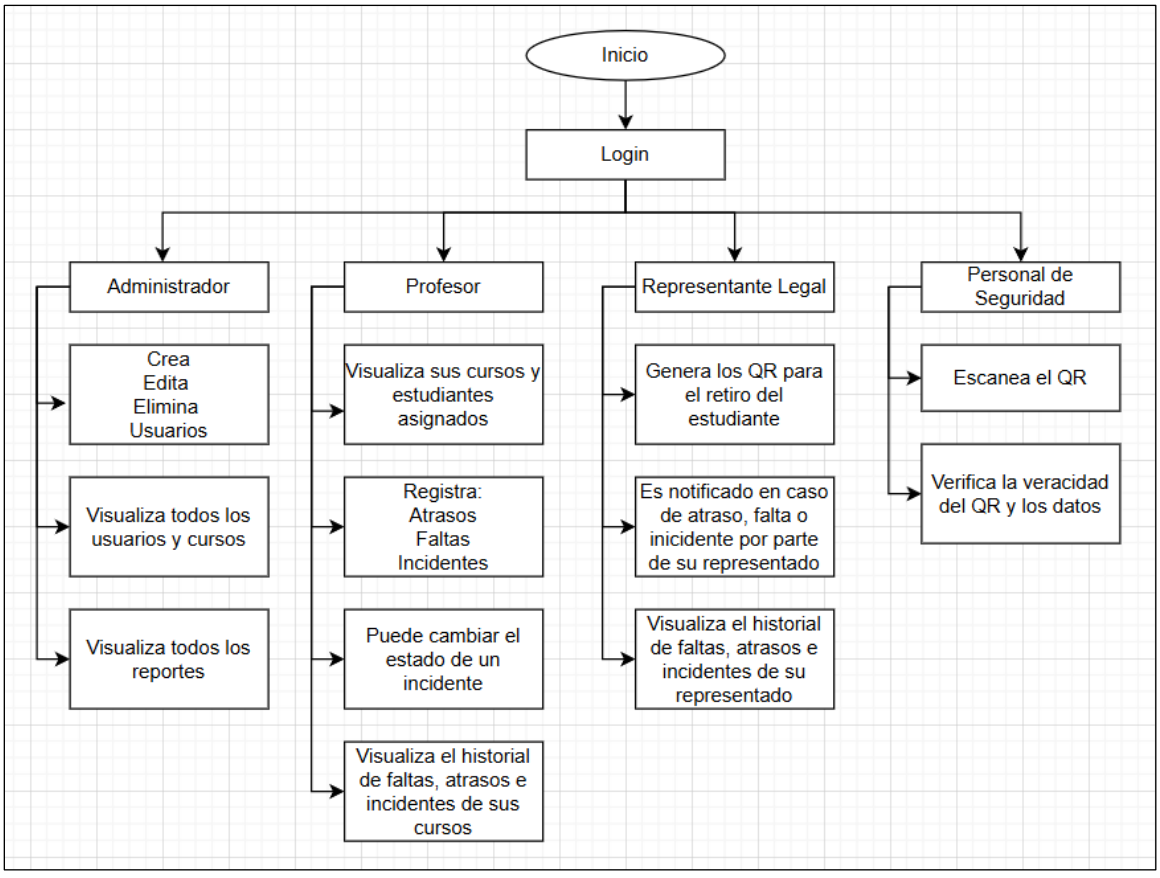
Mapa de navegación

Frontend

Para el desarrollo de la interfaz gráfica se tomaron en cuenta los roles y los módulos a implementar, dando como resultado el siguiente mapa de navegación.

Figura 4

Mapa de navegación del sistema



Nota: Proceso de logueo en sistema de acuerdo con el rol de usuario.

Diseño de Arquitectura

El diseño de la arquitectura de software es una fase que define la estructura, los componentes y las interacciones de un sistema para satisfacer los requisitos no funcionales. Una arquitectura bien diseñada garantiza que el sistema sea robusto, mantenible y escalable a lo largo del tiempo. Para el "Sistema de Gestión Escolar para la Unidad Educativa Jesús de Nazareth", se ha optado por una arquitectura que prioriza el modularidad y la separación de responsabilidades

Selección del Patrón Arquitectónico

El sistema se fundamenta en una arquitectura en capas (N-Tier), separando física y lógicamente la presentación, la lógica de negocio y la persistencia de datos. El objetivo principal es la separación de conceptos, donde cada componente tiene una responsabilidad única. Martin Fowler (2002), una autoridad en patrones de software destaca que esta separación permite que el modelo de datos y la lógica de negocio evolucionen independientemente de la interfaz de usuario. En el contexto de este proyecto:

Modelo: Representa la estructura de datos y la lógica de negocio. Se encarga de la interacción directa con la base de datos (PostgreSQL) para gestionar la información de estudiantes, cursos, asistencias e incidentes.

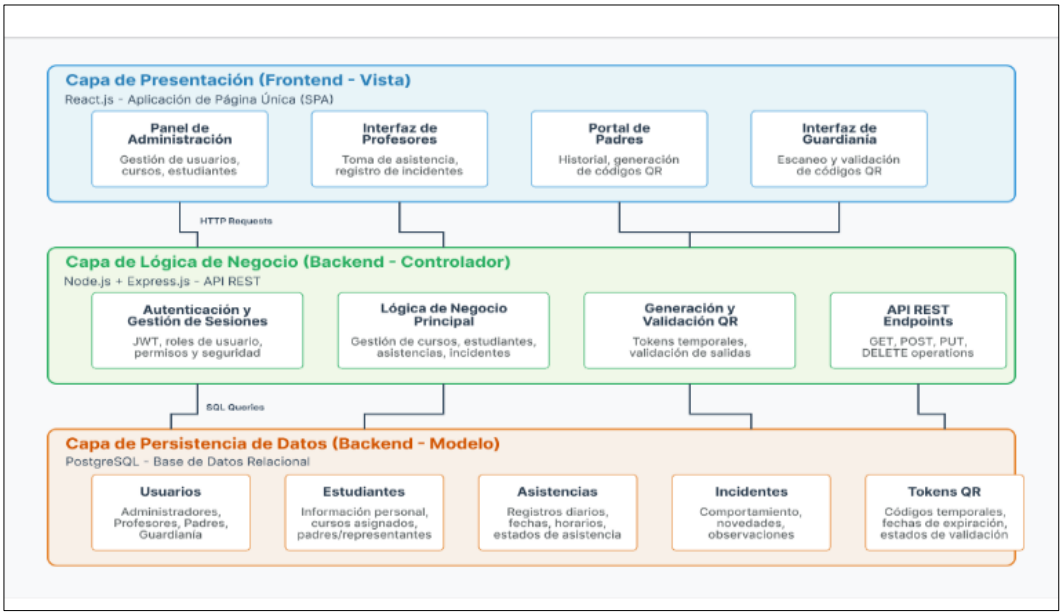
Vista: Es la interfaz de usuario con la que interactúan los distintos roles (administrativos, profesores, padres y guardianía). Se presenta en una aplicación web desarrollada con React, responsable de presentar los datos al usuario y capturar sus acciones.

Controlador: Se encarga de intermediar el Modelo y la Vista. Procesa las peticiones del usuario que vienen desde la Vista, ejecuta la lógica de negocio correspondiente interactuando con el Modelo y devuelve los resultados a la Vista para que sean mostrados. Esta función implementada en el backend con node.js y Express.

La organización de estos componentes y sus interacciones se detalla visualmente en el diagrama de arquitectura presentado en la Figura 2 *Diagrama de Arquitectura*

Figura 5

Diagrama de Arquitectura



Nota: Diagrama por capas datos, lógica y presentación.

Descripción de los Componentes

La arquitectura se compone de tres capas principales que operan de manera desacoplada:

Capa de Presentación (Frontend - Vista): Desarrollada como una Aplicación de Página Única (SPA) utilizando React.js. Esta capa es la única con la que los usuarios interactúan directamente. Es responsable de renderizar las interfaces gráficas para cada módulo: el panel de administración, la interfaz de los profesores para la toma de asistencia, el portal para padres donde pueden ver el historial de sus representados y generar los códigos QR, y la herramienta del personal de guardianía para el escaneo y validación de dichos códigos.

Capa de Lógica de Negocio (Backend - Controlador): Implementada en Node.js con el framework Express.js, esta capa constituye el cerebro del sistema. Su función es exponer una API REST (Representational State Transfer) que sirve como puente de comunicación entre el frontend y la base de datos. Una API REST es un estilo arquitectónico que utiliza los métodos estándar de HTTP (GET, POST, PUT, DELETE) para la comunicación, lo que la hace "simple y compatible con la infraestructura existente de Internet" (Richardson & Ruby, 2013). El backend se encarga de la autenticación de usuarios,

la gestión de sesiones, el procesamiento de toda la lógica de negocio (crear cursos, asignar estudiantes, registrar faltas, generar códigos QR) y la comunicación con la capa de datos.

Capa de Persistencia de Datos (Backend - Modelo): El almacenamiento de la información se gestiona a través de PostgreSQL, un sistema de gestión de bases de datos relacional de código abierto. Esta capa es responsable de garantizar la integridad, consistencia y seguridad de todos los datos críticos del sistema, incluyendo registros de usuarios, información de estudiantes, historiales de asistencia, novedades de comportamiento y los tokens para los códigos QR de salida.

Viabilidad Técnica

El estudio de viabilidad técnica es un análisis que determina si un proyecto puede ser desarrollado e implementado con los recursos tecnológicos disponibles en la institución. Su propósito es evaluar si la infraestructura existente es adecuada para el despliegue y funcionamiento óptimo del sistema.

Requisitos técnicos para el servidor On-Premise

Para garantizar el funcionamiento óptimo del sistema de gestión escolar en un entorno on-premise, el servidor debe cumplir con las siguientes especificaciones mínimas:

Procesador:

- Intel Core i3 de 8va generación o AMD Ryzen 3 equivalente
- Frecuencia mínima de 2.4 GHz
- Arquitectura: 64 bits

Memoria RAM

- Mínimo: 4GB DDR4
- Recomendado: 8GB DDR4
- Justificación: Node.js con Express.js requiere aproxiamdamente 1-2 GB, PostgreSQL necesita 1-2 GB, y el sistema operativo utiliza 2-3 GB

Almacenamiento

- Mínimo: 50 GB de espacio libre

- Recomendado: SSD para mejor rendimiento
- Distribución estimada:
    - Sistema operativo: 20GB
    - Aplicación Node.js: 5GB
    - Base de datos PostgreSQL: 10GB
    - Logs y respaldos: 15GB

Conectividad

- Puerto Ethernet 10/100/1000 Mbps
- Acceso a red local institucional

Requisitos de Software del Servidor

Sistema Operativo

- Windows Server 2019/2022
- Ubuntu Server 20.04 LTS o superior (alternativa Linux)

Software Base Requerido

- Node.js versión 18 LTS o superior
- PostgreSQL versión 13 o superior
- Git para control de versiones

Herramientas de Administración:

- PgAdmin 4 para gestión de base de datos
- Navegador web para acceso administrativo

Configuración de Red y Conectividad

Configuración de Red

- Dirección IP estática dentro de la red local
- Puerto 3000 o 8080 para aplicación web
- Puerto 5432 para base de datos PostgreSQL

- Configuración de firewall para permitir conexiones internas

Ancho de Banda:

- Mínimo: 10 Mbps para red local
- Recomendado: 50 Mbps para conexiones externas (representantes legales)

El sistema propuesto es técnicamente viable para la implementación en la Unidad Educativa "Jesús de Nazareth". Los requisitos mínimos de hardware y software son accesibles y la infraestructura institucional existente supera ampliamente las especificaciones necesarias.

Viabilidad Económica

El estudio de viabilidad económica es una estimación de los costos básicos asociados a la implementación y mantenimiento del proyecto. Es fundamental diferenciar entre los recursos existentes, que no generan un costo adicional, y los costos reales que deberán ser cubiertos.

El sistema propuesto es una aplicación web centralizada, lo que significa que no se instala localmente en cada computadora. En su lugar, se alojará en un servidor On-Premise. Esto permite a la Institución evitar gastos extras de los que ya tienen presupuestados.

Al tratarse de un trabajo de titulación, ciertos elementos serán aportados por los desarrolladores, y no se ha considerado una remuneración profesional en este análisis. El objetivo es demostrar que la inversión necesaria es razonable y accesible, garantizando la sostenibilidad del sistema una vez implementado.

Gastos de Mantenimiento de la Aplicación

Los gastos de mantenimiento corresponden a aquellos necesarios para asegurar el funcionamiento continuo y óptimo del sistema una vez puesto en marcha. Se ha considerado que el servidor será provisto por la institución (On-Premise), lo que reduce los costos iniciales de infraestructura.

Gastos de Mantenimiento de la Aplicación.

Estos gastos reflejan los costos operativos en los que incurre el desarrollador durante la fase de ejecución y desarrollo del proyecto de tesis. No incluyen remuneración profesional, ya que se asume como parte del esfuerzo académico.

Tabla 8

Gastos Asociados al Mantenimento de la Aplicación

Cantidad	Concepto	Detalle	Periodicidad	Costo
1	Servidor	Uso de la infraestructura existente (ON-PREMISE)	1 mes	$0.00
1	Internet	Conectividad para acceso al sistema	1 mes	$30.00
1	Suministros	Materiales de oficina, Energía eléctrica, etc.	1 mes	$0.00
1	Licencias de Software	Uso de Software de Código Abierto	1 mes	$0.00
Total		$30.00		

Nota: Detalle de los gastos básicos para el mantenimiento de la aplicación en línea.

Este consolidado presenta la suma total de los costos estimados para el desarrollo y el primer año de mantenimiento del sistema, ofreciendo una visión global de la inversión requerida.

Tabla 9

Resumen de Gastos del Equipo de Desarrollo en el proyecto

Cantidad	Concepto	Detalle	Periodicidad	Costo Mensual	Total
3	Desarrolladores	Desarrollo frontend y backend	4 meses	$0	$0
3	Alimentación	Gastos de alimentación durante el periodo de desarrollo	4 meses	$60.00	$180
3	Transporte	Desplazamientos relacionados con el proyecto	1 mes	$10.00	$30
3	Servicios Básicos	Electricidad, agua, internet en el hogar del desarrollador	4 meses	$40.00	$120
Total					$330

Nota: Detalle de gastos del equipo de desarrollo durante el proyecto.

Resumen General de los Costos

Según el análisis realizado, los costos asociados al desarrollo e implementación del sistema de gestión escolar son accesibles y se consideran razonables para la Unidad Educativa Jesús de Nazareth. La optimización en el uso de recursos existentes, como el servidor On-Premise, y la naturaleza del proyecto como tesis académica, que exime el pago de remuneración profesional al desarrollador, contribuyen a una

estructura de costos muy eficiente. Por lo tanto, se concluye que el proyecto es económicamente viable y no representa una barrera para su ejecución e implementación.

Tabla 10

Resumen General de Costos.

Descripción de gastos	**Valor**
Gastos de Mantenimiento en línea de la aplicación	$30.00
Gastos del Equipo de Desarrollado	$330.00
Total	$360.00

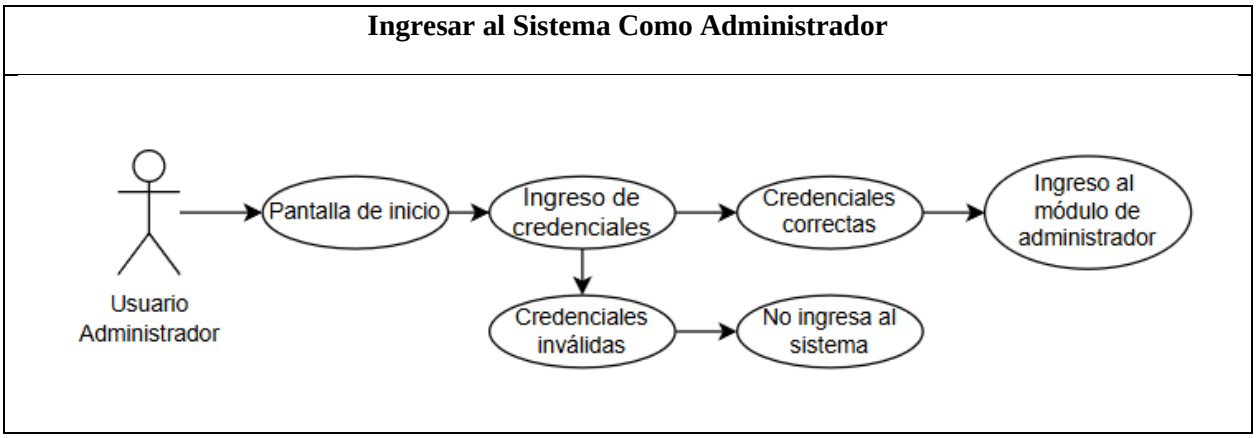
Nota: Detalles generales de los costos del proyecto

Casos de Uso

Casos de Uso Modulo Administrador

Figura 6

Caso de uso ingreso al sistema



Nota: Grafico UML de inicio de sesión con el rol de administrador.

Tabla 11

Ingreso al sistema como administrador

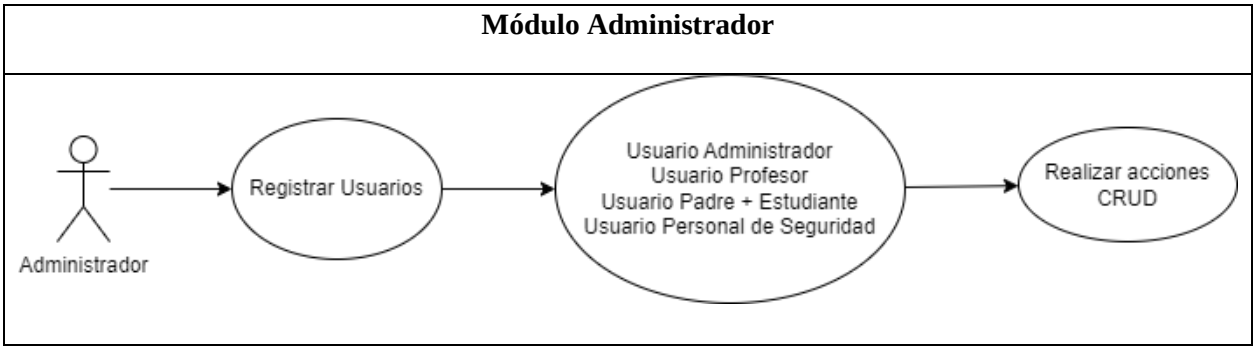
Nombre:	1.1 Ingresar al sistema como administrador
Descripción: El usuario debe ingresar con su nombre de usuario y su contraseña para poder acceder al sistema.	
Actores: Administrador	
Precondiciones: Usuario administrador ya registrado en el sistema	
Requisitos no Funcionales: Carga del módulo en 1 segundo	
Flujo de Eventos: 1. El administrador ingresa sus credenciales. 2. El sistema realiza la validación de las credenciales ingresadas. 3. Si son correctas, el administrador accede a su módulo.	

4. Si son incorrectas, el sistema le informa de sobre las credenciales invalidadas y no le permite ingresar al sistema.
Flujo Alterno: 4.1 El administrador puede recuperar su nombre de usuario y contraseña mediante la recuperación de usuario y contraseña.

Nota: Caso de uso para ingresar al módulo de administrador.

Figura 7

Gestión de usuarios.



Nota: Grafico UML de registro de nuevos usuarios.

Tabla 12

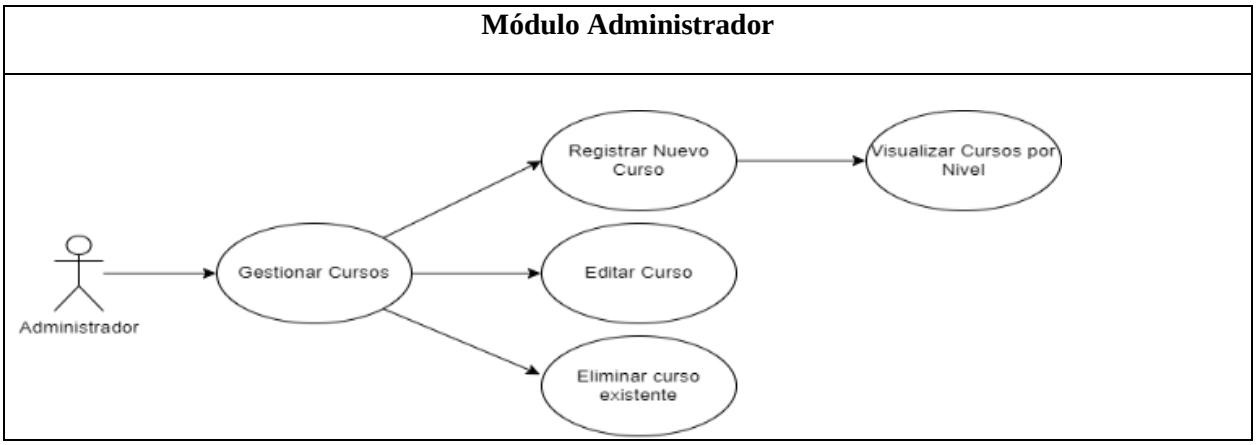
Gestión de usuarios

Nombre:	1.2 Gestión de usuarios
Descripción: El administrador puede registrar, visualizar, modificar y eliminar usuarios del sistema, incluyendo administrativos, profesores, representantes legales, estudiantes y personal de seguridad.	
Actores: Administrador	
Precondiciones: El administrador ha iniciado sesión en el sistema con credenciales válidas.	
Requisitos no Funcionales: • Validación de campos obligatorios • Control de roles permitidos • Interfaz intuitiva y rápida respuesta del sistema	
Flujo de Eventos: 1. El administrador accede a la opción de gestión de usuarios desde el panel principal. 2. Selecciona el tipo de usuario a gestionar (administrativo, profesor, representante legal + estudiante o guardia). 3. Puede registrar un nuevo usuario llenando los campos requeridos. 4. Puede visualizar la lista de usuarios existentes. 5. Puede editar la información de un usuario existente. 6. Puede eliminar un usuario del sistema. 7. El sistema valida los datos ingresados y realiza las acciones correspondientes.	
Flujo Alterno: 3.1. Si los campos están incompletos o inválidos, el sistema muestra mensajes de error y no permite guardar el registro.	

Nota: Caso de uso para registrar un nuevo usuario con el rol de profesor, estudiante, representante legal y guardia de seguridad.

Figura 8

Administración de cursos.



Nota: Grafico UML para la gestión de cursos – CRUD.

Tabla 13

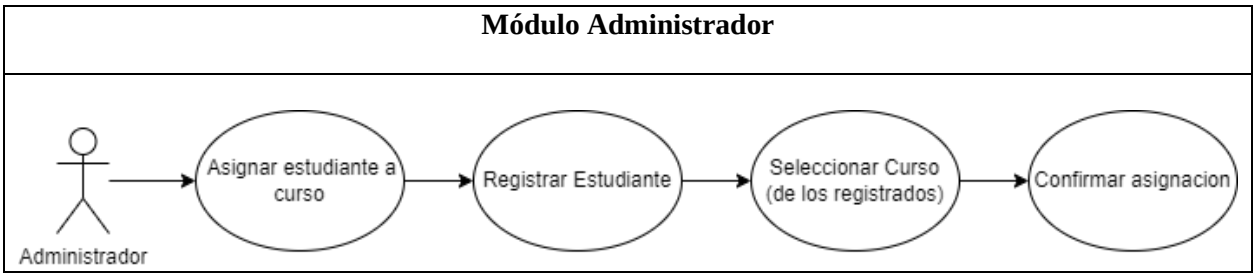
Gestión de Cursos

Nombre:	1.3 Gestión de cursos
Descripción: El administrador puede registrar, visualizar, modificar y eliminar cursos de la institución, especificando su nivel educativo y paralelo.	
Actores: Administrador	
Precondiciones: El administrador ha iniciado sesión en el sistema con credenciales válidas.	
Requisitos no Funcionales: • Validación de campos obligatorios • Prevención de duplicidad en registros • Interfaz intuitiva y rápida respuesta del sistema	
Flujo de Eventos: 1. El administrador accede a la sección de administración de cursos. 2. Visualiza la lista de cursos registrados organizados por nivel educativo. 3. Puede crear un nuevo curso ingresando nombre, nivel y paralelo. 4. Puede editar la información de un curso existente. 5. Puede eliminar un curso registrado. 6. El sistema valida los datos ingresados y actualiza la base de datos según la acción realizada.	
Flujo Alterno: 3.1. Si el nombre del curso ya existe en el mismo nivel y paralelo, el sistema muestra un mensaje de advertencia y no permite la duplicación.	

Nota: Caso de uso para el registro de nuevos usuarios y sus respectivos roles.

Figura 9

Asignación de estudiante a curso.



Nota: Grafico UML a de asignación de un estudiante a un curso.

Tabla 14

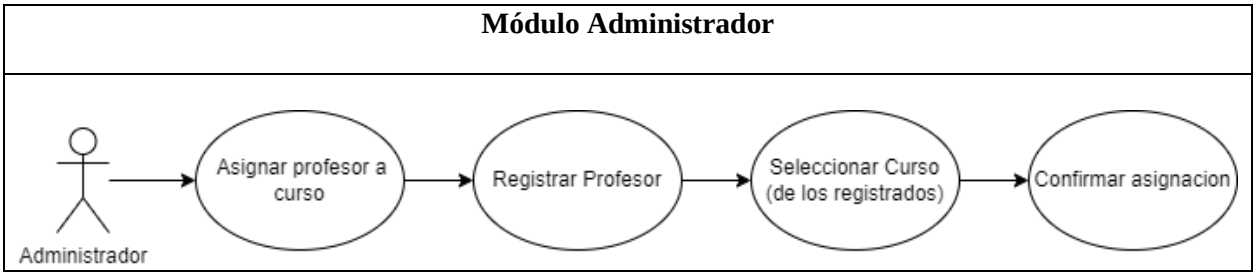
Asignación de estudiante a curso.

Nombre:	1.4 Asignación de estudiante a curso
Descripción: El administrador puede asignar un estudiante registrado a un curso específico dentro del sistema.	
Actores: Administrador	
Precondiciones: • El estudiante ya debe estar registrado en el sistema. • Deben existir cursos activos en el sistema.	
Requisitos no Funcionales: • Validación de existencia del estudiante y del curso • Prevención de asignaciones duplicadas • Interfaz ágil para búsqueda y selección	
Flujo de Eventos: 1. El administrador accede a la sección de gestión de estudiantes. 2. Selecciona un estudiante de la lista disponible. 3. Selecciona un curso activo del sistema. 4. Confirma la asignación del estudiante al curso seleccionado. 5. El sistema valida que el estudiante no esté ya asignado a otro curso y guarda la relación.	
Flujo Alterno: 5.1. Si el estudiante ya está asignado a un curso, el sistema muestra una advertencia y bloquea la duplicación. 5.2. Si no existen cursos disponibles, el sistema informa que debe registrarse al menos un curso.	

Nota: Caso de uso para la asignación de un estudiante a un curso.

Figura 10

Asignación de profesor a curso.



Nota: Grafico UML de uso de asignación de un profesor a un curso.

Tabla 15

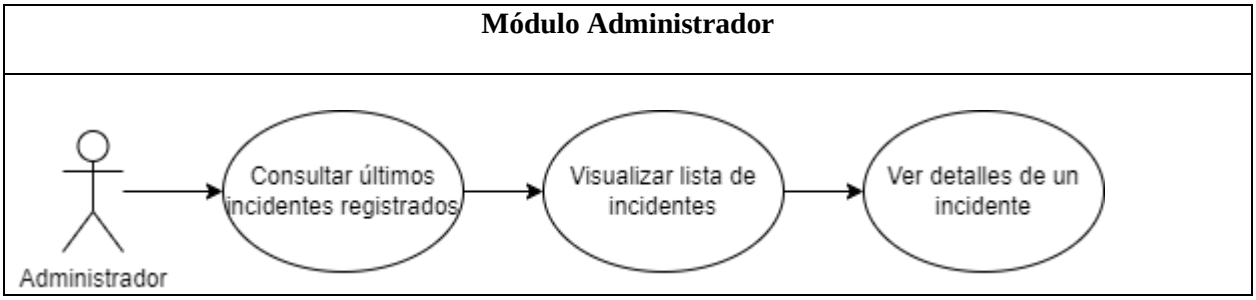
Asignación de profesor a curso.

Nombre:	1.5 Asignación de profesor a curso
Descripción: El administrador puede asignar uno o varios cursos a un profesor, de acuerdo con la planificación académica institucional.	
Actores: Administrador	
Precondiciones: • El profesor debe estar previamente registrado en el sistema. • Deben existir cursos registrados y activos.	
Requisitos no Funcionales: • Validación de relaciones ya existentes • Interfaz clara para visualizar asignaciones actuales	
Flujo de Eventos: 1. El administrador accede a la sección de gestión de profesores. 2. Selecciona un profesor de la lista disponible. 3. Visualiza los cursos actualmente asignados (si existieran). 4. Selecciona uno o varios cursos para asignar al profesor. 5. Confirma la acción. 6. El sistema registra la relación y actualiza los datos correspondientes.	
Flujo Alterno: 4.1. Si el curso ya está asignado al profesor, el sistema muestra un mensaje indicando la asignación previa.	

Nota: Caso de uso para asignar a un profesor a un curso.

Figura 11

Consulta de últimos incidentes registrados.



Nota: Grafico UML de incidentes registrados en la aplicación.

Tabla 16

Consulta de últimos incidentes registrados.

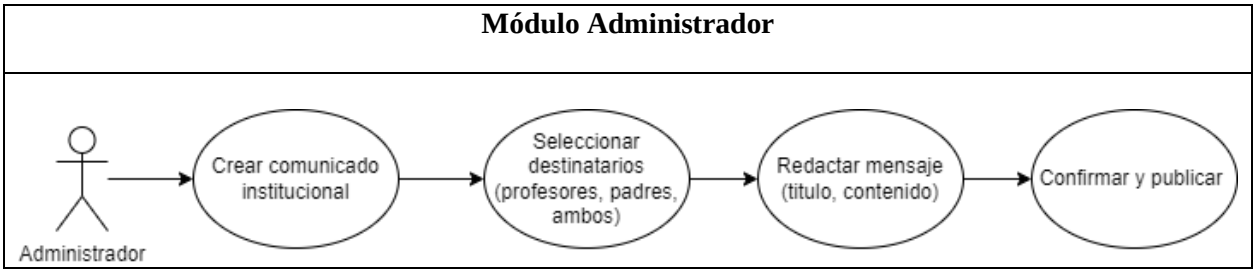
Nombre:	1.6 Consulta de últimos incidentes registrados
Descripción: El administrador puede visualizar un resumen de los incidentes disciplinarios más recientes registrados por los docentes.	
Actores: Administrador	
Precondiciones: • Deben existir incidentes registrados por parte de los profesores. • El administrador ha iniciado sesión con los permisos correspondientes.	

Requisitos no Funcionales: • Interfaz clara para facilitar la revisión rápida • Consulta optimizada para mostrar los registros más recientes
Flujo de Eventos: 1. El administrador accede a la sección de incidentes. 2. El sistema muestra por defecto los últimos incidentes registrados, ordenados por fecha descendente. 3. El administrador puede visualizar detalles como estudiante, curso, fecha y descripción del incidente. 4. Puede aplicar filtros si desea ver incidentes por curso o estudiante específico.
Flujo Alterno: 2.1. Si no existen incidentes registrados, el sistema muestra un mensaje indicando que no hay información disponible.

Nota: Caso de uso para visualizar los últimos incidentes registrados en la aplicación por parte del docente.

Figura 12

Creación de comunicados institucionales.



Nota: Grafico UML para la creación de un comunicado institucional.

Tabla 17

Creación de comunicados institucionales.

Nombre:	1.7 Creación de comunicados institucionales
Descripción: El administrador puede redactar y publicar comunicados dirigidos a profesores, padres de familia o a ambos, para informar novedades, avisos o eventos institucionales.	
Actores: Administrador	
Precondiciones: • El administrador ha iniciado sesión en el sistema. • Debe existir al menos un grupo destinatario (profesores o padres) registrado.	
Requisitos no Funcionales: • Interfaz de redacción intuitiva • Capacidad de seleccionar destinatarios por rol • Confirmación de publicación exitosa	
Flujo de Eventos: 1. El administrador accede a la sección de comunicados. 2. Redacta el título y contenido del comunicado. 3. Selecciona los destinatarios: profesores, padres de familia o ambos. 4. Confirma la publicación del comunicado. 5. El sistema registra y envía el comunicado a los usuarios correspondientes.	
Flujo Alterno: 2.1. Si el contenido está incompleto o vacío, el sistema muestra un mensaje de validación.	

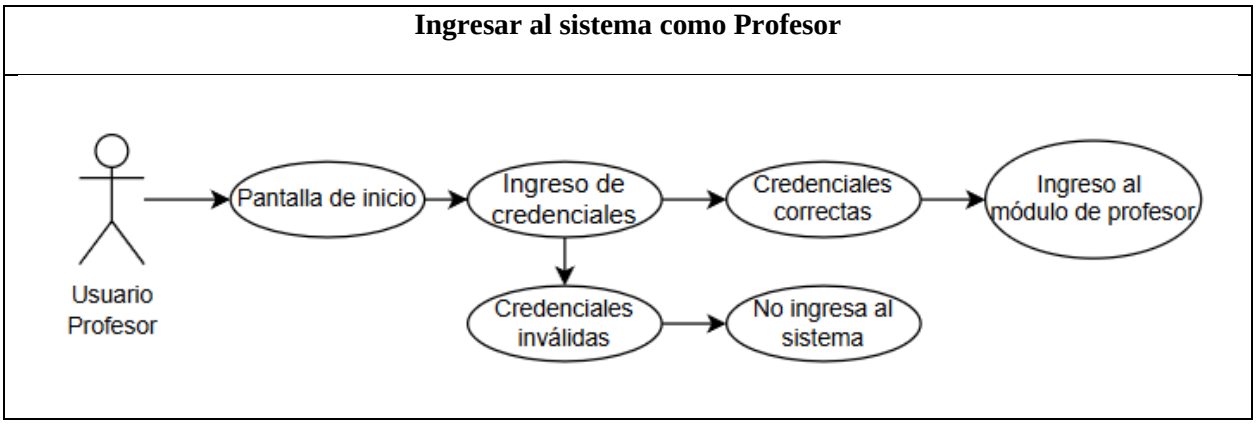
4.1. Si no hay usuarios del rol seleccionado, el sistema informa que no se pudo enviar el comunicado.

Nota: Caso de uso para la creación de un nuevo comunicado institucional.

Casos de Uso Módulo Profesor

Figura 13

Ingreso al módulo de profesor



Nota: Grafico UML de inicio de sesión con el rol de profesor.

Tabla 18

Ingreso al sistema como profesor

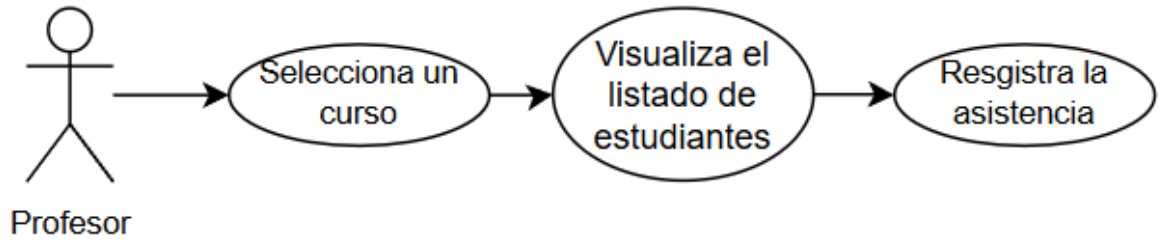
Nombre:	2.1 Ingresar al sistema como profesor
Descripción: El usuario debe ingresar con su nombre de usuario y su contraseña para poder acceder al sistema.	
Actores: Profesor	
Precondiciones: Usuario profesor ya registrado en el sistema	
Requisitos no Funcionales: Carga del sistema y visualización de los cursos en 1 segundo Vista responsive.	
Flujo de Eventos: 1. El profesor ingresa sus credenciales. 2. El sistema realiza la validación de las credenciales ingresadas. 3. Si son correctas, el profesor accede a su módulo. 4. Si son incorrectas, el sistema le informa de sobre las credenciales invalidadas y no le permite ingresar al sistema.	
Flujo Alterno: 4.1 El profesor puede recuperar su nombre de usuario y contraseña mediante la recuperación de usuario y contraseña.	

Nota: Caso de uso para ingresar al sistema con el rol de profesor

Figura 14

Registro de asistencias del estudiante

Módulo Profesor



Nota: Grafico UML de caso de uso de registro de las asistencias de estudiantes por curso.

Tabla 19

Registro de asistencia de estudiante

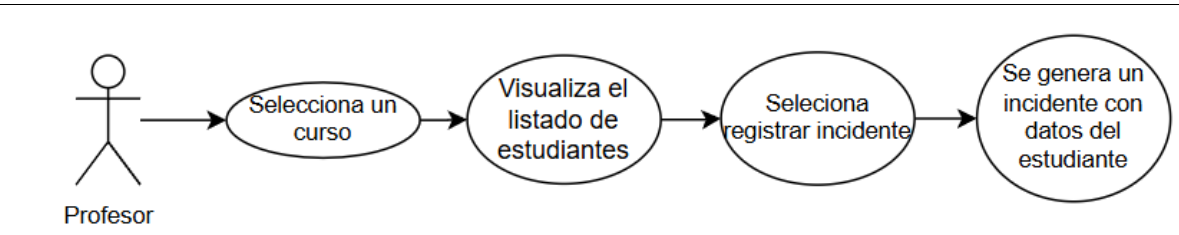
Nombre:	2.2 Registro de asistencia de estudiante
Descripción: El profesor debe ingresar con su nombre de usuario y su contraseña para poder acceder al sistema y visualizar sus cursos designados.	
Actores: Profesores	
Precondiciones: Ingreso al sistema con el rol de profesor	
Requisitos no Funcionales: Validación de datos ingresados	
Flujo de Eventos: 1. El profesor visualiza los cursos asignados y selecciona un curso. 2. Se despliega la lista de los estudiantes del curso. 3. Puede seleccionar los botones de asiste, falta o atraso, según sea necesario. 4. El sistema valida que todos los estudiantes tengan registrado un estado. 5. El sistema permite la toma de asistencia una vez por día.	
Flujo Alterno: 4.1 El sistema notifica al profesor que faltan estudiantes por registrar. 5.1 Al seleccionar el curso, el docente es notificado que ya se tomó lista en el curso.	

Nota: Caso de uso para registrar la asistencia de los estudiantes.

Figura 15

Diagrama de caso de uso: Registro de incidentes del estudiante.

Módulo Profesor



Nota: Grafico UML de caso de uso de registro de incidentes por curso y estudiante.

Tabla 20

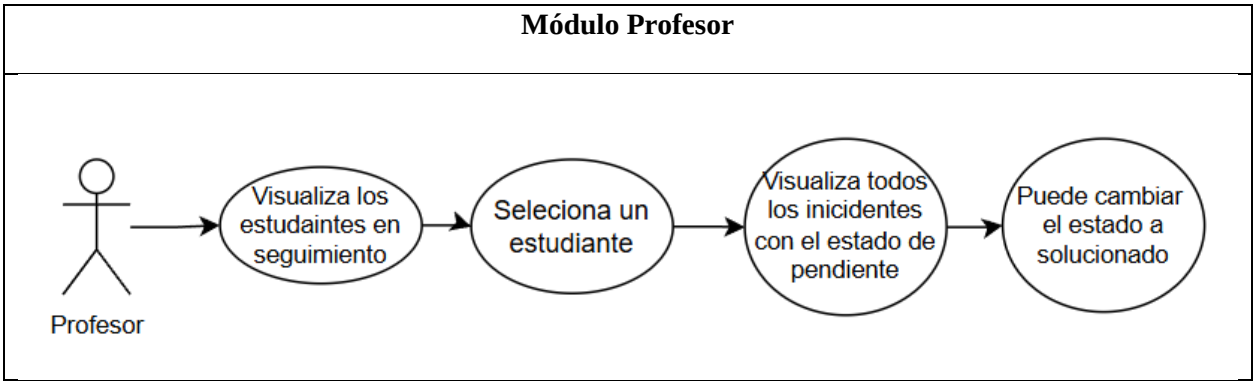
Registro de incidentes del estudiante.

Nombre:	2.3 Registro de incidentes del estudiante
Descripción: El profesor debe ingresar con su nombre de usuario y su contraseña para poder acceder al sistema y poder registrar incidentes.	
Actores: Profesores	
Precondiciones: Ingreso al sistema con el rol de profesor	
Requisitos no Funcionales: Validación de datos ingresados	
Flujo de Eventos: 1. El profesor visualiza los cursos asignados y selecciona un curso. 2. Se despliega la lista de los estudiantes del curso. 3. Selecciona el estudiante para el registro del incidente. 4. El sistema carga la información del estudiante, profesor, curso y fecha para el registro del incidente. 5. El profesor selecciona el tipo de incidente (medico, académico o disciplinario) 6. El profesor llena el campo de texto con información complementaria del incidente. 7. Al guardar el incidente el estudiante pasa a la ventana de seguimiento.	
Flujo Alterno: 6.1 El profesor debe llenar el campo de texto caso contario no podrá guardar el incidente. 7.1El listado de estudiantes de seguimiento permitirá visualizar todos los estudiantes en seguimiento identificando el id del profesor.	

Nota: Caso de uso para registrar los incidentes de los estudiantes por curso.

Figura 16

Seguimiento de incidentes del estudiante



Nota: Grafico UML de caso de uso actualización de estado de los incidentes.

Tabla 21

Seguimiento de incidentes del estudiante.

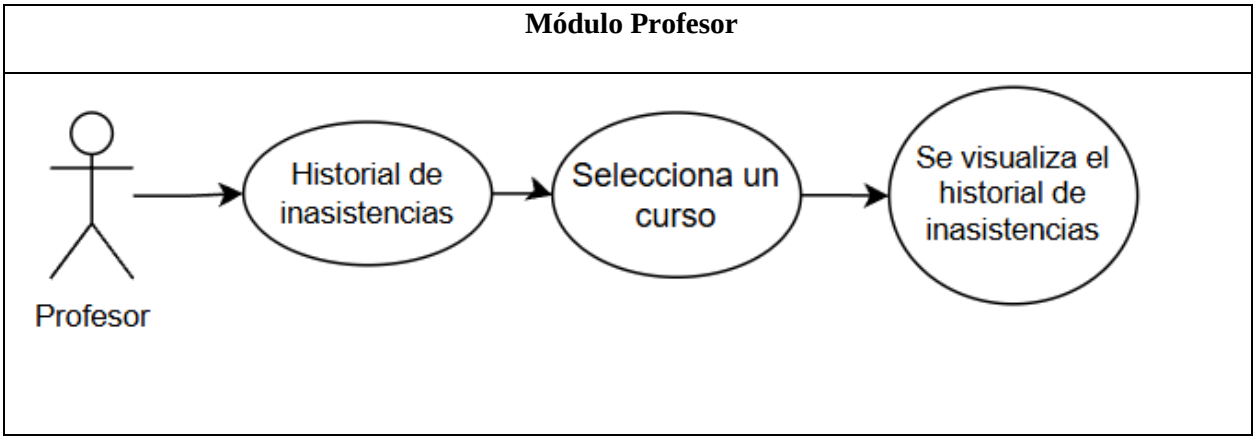
Nombre:	2.4 Seguimiento de incidentes del estudiante
Descripción: El profesor debe ingresar con su nombre de usuario y su contraseña para poder acceder al sistema y visualizar los estudiantes en seguimiento.	
Actores: Profesores	
Precondiciones: Ingreso al sistema con el rol de profesor	
Requisitos no Funcionales: Validación de datos ingresados	
Flujo de Eventos: 1. El profesor visualiza el listado de estudiantes en seguimiento.	

2. Selecciona un estudiante y visualiza todos sus incidentes con el estado pendiente. 3. Llena el campo de texto con la información de resolución del incidente. 4. Cambia el estado a resuelto y guarda la actualización del estado. 5. Si el estudiante no tiene incidentes pendientes, su nombre desaparece de la lista.
Flujo Alterno: 5.1 Si el estudiante tiene varios incidentes pendientes, el mismo seguirá en la lista de incidentes hasta que todos los estados cambien a resuelto.

Nota: Caso de uso para el seguimiento de estudiantes con incidentes.

Figura 17

Historial de inasistencias del estudiante.



Nota: Grafico UML de caso de uso para visualizar el historial de inasistencias.

Tabla 22

Historial de inasistencias del estudiante.

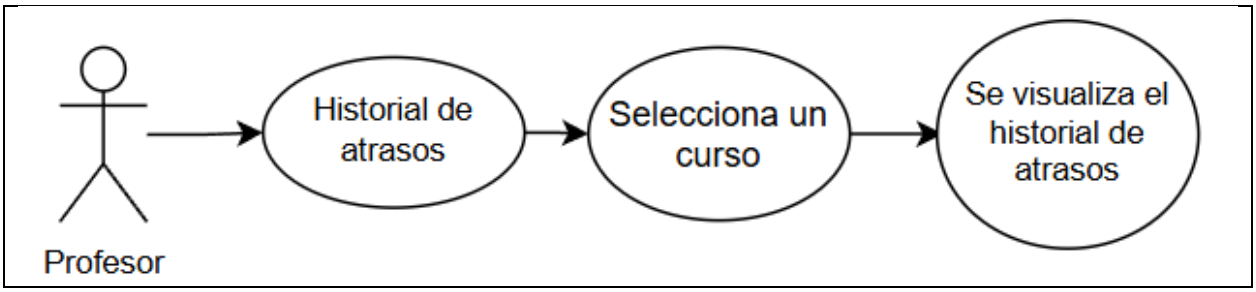
Nombre:	2.5 Historial de inasistencias del estudiante
Descripción: El profesor debe ingresar con su nombre de usuario y su contraseña para poder acceder al sistema y visualizar el historial de inasistencias.	
Actores: Profesores	
Precondiciones: Ingreso al sistema con el rol de profesor	
Requisitos no Funcionales: Que el historial se cargue en menos de 1 segundo.	
Flujo de Eventos: 1. El profesor selecciona el botón de inasistencias. 2. Selecciona el curso del cual quiere consultar. 3. Se visualizan únicamente los estudiantes que registran inasistencias. 4.Visualiza las fechas de inasistencia y un contador de inasistencias.	
Flujo Alterno: 2.1 Si no aparecen los datos de inasistencia debe comunicarse con el administrador.	

Nota: Caso de uso para visualizar el historial de inasistencias.

Figura 18

Historial de atrasos del estudiante

Módulo Profesor



Nota: Grafico UML de caso de uso para visualizar el historial de atrasos por curso y estudiante.

Tabla 23

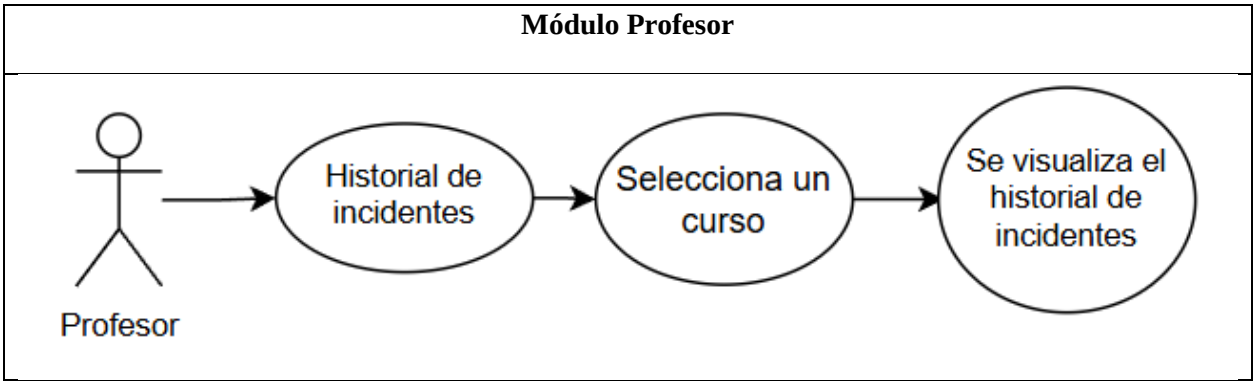
Historial de atrasos del estudiante.

Nombre:	2.6 Historial de atrasos del estudiante
Descripción: El profesor debe ingresar con su nombre de usuario y su contraseña para poder acceder al sistema y visualizar el historial de atrasos.	
Actores: Profesores	
Precondiciones: Ingreso al sistema con el rol de profesor	
Requisitos no Funcionales: Que el historial se cargue en menos de 1 segundo.	
Flujo de Eventos: 1. El profesor selecciona el botón de atrasos. 2. Selecciona el curso del cual quiere consultar. 3. Se visualizan únicamente los estudiantes que registran atrasos. 4.Visualiza las fechas de atrasos y un contador de atrasos.	
Flujo Alterno: 2.1 Si no aparecen los datos de los atrasos debe comunicarse con el administrador.	

Nota: Caso de uso para visualizar el historial de atrasos de los estudiantes.

Figura 19

Historial de incidentes del estudiante.



Nota: Grafico UML de caso de uso para visualizar el historial de incidentes del estudiante.

Tabla 24

Historial de incidentes del estudiante

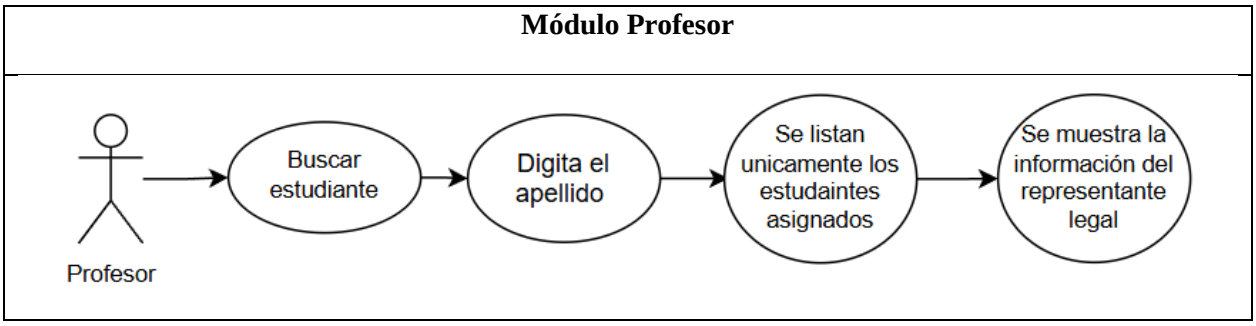
Nombre:	2.7 Historial de incidentes del estudiante

Descripción: El profesor debe ingresar con su nombre de usuario y su contraseña para poder acceder al sistema y visualizar el historial de incidentes.
Actores: Profesores
Precondiciones: Ingreso al sistema con el rol de profesor
Requisitos no Funcionales: Que el historial se cargue en menos de 1 segundo.
Flujo de Eventos: 1. El profesor selecciona el botón de incidentes. 2. Selecciona el curso del cual quiere consultar. 3. Se visualizan únicamente los estudiantes que registran incidentes. 4. Al seleccionar el estudiante se despliega el historial de todos sus incidentes donde se visualiza los datos del estudiante, la fecha de registro del incidente, el tipo y el comentario del incidente. Además, se pueden ver los incidentes con estado pendiente y resuelto.
Flujo Alterno: 2.1 Si no aparecen los datos de los atrasos debe comunicarse con el administrador.

Nota: Caso de uso para visualizar el historial de incidentes del estudiante por curso.

Figura 20

Visualizar información del representante legal.



Nota: Grafico UML de caso de uso buscar y visualizar la información de un representante legal.

Tabla 25

Visualizar información del representante legal.

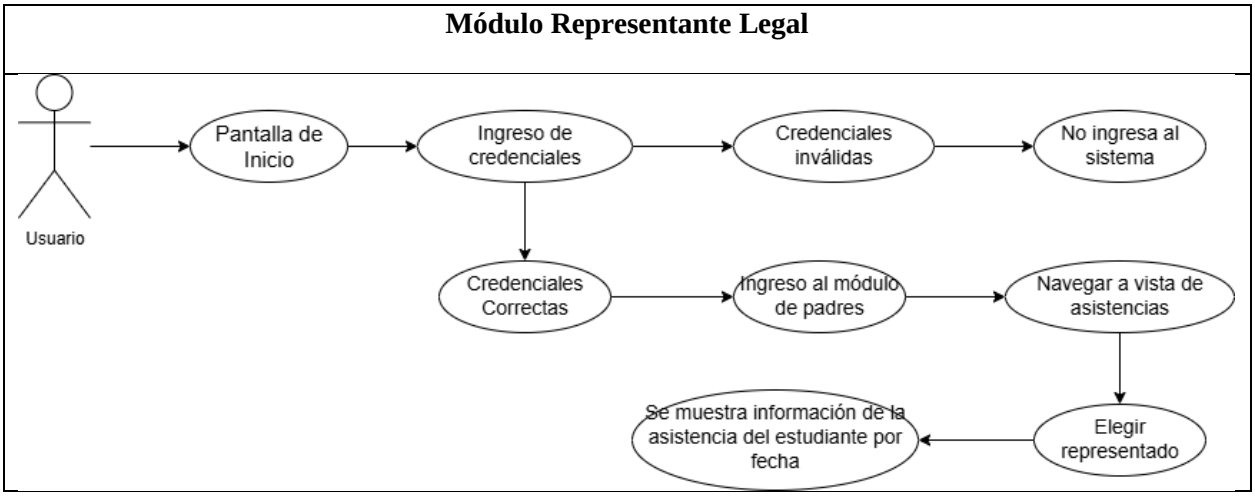
Nombre:	2.8 Seguimiento de incidentes del estudiante
Descripción: El profesor debe ingresar con su nombre de usuario y su contraseña para poder acceder al sistema y pueda buscar información básica de contacto del representante legal de un estudiante.	
Actores: Profesores	
Precondiciones: Ingreso al sistema con el rol de profesor	
Requisitos no Funcionales: Validación de datos ingresados	
Flujo de Eventos: 1. El profesor digita el apellido del estudiante. 2. El sistema refleja únicamente los estudiantes asignados a ese profesor. 3. El profesor selecciona el estudiante 4. Se muestra por pantalla la información básica del representante (nombre y número de contacto).	
Flujo Alterno: 1.1 si no aparece información del estudiante, verificar si se tipeo correctamente el apellido. 4.1 Si no aparece la información del representante legal debe comunicarse con el administrador.	

Nota: Caso de uso para buscar y visualizar la información de un padre de familia.

Casos de Uso Modulo Padre de Familia

Figura 21

Visualización de la asistencia del estudiante ligado al representante legal.



Nota: Grafico UML de uso para loguearse como representante legal y visualizar al módulo.

Tabla 26**6**

Visualización de asistencia de estudiante al que representa.

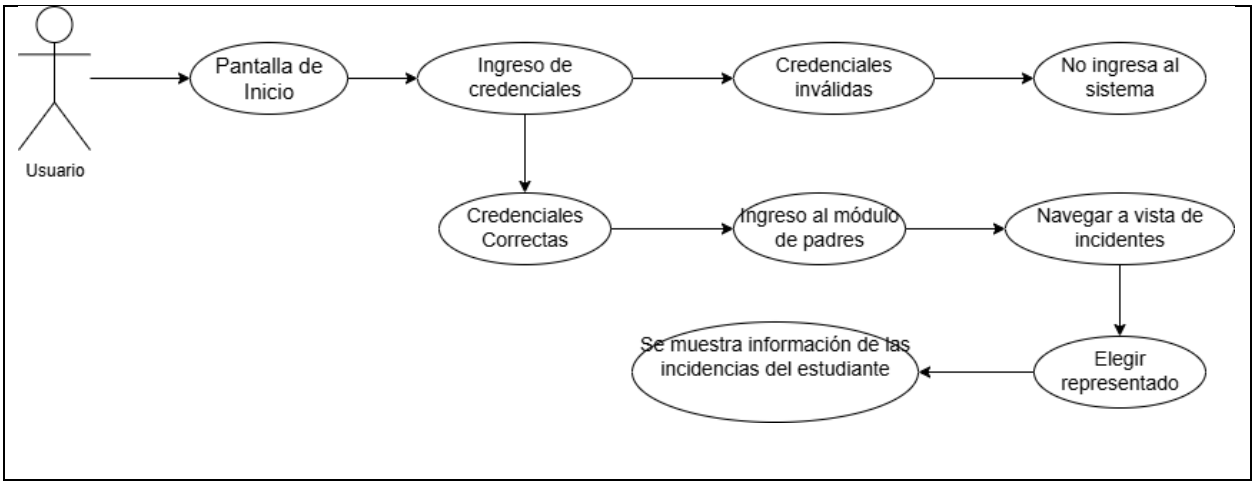
Nombre:	3.1 Visualización de asistencia de estudiante al que representa
Descripción: El Representante Legal ingresa con sus credenciales al sistema en el cual podrá observar la asistencia de su o sus representados según la fecha escogida.	
Actores: Representante Legal.	
Precondiciones: Ingreso al sistema con el rol de representante legal.	
Requisitos no Funcionales: Carga de asistencias en menos de 2 segundos.	
Flujo de Eventos: 1. El representante legal ingresa usuario y contraseña 2. El sistema valida sus credenciales 3. El sistema redirecciona a la página de inicio del módulo de representantes legales 4. El representante legal navega hacia el módulo de asistencia 5. Se muestra la página de asistencia del estudiante 6. El representante legal elige a su representado (si tiene más de uno) y la fecha en la que quiere consultar la asistencia 7. El sistema busca y muestra la información de asistencia del estudiante	
Flujo Alterno: 2.1 Si no son credenciales válidas el sistema lanza un error	

Nota: Caso de uso para loguearse como representante legal y visualizar al módulo.

Figura 22

Visualización de las incidencias del estudiante ligado al representante legal.

Módulo Representante Legal



Nota: Diagrama de uso para visualizar los incidentes del estudiante**.**

Tabla 27

Visualización de incidencias del estudiante al que representa.

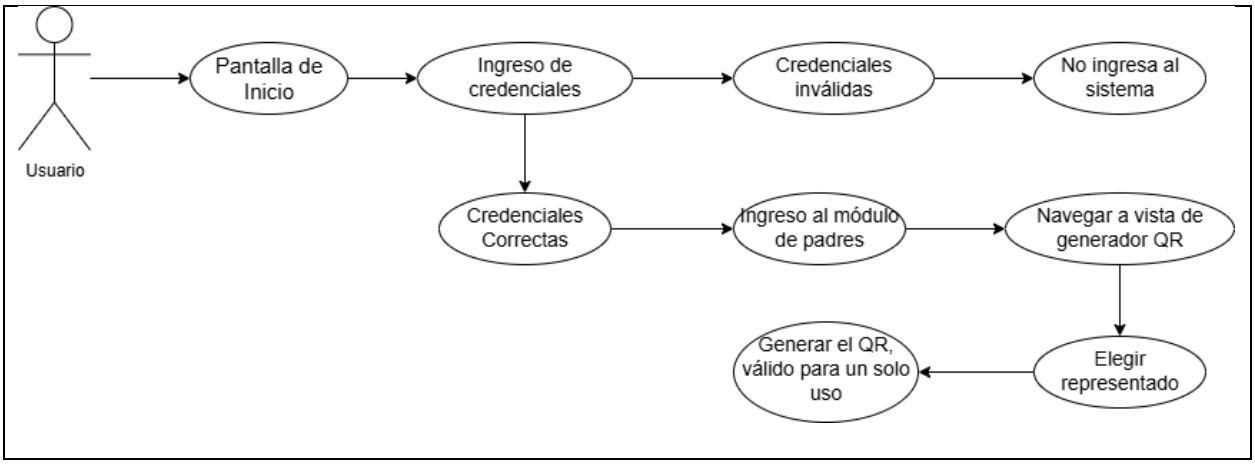
Nombre:	3.2 Visualización de incidencias de estudiante al que representa
Descripción: El Representante Legal ingresa con sus credenciales al sistema en el cual podrá observar las incidencias de su o sus representados.	
Actores: Representante Legal.	
Precondiciones: Ingreso al sistema con el rol de representante legal.	
Requisitos no Funcionales: Carga de incidencias en menos de 2 segundos.	
Flujo de Eventos: 1. El representante legal ingresa usuario y contraseña 2. El sistema valida sus credenciales 3. El sistema redirecciona a la página de inicio del módulo de representantes legales 4. El representante legal navega hacia el módulo de incidencias 5. Se muestra la página de incidencias del estudiante 6. El representante legal elige a su representado (si tiene más de uno). 7. El sistema busca y muestra la información de incidencias del estudiante	
Flujo Alterno: 2.1 Si no son credenciales válidas el sistema lanza un error	

Nota: Caso de uso para visualizar los incidentes de un estudiante asignado a un representante legal.

Figura 23

Generador QR para el retiro del estudiante.

Módulo Representante Legal



Nota: Grafico UML para generar un código qr para el retiro de un estudiante.

Tabla 28

Visualización de asistencia de estudiante al que representa.

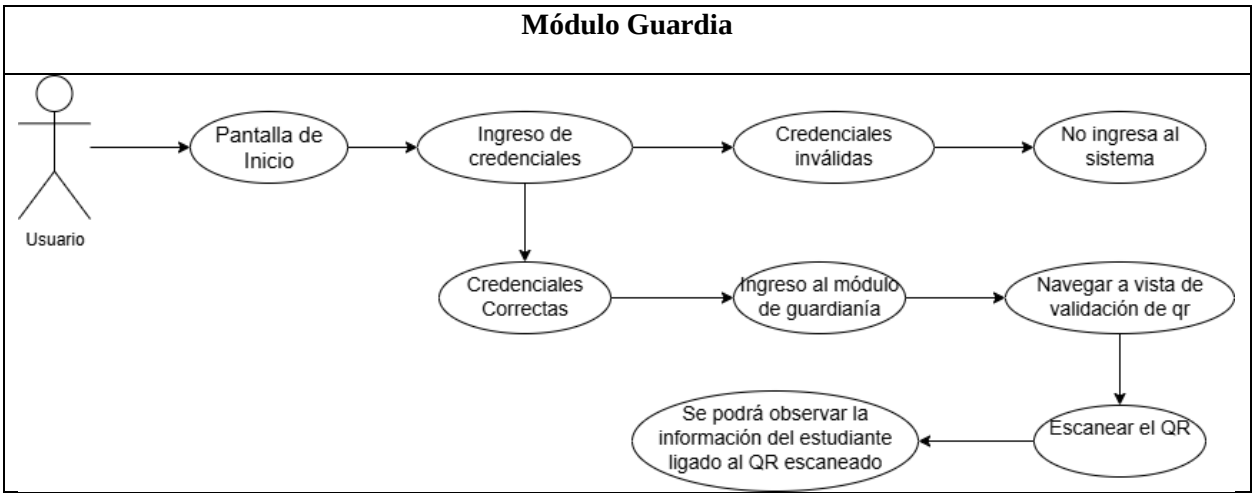
Nombre:	3.3 Generador QR para el retiro del estudiante
Descripción: El Representante Legal ingresa con sus credenciales al sistema en el cual podrá generar un qr para retirar a su representado.	
Actores: Representante Legal.	
Precondiciones: Ingreso al sistema con el rol de representante legal.	
Requisitos no Funcionales: Generar QR en menos de 3 segundos.	
Flujo de Eventos: 1. El representante legal ingresa usuario y contraseña 2. El sistema valida sus credenciales 3. El sistema redirecciona a la página de inicio del módulo de representantes legales 4. El representante legal navega hacia el módulo de generador de qr. 5. Se muestra la página de generador de qr para el retiro del estudiante. 6. El representante legal elige a su representado (si tiene más de uno). 7. El representante legal da click en el botón de Generar QR. 8. El sistema genera un QR de un solo uso ligado a un estudiante.	
Flujo Alterno: 2.1 Si no son credenciales válidas el sistema lanza un error	

Nota: Caso de uso para generar un nuevo código qr para retiro de un estudiante.

Casos de Uso Modulo Personal de Seguridad

Figura 24

Generador QR para el retiro del estudiante.



Nota: Grafico UML para loguearse como guardia y escanear el código qr generado por el representante legal del estudiante.

Tabla 29

Visualización de asistencia de estudiante al que representa.

Nombre:	4.1 Escaneo de QR
Descripción: El personal de seguridad ingresa al sistema para escanear el QR ligado al estudiante que dejará salir de la institución	
Actores: Personal de seguridad	
Precondiciones: Ingreso al sistema con el rol de guardia.	
Requisitos no Funcionales: Validar al estudiante que deberá dejar salir en menos de 2 segundos	
Flujo de Eventos: 9. El personal de seguridad ingresa usuario y contraseña. 10. El sistema valida sus credenciales. 11. El sistema redirecciona a la página de inicio del módulo de guardianía. 12. El personal de seguridad navega hacia el módulo de escaneo de qr. 13. Se muestra la página de escaneo de QR para la salida de la institución del estudiante. 14. El personal de seguridad escanea el QR. 15. El sistema valida el QR y redirecciona hacia la página de la información del estudiante. 16. El sistema actualiza el QR para que no pueda ser usado nuevamente.	
Flujo Alterno: 2.1 Si no son credenciales válidas el sistema lanza un error. 15. El sistema valida el QR y da un mensaje de error en caso de que haya sido usado antes.	

Nota: Caso de uso para ingresar al módulo de guardia y escanear el código qr generado por el representante legal del estudiante.

Capítulo III

Desarrollo

El sistema se construyó siguiendo un enfoque ágil basado en SCRUM, lo que permitió iterar rápidamente sobre los módulos principales y obtener retroalimentación continua de nuestro Scrum Master. Para el desarrollo se emplearon:

Frontend: React.js, creando componentes reutilizables para cada rol (Administrador, Profesor, Representante Legal y Guardia).

Backend: Node.js con Express.js, exponiendo una API REST para la gestión de estudiantes, asistencias, incidencias y códigos QR.

Gestión de datos: PostgreSQL como base relacional principal; MongoDB reservado para almacenar historiales de observaciones semiestructuradas.

Generación de QR: Biblioteca qrcode en Node.js para convertir tokens únicos en imágenes codificadas.

Autenticación y seguridad: JWT para sesiones y validación de peticiones, garantizando que cada operación esté debidamente autorizada.

Cada uno de estos componentes se integró de manera incremental, validando en cada sprint que los requisitos funcionales y no funcionales se cumplieran satisfactoriamente.

Interacción de los Componentes

Para ilustrar el flujo de trabajo, se describe a continuación el proceso de autorización de salida de un estudiante mediante código QR:

1. **Generación (Padre de Familia):** El padre de familia, a través de la interfaz de React (Vista), solicita un código QE para su representado. La Vista envía una petición POST a un endpoint específico de la API REST
2. **Procesamiento (Backend):** El Controlador en Express.js recibe la petición, valida la identidad del padre, genera un token único y temporal, lo asocia al estudiante correspondiente

y lo almacena en la base de datos PostgreSQL (Modelo). Finalmente, devuelve este token a la Vista.

3. **Visualización (Padre de Familia):** La aplicación en React recibe el token y lo renderiza como un código QR en la pantalla del dispositivo del padre.
4. **Verificación (Guardianía):** El personal de guardianía utiliza su interfaz en React para escanear el código QR. La aplicación extrae el token y envía una petición POST al endpoint de validación.
5. **Autorización (Backend):** El Controlador verifica el token contra la base de datos PostgreSQL. Si el token es válido, no ha expirado y corresponde a un estudiante, el sistema marca la salida como autorizada y devuelve una respuesta de éxito con los datos del estudiante a la Vista del guardia.
6. **Confirmación (Guardianía):** La aplicación del guardia recibe la confirmación y muestra en pantalla el mensaje "Salida Autorizada" junto con el nombre y foto del estudiante, completando el proceso de forma segura

Requerimientos del software

Requisitos del Módulo Administrador

Requisitos Funcionales

- **Gestión de Usuarios del Sistema:**

  El sistema debe permitir la creación, edición y eliminación de cursos.
  El administrador debe poder asignar profesores a cursos específicos.
  Debe poder ver agrupaciones de estudiantes por curso y nivel educativo.

- **Gestión de Cursos y Asignaciones:**

  El sistema debe permitir la creación, edición y eliminación de cursos.
  El administrador debe poder asignar profesores a cursos específicos.
  Debe poder ver agrupaciones de estudiantes por curso y nivel educativo.

- **Supervisión de Asistencias:**

El sistema debe mostrar registros de asistencias, atrasos e inasistencias.
El administrador debe poder filtrar estos registros por curso o estudiante.

- **Gestión de Incidentes Escolares:**

  El sistema debe guardar un historial por estudiante.

- **Monitoreo del Retiro de Estudiantes:**

  El administrador debe visualizar los retiros realizados mediante código QR.
  Debe poder ver intentos fallidos de retiro.

- **Gestión de Notificaciones:**

  El sistema debe permitir al administrador enviar notificaciones manuales a roles específicos (profesores, representantes, guardias).

Requisitos del Módulo de Profesores

Requisitos Funcionales

- **Registro de Asistencias por Curso:**

  El sistema debe permitir al profesor visualizar los cursos asignados.
  El profesor debe poder registrar el estado de asistencia de cada estudiante (asiste, falta, atraso).
  El registro de asistencia debe habilitarse una sola vez por día y por curso.

- **Gestión de Incidentes Estudiantiles:**

  El sistema debe permitir al profesor registrar incidentes por estudiante, categorizados como médicos, académicos o disciplinarios.
  Cada incidente debe permitir un campo de descripción obligatorio y estar asociado al curso, fecha y profesor.

- **Seguimiento y Resolución de Incidentes:**

  El sistema debe permitir actualizar el estado de los incidentes de "pendiente" a "resuelto".
  El profesor debe poder agregar comentarios de resolución al cerrar un incidente.

- **Consulta de Historiales:**

El sistema debe mostrar el historial de inasistencias, atrasos e incidentes por curso y estudiante.

El profesor debe poder filtrar la información por estudiante o por tipo de registro

- **Contacto con Representantes:**

  El sistema debe permitir consultar el nombre y número de contacto del representante legal de un estudiante.

Requisitos del Módulo de Padres de Familia

Requisitos Funcionales

- **Generación y Envío de Código QR:**

  El aplicativo debe generar el código QR.

  El aplicativo debe verificar la correcta generación del código QR.

- **Gestión de Caducidad y Uso del Código QR:**

  El código QR debe caducar después de su uso.

  El aplicativo debe evitar la generación de códigos duplicados.

  El sistema debe invalidar el código QR inmediatamente después de su primer uso exitoso.

  Se debe impedir la reutilización del mismo código para un segundo retiro.

- **Escaneo y Validación del Código QR:**

  El sistema debe validar la autenticidad del código y la identidad del estudiante.

  Una vez validado y aprobado el retiro del estudiante, el código QR se debe registrar como "usado".

- **Regeneración del Código QR:**

  El representante debería poder generar un nuevo código en caso de error o caducidad desde el aplicativo.

  Se debe permitir la regeneración inmediata del código si es inválido.

  Se debe permitir la regeneración del código QR de forma rápida desde la aplicación.

- **Notificaciones de Atrasos o Faltas:**

  El sistema debe enviar notificaciones a la aplicación del representante legal.

  Si se trata de un atraso, el sistema debe especificar la hora de ingreso después de la hora de inicio de clases.

Si se trata de una falta, el sistema debe detallar que el estudiante no asistió.

- **Notificaciones de Incidentes Escolares:**

  El sistema debe generar una notificación automática para el representante legal del estudiante, informando sobre el tipo de incidente.

Requisitos del Módulo de Guardianía

Requisitos Funcionales

- **Validación de Código QR para Retiro de Estudiantes:**

  El sistema debe permitir al personal de seguridad iniciar sesión con sus credenciales.
  Debe activar la cámara del dispositivo para escanear códigos QR entregados por los representantes legales.
  El sistema debe verificar si el código QR es válido, no caducado, y corresponde a un estudiante autorizado.

- **Autorización de Salida:**

  Al validar el código, el sistema debe mostrar los datos del estudiante y del representante autorizado.
  El sistema debe registrar la fecha y hora del retiro exitoso.

- **Gestión de Retiros:**

  El sistema debe bloquear el uso de códigos QR duplicados o expirados.
  Debe mostrar un mensaje claro si el código es inválido o el estudiante ya fue retirado.

Requisitos No Funcionales

- **Rendimiento:**

  La carga de cursos e historial de estudiantes debe completarse en menos de 2 segundos.

  Los registros de asistencia e incidentes deben guardarse sin retrasos perceptibles.

  La notificación de atraso/falta debe enviarse al representante legal inmediatamente

  después de registrar la falta o atraso.

  La notificación de incidente debe enviarse tan pronto como se registre el incidente.

La carga de datos como listas de usuarios, cursos o asistencias debe realizarse en menos de 3 segundos.

Los filtros por fecha, curso o nombre deben responder de forma inmediata.

La validación del código QR debe completarse en menos de 2 segundos desde su escaneo.

- **Seguridad:**

  El módulo debe contar con autenticación basada en roles.

  Toda la información sensible (contraseñas, tokens) debe estar cifrada.

  Solo el administrador debe tener acceso a la gestión completa del sistema.

  Solo el profesor autenticado debe acceder a los datos de sus cursos y estudiantes asignados.

  Almacenamiento cifrado de tokens.

  Rutas protegidas en el frontend.

- **Usabilidad:**

  La interfaz debe ser clara, organizada por secciones y permitir acciones rápidas (CRUD).

  Los formularios deben contar con validaciones automáticas y mensajes de error amigables.

  El sistema debe adaptarse tanto a escritorio como a teléfono según el rol, priorizando la vista administrativa.

  Los botones de asistencia e incidentes deben estar claramente diferenciados y accesibles.

  Los formularios deben validar campos requeridos y mostrar mensajes de ayuda al usuario.

  La generación del código QR debe ser sencilla.

Cronograma

Tabla 30

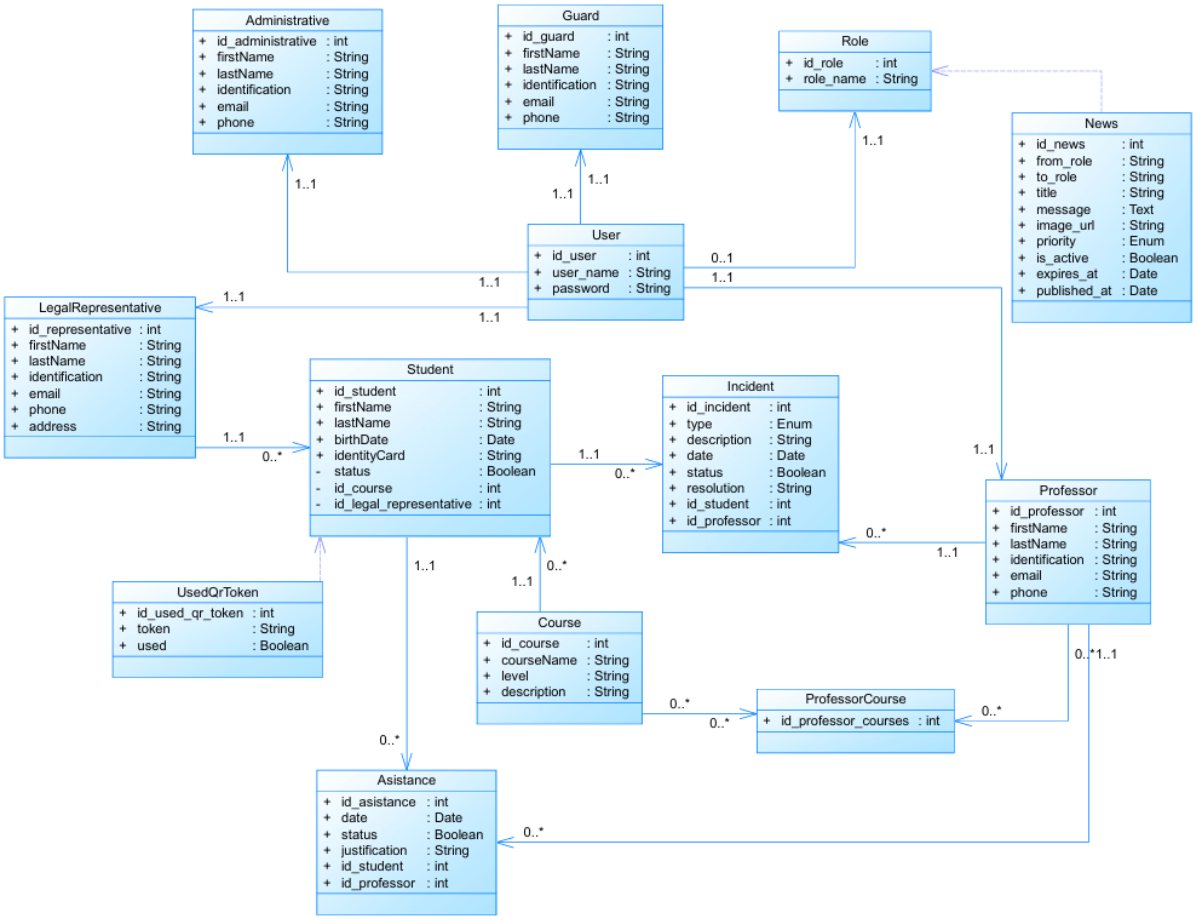
Cronograma de actividades

Sprint	Iteración	Duración	Actividad principal

Sprint 1	Levantamiento de Requerimientos	2 semanas	Planificación, entrevistas y diseño de BD conceptual
Sprint 2	Desarrollo del Sistema Web Backend	2 semanas	Creación de modelos, controladores y rutas
Sprint 3	Diseño y Desarrollo del Sistema Frontend	2 semanas	Diseño de vistas y autenticación de usuarios
Sprint 4	Desarrollo de Módulos del Sistema Frontend	2 semanas	Crear las vistas de los módulos y navegación entre vistas
Sprint 5	Conexión a APIs Frontend	2 semanas	Conexión del Frontend con el Backend para funcionalidades de la aplicación
Sprint 6	Pruebas y Features	2 semanas	Probar el funcionamiento del sistema, corrección de bugs y desarrollo de características adicionales.

Estos seis sprints, de dos semanas cada uno, abarcaron desde el levantamiento de requisitos hasta las pruebas finales.

Diagrama de clases



El diagrama de clases modela las principales entidades del sistema y muestra sus atributos más relevantes, así como las relaciones que existen entre ellas. En el nivel superior aparecen las clases Usuario y Rol, donde cada usuario se asocia a un rol que define su nivel de acceso. A continuación, se definen las especializaciones de usuario: Administrativo, Profesor, Representante Legal y Guardia, cada una con sus datos de identificación y contacto.

Por otro lado, las entidades Curso y Estudiante reflejan la estructura académica: cada curso puede agrupar a varios estudiantes, y cada estudiante está vinculado a un representante legal.

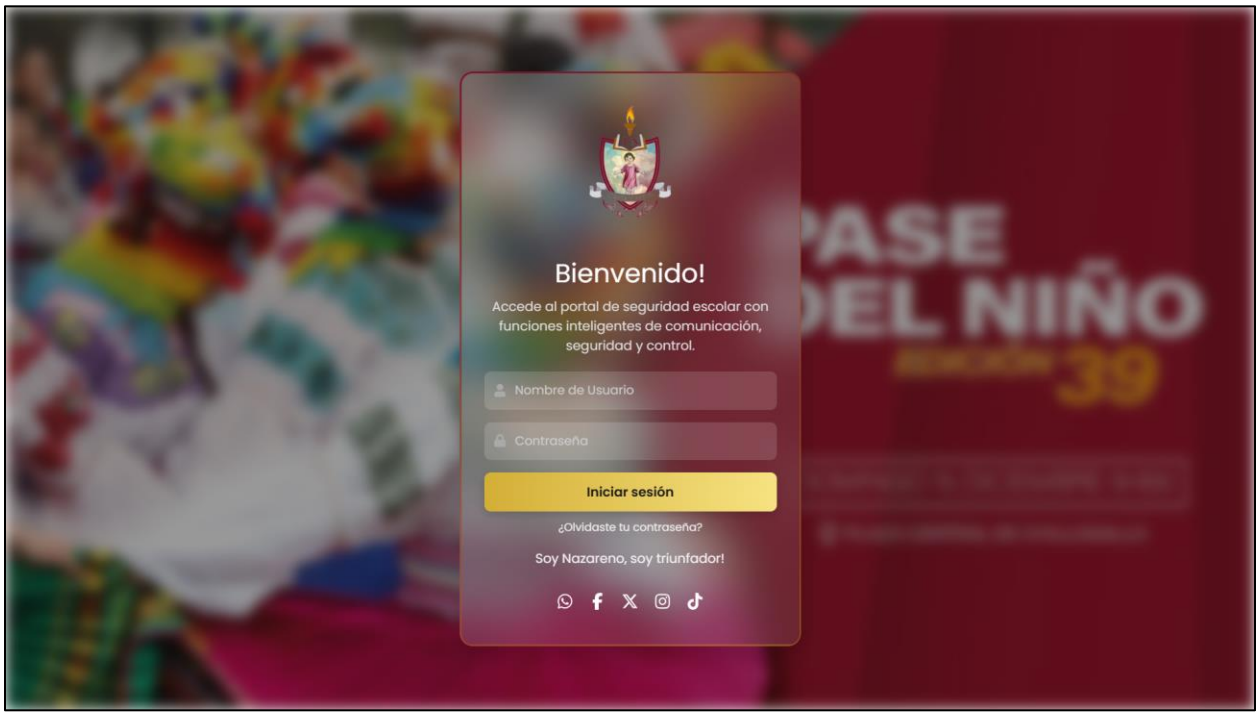
Código fuente

A continuación, se presentan la pantalla de inicio y las pantallas de cada módulo y como estas se ven relacionadas con el rol y el inicio de sesión.

En la pantalla de inicio de sesión se presentan los campos de relleno del formulario, el usuario es el nombre de usuario, no se debe agregar el @ ni el dominio. La contraseña será enviada al correo electrónico personal luego del registro del usuario en el sistema. Si las credenciales son válidas se podrá avanzar a las siguientes pantallas.

Figura 25

Pantalla de inicio de sesión.



Nota: Captura de pantalla de la pantalla inicial, el ingreso de las credenciales permite el acceso por rol.

Figura 26

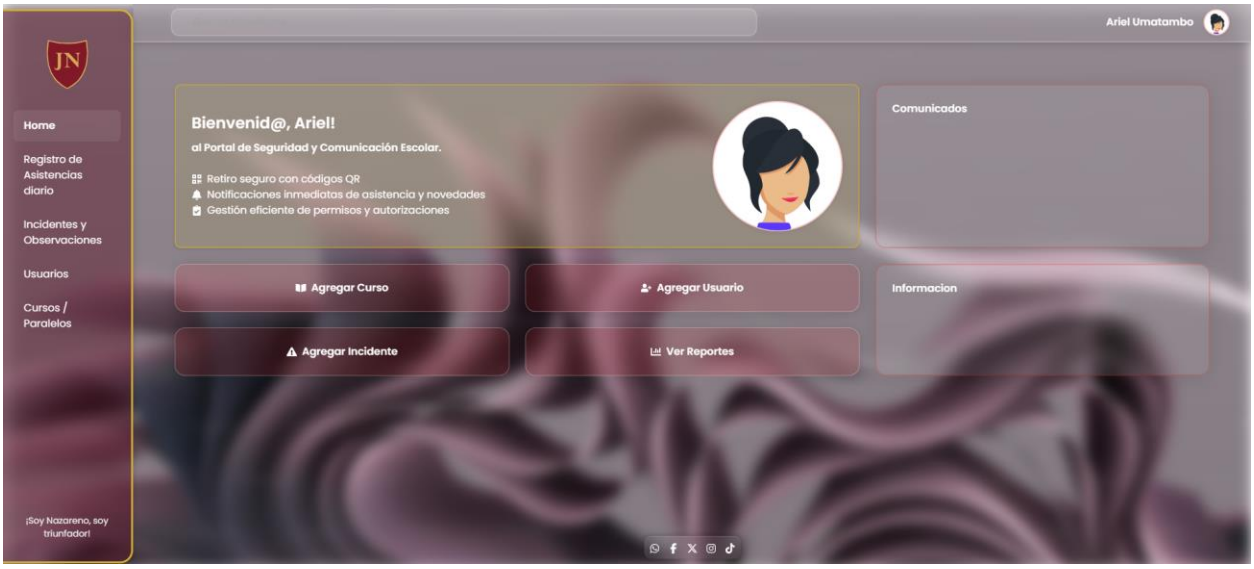
Código de inicio de sesión.

```
const handleSubmit = async (e) => {
  e.preventDefault();

  try {
    const response = await axios.post('http://localhost:3000/api/auth/login', {
      user_name: email,
      password
    });

    const { user, token } = response.data;
    login(user, token);

    switch (user.role) {
      case 'administrative':
        navigate('/admin');
        break;
      case 'professor':
        navigate('/professor');
        break;
      case 'legal_representative':
        navigate('/legal-representantive/');
        break;
      case 'guard':
        navigate('/guard');
        break;
      default:
        alert('Rol no reconocido');
    }
  } catch (err) {
    const backendMessage = err.response?.data?.message || 'Error desconocido';
    alert(backendMessage);
    console.error('Login error:', err);
  }
};
```

Nota: Captura del código de inicio de sesión que permite el logue por rol.

Figura 27

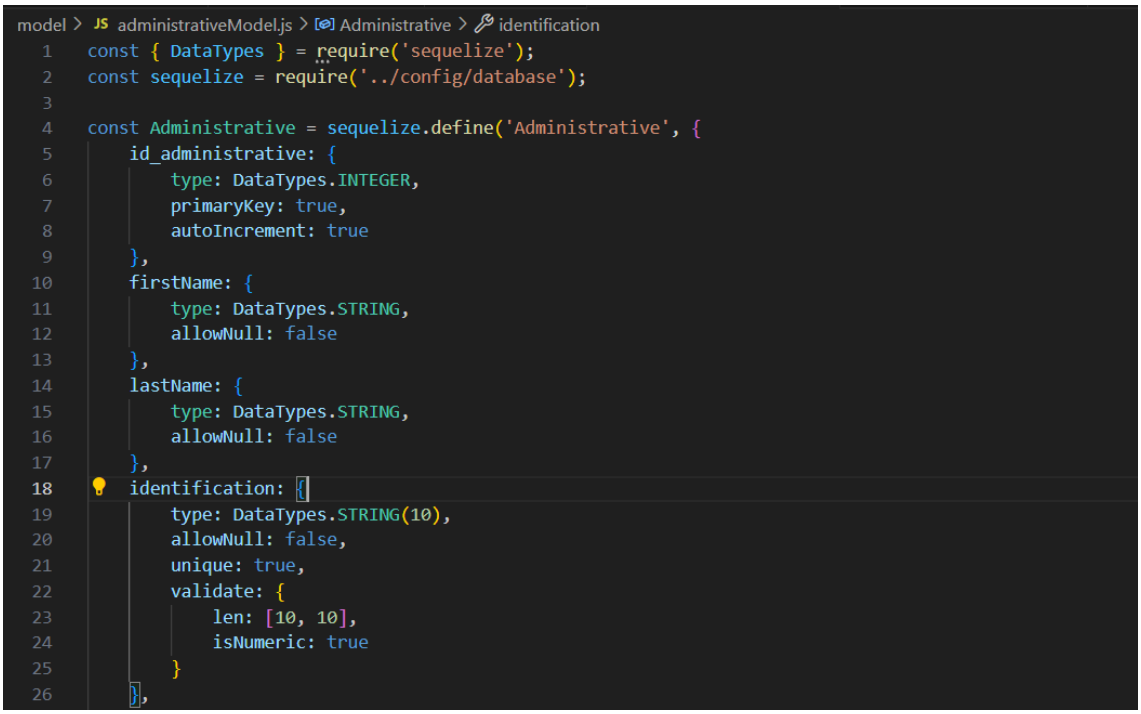
Interfaz del usuario Administrador



Nota: Captura de pantalla del módulo de administrador.

Figura 28

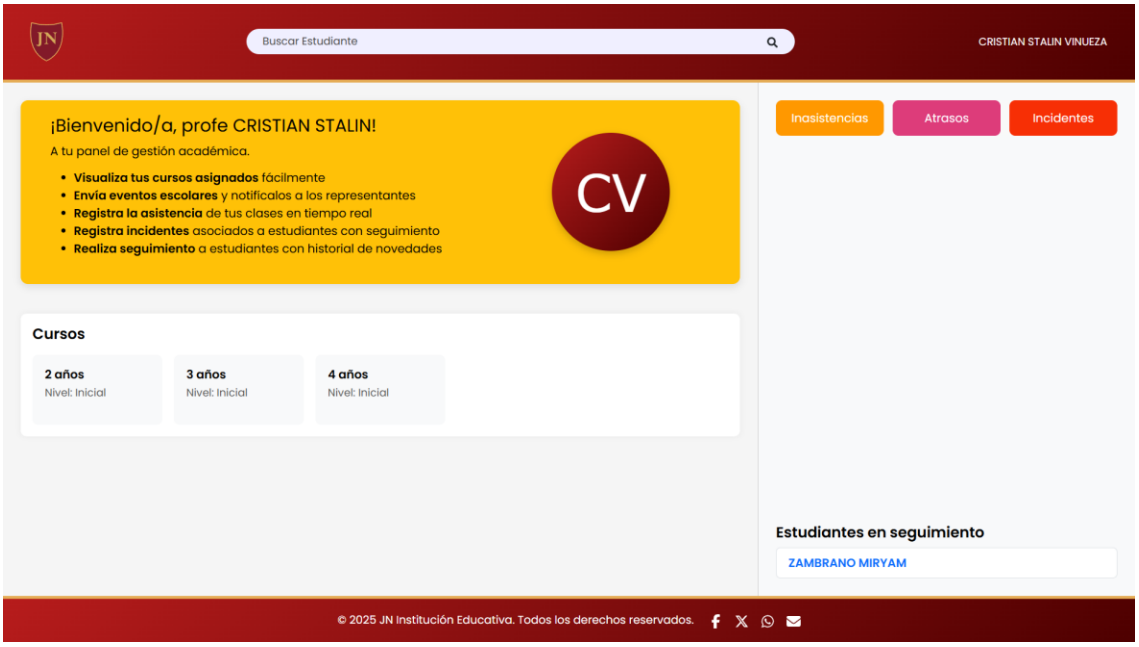
Código del modelo admisnitrative



Nota: Captura del código del modelo administrative.

Figura 29

Interfaz del usuario Profesor



Nota: Captura de pantalla del módulo de profesor.

Figura 30

Código para obtener cursos por id de profesor

```
exports.getCoursesByProfessor = async (req, res) => {
  const { id } = req.params;

  try {
    const professor = await Professor.findByPk(id, {
      include: {
        model: Course,
        as: 'courses',
        through: { attributes: [] }
      }
    });

    if (!professor) {
      return res.status(404).json({ message: 'Profesor no encontrado' });
    }

    return res.status(200).json(professor.courses);
  } catch (error) {
    console.error('Error al obtener cursos del profesor:', error);
    return res.status(500).json({ message: 'Error al obtener cursos del profesor', error: error.message });
  }
};
```

Nota: Captura del controlador para buscar los cursos asignados al profesor mediante su id.

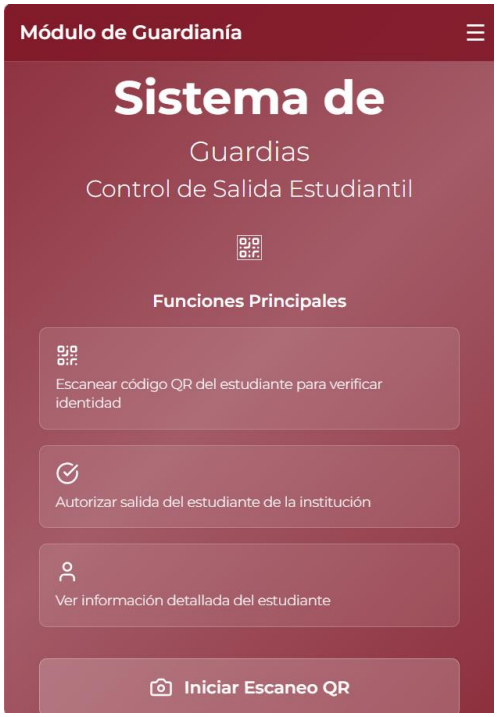
Figura 31

Interfaz del usuario Representante Legal

Nota: Captura de pantalla del módulo del representante legal en un dispositivo móvil.

Figura 32

Interfaz del usuario Guardia de Seguridad



Nota: Captura de pantalla del módulo del guardia de seguridad en un dispositivo móvil.

Figura 33

Captura del código de las rutas para las APIs

```
33
34  // ------------------- RUTAS API --------------------
35
36  app.use('/api/professor-courses', professorCourseRoute);
37  app.use('/api/students', studentRoute);
38  app.use('/api', administrativeRoute);
39  app.use('/api/courses', courseRoute);
40  app.use('/api/asistances', asistanceRoutes);
41  app.use('/api/professors', professorRoute);
42  app.use('/api/legal-representatives', legalRepresentativeRoute);
43  app.use('/api/incidents', incidentsRoute);
44  app.use('/api', roleRoute);
45  app.use('/api/auth', authRoute);
46  app.use('/api/news', newsRoute);
47  app.use('/api/guards', guardRoute);
48  app.use('/api', userRoutes);
```

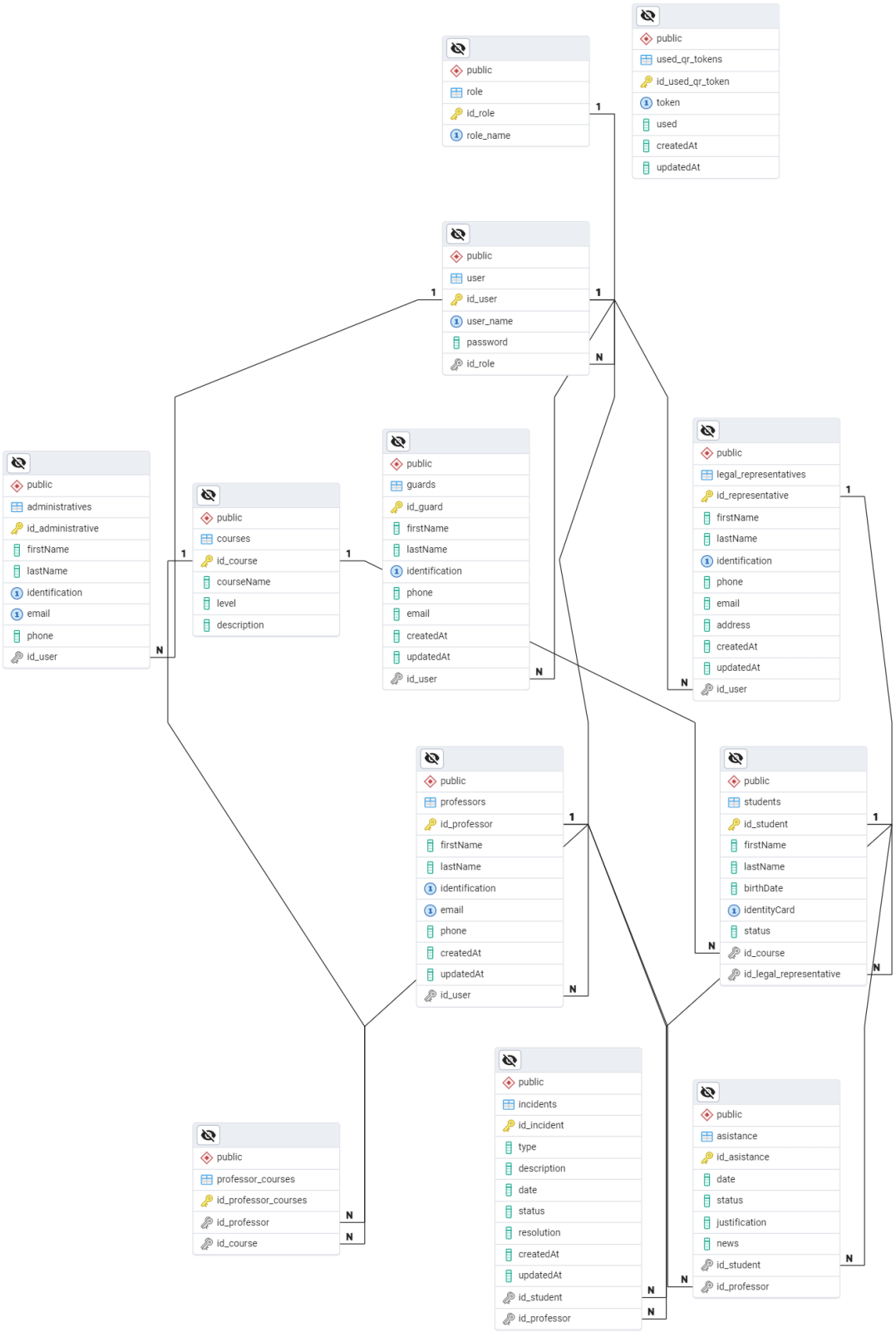
Nota: Captura de pantalla de index.js.

Modelo Base de datos

Tablas y columnas

- **user** (id_user PK, user_name, password, id_role FK)
- **role** (id_role PK, role_name)
- **administratives** (id_administrative PK, firstName, lastName, identification, email, phone, id_user FK)
- **professors** (id_professor PK, firstName, lastName, identification, email, phone, id_user FK)
- **legal_representatives** (id_representative PK, firstName, lastName, identification, phone, email, address, id_user FK)
- **guards** (id_guard PK, firstName, lastName, identification, phone, email, id_user FK)
- **courses** (id_course PK, courseName, level, description)
- **students** (id_student PK, firstName, lastName, birthDate, identityCard, status, id_course FK, id_legal_representative FK)
- **professor_courses** (id_professor_courses PK, id_professor FK, id_course FK)
- **asistance** (id_asistance PK, date, status, justification, news, id_student FK, id_professor FK)
- **incidents** (id_incident PK, type, description, date, status, resolution, id_student FK, id_professor FK)
- **used_qr_tokens** (id_used_qr_token PK, token, used)
- **news** (id_news PK, from_role, to_role, title, message, image_url, priority, is_active, expires_at, published_at, createdAt, updatedAt)

MER


public
used_qr_tokens
id_used_qr_token
token
used
createdAt
updatedAt
public
role
id_role
role_name
public
user
id_user
user_name
password
id_role
public
administratives
id_administrative
firstName
lastName
identification
email
phone
id_user
public
courses
id_course
courseName
level
description
public
guards
id_guard
firstName
lastName
identification
phone
email
createdAt
updatedAt
id_user
public
legal_representatives
id_representative
firstName
lastName
identification
phone
email
address
createdAt
updatedAt
id_user
public
professors
id_professor
firstName
lastName
identification
email
phone
createdAt
updatedAt
id_user
public
students
id_student
firstName
lastName
birthDate
identityCard
status
id_course
id_legal_representative
public
professor_courses
id_professor_courses
id_professor
id_course
public
incidents
id_incident
type
description
date
status
resolution
createdAt
updatedAt
id_student
id_professor
public
asistance
id_asistance
date
status
justification
news
id_student
id_professor

El MER representa gráficamente las tablas anteriores y sus claves foráneas. Destacan las siguientes relaciones:

- Un Rol puede estar asignado a varios Usuarios.
- Un Usuario se especializa en uno de los perfiles: Administrativo, Profesor, Representante Legal o Guardia.
- Un Curso agrupa a múltiples Estudiantes, y cada estudiante se vincula a un representante legal.
- Los Profesores y los Cursos se relacionan en forma de muchos-a-muchos mediante la tabla intermedia professor_courses.
- Cada Estudiante puede tener múltiples registros en Asistencia e Incidente, y cada uno de estos eventos es registrado por un profesor.
- La tabla used_qr_tokens se relaciona opcionalmente con el estudiante que utiliza el código QR.
- Los News se dirigen a los roles definidos para controlar quién recibe cada comunicado.
- Este modelo garantiza la integridad referencial y facilita la consulta eficiente de la información necesaria para el funcionamiento de la plataforma.

Diccionario de datos

Tabla 31

Usuarios

Columna	Tipo de Dato	Condiciones	Detalle
id_user	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Llave primaria del usuario
user_name	VARCHAR(50)	NOT NULL, UNIQUE	Nombre de usuario
password	VARCHAR(255)	NOT NULL	Contraseña del usuario

Nota: Diccionario de datos de la tabla usuarios

Tabla 32

Rol

Columna	Tipo de Dato	Condiciones	Detalle
id_role	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Llave primaria del rol
role_name	VARCHAR(30)	NOT NULL, UNIQUE	Nombre del rol

Nota: Diccionario de datos de la tabla rol

Tabla 33

Administrador

Columna	Tipo de Dato	Condiciones	Detalle
id_administrative	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Llave primaria del administrador
firstName	VARCHAR(255)	NOT NULL	Nombre del administrador
lastName	VARCHAR(255)	NOT NULL	Apellido del administrador
identification	VARCHAR(10)	NOT NULL, UNIQUE	Número de cédula
email	VARCHAR(255)	NOT NULL, UNIQUE	Dirección de correo electrónico
phone	VARCHAR(255)	NULLABLE	Número de teléfono

Nota: Diccionario de datos de la tabla administrative

Tabla 34

Profesor

Columna	Tipo de Dato	Condiciones	Detalle
id_professor	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Llave primaria del profesor
firstName	VARCHAR(255)	NOT NULL	Nombre del profesor
lastName	VARCHAR(255)	NOT NULL	Apellido del profesor
identification	VARCHAR(10)	NOT NULL, UNIQUE	Número de cédula
email	VARCHAR(255)	NOT NULL, UNIQUE	Dirección de correo electrónico
phone	VARCHAR(255)	NULL	Número de teléfono (opcional)

Nota: Diccionario de datos de la tabla professor

Tabla 35

Estudiante

Columna	Tipo de Dato	Condiciones	Detalle
id_student	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Llave primaria del estudiante
firstName	VARCHAR(255)	NOT NULL	Nombre del estudiante
lastName	VARCHAR(255)	NOT NULL	Apellido del estudiante
birthDate	DATE	NOT NULL	Fecha de nacimiento

identityCard	VARCHAR(10)	NOT NULL, UNIQUE	Número de cédula
status	ENUM ('active','inactive')	NOT NULL, DEFAULT 'active'	Estado del estudiante
id_course	INTEGER	NOT NULL, FOREIGN KEY	Curso al que pertenece el estudiante

Nota: Diccionario de datos de la tabla student

Tabla 36

Representante legal

Columna	Tipo de Dato	Condiciones	Detalle
id_representative	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Llave primaria del representante legal
firstName	VARCHAR(255)	NOT NULL	Nombre del representante legal
lastName	VARCHAR(255)	NOT NULL	Apellido del representante legal
identification	VARCHAR(10)	NOT NULL, UNIQUE	Número de cédula
phone	VARCHAR(255)	NULLABLE	Número de teléfono
email	VARCHAR(255)	NOT NULL	Dirección de correo electrónico
address	VARCHAR(255)	NULLABLE	Dirección

Nota: Diccionario de datos de la tabla legal_representatives

Tabla 377

Guardia

Columna	Tipo de Dato	Condiciones	Detalle
id_guard	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Llave primaria del guardia
firstName	VARCHAR(255)	NOT NULL	Nombre del guardia
lastName	VARCHAR(255)	NOT NULL	Apellido del guardia
identification	VARCHAR(10)	NOT NULL, UNIQUE	Número de cédula
phone	VARCHAR(255)	NULLABLE	Número de teléfono
email	VARCHAR(255)	NOT NULL	Dirección de correo electrónico

Nota: Diccionario de datos de la tabla guard

Tabla 38

Curso

Columna	Tipo de Dato	Condiciones	Detalle
id_course	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Llave primaria del curso
courseName	VARCHAR(255)	NOT NULL	Nombre del curso
level	VARCHAR(255)	NOT NULL	Nivel del curso
description	VARCHAR(255)	NOT NULL	Descripción del curso

Nota: Diccionario de datos de la tabla course

Tabla 39

Profesor_curso

Columna	Tipo de Dato	Condiciones	Detalle
id_professor_courses	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Llave primaria
ProfessorId	INTEGER	FOREIGN KEY (professors)	Llave foranea Id Profesor
CourseId	INTEGER	FOREIGN KEY (courses)	Llave foranea Id Estudiante

Nota: Diccionario de datos de la tabla profesor_courses

Tabla 40

Asistencia

Columna	Tipo de Dato	Condiciones	Detalle
id_asistance	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Llave primaria de asistencia
date	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Fecha y hora de la asistencia
status	ENUM ('present','absent', 'late')	NOT NULL	Estado de la asistencia
justification	VARCHAR(200)	NULLABLE	Justificación de atraso/falta
news	VARCHAR(200)	NULLABLE	Notificación de atraso/falta

Nota: Diccionario de datos de la tabla asistance

Tabla 41

Incidentes

Columna	Tipo de Dato	Condiciones	Detalle
id_incident	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Llave primaria del incidente
type	ENUM('disciplinary', 'academic','medical','security')	NOT NULL	Tipo de incidente
description	TEXT	NOT NULL	Descripción detallada del incidente
date	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Fecha y hora del incidente

status	ENUM('pending','resolved')	NOT NULL, DEFAULT 'pending'	Estado de resolución del incidente
resolution	TEXT	NULLABLE	Resolución del incidente

Nota: Diccionario de datos de la tabla incident

Capitulo IV

Pruebas y resultado

Este capítulo detalla las pruebas realizadas al sistema de gestión escolar desarrollado para la Unidad Educativa Jesús de Nazareth, con el propósito de validar su funcionalidad, estabilidad y cumplimiento de los requisitos planteados en los capítulos anteriores. Se aplicaron pruebas funcionales por módulo y pruebas de validación con usuarios finales, registrando los resultados mediante tablas de iteración por versión.

Pruebas realizadas al software

Se aplicaron pruebas funcionales a cada uno de los módulos del sistema (Administrador, Profesor, Representante Legal y Guardia), considerando tanto los procesos principales como las tareas que realiza cada actor. Las pruebas se desarrollaron en iteraciones incrementales, incorporando mejoras y ajustes con base en observaciones de los usuarios durante el proceso.

Las principales funcionalidades evaluadas fueron:

- Registro y gestión de usuarios.
- Registro de asistencia estudiantil.
- Registro de comportamiento.
- Solicitudes y validación de retiro de estudiantes.
- Acceso diferenciado por rol.

A continuación, se presentan los resultados organizados por módulo y versión.

Tablas de iteración de cada prueba con sus versiones

Tabla 42

Prueba de roles e interfaces diferenciadas

Módulo	**Iteración**	**Prerrequisito**	**Actor**	**Proceso**	**Subproceso**	**Tarea**	**Resultado alcanzado**	**Observaciones**
Usuarios	5	Arreglo de roles en la aplicación	Usuario	Diferenciar roles	Visualización por rol	Visualización de interfaces	100%	Se visualizan correctame

						para usuario y administrador		nte ambas interfaces

Nota: Esta tabla muestra la iteración 5 roles e interfaces diferenciadas

Tabla 43

Prueba del Registro de Asistencia

Módulo	Iteración	Prerrequisito	Actor	Proceso	Subproceso	Tarea	Resultado alcanzado	Observaciones
Profesores	6	Ingresar al sistema con el rol de Profesor	Profesor	Registrar asistencias en la interfaz	Visualización de asistencia de todo el curso	Registrar asistencia de estudiantes	100%	Se registra correctamente la asistencia de cada estudiante.

Nota: Esta tabla muestra la iteración 6 registro de asistencia

Tabla 44

Prueba del Registro de Incidencias

Módulo	Iteración	Prerrequisito	Actor	Proceso	Subproceso	Tarea	Resultado alcanzado	Observaciones
Profesores	6	Ingresar al sistema con el rol de Profesor	Profesor	Registrar incidencias en la interfaz	Visualización de incidencias de todo el curso	Registrar incidencias de estudiantes	100%	Se registra correctamente la incidencia de un estudiante

Nota: Esta tabla muestra la iteración 6 registro de incidencias

Tabla 45

Prueba del Registro y Gestión de Usuarios en el módulo de Administrador

Módulo	Iteración	Prerrequisito	Actor	Proceso	Subproceso	Tarea	Resultado alcanzado	Observaciones
Administrador	5	Ingresar al sistema con el rol de Administrador	Administrador	Registrar Cursos	Validación de campos	Registrar cursos y paralelos	100%	Se registra correctamente los cursos y paralelos

Administrador	5	Ingresar al sistema con el rol de Administrador y haber creado al menos un curso.	Administrador	Registrar Profesores	Validación de creación de profesor con cursos y envío de credenciales por correo electrónico	Registra un profesor asociado a un curso	100%	Se registra correctamente el profesor
Administrador	6	Ingresar al sistema con el rol de Administrador y haber creado al menos un curso.	Administrador	Registrar Estudiante y Representante Legal asociado	Validación de creación de estudiante asociado a un profesor y a un curso y envío de credenciales por correo electrónico	Registra estudiante y representante legal	100%	Se registra correctamente el estudiante y el representante legal
Administrador	6	Ingresar al sistema con el rol de Administrador.	Administrador	Registrar Personal de Seguridad de la institución	Envío de credenciales por correo electrónico	Registra a una persona de seguridad	100%	Se registra correctamente el personal de seguridad

Nota: Esta tabla muestra la iteración 7 gestión de usuarios y roles

Tabla 46

Prueba de la generación de QR para el retiro del estudiante

Módulo	**Iteración**	**Prerrequisito**	**Actor**	**Proceso**	**Subproceso**	**Tarea**	**Resultado alcanzado**	**Observaciones**
Representante Legal	6	Ingresar al sistema con el rol de	Representante Legal	Generar un QR de uso	Visualización del QR	Generar un QR con informa	100%	Se genera correctamente el QR para la

		Representante Legal		único para la salida de la institución del estudiante		ción del estudiante		visualización y descarga por parte del Representante Legal

Nota: Esta tabla muestra la iteración 6 generación del QR

Tabla 47

Prueba del escaneo y validación del QR para la salida del estudiante

Módulo	**Iteración**	**Prerrequisito**	**Actor**	**Proceso**	**Subproceso**	**Tarea**	**Resultado alcanzado**	**Observaciones**
Personal de Seguridad	6	Ingresar al sistema con el rol de Personal de Seguridad	Personal de Seguridad	Escanear el QR	Validación del QR y visualización de la información básica del Estudiante	Escanear un QR asociado a un estudiante para la salida de la institución	100%	Se valida correctamente el QR, se actualiza su uso en la base de datos y se muestra la información del estudiante.

Nota: Esta tabla muestra la iteración 7 validación de QR

Pruebas de rendimiento

Para la realización de pruebas de rendimiento se ha realizado una simulación donde se puede observar la cantidad de estrés que soporta en nuestro equipo para pruebas. La prueba consiste en la generación de los códigos QR, actividad que se realizara diariamente para retirar al estudiante de la institución. Para esta prueba se usaron 300 usuarios con el rol de representante legal tratando de generar 300 códigos QR para el retiro de cada estudiante.

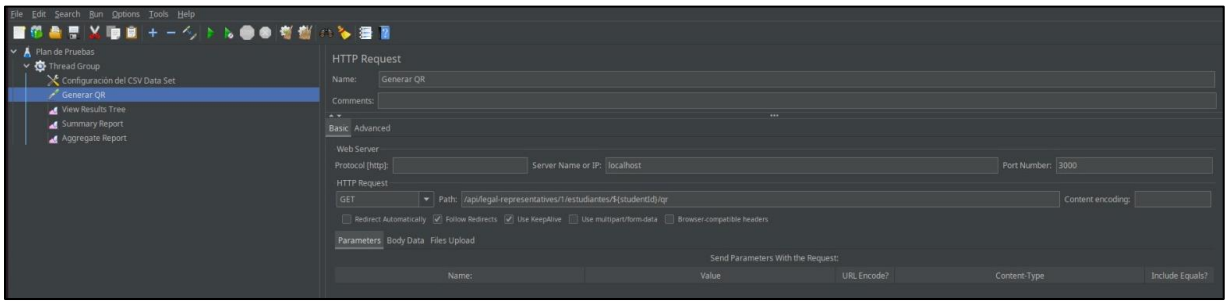
Figura 34

Prueba de saturación y estrés – Seteo de 300 usuarios

Nota: Carga de la aplicación se configura el thead group con un numero de 300.

Figura 35

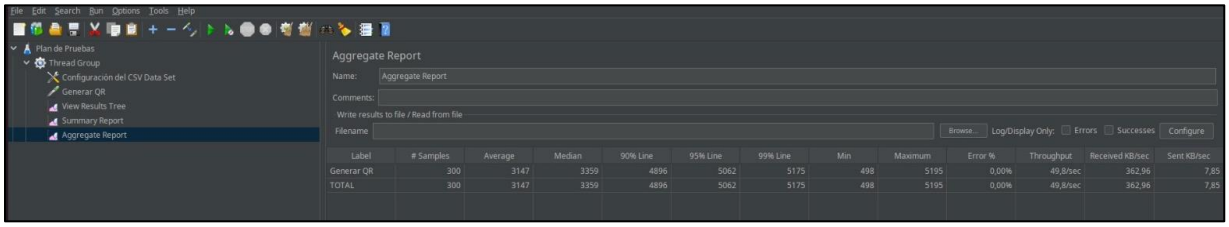
Prueba de saturación y estrés – Seteo de la API



Nota: Se utiliza la API para generar los códigos QR, el padre de familia debe estar registrado en la base y tener asignado un estudiante.

Figura 36

Prueba de carga – Resumen de los 300 métodos GET a la API

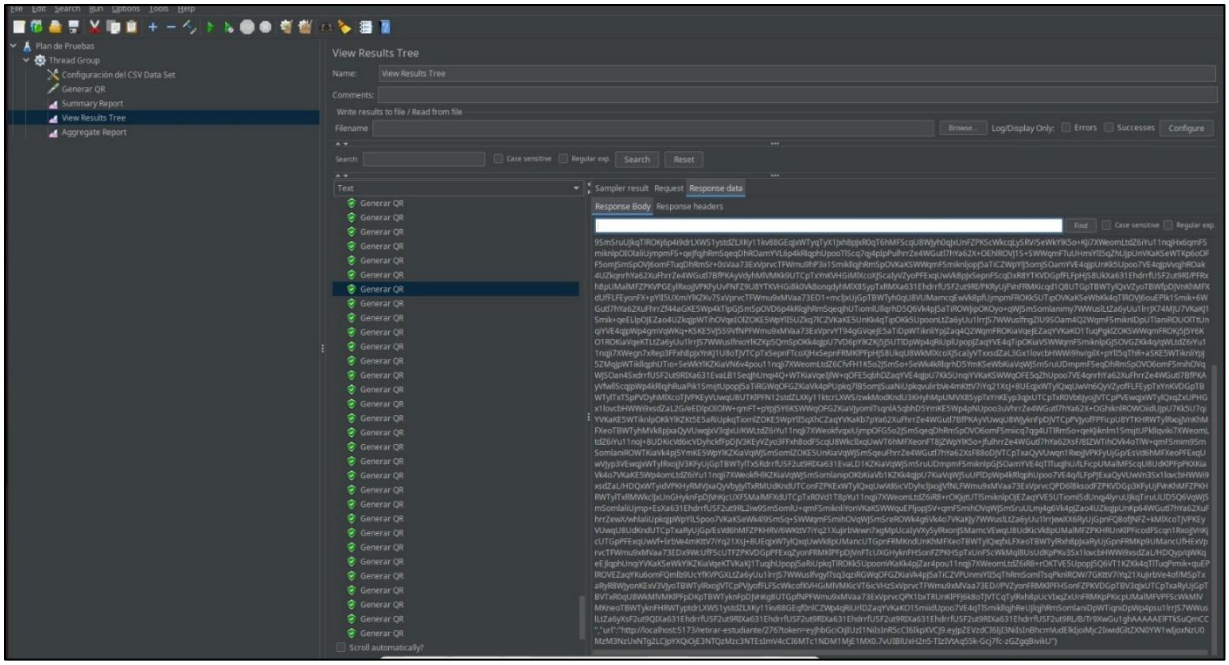


Nota: Resumen de los 300 métodos consumidos en la API.

En los resultados de la prueba se puede observar que el tiempo de respuesta de la página con 300 usuarios es de 3147 milisegundos. Además, se verifica el factor de Error es de 0%. Es decir, la aplicación reacciona de manera normal con el número de concurrencia aplicado

Figura 37

Prueba de saturación y estrés – Verificación de la creación del JWT



Nota: Se selecciona la prueba del estudiante 276 para verificar la creación del JWT, en cual se puede ver la url de la API y el Id del estudiante.

Figura 38

Prueba de saturación y estrés – Verificación del código QR y escaneo del mismo

Nota: Para verificar la autenticidad del JWT generado en la prueba, se selecciona la prueba 299, se copia el código del JWT y se lo inserta en un HTML. Posteriormente se escanea el código QR con el celular para verificar su autenticidad.

Conclusiones

La identificación y análisis de los requerimientos técnicos y administrativos permitieron comprender las particularidades operativas de la Unidad Educativa Jesús de Nazareth. Gracias a este análisis se diseñó una solución que responde a las necesidades específicas de control de asistencias, autorizaciones de retiro y gestión de comportamiento estudiantil, alineándose con la realidad tecnológica y organizacional de la institución.

La implementación del módulo de autorización de retiros con códigos QR mejoró la seguridad del proceso, reduciendo la posibilidad de retiros no autorizados. Este mecanismo no solo facilita el control por parte del personal de seguridad, sino que también genera confianza en los padres al tener un sistema confiable y moderno para validar la autorización de retiro de los estudiantes.

El desarrollo del módulo de asistencias e incidentes logró establecer una trazabilidad efectiva del comportamiento estudiantil. La incorporación de notificaciones automáticas y formularios de justificación para padres agilizó la comunicación institución-representante legal, permitiendo una gestión más proactiva y eficiente ante faltas o situaciones disciplinarias.

Recomendaciones

Se recomienda que la institución realice capacitaciones periódicas a los usuarios del sistema, especialmente a docentes y representantes legales, para asegurar un uso adecuado y continuo de la plataforma. Asimismo, se sugiere establecer un proceso de retroalimentación que permita recoger opiniones de los usuarios y así orientar futuras mejoras o nuevas funcionalidades.

Se recomienda continuar el desarrollo evolutivo del sistema mediante la integración de funcionalidades adicionales como análisis estadístico de asistencias, generación de reportes automáticos, y una app móvil nativa para representantes. Esto permitirá una mayor adopción del sistema, mejorando la experiencia del usuario y facilitando la toma de decisiones por parte de directivos.

Se recomienda integrar pruebas automatizadas (unitarias y de integración) utilizando herramientas como Jest para el backend y Cypress para el frontend. Estas mejoras no solo facilitarán el

mantenimiento y evolución del sistema, sino que también aumentarán su fiabilidad ante futuras ampliaciones.

Referencias bibliográficas

Cohn, M. (2010). Succeeding with Agile: Software Development Using Scrum [Tener éxito con ágil: Desarrollo de software utilizando Scrum]. Addison-Wesley Professional.

Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th ed.) [Ingeniería de software: Un enfoque práctico (9.ª ed.)]. McGraw-Hill Education.

Rubin, K. S. (2013). Essential Scrum: A Practical Guide to the Most Popular Agile Process [Scrum esencial: Una guía práctica para el proceso ágil más popular]. Addison-Wesley Professional.

Fowler, M. (2002). Patterns of Enterprise Application Architecture [Patrones de arquitectura para aplicaciones empresariales]. Addison-Wesley Professional.

Richardson, L., & Ruby, S. (2013). RESTful Web APIs [APIs web RESTful]. O'Reilly Media.