PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

INGENIERÍA EN SISTEMAS DE INFORMACIÓN

TEMA:

DESARROLLO DE UNA APLICACIÓN WEB PARA EL CONSULTORIO JURÍDICO GRATUITO DE LA PUCE – GESTIÓN DE USUARIOS Y PRIMERAS CONSULTAS

AUTOR:

FRANCISCO JOSÉ VELASCO REYES

DIRECTOR:

ING. GUIDO OCHOA

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS DE INFORMACIÓN

QUITO DM, ABRIL DE 2025

DEDICATORA

Este trabajo va dirigido a mis padres, quienes me han guiado en toda mi vida y por ellos he tenido la oportunidad de estudiar lo que me ha apasionado desde pequeño. También quiero dedicar este trabajo a mis hermanos y mi amigo Elías Guerrero, que en el poco tiempo que pude compartir con él en este mundo, me dio su apoyo y amistad.

AGRADECIMIENTO

En primer lugar, agradezco a Dios por guiarme durante todas las etapas de mi vida hasta este momento y por su bendición en mi vida. A mis padres, quienes me han educado con sabiduría e inteligencia para poder saber cómo actuar y por ellos soy lo que soy.

Quiero agradecer al Ing. Guido Ochoa por su guía y ayuda como el director de este trabajo de titulación, al Ing. Nelson Salgado por su gestión durante el desarrollo de todo el proyecto, a todos mis profesores de la carrera quienes me han trasmitido su conocimiento y sobre todo experiencias que servirán en la vida profesional. A mis amigos de la carrera quienes me inspiraron a continuar con este proyecto y todos mis compañeros.

Finalmente, gracias a Alejandro Taboada Sánchez (ATS) por brindar su conocimiento.

RESUMEN

En los últimos años, el Consultorio Jurídico Gratuito de la PUCE (CJGP) enfrenta diferentes retos en la gestión de sus procesos internos, en este sentido se comprende la necesidad de una aplicación web que les permita gestionar todas sus áreas fácil y eficientemente; a pesar de que el Consultorio Jurídico ya cuenta con un sistema, hay ciertos procesos que han ido evolucionando a lo largo del tiempo o directamente no están contemplados en éste, incrementando su complejidad. Naturalmente, al ser una aplicación web que busca reemplazar al sistema actual, existen varios procesos para automatizar, por lo que este trabajo será un módulo que abordará el diseño general U.I (User Interface) de la aplicación, la Gestión de Usuarios donde se controla a todos los usuarios del sistema, Gestión de Roles/Permisos que permite autorizar o revocar el acceso de interfaz a un determinado rol de la aplicación y el área de Primeras Consultas, siendo el primer paso donde se recibe al usuario, se registra su información y verifica si su caso es apto para darle patrocinio o, por el contrario, brindarle una asesoría respectiva.

PALABRAS CLAVE: Aplicación Web; Consultorio Jurídico Gratuito; Gestión de Usuarios; Gestión de Roles y Permisos; Primeras Consultas; Diseño de Interfaz (UI).

ABSTRACT

Nowadays, the PUCE Free Legal Clinic (CJGP) has faced several challenges in managing its internal processes. While a system is already in place, certain processes have evolved or are not currently supported, increasing its operational complexity. This has led to the need for a web-based application to enable efficient management of all areas. As this new application seeks to replace the current system, several processes require automation. Therefore, this work addresses the development of a foundational module that includes: the application's overall UI (User Interface) design; the User Management to control all system users; a Role-Based Permission Management to authorize or revoke interface access for specific roles; and the Initial Consultations area. This final component serves as the first step in the client workflow, responsible for registering user information and verifying whether a case qualifies for legal sponsorship or, alternatively, for advisory services.

KEY WORDS: Web Application; PUCE Free Legal Clinic; User Management; Role-Based Permission Management; Initial Consultations; User Interface Design.

ÍNDICE

CONTENIDO

ÍNDICE DE FIGURAS Y GRÁFICOS

ÍNDICE DE TABLAS

CAPÍTULO I: INTRODUCCIÓN

1.1 Planteamiento del problema

Actualmente, el Consultorio Jurídico Gratuito de la PUCE enfrenta desafíos en la gestión de sus procesos internos, como resultado, la resolución de estos casos se convierte en una carga de tiempo para los estudiantes incluyendo el hecho de que cada caso en particular tiene respectivo nivel de complejidad. Si bien existe un sistema en los CJGP, hay ciertos aspectos que no están contemplados en éste, lo que puede limitar en cierto grado la calidad del servicio, siendo un problema que se complica más debido a la alta demanda de personas en esta área y que buscan soluciones rápidas y efectivas.

En función de esta problemática se plantea la siguiente pregunta principal de investigación:

- ¿Cuál sería la forma más efectiva de desarrollar una aplicación web en el Consultorio Jurídico Gratuito de la PUCE?

Y las siguientes preguntas secundarias:

- ¿Qué características debe tener la aplicación web para asegurar un seguimiento eficiente de los casos?
- ¿Cuáles son las principales necesidades de los usuarios en el consultorio jurídico que la aplicación web debe abordar?
- ¿Qué dificultades enfrentan actualmente los asesores legales al gestionar casos y cómo puede la aplicación web propuesta resolverlas?

1.2 Justificación

El Consultorio Jurídico Gratuito (CJG) ha sido un medio importante para los estudiantes de la facultad de Derecho en la PUCE, no solo permite mejorar sus habilidades, sino que también

ayuda a personas que no tienen los suficientes recursos económicos para recibir una asistencia y asesoramiento adecuado. Naturalmente, el consultorio jurídico de la PUCE cuenta con una alta demanda de casos y/o situaciones legales que afectan directamente la vida cotidiana de la persona involucrada, en consecuencia, la gestión interna de esta área en particular es un reto que debe ser abordado. En este sentido se comprende la necesidad del desarrollo de una aplicación web para los CJGP con el fin de convertirse en una herramienta de apoyo para los futuros profesionales de esta Facultad, mejorando sus servicios y las necesidades de los usuarios que más lo necesiten.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un módulo de la aplicación web para los Consultorios Jurídicos Gratuitos de la PUCE que permita gestionar eficientemente los casos en el área de Primeras Consultas, los usuarios del sistema y sus permisos.

1.3.2 Objetivos Específicos

- Identificar las nuevas necesidades de automatización y mejora para facilitar la gestión en Primeras Consultas.
- Diseñar y codificar una vista intuitiva para la aplicación web, gestión de usuarios, roles y la creación de los casos en primeras consultas, incluyendo los elementos para llevar el seguimiento de estos hasta llegar al patrocinio y el reporte de ésta.
- Asegurar la protección de la aplicación web mediante la implementación de un sistema de autenticación y control de acceso, utilizando Json Web Token (JWT) para la gestión de sesiones, Bcrypt para la encriptación de contraseñas y Pinia para la autorización o prohibición de vistas.

- Mejorar la funcionalidad y usabilidad de la aplicación a través de un proceso iterativo de prototipado, incorporando la retroalimentación del usuario final hasta obtener la aprobación del prototipo final.
- Probar las funcionalidades del módulo de primeras consultas, el sistema de autenticación (Gestión de usuarios) y los roles/permisos que tiene cada usuario del sistema (Gestión de roles).

1.4 Alcance

Este trabajo de titulación tiene como alcance el diseño U.I (User Interface) inicial de la aplicación web (inicio, menú, login o inicio de sesión, configuración, etc.), el desarrollo del módulo de Primeras Consultas, la gestión de usuarios y sus roles/permisos para el Consultorio Jurídico Gratuito de la PUCE, utilizando la metodología del prototipado evolutivo; además de los estándares, herramientas de desarrollo y lineamientos por parte de la Dirección Informática de la PUCE (Departamento de Desarrollo), para cumplir correctamente con las fases de la metodología y el objetivo general del proyecto. Finalmente, este trabajo de titulación no contempla la puesta en producción dentro de los Consultorios Jurídicos y el diseño responsive de la aplicación web.

CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA

2.1 Base teoría de Consultorios Jurídicos PUCE

El Consultorio Jurídico de la PUCE es una dependencia de la Facultad de Jurisprudencia la cual se encarga de brindar servicios de asesoría y patrocinio legal gratuito en diversas áreas (Civil, Penal, Familiar, Movilidad Humana), con el objetivo de ayudar a aquellas personas que estén pasando por situaciones económicas delicadas, culturales o de carácter social, acudiendo al servicio en defensa de sus derechos. Es así como también se encargan de la formación en experiencia práctica a los estudiantes de la facultad que busquen realizar sus prácticas preprofesionales, mejorando sus habilidades como un paso previo a su vida laboral (Pontificia Universidad Católica del Ecuador [PUCE], 2023).

2.2 Metodología de desarrollo de software

Una metodología de desarrollo de software consiste en una serie de pasos o principios que permiten el desarrollo riguroso de un producto de software, guiando el proceso de creación desde la ingeniería de requerimientos hasta el despliegue de forma general (Santander Universidades, 2020). Cabe destacar que una metodología de software bien realizada (siempre que cumpla los requisitos estrictamente) permite garantizar la calidad del producto de software, esto se debe a la organización estructurada que le da al proyecto de desarrollo ya que se divide por fases y dependiendo la metodología, es posible iterar cierta fase (revisar y corregir) o la posibilidad de regresar a una fase anterior según las circunstancias del proyecto y las necesidades del cliente. Finalmente, existen metodologías de tipo tradicional y ágiles, donde las primeras se usan en proyectos con requerimientos claros y definidos, haciendo énfasis en el diseño y la documentación, pues lo que se busca es la baja probabilidad de cambios; por otra parte, en las metodologías ágiles

se prioriza la flexibilidad e involucramiento del cliente en las fases del desarrollo en iteraciones cortas.

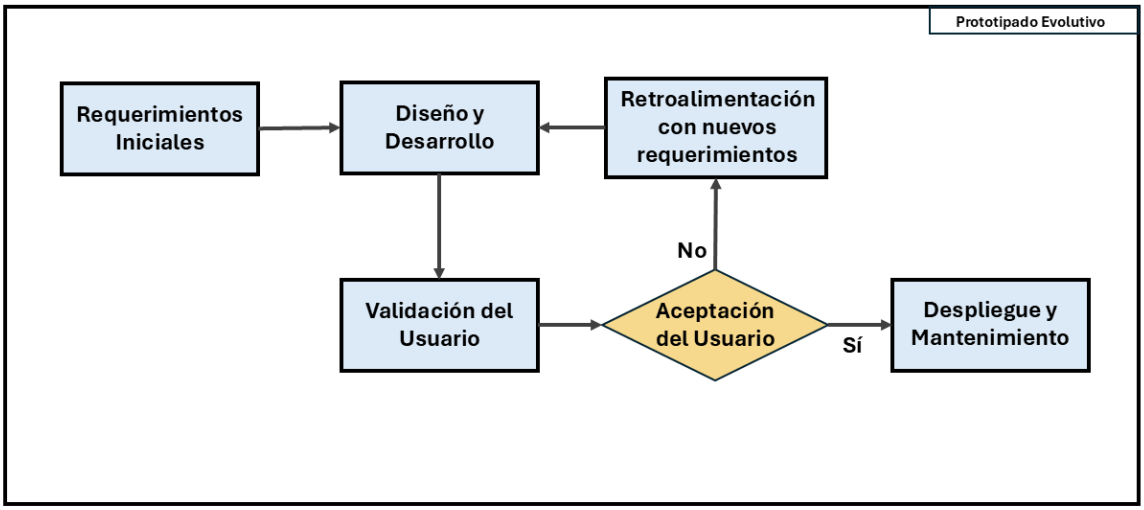
2.2.1 Prototipado Evolutivo

Es una metodología de desarrollo de software ágil en la cual se desarrolla un prototipo inicial de rápida construcción que gradualmente va evolucionando conforme se tienen comentarios de los stakeholders (todos los involucrados), donde se realizan las respectivas correcciones de mejora o se desarrollan los nuevos requerimientos solicitados y se le presenta al usuario hasta llegar al producto de software final (Delgado y Diaz, 2021). De este modo, el prototipo se refina con múltiples iteraciones gracias a la retroalimentación (feedback), siendo flexible a los cambios e involucrando constantemente al cliente en el proceso.

Dentro de este marco, se crea una versión inicial de un sistema o aplicación con fines de demostración o prueba; este es un prototipo de requerimientos que implica la creación de una implementación parcial del sistema para aprender sobre sus requisitos. De esa forma, se construye lo más rápido posible y se entrega a usuarios, clientes o stakeholders para que lo experimenten y a partir de esta presentación, ellos proporcionan una retroalimentación sobre lo que les gustó, no les gustó o qué se podría añadir como nuevo requerimiento, información que se captura en la documentación de especificación de requerimientos para el desarrollo del sistema real, siendo una característica útil cuando los requerimientos no se conocen bien o cambian constantemente. De esa forma, el prototipo evoluciona y mejora (en concordancia con su nombre), siendo de forma iterativa el proceso y llegando a convertirse en el producto final. Si bien tiene desventajas como no saber cuántas iteraciones son necesarias en caso de que no haya formalidad en el alcance y planificación del proyecto, esta metodología es eficiente por su enfoque y definición clara de roles y flujos de trabajo.

Figura 1

Modelo del Prototipado Evolutivo



Nota: La figura representa las fases del Prototipado Evolutivo. Fuente: Elaboración propia.

2.3 Lenguajes de programación

Consiste en una serie de instrucciones/operaciones que permiten expresar algoritmos en forma de código y así la computadora pueda procesarlo en forma de un programa informático, con la posibilidad de crear soluciones simples en consola (como una calculadora simple) hasta software robusto y complejo lleno de varias funcionalidades que facilitan la vida de los usuarios finales dependiendo su propósito (aplicaciones, juegos, sistemas de información, etc.), creados por medio de un entorno de desarrollo integrado (IDE) (Cilsa, 2023). Naturalmente, los lenguajes de programación están definidos por su sintaxis y semántica, donde la sintaxis consiste en las reglas de escritura que tiene el código y la semántica el significado de cada instrucción, por lo que cada lenguaje cuenta con sus propias herramientas, ecosistema y ventajas que lo destacan para determinados propósitos, ya se sea desarrollo web, bases de datos, aplicaciones móviles. Adicionalmente, existen lenguajes de programación compilados (C++, Java, C#) e interpretados

(Python, JavaScript) y según sus características algunos pueden ser tipados (donde el tipo de dato en la variable es definido y no cambia), pero en esencia todos cumplen su objetivo inicial, el cual es la comunicación con la máquina y al darle instrucciones para solucionar problemas de forma estructurada y sistemática.

2.3.1 JavaScript

Es un lenguaje de programación de alto nivel de código abierto (ECMAScript), donde sus instrucciones son interpretadas por el navegador, de esa forma, permite crear software sin necesidad de compilar el código y darles funcionalidad a las páginas web (desde efectos, animaciones e interacciones) o programar el lado del servidor back-end (usando el entorno de NodeJS) lo que la convierte en una herramienta robusta y completa (Lopez, 2023). En ese sentido, JavaScript es un lenguaje muy versátil en cuanto a sus características técnicas, debido a que es multiparadigma ya que ajusta a la programación orientada a objetos, funcional, y otras menos comunes como la imperativa/procedimental, siendo una característica que hace el código más flexible y reutilizable.

Cabe considerar, por otra parte que JavaScript destaca por su capacidad de manipular las tareas asíncronas por medio de promesas (promises), devolución de llamada (callback), así como las funciones asíncronas y de espera (async/await) donde se hacen peticiones al servidor sin la necesidad de que todo el programa se tenga mientras espera una respuesta, lo que es una característica fundamental en cualquier software moderno, debido a que mientras el sistema genera un reporte el usuario puede realizar otra actividad dentro del sistema. Otra característica importante es el control del modelo de objeto de documento DOM (Document Object Model), siendo la clave para poder actualizar el contenido de la página de forma sincrónica, es decir, en tiempo real conforme el usuario haga una interacción.

Finalmente, el ecosistema de este lenguaje es muy grande, contando con una gran comunidad que proporciona librerías, frameworks del lado del servidor y el cliente, APIs integradas desde el navegador como Fetch (para peticiones HTTP), herramientas como la geolocalización, control del caché, cookies y local storage para almacenamiento de datos desde el navegador en la parte del cliente.

2.3.2 TypeScript

TypeScript es un lenguaje de programación desarrollado por Microsoft Corporation en 2012, siendo una versión mejorada de JavaScript que incluye el tipado estático, lo que significa que se detectan errores antes de ejecutar el código y se define que tipos de datos debe tener la variable (string, number, boolean, blob, etc.), haciéndolo más robusto y consecuentemente incrementando la calidad del código y su mantenibilidad (López, 2023).

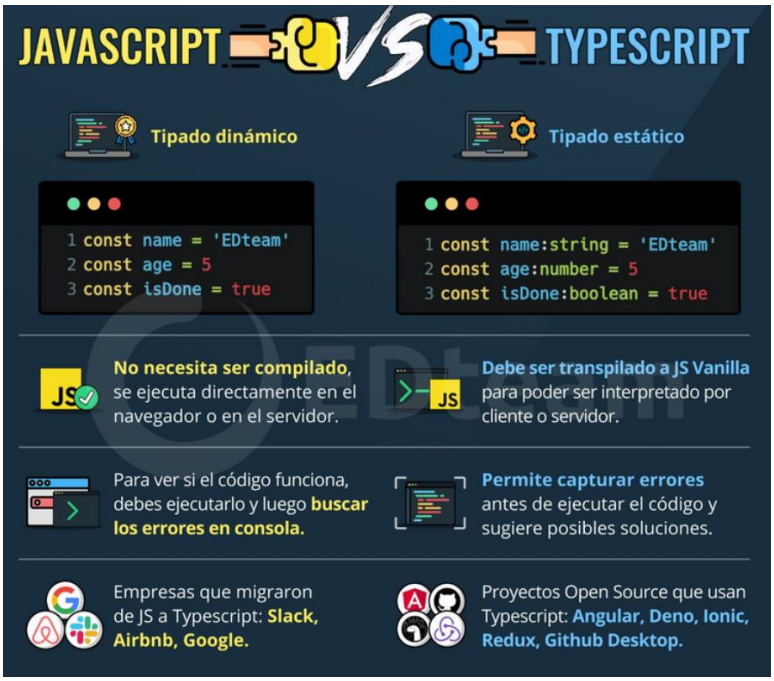
Uno de los aspectos más importantes de usar TypeScript en lugar de JavaScript (en el lado del cliente) es la experiencia de desarrollo que ofrece, siendo mucho más sólida en proyectos de gran escala y con varios desarrolladores porque el tipado es un control de calidad implícito en el código, lo que evita muchos errores e inconsistencias cuando se trata de saber por qué el programa no funciona de la forma esperada y que dichos errores son detectados al momento de la ejecución y no antes como en TypeScript (López, 2023). Por otra parte, ofrece características que favorecen a la reutilización de código como lo son las interfaces, mejor soporte para la POO (clases, herencia y métodos), característica de genéricos (generics), de esa forma se tiene un código escalable que se convierte posteriormente a JavaScript, garantizando la compatibilidad con todos los navegadores, frameworks y librerías.

En resumidas cuentas, TypeScript mejora la productividad de los desarrolladores y la mantenibilidad del código en proyectos de gran escala, por lo que busca mejorar al máximo las

características del ya robusto JavaScript y complementarlas para poder mejorar la calidad del código, darle al programa una estructura definida en apoyo con un framework y finalmente controlar mejor los errores antes de su ejecución.

Figura 2

Diferencias entre JavaScript & TypeScript



Nota: La figura representa las diferencias entre JavaScript & TypeScript. Tomado de *JavaScript vs. TypeScript: ¿Cuál es la diferencia?*, por Lopez, Y, 2023, EDteam.

2.4 Entorno de Desarrollo Integrado (IDE)

El entorno de desarrollo integrado (IDE), es un software que permite la construcción de aplicaciones a través de una interfaz (U.I), proporcionando herramientas útiles que agilizan el proceso de codificación tales como: compilador, depurador, editor de código fuente, extensiones de terceros y últimamente la integración de la inteligencia artificial con GitHub Copilot por ejemplo (Red Hat, 2023).

2.4.1 Visual Studio Code

Es un editor de código multiplataforma de código abierto, ligero, gratuito, creado por Microsoft Corporation en el año 2015. Su flexibilidad, extensiones y herramientas hacen que funcione a nivel de un IDE con la posibilidad de desarrollar software robusto. Entre sus principales características se destaca su soporte para múltiples lenguajes (Node.js, JavaScript, Typescript, etc.), compatibilidad con Git, depuración integrada, personalización y tiene un diseño ligero que ofrece buen rendimiento en equipos con características de bajos recursos (García, 2024).

2.5 Base de Datos

Las bases de datos son un conjunto de datos que están interrelacionados y organizados por medio de una estructura de datos. De ese modo, todos los datos se integran con poca duplicidad, lo que significa que toda la organización puede acceder a la información según sus roles y permisos, almacenando además la descripción de los datos (metadatos) en diccionarios (Marqués, 2011). Naturalmente, se diseñan con el fin de cumplir con los requerimientos de información de cada empresa según su lógica de negocio y la forma en la que opera sus procesos la organización.

2.5.1 Sistema Gestor de Base de Datos (SGBD)

Es una aplicación en la cual los usuarios pueden ejecutar diferentes operaciones sobre la base de datos: (crear, eliminar, mantener, etc.), adicionalmente permite operaciones sobre los registros de datos a través de un lenguaje de manejo de datos, siendo el estándar SQL que es un lenguaje no procedural (opera sobre colecciones de registros) (Marqués, 2011). En ese sentido, los SGBD proporcionan herramientas completas que facilitan el mantenimiento de la base de datos, haciéndola más flexible en su manejo con funciones como: un sistema de seguridad de la BD, control, integridad y documentación.

2.5.2 Lenguaje de Consulta de BD (SQL)

El lenguaje de consulta de BD o mejor conocido como SQL (Structured Query Language), es el lenguaje no procedural estándar para poder operar sobre los datos en las bases de datos relacionales. Dentro del SQL existen cuatro sentencias que permiten realizar operaciones, en primer lugar, el DDL (Data Definition Language) siendo comandos que permiten definir la estructura de la BD, el DML (Data Manipulation Language) permiten la manipulación sobre los registros de datos, el DCL (Data Control Language) controla los privilegios y permisos de los usuarios en la BD y el TCL (Transaction Control Language) permite operar las transacciones. Cabe destacar que esto no aplica para las bases de datos no relacionales, siendo otro tipo de BD en la cual sus tablas no están relacionadas y sus datos se organizan de forma escalable (Marqués, 2011).

2.5.3 MySQL

Es un SGBD (Sistema Gestor de Base de Datos) relacional desarrollada por MySQL AB y posteriormente adquirida por Oracle Corporation en 2010. Hasta la fecha, sigue siendo la base de datos más popular del mundo de código abierto, ofreciendo herramientas robustas que permiten el cumplimiento del modelo ACID en las transacciones y una gestión completa (Erickson, 2024).

2.6 Control de Versiones

Hace referencia a un sistema que permite la gestión y rastreo de cualquier cambio en una colección de archivos o un solo archivo conforme avanza el tiempo, siendo de gran ayuda en el desarrollo de software debido a la interacción que ocurre naturalmente del grupo de trabajo, permitiendo ver un historial de modificaciones y la posibilidad de recuperar versiones especificas según se requiera (Unity, 2025).

2.6.1 Importancia del control de versiones

El control de versiones es importante por sus propias funciones ya mencionadas, por lo que en un ambiente profesional la falta de un control de versiones hace muy difícil el manejo de qué cambios realizaron y cuando se hicieron, creando conflictos en el código (GitHub, 2025).

2.6.2 Git

Git es un sistema de control de versiones (distribuido) de código abierto y gratuito creado por Linus Torvalds en 2005; es usado ampliamente y cuenta con las funciones ya mencionadas, además de seguridad con hash criptográfico (GitHub, 2025). Cabe resaltar que Git es distribuido, siendo una característica que permite que el desarrollador tenga una copia en el repositorio con todos los sus cambios en la nube vía GitHub y también localmente, permitiendo operaciones de sincronización, modificación, revertir cambios, entre otros. Como se mencionó anteriormente, el sistema criptográfico de hash SHA-1 permite asignar un identificador único a cada cambio (commit) realizado por el desarrollador, garantizando la integridad de que usuario modificó una determinada línea de código, fechas, información del cambio, sirviendo como auditoría para mejorar el trabajo en equipo y rastrear el progreso en general de un software en desarrollo.

Otro de los componentes más importantes Git es su funcionalidad de ramas (branch), donde se puede trabajar de forma aislada con una versión del código específica, facilitando entornos de pruebas o el desarrollo de módulos, para finalmente realizar una integración del código con la operación de unión (merge) de forma automática y también controlada a la rama principal del proyecto (main o master). Finalmente, su integración con varias plataformas como GitHub, GitlLab hacen de esta una herramienta poderosa, que permite hacer más que gestionar los cambios en el código, mejorando la productividad y garantizando una construcción del software estructurada.

2.6.3 GitHub

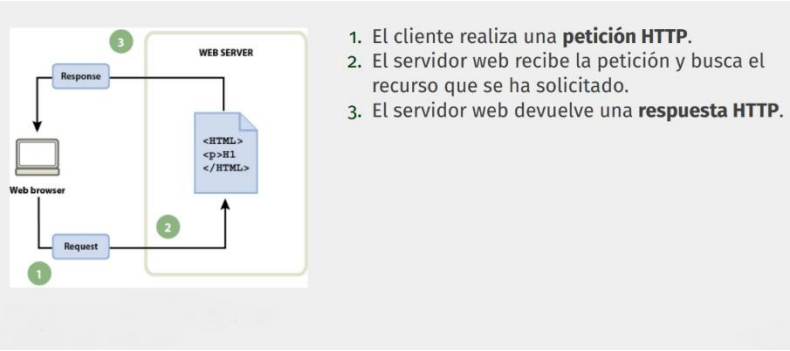
Una plataforma en la nube donde es posible subir código en repositorios usando Git (sistema de control de versión), siendo popular mundialmente en el desarrollo de software por su facilidad de colaboración y gestión de cambios (GitHub, 2025).

2.7 Aplicación Web

Una aplicación web es un software desarrollado en un lenguaje de programación web el cual tiene un comportamiento dinámico, es decir que el contenido cambiará según las funcionalidades que se hayan desarrollado (carga de contenido según perfiles, operaciones CRUD a los datos, …) para poder gestionar un determinado problema, funcionando a través de un navegador web usando el protocolo Http/Https; se diferencia de una página web normal la cual carece de funcionalidades complejas (debido a que solo tiene acceso al JavaScript del lado del cliente) y su contenido es estático, lo que significa que no cambiará dinámicamente, pues su fin es informativo. En concordancia, las aplicaciones web utilizan no solo tecnologías del lado del cliente Front-End (HTML, CSS, JS), sino también lenguajes de programación de alto nivel en el lado del servidor Back-End (Python, PHP, Java, …), interactuando con la base de datos y poder programar las funcionalidades requeridas (Muñoz, 2023).

Figura 3

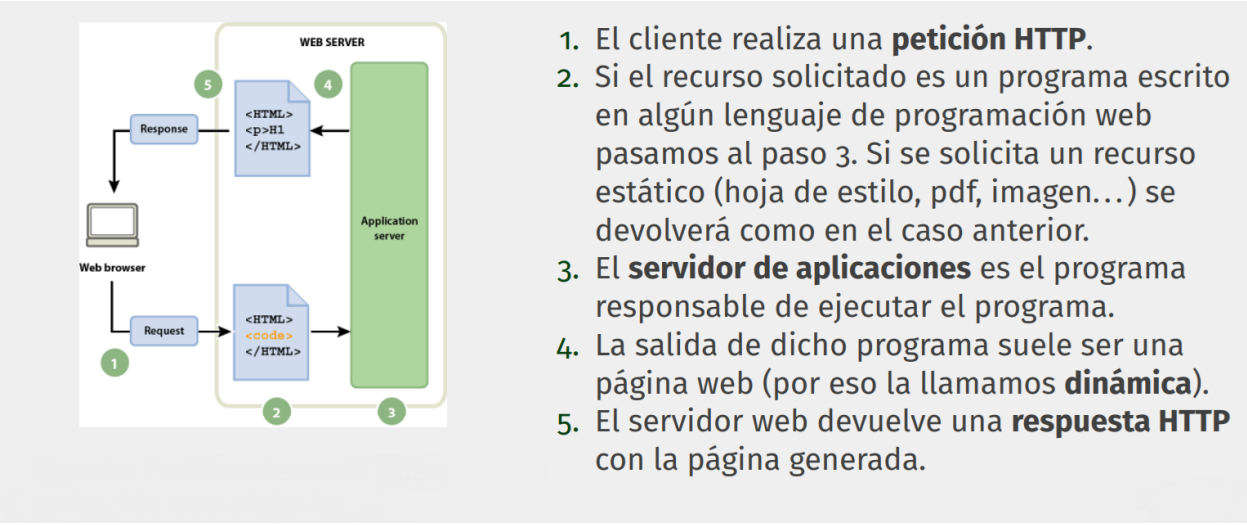
Procesamiento de una página web estática



Nota: La figura representa el acceso a páginas web estáticas especificando su proceso. Tomado de *Introducción a las aplicaciones web* (p.4), por J. Muñoz, 2023, IES Gonzalo Nazareno.

Figura 4

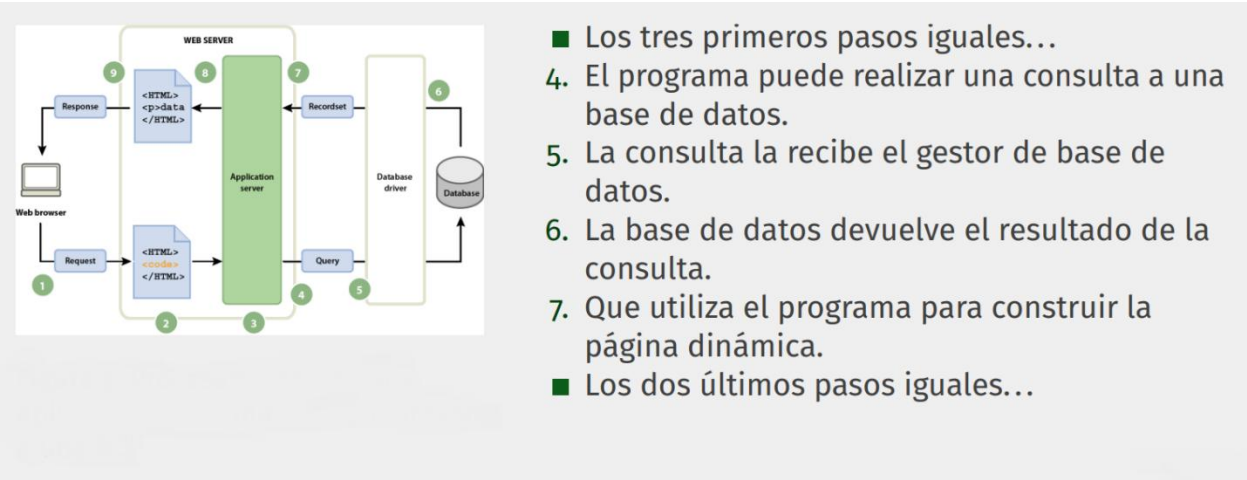
Procesamiento de una aplicación web dinámica



Nota: La figura representa el acceso a páginas web dinámicas. Tomado de *Introducción a las aplicaciones web* (p.5), por J. Muñoz, 2023, IES Gonzalo Nazareno.

Figura 5

Procesamiento de una aplicación web dinámica con acceso a una BD



Nota: La figura representa el acceso a páginas web dinámicas con acceso a base de datos. Tomado de *Introducción a las aplicaciones web* (p.6), por J. Muñoz, 2023, IES Gonzalo Nazareno.

2.7.1 Back-End

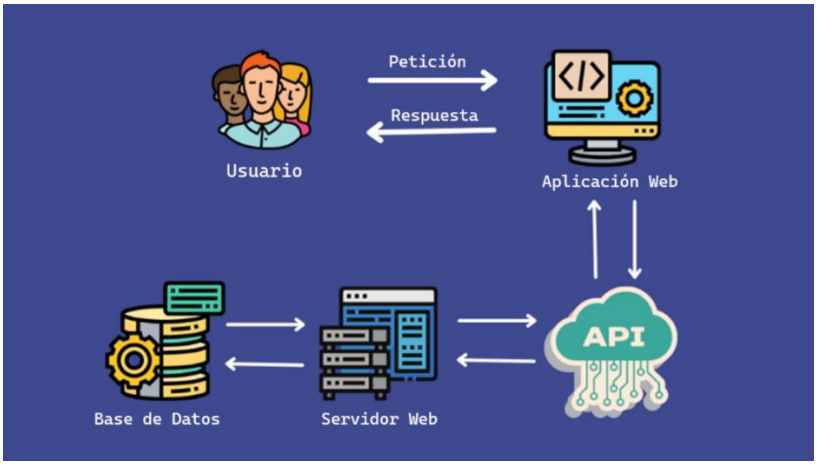
Hace referencia a la capa de acceso a los datos de un software, siendo un componente no visible por el usuario final. Naturalmente, se encarga de manejar la lógica de la aplicación, interactúa directamente con el frontend a través de APIs, controla la base de datos, usuarios, servicios externos (Zelaya, 2020).

2.7.1.1 API REST

En primer lugar, una API (Application Programming Interface), es un conjunto de protocolos, funciones, reglas que permiten comunicar varios tipos de software mutuamente y de esa forma ser reusadas por otras aplicaciones. Por otra parte, REST (Representational State Transfer) es un estilo de arquitectura para diseño de servicios web, usando operaciones sobre los recursos (GET, POST, PUT, DELETE). Dentro de ese orden de ideas, vemos que una API REST es una API que sigue los principios de arquitectura REST, facilitando la comunicación entre sistemas basado en el protocolo HTTP (International Business Machines [IBM], 2025).

Figura 6

Arquitectura de una API REST



Nota: La figura representa el proceso de solicitud y respuesta de una API REST. Tomado de *What is rest API and how to use it in Python,* por K. Hazra, 2024, Linkedin.

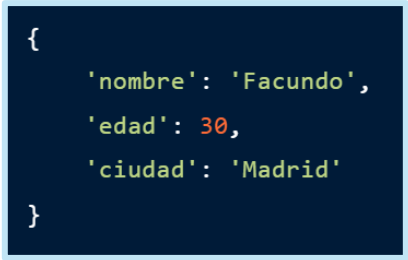
2.7.1.2 JSON

Conocido como JSON (JavaScript Object Notation) un tipo de archivo con formato semiestructurado usado ampliamente en el desarrollo web y aplicaciones en general, destacando por su función de comunicación entre sistemas para recibir y enviar datos entre el lado del cliente y servidor, además de poder almacenar información de forma sencilla y más ligera que otros formatos como el XML. En ese sentido, su sintaxis es simple, pues se compone de pares clave y valor ordenados de forma jerárquica con el uso de llaves "{}" y corchetes "[]", soporta tipos de datos básicos como cadenas de texto (string), números, booleanos, vacíos (null), arreglos, objetos; también es completamente independiente a su lenguaje JavaScript, pues funciona en otros lenguajes de programación como Java, C#, PHP, Python, etc.

Retomando que es un formato ligero, esto significa que su capacidad de transmisión de datos es rápida y ocupa poco espacio, siendo características claves en un sistema actual donde se prima la velocidad de respuesta para las peticiones en servicios web (API REST), móviles y peticiones en general al servidor. De este modo, cuando una aplicación necesita obtener datos de una API, dicha información llega en JSON generalmente, y es procesada en el lado del cliente para poder mostrarla al usuario, siendo un estándar en el intercambio de datos por su simplicidad, rendimiento y compatibilidad general (Deyimar, 2023).

Figura 7

Ejemplo de JSON



Nota: La figura representa un ejemplo de JSON, donde se puede ver el objeto que tiene 3 pares de clave y valor. Tomado de *Formato JSON: qué es y para qué sirve,* por F. García, 2024, arsys.

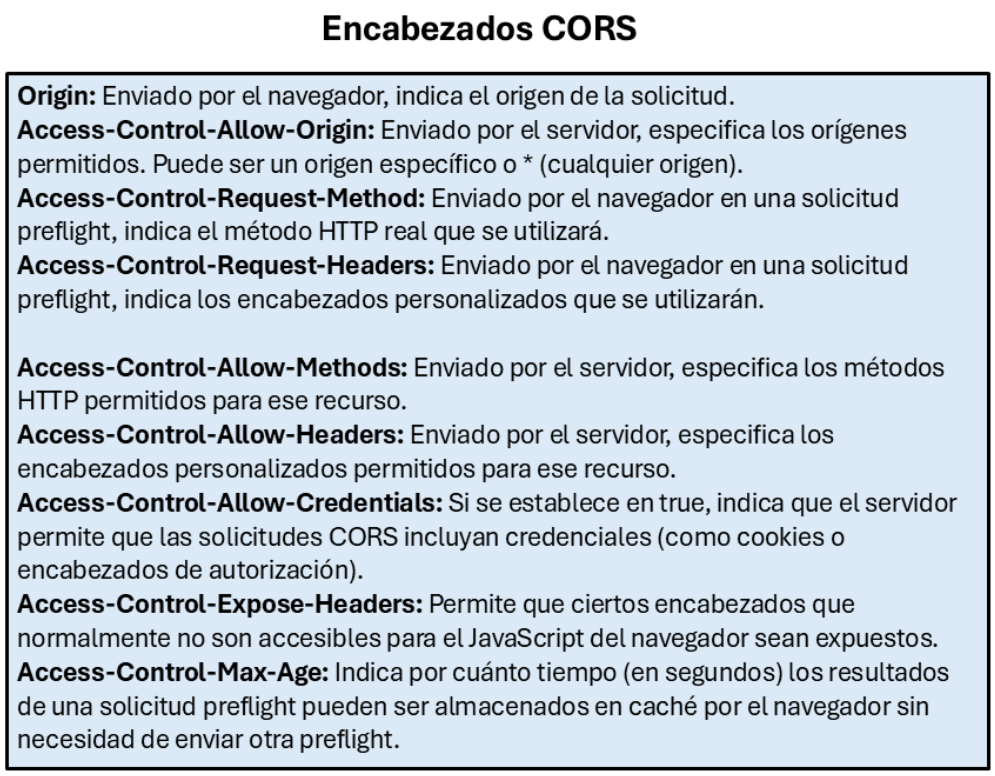
2.7.1.3 CORS

Conocido como intercambio de recursos de origen cruzado, siendo un mecanismo de seguridad en los navegadores que controla cómo el origen puede solicitar recursos a una fuente externa (dominio, protocolo y puerto). Su existencia es crucial para evitar ataques maliciosos como el CSRF (Falsificación de Solicitud entre sitios), de esa forma la política de seguridad de mismo origen es la medida en la que se ha llegado a acordar el uso de los CORS de forma estandarizada en cualquier navegador (Deyimar, 2023).

Dentro de ese orden de ideas, su funcionamiento se basa en intercambiar encabezados HTTP entre navegador y servidor, siendo solicitudes simples donde no hay encabezados personalizados (GET, POST, HEAD), verificando que el origen coincida con el valor aceptado en el servidor, siendo personalizado según la etapa de desarrollo. Por otra parte, las solicitudes "preflight" son complejas, pues requieren de encabezados personalizados con métodos como (PUT, DELETE), usándose la solicitud del tipo "OPTIONS" informando la solicitud que se busca realizar.

Figura 8

Encabezados de CORS



Nota: La figura representa los principales encabezados HTTP de CORS. Fuente: Elaboración propia.

2.7.2 Front-End

Es la capa visual del software que consume los servicios del lado del servidor backend, es decir, las interfaces (pantallas) de una aplicación o sistema en donde el usuario final interactúa directamente. En ese sentido, utiliza diferentes tecnologías del lado del cliente ya mencionadas como Html, CSS, Js y actualmente frameworks + librerías de desarrollo web para agilizar el proceso (Zelaya, 2020).

2.8 Vue.js

Un framework de frontend caracterizado por ser progresivo, es decir que se puede adaptar el proyecto de forma incremental dependiendo las necesidades. De esa forma, destaca por su facilidad para crear aplicaciones desde baja a alta escala, con interfaces dinámicas enfocadas al uso de una sola página SPA (Single Page Application) lo que permite que el contenido sea ejecutado en una sola página web, evitando que se recargue la página completa y actualizando o reescribiendo de forma dinámica la página web actual según las acciones del usuario.

En función de lo planteado, su punto fuerte es la simplicidad, y sus funcionalidades robustas como la recarga en caliente (hot reload), permitiendo recargar cambios en tiempo real, además de usar un sistema reactivo que detecta automáticamente cualquier cambio en los datos y actualizando la interfaz; por otra parte, permite utilizar tecnologías como Vite.js e integrar sus herramientas a Vue, facilidad en la elección del lenguaje de programación (JavaScript o TypeScript), recarga del proyecto en caliente y una arquitectura basada en componentes que facilita la organización del código así como su escalabilidad (Kumar, 2024).

En los últimos 5 años se ha posicionado como uno de los frameworks de desarrollo web más usados junto a React y Angular gracias a su gran ecosistema y su robustez, siendo usado por grandes empresas como Netflix, Nintendo, Adobe, Xiami, entre otros.

2.9 Biblioteca de Herramientas

2.9.1 Node.js

Lanzado en 2009, es un entorno de ejecución de JavaScript que posibilita la ejecución del código de este lenguaje en el lado del servidor. Es así como permite la construcción de aplicaciones backend, web, script, etc.

2.9.2 Express.js

Un framework web para Node.js que proporciona una estructura para el desarrollo de aplicaciones web, además de ofrecer diferentes herramientas robustas que permiten gestionar y construir APIs de forma eficiente, manejo de solicitudes, respuestas, middleware y enrutamiento.

2.9.3 Vite

Vite es una herramienta (build tool) que permite desarrollar aplicaciones web de forma rápida a través de un servidor de desarrollo y una compilación mejorada. Entre sus características se encuentra la ya mencionada recarga en caliente (hot reload), compatibilidad con otros frameworks (Vue.js, React, etc.), plugins y lenguajes (Vite, 2025).

2.9.4 Tailwind CSS

Es un framework (marco de trabajo) de CSS, el cual ayuda a desarrollar diseños U.I de forma rápida y fácil, a través de clases individuales robustas para mayor personalización y flexibilidad a diferencia de otros frameworks como Bootstrap los cuales ofrecen componentes ya construidos. Estas características más su integración con frameworks de desarrollo web hacen que el diseño y la experiencia de usuario UX mejore significativamente (De Roy, 2023).

2.9.5 PrimeVue

Desarrollado por PrimeTek, es una biblioteca de componentes U.I (User Interface) diseñado para trabajar con el framework de desarrollo web Vue.js (PrimeVue, 2025). Cuenta con

una amplia gama de componentes ya creados (tablas, botones, formularios) que facilitan el desarrollo frontend, ahorrando el trabajo de diseñar desde cero. Finalmente ofrece compatibilidad con Tailwind CSS, estándares de la WCAG (Web Content Accessibility Guidelines) y estilos personalizados.

2.9.6 Pinia

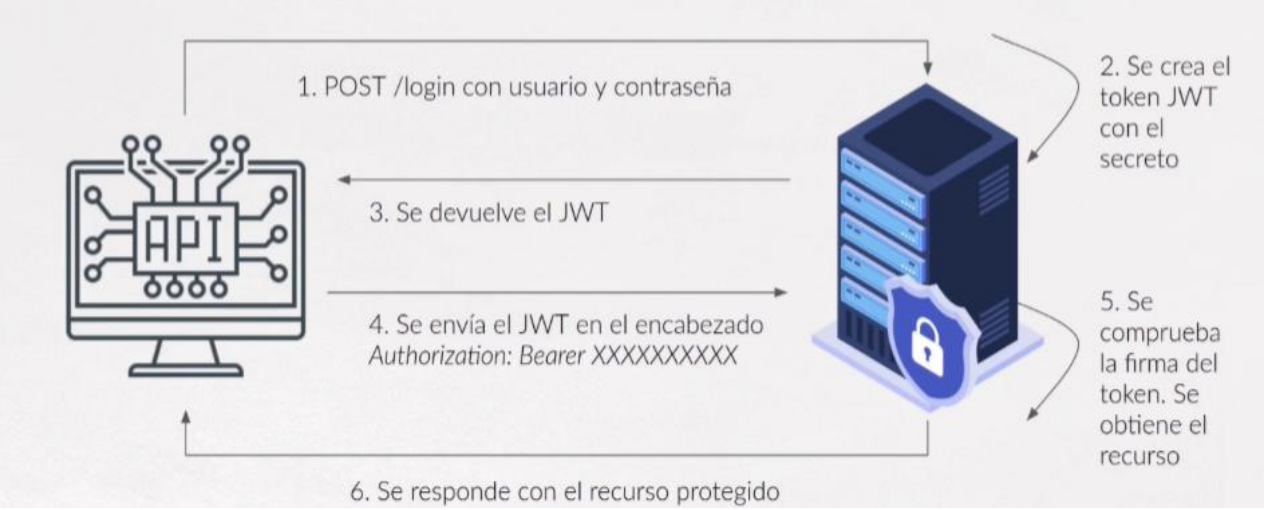
Es una biblioteca de estados para Vue, eso significa que permite gestionar de forma flexible y fácil estados de forma global en toda la aplicación entre componentes y páginas, siendo una opción ligera, modular y que cuenta con un soporte grande para typescript (Pinia, 2025).

2.9.7 JWT (Json Web Token)

Json Web Token es un estándar abierto (RFC 7519) basado en Json para la creación de "tokens" (identificador) de acceso en base 64 codificado e intercambiar información de forma segura entre las dos partes. Funciona de forma en la que el token se envía al servidor para ser verificado (usando un código secreto) y si es válido se realiza la acción correspondiente; por lo que generalmente su uso es para fines de autenticación y acceso autorizado a aplicaciones, APIs, entre otros (Lopez, 2020). Finalmente está compuesto de un header (indica el algoritmo y tipo de token), el payload (la información que se busque guardar en el token como datos del usuario) y signature (una firma que permite comprobar la validez del token).

Figura 9

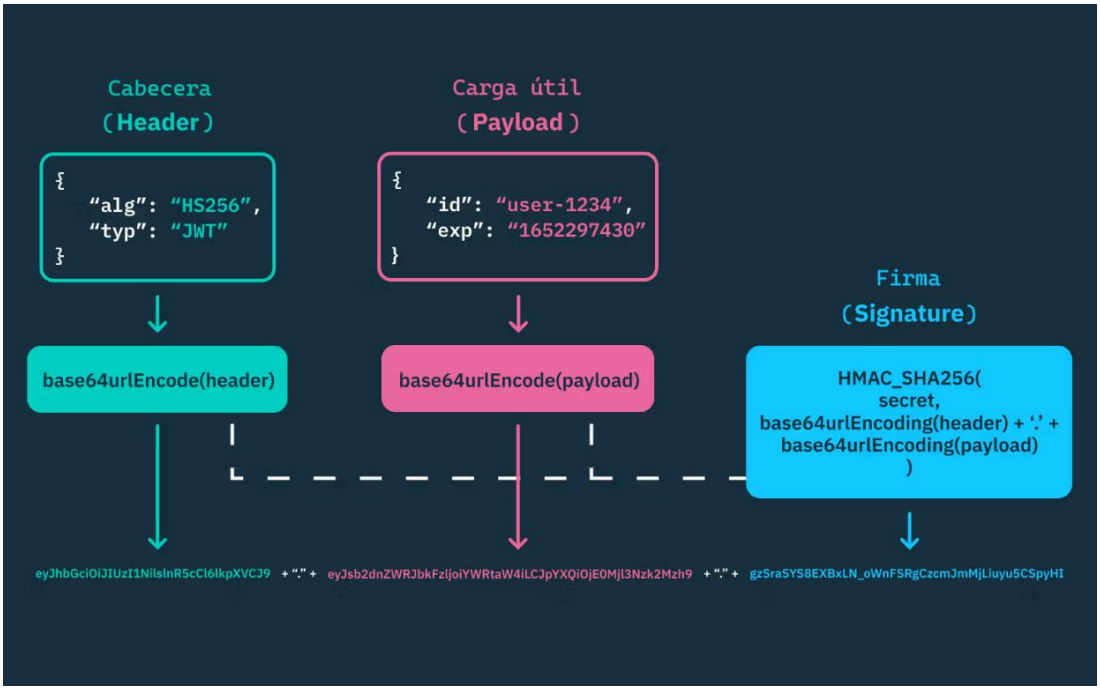
Funcionamiento de un token JWT (Json Web Token)



Nota: La figura representa el funcionamiento de un token JWT. Tomado de *Qué es Json Web Token y cómo funciona*, por Lopez, L, 2020, OpenWebinars.

Figura 10

Estructura de un JWT



Nota: La figura representa estructura de tres partes que tiene un token JWT. Tomado de *JWT claims*, por Rich, A, 2023, Stytch.

2.9.8 Bcrypt

Es un algoritmo de hashing de contraseña basado en blowfish (permite múltiples iteraciones de hash) y salting (cadena aleatoria que evita que dos contraseñas iguales generen el mismo hash) por lo que al realizar acciones como el login (inicio de sesión) no se hace la comparación con el texto plano, sino que se vuelve a realizar el proceso de encriptación hashing y comparar el resultado con el almacenado. A diferencia de otros algoritmos como MD5 o SHA-1, bcrypt tarda más en generar el hash, siendo una característica intencional ya que es ajustable y dificulta los ataques de fuerza bruta (Arias, 2021).

2.9.9 Axios

Es una librería de Node.js basada en promesas (Promise API) que permite hacer solicitudes HTTP en el navegador y en Node.js de forma asíncrona, estructurada, cuenta con soporte automático para JSON (usando API Rest), maneja errores e intercepta petición y resultado (García, 2023).

2.9.10 Node Mailer

Una librería diseñada para Node.js cuya función principal es él envío de correos electrónicos (emails) desde una aplicación web de forma simple, adicionalmente, se usa para enviar notificaciones de confirmación, códigos de recuperación y notificaciones automáticas.

2.9.11 File saver

Consiste en una librería que permite descargar archivos generados desde el lado del cliente en su navegador web, de esa forma, es capaz de exportar imágenes, documentos, archivos, etc; todo esto sin la necesidad de enviar ni guardar datos al servidor.

2.10 Buenas Prácticas en el Desarrollo Web

Una serie de estándares y principios que permiten construir un producto de software de calidad, pues aplicándolas se logra construir aplicaciones más accesibles, seguras y fáciles de usar. Según (MDN Web Docs, 2025) estos son los estándares de desarrollo web actuales:

Tabla 1

Buenas prácticas y estándares de desarrollo web

Categoría	Buena práctica	Descripción
Accesibilidad (WCAG)	Aplicar los estándares WCAG 2.1 en lo posible	Hacer diseños U.I amigable, colores adecuados, navegación por teclado.
Código limpio	Usar los principios de programación como KISS, DRY.	Mantener el código simple, reutilizable. Separar lógica en funciones pequeñas
Semántica HTML	Usar etiquetas semánticas (<header>, <main>, etc.)	Mejora la accesibilidad, SEO y comprensión del DOM.
Estructura CSS	Uso de frameworks CSS robusto como Tailwind	Ayuda con la modularidad y evita conflictos de estilos.
Responsive Design (Diseño responsivo)	Diseñar para usar la aplicación en varios dispositivos (PC, TV, Móvil)	El uso de flexbox, grid, unidades relativas, media query.
Optimización de imágenes	Comprimir imágenes y usar formatos como (WebP, AVIF)	Mejora el tiempo de carga de la imagen y su rendimiento.
Versionamiento y control	Usar un Sistema de control de versiones como Git	Simplifica el trabajo en equipo y permite un control eficiente del historial de cambios.
Performance web (Rendimiento web)	Optimizar los scripts y estilos, usar “axios” para peticiones.	Reduce tiempos de carga, mejora UX.
SEO básico	Usar etiquetas <title>, <meta>, <alt>, <h1-h6>	Mejora visibilidad en buscadores y accesibilidad.

Documentación	Documentar componentes, endpoints y funciones.	Ayuda a entender el proyecto a los demás desarrolladores.

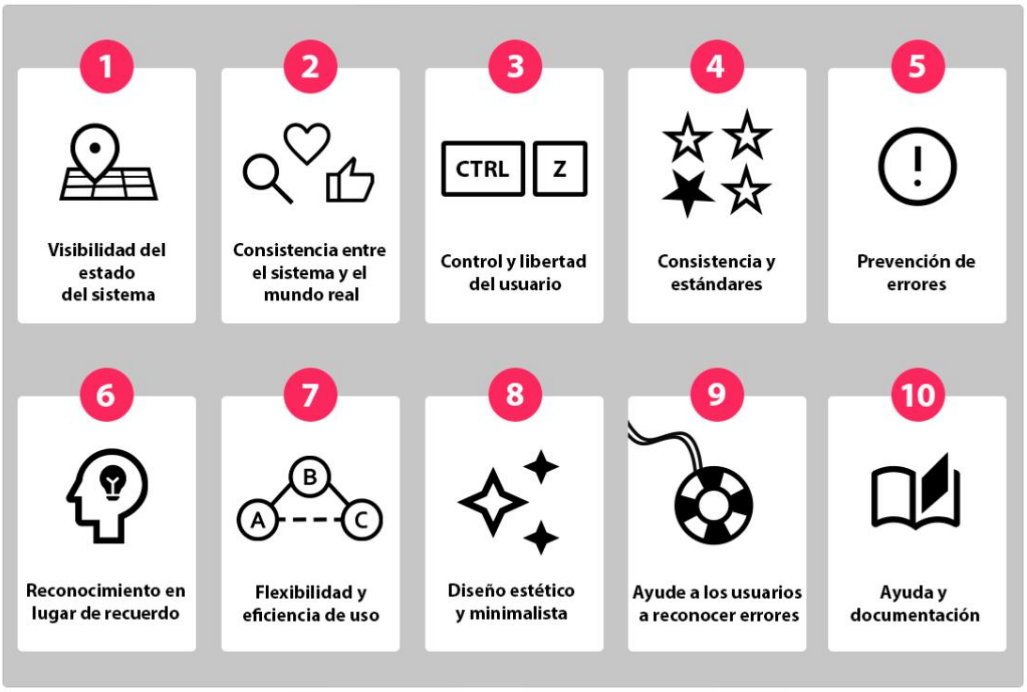
Nota: Esta tabla muestra un resumen de los estándares y buenas prácticas en el desarrollo web moderno, dividido por categorías, la buena práctica/estándar y su descripción. Fuente: Elaboración propia, apoyado con herramientas de Inteligencia Artificial.

2.10.1 Principios de diseño y usabilidad

Consiste en una serie de diez heurísticas propuestas por el Doctor Jacob Nielsen, son muy usados en la industria para resolver problemas de usabilidad, incrementando el UX (User Experience) de forma que los usuarios puedan interactuar fácilmente con la aplicación (Amorin, 2024).

Figura 11

Las diez heurísticas de Jacob Nielsen



Nota: La figura representa las diez heurísticas de Jacob Nielsen de forma resumida. Tomado de *10 principios de usabilidad de Jakob Nielsen (con ejemplos)*, por D. Amorin, 2024, Diego Amorin.

2.10.2 Seguridad en las aplicaciones web

Hace referencia al proceso de proteger las aplicaciones web a través de buenas prácticas y tecnologías que permitan aplicar medidas preventivas y paliativas contra los ciberataques, de lo contrario, las amenazas pueden afectar considerablemente a la organización, sus datos y el servicio. Según (De Luca, 2022), a continuación, se muestra las mejores prácticas en desarrollo web para mejorar la seguridad de estas aplicaciones.

Tabla 2

Buenas prácticas de seguridad en el desarrollo web

Categoría	Buena práctica	Descripción
Gestión de sesiones	Usar JWT en cookies seguras (HttpOnly + SameSite)	Protege contra los ataques XSS y CSRF al ser configurado correctamente.
Control de acceso	Aplicar roles y permisos en el frontend y backend	Usar middlewares para proteger rutas si el usuario tiene el token de la sesión.
Evitar XSS y SQLi	Usar frameworks con protección integrada y evitar acceso directo al DOM de la aplicación.	Frameworks como React, Angular y Vue controlan entradas automáticamente.
CSRF (Cross-Site Request Forgery)	Configurar correctamente cookies con “SameSite” o usar tokens antifalsificación	Evita que el navegador envíe cookies en solicitudes maliciosas.
Componentes y dependencias	Mantener librerías y frameworks actualizados	Evitar dependencias obsoletas. Usar “npm audit” para controlar las versiones.

Información sensible	No exponer el secreto del token en el código ni en repositorios públicos.	Usar archivos “.env” y agregarlos a “.gitignore”.
Auditoría y escaneo	Realizar auditorías con herramientas específicas.	Detectar fallos de seguridad antes de que lleguen a producción.
Diseño de la API	Seguir principios REST y aplicar versionado.	Las APIs bien diseñadas son mejor mantenibles y menos vulnerables.
HTTPS obligatorio	Encriptar toda la comunicación con SSL/TLS	Nunca usar HTTP en producción.

Nota: Esta tabla muestra un resumen de las buenas prácticas de seguridad en el desarrollo web, dividido por categorías, la buena práctica/estándar y su descripción. Fuente: Elaboración propia, apoyado con herramientas de Inteligencia Artificial.

CAPÍTULO III: ANÁLISIS Y DISEÑO DE LA APLICACIÓN

3.1 Análisis de Requerimientos

Los requerimientos son las necesidades o expectativas que debe cumplir la aplicación web desarrollada, en ese sentido se dividen en dos grandes grupos: Funcionales y No Funcionales.

3.1.1 Requerimientos Funcionales

3.1.1.1 Gestión de Usuarios

- **Inicio de Sesión:** La aplicación permitirá iniciar sesión a la cuenta del usuario por medio de su correo y contraseña. De esa forma, la aplicación permitirá que solo los usuarios autenticados (token JWT válido) accedan a la página de inicio.
- **Recuperación de contraseña:** La aplicación deberá tener una sección para poder recuperar la contraseña a través de un código de verificación temporal vía correo electrónico.
- **Configuración de usuario:** El usuario podrá ver sus datos registrados, cambiar su contraseña y actualizar su foto de perfil.
- **CRUD de usuario:** El administrador y el rol con permisos podrán: crear, ver, actualizar e inhabilitar a los usuarios del sistema. Adicionalmente, la aplicación deberá enviar un correo electrónico de notificación al usuario que se ha creado con sus respectivas credenciales para acceder al sistema, de forma similar cuando se cambie el correo electrónico del usuario a uno nuevo.

3.1.1.2 Gestión de Roles/Permisos

- **Creación de roles:** La aplicación permitirá un CRUD de los roles que tendrá cada usuario en una vista de gestión de parámetros.

- **Asignación de permisos:** Dentro de la gestión de roles, el administrador puede elegir los permisos que tendrá cada rol para acceder a una determinada vista de la aplicación de forma modular, de lo contrario no se le permitirá el acceso.
- **Redirección:** La vista de gestión de parámetros deberá tener un botón que permita redireccionar al usuario al apartado de creación de roles y viceversa a su gestión.
- **Protección de interfaz:** La aplicación restringirá el acceso de interfaces a los perfiles sin el permiso correspondiente.

3.1.1.3 Primeras Consultas

- **Búsqueda de usuario:** La interfaz de primeras consultas tendrá un cuadro de búsqueda de usuarios (cliente) con su cédula de identidad. En caso de estar registrado el valor, se cargarán los datos del cliente.
- **CRUD de primeras consultas:** La aplicación permitirá crear los datos del usuario y de la ficha de atención de primeras consultas en caso de no existir, de lo contrario, es posible crear una nueva ficha de atención (para nuevas consultas), actualizar y ver otras fichas de atenciones pasadas.
- **Alertas de viabilidad:** La aplicación deberá mostrar alertas de viabilidad sobre un caso cuando el usuario no cumpla con los requisitos específicos socioeconómicos y demográficos. De ese modo, se mostrará que aspecto no cumple y una opción para rechazar su patrocinio del caso.
- **Generación de ficha de técnica:** Dentro de cada ficha de atención, habrá un botón que permitirá la generación un documento PDF correspondiente a la información de la ficha (el caso) como un resumen de esos datos y los acuerdos que debe cumplir

el usuario. El documento se actualizará si la información de la ficha o el usuario cambian.

- **Tabla de actividades:** En la sección de actividades de un caso, habrá una tabla que permita ver el resumen del módulo de "Gestión de casos y actividades" con toda la información que ha desarrollado el estudiante del caso y un botón que permita ver el documento de evidencias.
- **Revisión de casos:** La aplicación tendrá una interfaz donde se carguen todos los casos que han sido llenados por los estudiantes de primeras consultas y que necesitan una revisión obligatoria por parte del coordinador. La información se cargará en una tabla indicando datos resumidos, filtros, búsqueda general, un botón para visualizar la ficha técnica y una redirección al caso en particular; una vez el caso ha sido revisado y aprobado a patrocinio o marcado como asesoría, ya no se mostrará en la tabla.
- **Interfaz general de casos:** La aplicación tendrá una vista general de todos los casos que han creado, recordando de que un usuario puede tener varios casos. Se manejarán filtros individuales en cada columna de datos, búsqueda general, visualización de ficha técnica y redirección al caso.

3.1.1.4 Reportes

- **Primeras Consultas:** La aplicación tendrá una interfaz donde se seleccionará el tipo de reporte que se desee, se ingresará una fecha de inicio y fin, para generar un reporte en Excel de todos los casos y el usuario del caso en ese rango de tiempo.

3.1.2 Requerimientos No Funcionales

3.1.2.1 Usabilidad

- La aplicación deberá tener un diseño de interfaz de usuario (U.I) versátil, intuitivo y fácil de usar.
- El diseño seguirá los principios de usabilidad y experiencia de usuario (UX), como las heurísticas de Jakob Nielsen.
- La aplicación se desarrollará como Single Page Application (SPA)

3.1.2.2 Seguridad

- La aplicación usará Json Web Token (JWT) como mecanismo de autenticación de usuarios.
- Las contraseñas de los usuarios deben almacenarse de forma segura en la base de datos, utilizando el algoritmo de hashing Bcrypt.
- Se debe implementar un control de acceso basado en roles (RBAC) para la aplicación.

3.1.2.3 Mantenibilidad y Calidad del código

- El desarrollo deberá seguir los estándares y lineamientos proporcionados por la Dirección Informática (área de desarrollo) de la PUCE.
- El código fuente de la aplicación deberá estar escrito en inglés.
- El backend deberá ser desarrollado en un entorno de Node.js
- El frontend deberá ser desarrollado usando el framework Vue.js
- Se usará el lenguaje de programación TypeScript en el lado del cliente.
- La base de datos será MySQL.

- El servidor de aplicación será administrado en la universidad (D.I) y podría tener un S.O Ubuntu Server (versión LTS), 4GB RAM y 2vCPU como configuración inicial propuesta.

3.1.2.4 Portabilidad

- La aplicación web deberá ser multiplataforma entre S.O (Microsoft Windows 10 y 11, Linux Ubuntu, Oracle y Debian)
- Será compatible con los navegadores web más populares actualmente (Firefox, Chrome, Edge, Opera, etc.)

3.2 Especificación de requerimientos

Para el desarrollo de esta sección se delimitarán las funcionalidades del módulo de la aplicación en función de los requerimientos, especificando en los casos de uso únicamente los más importantes:

F1. Gestión de Usuarios

F2. Gestión de Roles/Permisos

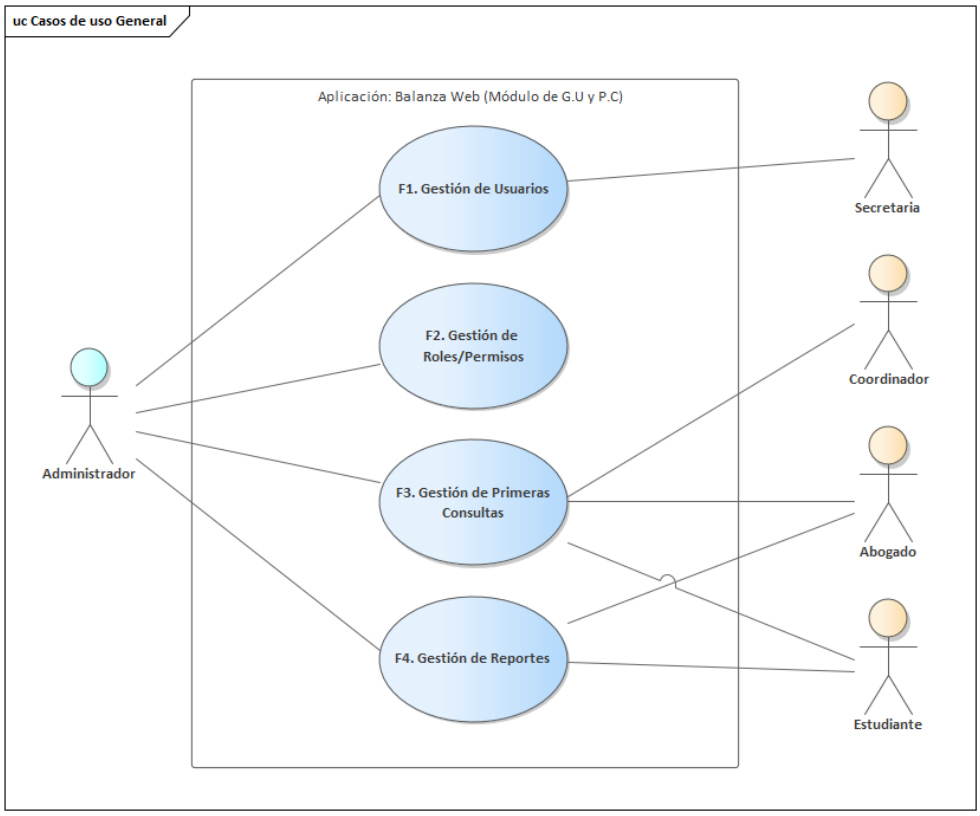
F3. Gestión de Primeras Consultas

F4. Gestión de Reportes

3.2.1 Diagrama de Casos de uso general

Figura 12

Diagrama de Casos de Uso General



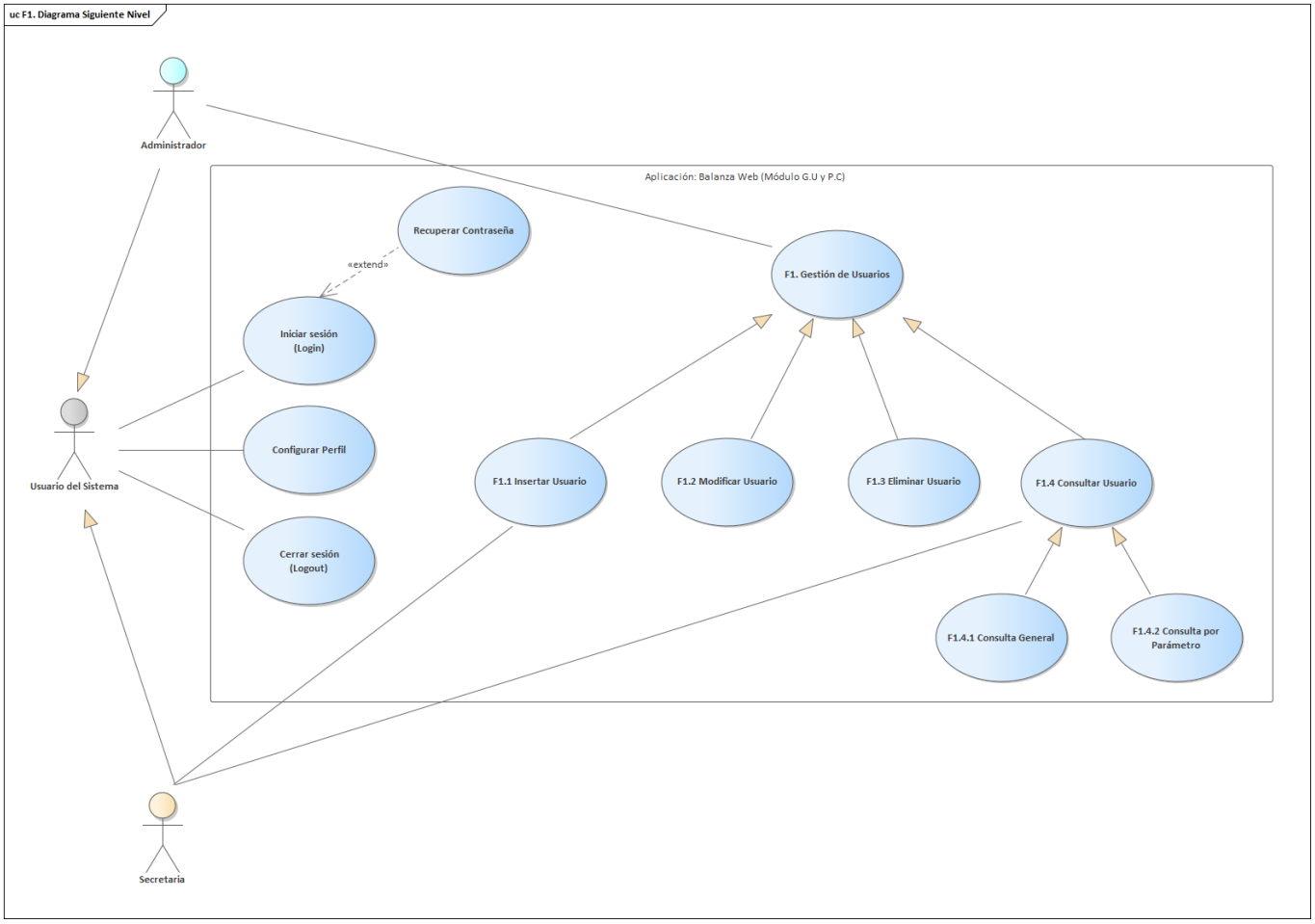
Nota: La figura representa el diagrama de casos de uso general del módulo propuesto, junto con los 5 roles que interactúan con las funcionalidades descritas, excluyendo al rol de "Trabajo Social", debido a que no forma parte de la interacción del módulo. Fuente: Elaboración propia.

3.2.2 Diagrama de Casos de uso: Siguiente Nivel

3.2.2.1 Siguiente Nivel: F1. Gestión de Usuarios

Figura 13

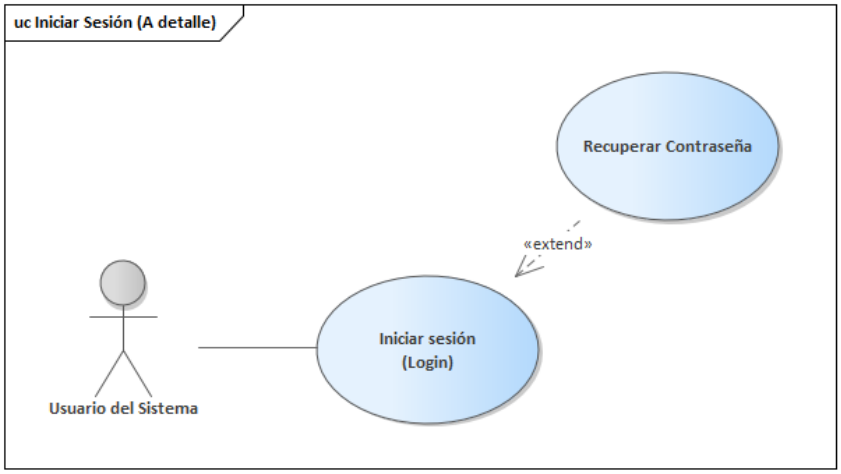
F1. Gestión de Usuarios - Diagrama Siguiente Nivel



Nota: La figura representa el diagrama siguiente nivel para el F1. Gestión de Usuarios, donde se denota un rol general llamado "Usuario del Sistema" el cual tendrá varios casos de uso en común, sin embargo, una vez iniciada la sesión el rol asignado (Administrador, Secretaria), es el encargado de ejecutar cierta operación según sus permisos. Fuente: Elaboración propia.

Figura 14

Caso de uso: Iniciar Sesión (A detalle)



Nota: La figura representa el diagrama a detalle para el caso de uso "Iniciar sesión (Login)", donde el rol general (Usuario del Sistema) tiene los permisos para poder iniciar sesión o recuperar su contraseña en la aplicación. Fuente: Elaboración propia.

Tabla 3

Caso de uso: Iniciar Sesión (A detalle)

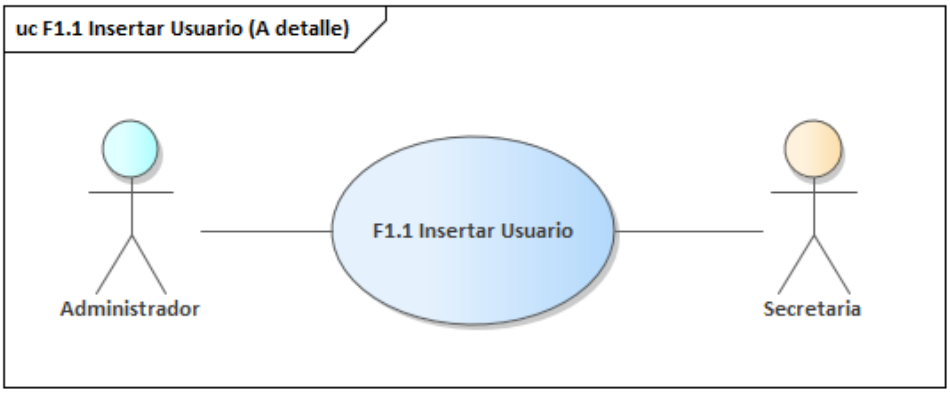
Nombre del caso de uso: Iniciar Sesión (Login)	**ID Único:** GU-001
Área: -	
Actor(es): Usuario del Sistema, Administrador, Secretaria	
Interesados: Usuario del Sistema	
Nivel: -	
Descripción: Permitir que un usuario con credenciales válidas y una cuenta activa se pueda autenticar en el sistema para acceder a las funcionalidades de su respectivo rol.	
Evento desencadenador: El actor navega a la página de Balanza Web (Login), introduce su correo electrónico y contraseña, y hace clic en el botón "Iniciar Sesión"	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal):	**Información para los pasos**
1. El actor ingresa su correo y contraseña en el formulario de la página de login.	

2. El sistema valida localmente que los campos no estén vacíos.	
3. El actor hace clic en el botón "Iniciar Sesión".	
4. El sistema envía las credenciales al servidor mediante una petición a la API.	Correo electrónico, Contraseña
5. El servidor valida las credenciales contra la base de datos y verifica que el estado del usuario sea "Activo".	
6. El servidor genera un token de sesión (JWT) y lo establece en una cookie HttpOnly	Token JWT, Cookie de sesión
7. El sistema redirige al actor a la página de inicio (/) de la aplicación.	
Precondiciones: El actor debe tener una cuenta de usuario interna creada previamente en el sistema.	
Postcondiciones: El actor está autenticado en el sistema, se crea una sesión válida y se cargan sus permisos de rol.	
Suposiciones: El actor tiene acceso a un navegador web compatible y conexión a la red de la PUCE.	
Garantía de éxito: El actor es redirigido a la página de inicio, visualiza el menú y las opciones correspondientes a su rol.	
Garantía mínima: El sistema envía un mensaje de error al intento de login (“Credenciales incorrectas", "Cuenta inactiva") o el actor pudo iniciar sesión.	
Requerimientos cumplidos: Permitir que el actor pueda iniciar sesión eficientemente.	
Cuestiones pendientes: ¿Se implementará un límite de intentos fallidos antes de bloquear la cuenta temporalmente?	
Prioridad: Alta	
Riesgo: Medio	

Nota: Esta tabla describe el caso de uso de “Iniciar Sesión o Login” a detalle. Fuente: Elaboración propia.

Figura 15

Caso de uso: F1.1 Insertar Usuario



Nota: La figura representa el diagrama a detalle para el caso de uso "F1.1 Insertar Usuario", siendo una funcionalidad exclusiva del Administrador y Secretaria, sin embargo, de ser autorizado otro rol podría acceder a la funcionalidad. Fuente: Elaboración propia.

Tabla 4

Caso de uso: F1.1 Insertar Usuario (A detalle)

Nombre del caso de uso: F1.1 Insertar Usuario	**ID Único:** GU-002
Área: -	
Actor(es): Administrador, Secretaria	
Interesados: Administrador, Nuevo Usuario	
Nivel: -	
Descripción: Permite a un actor con privilegios (Administrador o Secretaria) registrar un nuevo usuario interno en el sistema, asignándole un área, rol y credenciales de acceso. El sistema notifica automáticamente al nuevo usuario por correo electrónico	
Evento desencadenador: El actor decide agregar un nuevo usuario al sistema. Solo usuarios tipo "Estudiante" en el caso de la Secretaria y en otra vista diseñada solo para su rol.	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal):	**Información para los pasos**
1. El actor navega en el menú lateral y selecciona la opción "Gestión de Usuarios".	

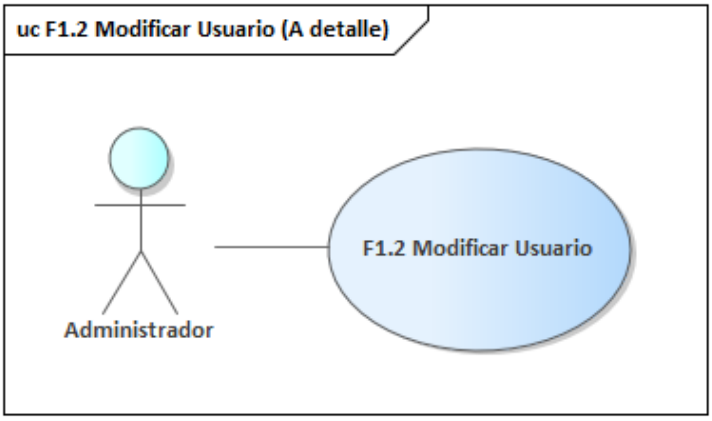
2. El sistema muestra la interfaz respectiva con una lista de usuarios general de la aplicación (/Usuarios).	
3. El actor hace clic en el botón "Crear Usuario".	
4. El sistema redirige al actor al formulario de creación de nuevo usuario en la ruta (/NuevoUsuario).	
5. El actor completa todos los datos requeridos del nuevo usuario en el formulario.	Datos del nuevo usuario (cédula o pasaporte, nombre, correo, contraseña rol, área, estado)
6. El actor hace clic en el botón "Crear Usuario" para enviar el formulario.	
7. El sistema valida los datos del formulario en el lado del cliente (frontend) y envía los datos del nuevo usuario al servidor a través de una petición a la API.	
8. El servidor valida los datos, verifica que no exista un usuario duplicado, y encripta la contraseña y guarda el nuevo registro de usuario en la base de datos MySQL con estado "Activo".	
9. El servidor envía automáticamente un correo electrónico al nuevo usuario con sus credenciales de acceso (correo y contraseña).	
10. El sistema muestra un mensaje de éxito en la interfaz.	
Precondiciones: El actor debe haber iniciado sesión en el sistema y tener los permisos de rol asignados para poder crear usuarios.	
Postcondiciones: Se crea un nuevo registro en la tabla Internal_User. El nuevo usuario recibe un correo electrónico con sus credenciales y puede iniciar sesión.	
Suposiciones: El correo electrónico proporcionado para el nuevo usuario es válido para recibir la notificación de sus credenciales.	
Garantía de éxito: El nuevo usuario es creado exitosamente en la base de datos, esta notificado, y puede acceder al sistema. El actor puede regresar a la lista de usuarios y ver a la nueva persona registrada.	
Garantía mínima: En caso de haber un error en algún dato, el sistema no crea al usuario y muestra un mensaje claro al actor, permitiéndole corregir la información.	

Requerimientos cumplidos: El actor con los respectivos permisos podrá crear a los usuarios del sistema y se les notificará vía correo electrónico sus credenciales.
Cuestiones pendientes: Ninguna
Prioridad: Alta
Riesgo: Bajo

Nota: Esta tabla describe el caso de uso "F1.1 Insertar Usuario" a detalle. Fuente: Elaboración propia.

Figura 16

Caso de uso: F1.2 Modificar Usuario



Nota: La figura representa el diagrama a detalle para el caso de uso "F1.2 Modificar Usuario". Fuente: Elaboración propia.

Tabla 5

Caso de uso: F1.2 Modificar Usuario (A detalle)

Nombre del caso de uso: F1.2 Modificar Usuario	**ID Único:** GU-003
Área: -	
Actor(es): Administrador	
Interesados: Administrador, Usuario del Sistema	
Nivel: -	
Descripción: Permite al Administrador modificar la información y el estado de un usuario existente en el sistema (nombre, rol, área o si la cuenta está activa o inactiva).	

Evento desencadenador: El actor necesita actualizar los datos o el rol de un usuario existente en el sistema.	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal):	**Información para los pasos**
1. El actor busca y localiza al usuario que desea modificar en la tabla y hace clic en el icono de "Editar" (lápiz) de la fila correspondiente.	
2. El sistema captura el ID del usuario y despliega un cuadro de diálogo en modo edición, precargando todos los datos del usuario en los campos correspondientes.	
3. El actor modifica los campos deseados en el formulario.	Datos del usuario modificado (Nombre, Apellido, Correo, Número de teléfono, Estado, Tipo, Área)
4. El actor hace clic en el botón "Guardar" y el sistema envía la información actualizada al servidor a través de una petición a la API.	
5. El servidor encuentra al usuario por su ID, valida los datos y actualiza el registro correspondiente en la tabla Internal_User de la base de datos, si el campo de correo electrónico fue modificado, el servidor envía una notificación al nuevo correo informando del cambio.	
6. El sistema cierra el cuadro de diálogo y muestra un mensaje de éxito en la interfaz.	
Precondiciones: El actor debe haber iniciado sesión con un rol de Administrador. El usuario que se desee modificar debe existir en el sistema.	
Postcondiciones: La información del usuario es actualizada en la base de datos. Si se cambió el correo, se envía una notificación.	
Suposiciones: El actor tiene conocimiento claro de los cambios que debe realizar y sus implicaciones posteriores (como cambiar un rol).	
Garantía de éxito: Los datos del usuario se actualizan correctamente en el sistema y los cambios se reflejan inmediatamente en la lista de usuarios.	
Garantía mínima: Si ocurre un error durante la actualización, los datos del usuario permanecen sin cambios en la base de datos y el sistema informa al actor del problema mediante una notificación.	

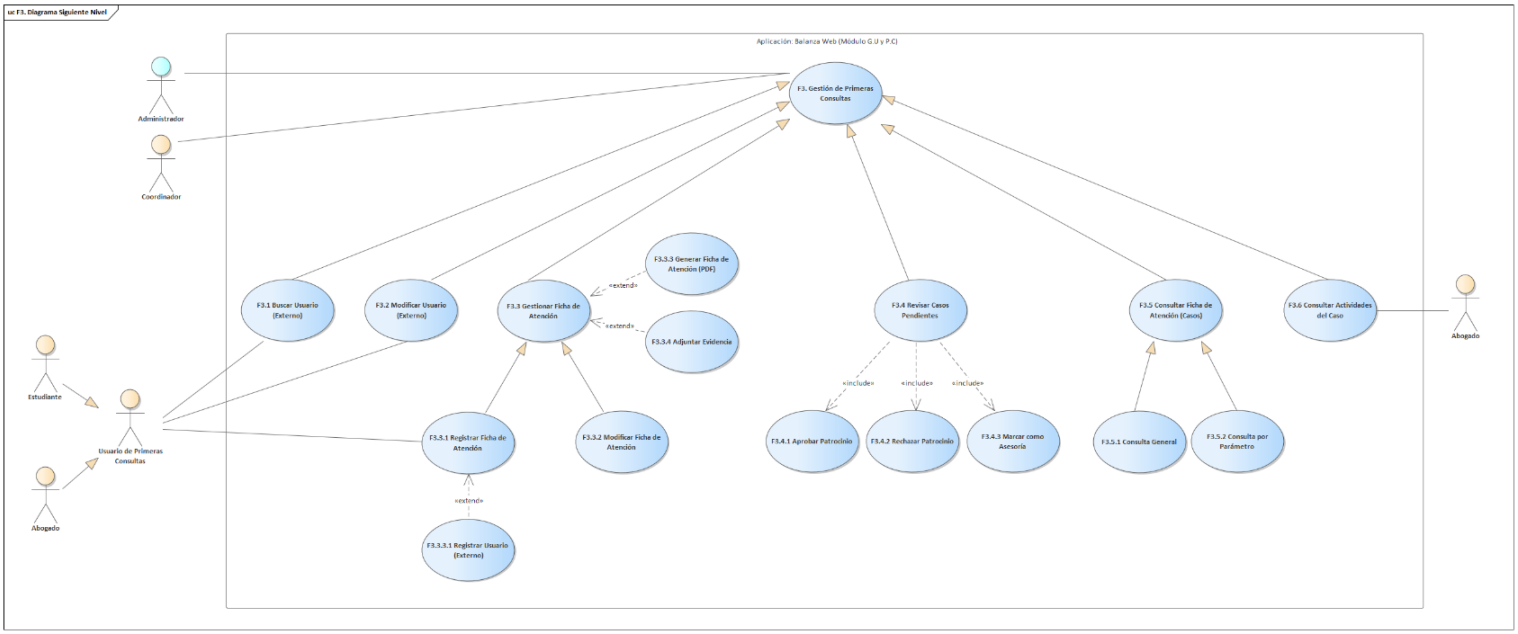
Requerimientos cumplidos: El actor con los respectivos permisos podrá editar a los usuarios existentes del sistema y se les enviará una notificación en caso de que el campo de correo electrónico haya sido modificado a su nuevo correo.
Cuestiones pendientes: Ninguna
Prioridad: Alta
Riesgo: Bajo

Nota: Esta tabla describe el caso de uso "F1.2 Modificar Usuario" a detalle. Fuente: Elaboración propia.

3.2.2.1 Siguiente Nivel: F3. Gestión de Primeras Consultas

Figura 17

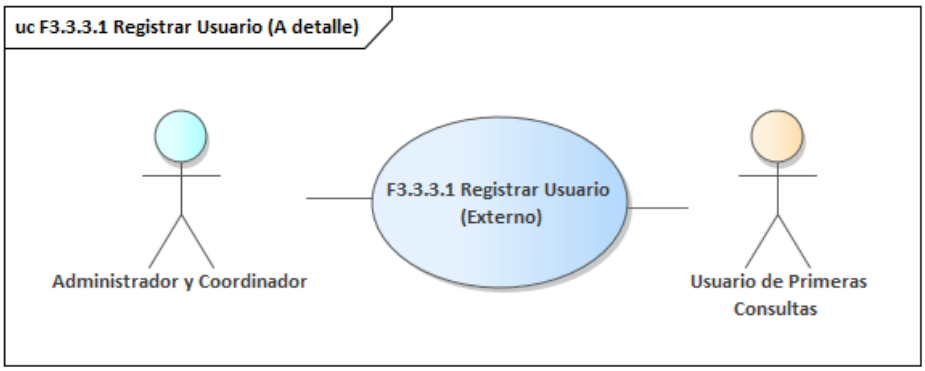
F3. Gestión de Primeras Consultas - Diagrama Siguiente Nivel



Nota: La figura representa el diagrama siguiente nivel para la F3. Gestión de Primeras Consultas, donde se denota un rol general llamado "Usuario de Primeras Consultas" el cual tendrá varios casos de uso en común. Por otra parte, se puede observar las relaciones (include y extend) y conexiones con los actores, además de explicar la herencia en algunos casos de uso.

Figura 18

Caso de uso: F3.3.3.1 Registrar Usuario (Externo)



Nota: La figura representa el diagrama a detalle para el caso de uso "F3.3.3.1 Registrar Usuario (Externo)". Fuente: Elaboración propia.

Tabla 6

Caso de uso: F3.3.3.1 Registrar Usuario Externo (A detalle)

Nombre del caso de uso: F3.3.3.1 Registrar Usuario (Externo)	**ID Único:** PC-001
Área: Primeras Consultas	
Actor(es): Usuario de Primeras Consultas (Estudiante, Abogado), Coordinador, Administrador.	
Interesados: Coordinador, Estudiante, Abogado	
Nivel: -	
Descripción: Permite a un actor registrar a un nuevo usuario externo (cliente) y su primera Ficha de Atención de manera simultánea, siempre y cuando este no se encuentra registrado en la base de datos.	
Evento desencadenador: Luego de buscar a un usuario por su ID en la interfaz, el sistema informa que no existe, y el actor procede a registrarlo junto con su consulta.	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal)	**Información para los pasos**
1. El actor navega en el menú lateral, hace clic en la opción de "Nuevo Caso" y muestra la interfaz de nuevo caso (/NuevoCaso)	

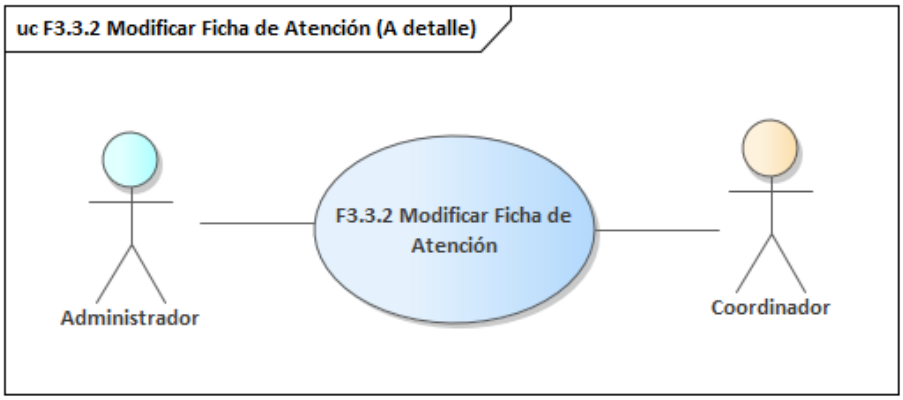
2. El actor escribe en cuadro de texto el ID del usuario (cédula o pasaporte) y hace clic en botón verde de buscar (lupa).	ID del Usuario Externo
4. El sistema muestra un mensaje ("Usuario no encontrado, por favor registrar sus datos") y habilita el formulario completo.	
5. El actor completa tanto los datos personales del nuevo usuario (datos personales, datos demográficos, etc.) como los datos de la Ficha de Atención.	Datos completos del usuario y la consulta
6. El actor hace clic en el botón "Crear Nuevo Caso".	
7. El servidor valida los datos, crea un nuevo registro en la tabla User y un nuevo registro en la tabla Initial_Consultations, y los enlaza (por la relación FK).	
8. El sistema muestra un mensaje de éxito y reinicia el formulario, dejándolo listo para una nueva búsqueda.	
Precondiciones: El actor ha iniciado sesión y se ha completado el caso de uso "F3.1 Buscar Usuario por ID" con el resultado "usuario no encontrado".	
Postcondiciones: Se crea un nuevo registro en la tabla User y un nuevo registro en la tabla Initial_Consultations en la base de datos.	
Suposiciones: El actor tiene toda la información necesaria para realizar el registro completo (tanto del cliente como de la consulta).	
Garantía de éxito: El nuevo cliente y su primera consulta son registrados exitosamente en el sistema, confirmándose a través de la notificación de éxito.	
Garantía mínima: En caso de ocurrir un error, no se crea ni el usuario ni la ficha, y el sistema informa al actor del problema, permitiéndole corregir la información.	
Requerimientos cumplidos: Creación de Usuario externo y Consulta Inicial.	
Cuestiones pendientes: Ninguna.	
Prioridad: Alta	
Riesgo: Bajo	

Nota: Esta tabla describe el caso de uso “F3.3.3.1 Registrar Usuario Externo” a detalle.

Fuente: Elaboración propia.

Figura 19

Caso de uso: F3.3.2 Modificar Ficha de Atención



Nota: La figura representa el diagrama a detalle para el caso de uso "F3.3.2 Modificar Ficha de Atención". Fuente: Elaboración propia.

Tabla 7

Caso de uso: F3.3.2 Modificar Ficha de Atención (A detalle)

Nombre del caso de uso: F3.3.2 Modificar Ficha de Atención	**ID Único:** PC-002
Área: Primeras Consultas	
Actor(es): Coordinador, Administrador.	
Interesados: Coordinador, Estudiante, Abogado	
Nivel: -	
Descripción: Permite actualizar una Ficha de Atención existente para un usuario externo previamente registrado en el sistema.	
Evento desencadenador: El actor decide editar la información de una ficha de atención en particular.	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal)	**Información para los pasos**
1. El actor navega en el menú lateral, hace clic en la opción de "Nuevo Caso" y el sistema muestra la interfaz de nuevo caso (/NuevoCaso)	

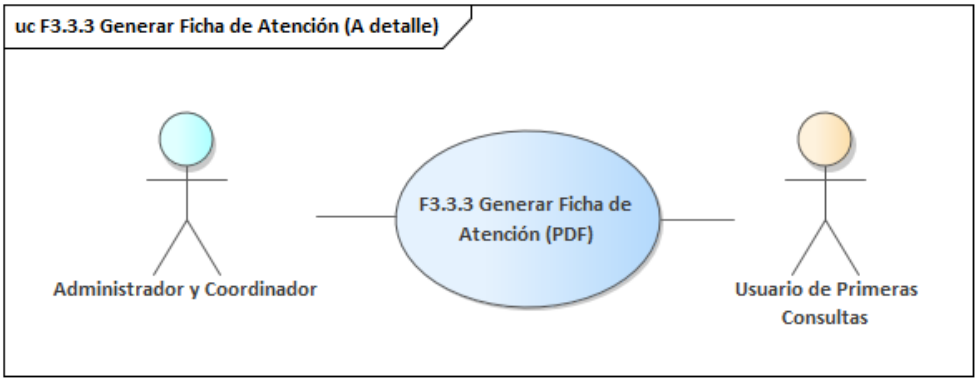
2. El actor escribe en cuadro de texto el ID del usuario (cédula o pasaporte) y hace clic en botón verde de buscar (lupa).	ID del Usuario Externo
3. El sistema muestra un mensaje y carga los datos del Usuario, así como todas las consultas que haya tenido.	
4. El actor hace clic en el botón “Editar Ficha” (ícono de papel con lápiz)	
5. El actor procede a editar los campos requeridos de la ficha	Campos de la consulta (Área, Tema del caso, Observación, etc.)
6. El actor hace clic en el botón "Guardar" para almacenar los nuevos cambios y el sistema valida los datos del formulario en el frontend.	
7. El sistema envía los datos de la ficha al servidor, valida la información y actualiza la ficha de atención	
8. El sistema muestra un mensaje de confirmación de éxito y redirige al paginado la ficha modificada.	
Precondiciones: El actor ha iniciado sesión y se ha completado el caso de uso "F3.1 Buscar Usuario por ID" con el resultado "usuario encontrado".	
Postcondiciones: Se actualiza el registro en la tabla Initial_Consultations de la base de datos, asociado al usuario externo.	
Suposiciones: El actor tiene toda la información necesaria para realizar la modificación.	
Garantía de éxito: La consulta es actualizada exitosamente en el sistema, confirmándose a través de la notificación de éxito.	
Garantía mínima: En caso de ocurrir un error, no se edita la ficha, y el sistema informa al actor del problema, permitiéndole corregir la información.	
Requerimientos cumplidos: Actualización de nueva Ficha de Atención.	
Cuestiones pendientes: Ninguna	
Prioridad: Alta	
Riesgo: Bajo	

Nota: Esta tabla describe el caso de uso “F3.3.2 Modificar Ficha de Atención” a detalle.

Fuente: Elaboración propia.

Figura 20

Caso de uso: F3.3.3 Generar Ficha de Atención



Nota: La figura representa el diagrama a detalle para el caso de uso "F3.3.3 Generar Ficha de Atención (PDF)". Fuente: Elaboración propia.

Tabla 8

Caso de uso: F3.3.3 Generar ficha de Atención (A detalle)

Nombre del caso de uso: F3.3.3 Generar ficha de Atención	**ID Único:** PC-003
Área: Primeras Consultas	
Actor(es): Usuario de Primeras Consultas (Estudiante, Abogado), Coordinador, Administrador.	
Interesados: Coordinador, Estudiante, Abogado	
Nivel: -	
Descripción: Permite al actor generar un documento PDF formateado que contiene un resumen de los datos de una Ficha de Atención específica que está siendo visualizada, siendo un documento que debe ser firmado por el usuario externo, ya que contiene los acuerdos, términos y condiciones al usar el servicio de los consultorios jurídicos.	
Evento desencadenador: El actor necesita una copia física o digital de la ficha de atención para archivar, imprimir o compartir y hace clic en el botón correspondiente.	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal)	**Información para los pasos**
1. Mientras el actor visualiza los detalles de una Ficha de Atención existente en la interfaz,	

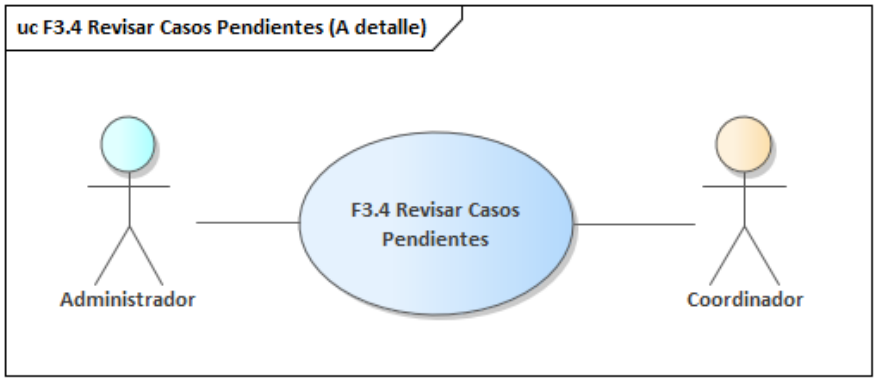
hace clic en el botón "Generar PDF" (icono de PDF).	
2. El sistema envía una solicitud al servidor, incluyendo el ID de la consulta que se está visualizando.	
3. El servidor recupera todos los datos asociados a esa consulta de la base de datos MySQL y genera dinámicamente un documento PDF con los datos, utilizando una plantilla predefinida para la ficha de atención.	ID de la Consulta
4. El sistema (en el frontend) recibe el archivo y utiliza la librería file-saver para abrir un cuadro de dialogo con el documento adjunto.	
5. El actor hace clic en “descargar” y elige una ubicación en su equipo y guarda el archivo PDF con el nombre que desee.	
Precondiciones: El actor ha iniciado sesión y se ha completado el caso de uso "F3.1 Buscar Usuario por ID" con el resultado "usuario encontrado".	
Postcondiciones: El actor tiene una copia local del archivo PDF de la ficha de atención. El estado del sistema no se modifica.	
Suposiciones: El navegador del actor tiene permisos para descargar y guardar archivos.	
Garantía de éxito: El actor visualiza o descarga exitosamente el archivo PDF con todos los datos relevantes de la consulta seleccionada.	
Garantía mínima: En caso de ocurrir un error, el documento se intentará cargar e informará sobre el problema al actor.	
Requerimientos cumplidos: Generación de ficha de técnica.	
Cuestiones pendientes: Ninguna.	
Prioridad: Media	
Riesgo: Bajo	

Nota: Esta tabla describe el caso de uso “F3.3.3 Generar ficha de Atención” a detalle.

Fuente: Elaboración propia.

Figura 21

Caso de uso: F3.4 Revisar Casos Pendientes



Nota: La figura representa el diagrama a detalle para el caso de uso "F3.4 Revisar Casos Pendientes". Fuente: Elaboración propia.

Tabla 9

Caso de uso: F3.4 Revisar Casos Pendientes (A detalle)

Nombre del caso de uso: F3.4 Revisar Casos Pendientes	**ID Único:** PC-004
Área: Primeras Consultas	
Actor(es): Coordinador, Administrador.	
Interesados: Coordinador, Estudiante, Abogado	
Nivel: -	
Descripción: Permite a un actor con rol de supervisor visualizar la lista de casos de primeras consultas que están pendientes de revisión para tomar una decisión sobre su viabilidad y siguiente paso (Aprobar para patrocinio o Marcar como asesoría).	
Evento desencadenador: El actor accede a la sección de revisión para procesar los nuevos casos que han sido registrados en el sistema.	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal)	**Información para los pasos**
1. El actor navega en el menú lateral, hace clic en la opción de "Revisar Casos" y el sistema muestra la interfaz de revisión de casos (/RevisarCaso).	

2. El sistema muestra los casos pendientes por revisar en una tabla con la información principal, siendo lo principal la alerta de si el caso tiene algún problema de viabilidad.	
3. El actor selecciona un caso de la lista para analizarlo. Puede hacer clic en el botón de "Ir al caso" para navegar a la ficha completa y revisar todos los detalles (incluyendo alertas de viabilidad generadas en cada ficha de atención).	
4. Tras analizar la información, el actor decide el siguiente paso para el caso. Este caso de uso incluye (<<include>>) a los casos de uso "Aprobar Patrocinio", "Rechazar el Patrocinio" o "Marcar como Asesoría".	
5. El actor ejecuta una de las acciones, correspondientes a su respectivo caso de uso. En caso de aprobar, el actor debe ejecutar el caso de uso F3.3.2 y seleccionar “Patrocinio”. Para marcar como asesoría, el actor debe aprobar la opción desde la interfaz de revisión y finalmente para rechazar el patrocinio, el actor debe visualizar si el caso tiene alertas y desde ahí ejecutar la opción.	
6. El sistema actualiza el estado del caso en la base de datos a “Por Asignar”, el cual llegará a la bandeja de casos que deben ser asignados a su respectiva área y reinicia la lista de casos pendientes, eliminando de la tabla el caso que acaba de ser procesado.	
Precondiciones: El actor debe haber iniciado sesión y tener los permisos para ver la interfaz. Deben existir casos en estado "Pendiente" en el sistema.	
Postcondiciones: El caso revisado ya no se encuentra en la lista de pendientes. Su estado ha sido actualizado a "Por Revisar" o "Asesoría", permitiendo que continúe en el siguiente flujo de trabajo.	
Suposiciones: El actor tiene el criterio y la autoridad para decidir sobre la viabilidad de un caso.	
Garantía de éxito: El caso es procesado exitosamente, su estado se actualiza y avanza correctamente en el flujo de trabajo del consultorio	

Garantía mínima: Si ocurre un error, el estado del caso no cambia y el sistema informa al actor del problema, manteniendo el caso en la lista de pendientes para un nuevo intento.
Requerimientos cumplidos: Revisión de todos los casos pendientes y continuación del flujo de trabajo.
Cuestiones pendientes: Ninguna.
Prioridad: Alta
Riesgo: Bajo

Nota: Esta tabla describe el caso de uso "F3.4 Revisar Casos Pendientes" a detalle. Fuente: Elaboración propia.

3.3 Diagrama Entidad-Relación (ERD)

En este caso, al ser una aplicación web que abarca múltiples módulos como ya se ha mencionado, el análisis principal estará centrado en las entidades necesarias para la construcción del módulo de Gestión de Usuarios, Primeras Consultas, Roles y Permisos. Naturalmente, el modelo define todas las entidades de la aplicación, sus atributos (campos) y las relaciones que hay entre cada tabla.

Dentro de este orden de ideas, se tiene en primer lugar a la entidad "Internal_Users", siendo la tabla central que se encarga de almacenar la información de todos los usuarios de consultorios jurídicos que hagan uso de la aplicación (Administrador, Coordinador, Abogados, Estudiantes, Secretaria), siendo su clave primaria "Internal_ID" que corresponde a la cédula o pasaporte de la persona, además de campos como su correo, contraseña, área dentro del Consultorio Jurídico y otros. Por consiguiente, se relaciona con gran parte del sistema (como Perfiles, Primeras Consultas, Evidencias, …), pues es una parte central de la lógica que maneja la aplicación.

Ahora bien, la tabla "Users" que contiene representa a los usuarios externos o clientes que acuden al consultorio jurídico en busca de una asesoría o patrocinio legal, almacenando información personal, demográfica, socioeconómica, situación vulnerabilidad, entre otros; por

consiguiente, su relación principal radica en que un usuario puede tener varias primeras consultas (atenciones o casos), siendo una relación uno a muchos (1:N) con Initial_Consultations.

En cuanto a la entidad de Primeras Consultas, llamándose “Initial_Consultations”, es una parte indispensable de toda la aplicación web, pues sin esta no hay casos por los que se puedan gestionar en los otros módulos, pues cada registro de esta tabla es una ficha de atención que documenta un caso (ya sea asesoría o patrocinio), almacenando información como el tipo de servicio, la materia (área) del caso y su respectivo tema; por ende, se vincula con “Users” con la relación descrita antes, recibiendo la clave foránea (para saber de qué cliente es el caso) y también tiene una relación con el usuario del sistema “Internal_Users” (para saber qué usuario registro el caso).

Con respecto a la Gestión de Roles y Permisos, se maneja una tabla de parámetros llamada “Profile”, siendo una entidad simple que contiene únicamente el nombre de cada perfil que se puede usar dentro de la aplicación web (Administrador, Estudiante, …), así esta entidad se relaciona directamente con otra tabla llamada “ProfileViewPermission”, donde la relación consiste en que un perfil puede tener muchos permisos de vista (las interfaces dentro del frontend), y un permiso de vista pertenece a un perfil. Naturalmente, al ser una relación (1:N) esta tabla que controla todos los permisos de un cierto rol o perfil contiene la clave foránea de ésta (Profile_ID) y la entidad del perfil (Profile) se relaciona con los usuarios del sistema (Internal_Users), haciendo que un perfil pueda tener muchos usuarios internos, y un usuario interno tenga un solo perfil (1:N). Cabe destacar que el diseño E/R final se encuentra dentro de la sección “Anexos” para mejor visibilidad.

3.4 Diseño de la Arquitectura del Sistema

La arquitectura del software propuesto sigue un patrón de 3 capas, siendo una estructura robusta que separa las responsabilidades y satisface las necesidades de los consultorios jurídicos, además de facilitar el mantenimiento y la escalabilidad de la aplicación web. En primer lugar, la capa de presentación (Frontend) es la parte principal con la que el usuario interactúa a través de un navegador web y que consume los servicios API REST del servidor de aplicación (Backend); en cuanto las tecnologías, son las mencionadas en el capítulo II, siendo una aplicación de una sola página (SPA) con el framework Vue.js, Typescript, Pinia para manejar los estados de la sesión, Primevue para todo el diseño de la interfaz UI y Vite como plantilla y servidor de desarrollo.

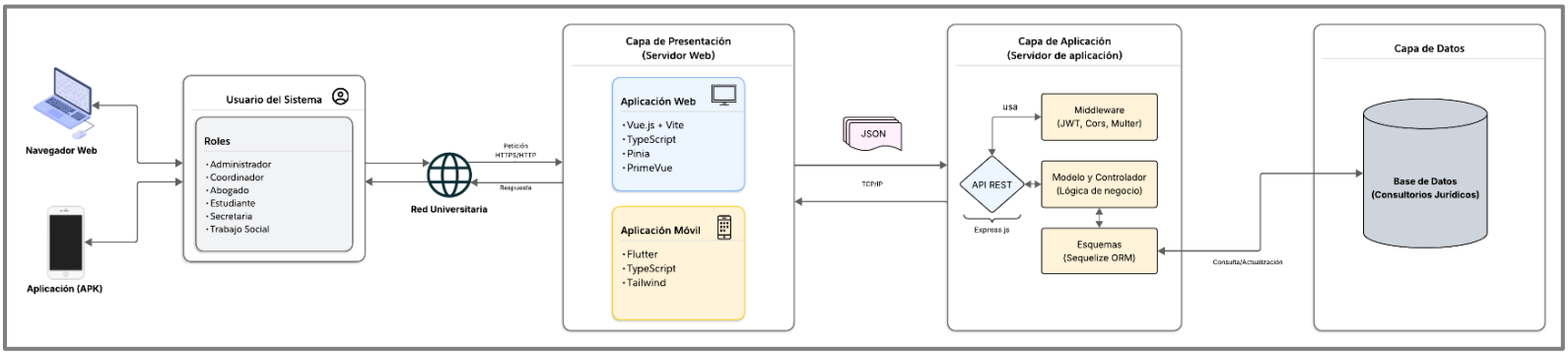
Por otra parte, la capa de aplicación contiene toda la lógica de negocio y funcionalidades que consumirá el frontend a través de una API REST, en el servidor de aplicación que usará Node.js y Express.js permitiendo gestionar las rutas y peticiones HTTP/HTTPS en el archivo (app.js). En ese sentido, el backend está construido usando el patrón MVC (sin vista), definiendo el esquema de datos que usará el ORM Sequalize para mapear las estructuras definidas a las tablas de MySQL. Los modelos contienen la lógica de negocio que se conecta e interactúa con los esquemas de datos (tablas) y el controlador el cual interactúa con los modelos para procesar las solicitudes que llegan desde las rutas y de ese modo enviar la respuesta al cliente en formato JSON. Naturalmente, se hace uso de middlewares que permiten procesar solicitudes como la autenticación con JWT (para proteger rutas y permitir accesos a ciertos recursos), el manejo de CORS para dar acceso al frontend a ejecutar las solicitudes API de forma segura y Multer para controlar el manejo de archivos que usa la aplicación.

Finalmente, la capa de datos es responsable del almacenamiento de toda la información en los consultorios jurídicos, utilizando MySQL que se comunica directamente con la capa de

aplicación por medio de Sequalize, lo que traduce las llamadas a funciones de JavaScript en consultas SQL. Se maneja un archivo para permitir la conexión e iniciar correctamente el servidor de aplicación (index.js).

Figura 22

Arquitectura del sistema



Nota: La figura representa la arquitectura del sistema propuesto para el Consultorio Jurídico de la PUCE. Fuente: Elaboración propia.

3.4.1 Diseño inicial de la Interfaz de Usuario (U.I)

En cuanto al diseño de la interfaz de usuario U.I se aplicarán las principales heurísticas de usabilidad mencionadas anteriormente, además elaborar el diseño principal de la aplicación web debido a que el módulo de gestión de usuarios y primeras consultas contiene la parte inicial del flujo de trabajo operativo del consultorio jurídico. En ese sentido, se diseñará la interfaz (U.I) de inicio de sesión (login), olvidé mi contraseña, menú principal que contiene todas las rutas de la aplicación web, página de inicio (home), vista de primeras consultas (nuevo caso), reportes, administración del sistema y configuración de usuarios. Se profundizará en el diseño en el capítulo IV, en concordancia con el desarrollo siguiendo la metodología del prototipado evolutivo.

Por otra parte, la idea general será mejorar y rediseñar las interfaces del actual sistema, haciendo que la aplicación web sea más fácil de usar por parte de los usuarios finales, utilizando colores simples, íconos, secciones de contenido y un diseño moderno o minimalista orientado a la fácil visualización y llenado de los campos de formularios, así como el acceso a las interfaces deseadas eficazmente.

Figura 23

Diseño mockup (bosquejo): Inicio de sesión

Nota: La figura representa un bosquejo inicial (mockup) de la pantalla de inicio de sesión que usarán los usuarios para acceder a la aplicación web. Fuente: Elaboración propia.

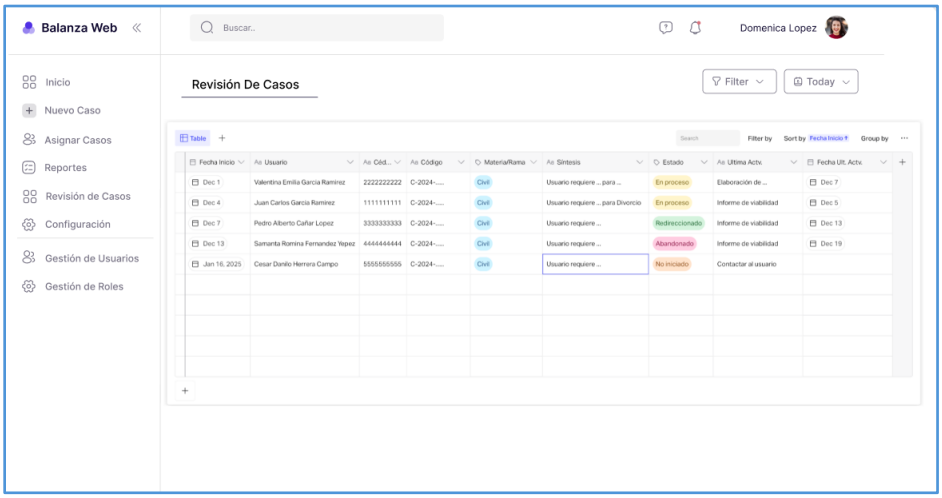
Figura 24

Diseño mockup (bosquejo): Inicio de sesión

Nota: La figura representa un bosquejo inicial (mockup) del menú lateral que usarán los usuarios para navegar entre interfaces en la aplicación web. Fuente: Elaboración propia.

Figura 25

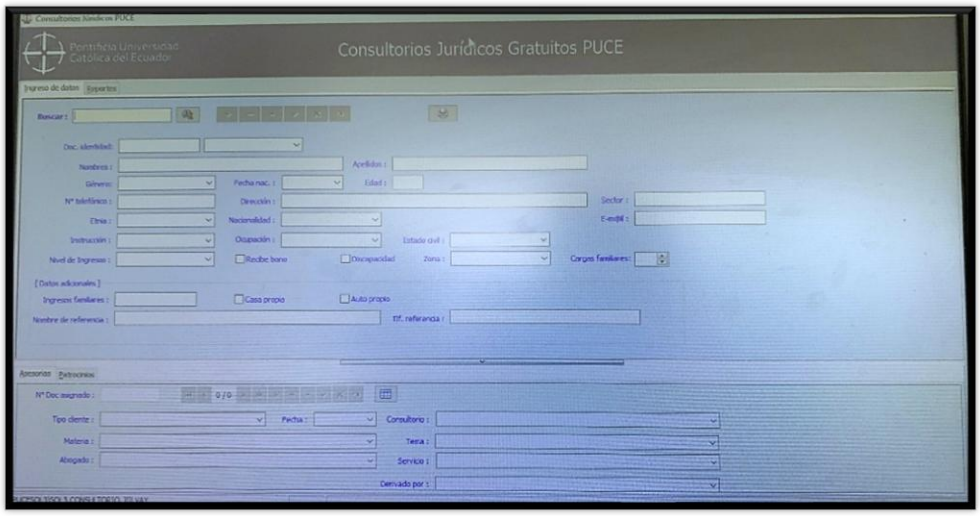
Diseño mockup (bosquejo): Concepto inicial de la aplicación web



Nota: La figura representa un concepto inicial general de cómo se podría ver la aplicación web y su disposición de elementos en la interfaz. Fuente: Elaboración propia.

Figura 26

Sistema Balanza: Primeras Consultas

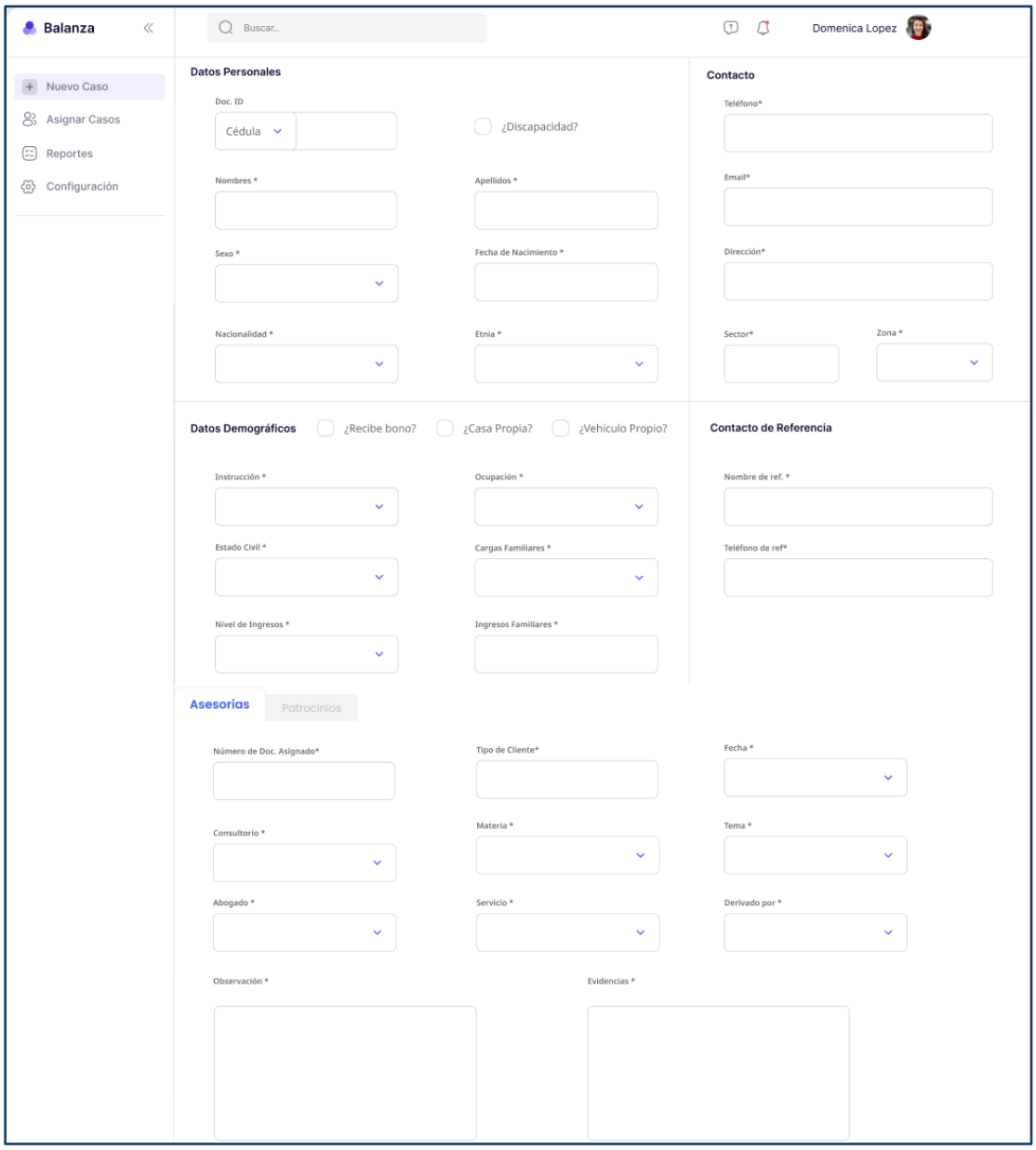


Nota: La figura muestra la pantalla principal del Sistema Balanza usado actualmente por Consultorios Jurídicos, contemplando el área de Primeras Consultas y los reportes de esta.

Fuente: Elaboración propia.

Figura 27

Diseño mockup (bosquejo): Primeras Consultas



Nota: La figura muestra un concepto inicial de cómo se podría ver la interfaz de primeras consultas rediseñada, junto con una nueva disposición de elementos y dividida por secciones de contenido. Fuente: Elaboración propia.

3.4.2 Roles y Permisos de la aplicación web

Como se ha venido mencionando, esta es una parte principal debido a que va relacionado con la gestión de usuarios y la gestión de parámetros, siendo la última un módulo que en resumidas cuentas contiene un CRUD básico de diferentes tablas que almacena los valores y campos que se usan y consumen en toda la aplicación web gestionados por un rol administrador (creado de forma predeterminada junto con un usuario), y entre dichos campos o tablas están los roles que se asignan a cada usuario del sistema.

Naturalmente, los roles son creados, actualizados y eliminados desde parámetros, sin embargo, se creará una interfaz que permita gestionar eficientemente los permisos de cada rol en el sistema, en otras palabras, las vistas que tendrá acceso un determinado rol según lo requiera el administrador del sistema en concordancia con la lógica de negocio de los Consultorios Jurídicos. Cabe recalcar que existen ciertos puntos dentro de cada vista que se gestionarán de forma manual dentro del código (y que serán visibles u ocultados en la interfaz según el rol), debido a que esa es la lógica que busca manejar el Consultorio Jurídico respecto a ciertas funciones que serán accesibles para un determinado rol (tal como se describe en los diagramas de casos de uso), por ejemplo, en las alertas de viabilidad la opción para rechazar el patrocinio será solo visible para el administrador y coordinador.

CAPÍTULO IV: DESARROLLO DE LA APLICACIÓN

4.1 Preparación del Entorno de Desarrollo

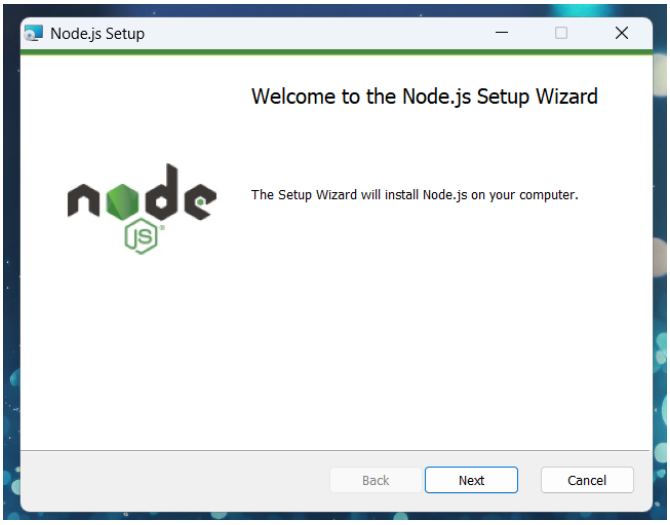
4.1.1 Entorno del Backend (Servidor)

Como punto inicial, se comenzó con la instalación de Node.js en el servidor de aplicación (equipo PC local), configurando un proyecto de cero con el framework Express.js para la creación del API REST, los middlewares, la gestión de rutas y seleccionando JavaScript como lenguaje de programación. En segundo lugar, la instalación de las principales dependencias como Sequelize (ORM), para mapear los objetos en esquemas de tabla y realizar todas las operaciones SQL directamente en métodos integrados. La librería de Json Web Token (JWT) para manejar las sesiones del usuario en tokens que se almacenaran en la cookie del cliente de forma segura, Bcrypt que permite la encriptación de contraseñas antes de almacenarse en la base de datos, el controlador de conexión con la base de datos (mysql2), Nodemailer para la funcionalidad de envío de correos y un archivo de configuración (config.js) para controlar variables globales que se usen dentro del backend como la contraseña segura para desencriptar el JWT.

Posteriormente, la creación y organización de las carpetas principales del proyecto siendo los Esquemas, Modelos, Controladores y Rutas, así como una carpeta que contiene los middlewares de autenticación, manejo de cors y archivos usando multer. Finalmente, la instalación de la base de datos MySQL en un entorno de máquina virtual (VM) con un sistema operativo AnduinOS que está basado en Linux Ubuntu (v25.04), configurando además 4GB de RAM y 4 núcleos de procesamiento para tener tiempos de respuesta óptimos en las consultas SQL.

Figura 28

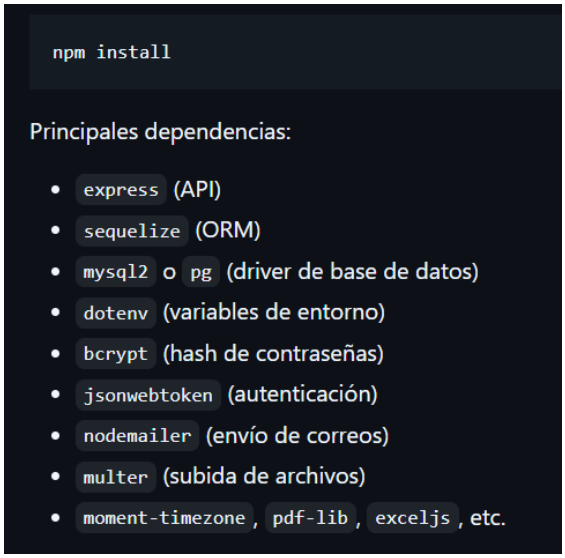
Instalación de Node.js en PC



Nota: La figura muestra el menú de instalación de Node.js en mi PC local. Fuente: Elaboración Propia.

Figura 29

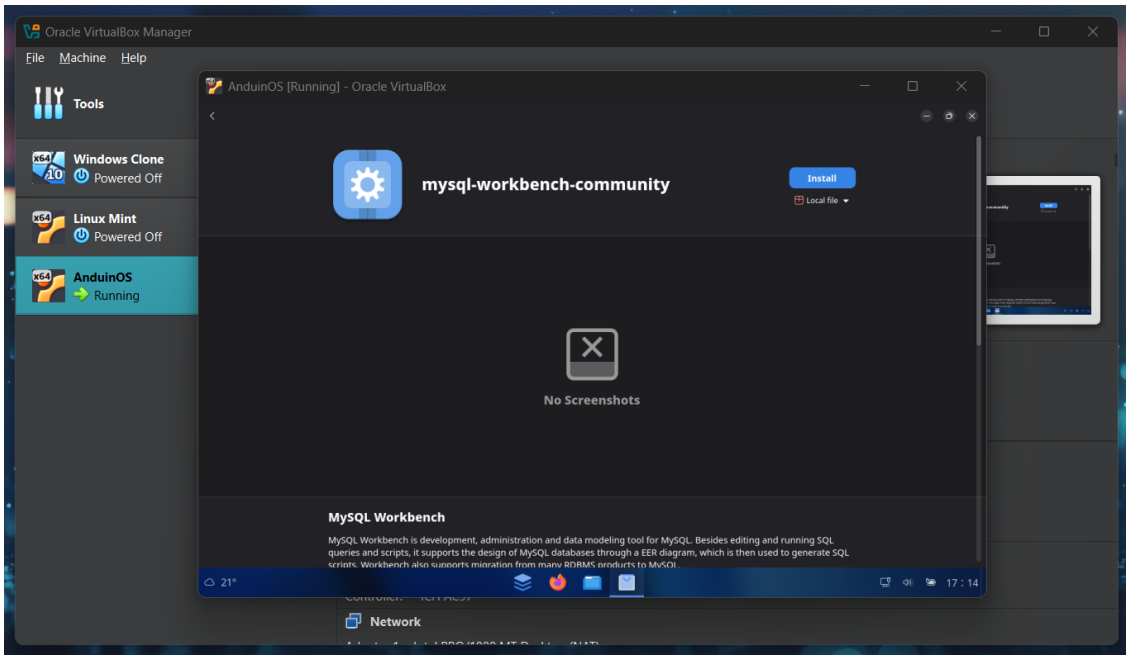
Instalación de Dependencias (Backend)



Nota: La figura muestra las principales dependencias que usa el backend, junto con su comando de instalación. Fuente: Elaboración Propia.

Figura 30

Instalación de MySQL Worbench



Nota: La figura muestra la instalación de la base de datos, junto con su SGBD MySQL Workbench dentro de un entorno virtual con AnduinOS. Fuente: Elaboración Propia.

4.1.2 Entorno del Frontend (Cliente)

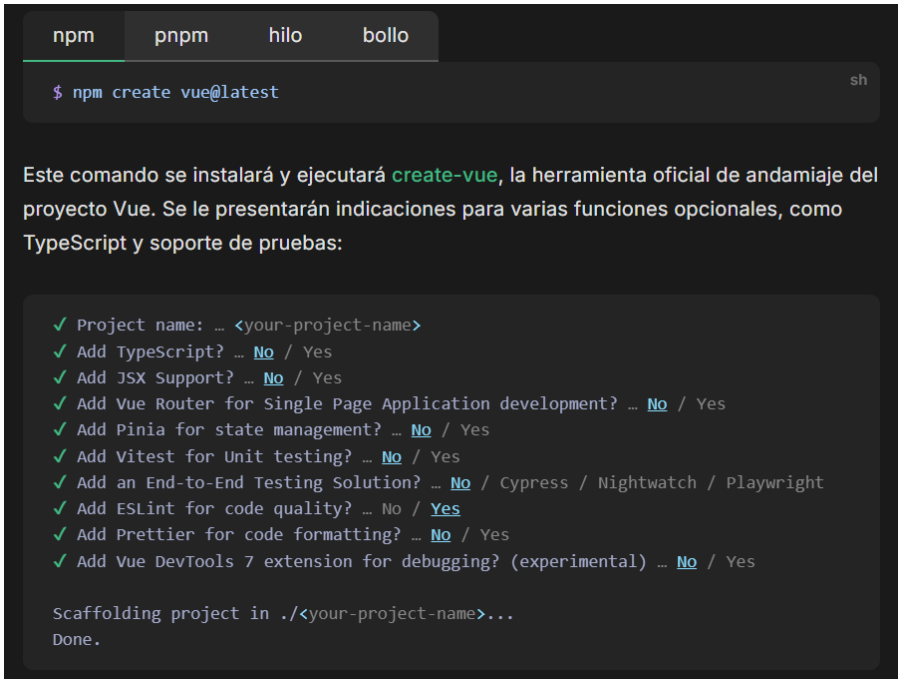
El siguiente punto trata del entorno que se construyó para el desarrollo del frontend, siendo una aplicación de una sola página SPA (Single Page Application) utilizando el framework Vue.js (v3), además se configuró el proyecto con la tecnología de Vite y sus características que ya se han mencionado como lo es la recarga en caliente, un servidor de desarrollo rápido. Se usó TypeScript para aprovechar sus ventajas de tipado y mejorar la mantenibilidad del código en general y finalmente se incluyó a Pinia en la configuración que es la biblioteca para manejar los estados de

la aplicación y la sesión del usuario. Por otra parte, se usó Vue-Router para poder gestionar todas las rutas de navegación dentro de la aplicación web.

Una vez fue configurado el proyecto, se instaló PrimeVue y Tailwind como las librerías principales para el diseño general de componentes de interfaz, garantizando un aspecto profesional, moderno y minimalista a la aplicación web. En ese sentido, para hacer uso de PrimeVue, se configuró un plugin necesario para la instalación y configuración inicial dentro del archivo "main.ts", además de configurar el tema que usará la aplicación (Aura en este caso). Naturalmente, el proceso fue bastante similar en cuanto a la configuración de Tailwind en el proyecto, siendo necesario indicarle a Vue que debe usar los plugins de ambas librerías en el archivo "vite.conf.ts" y finalmente importar los estilos dentro del archivo global de CSS (main.css).

Figura 31

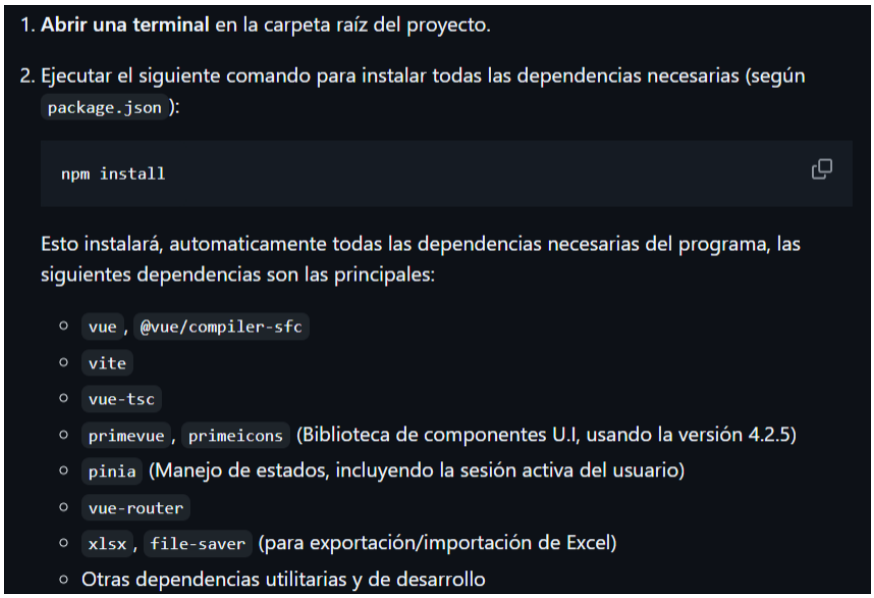
Ejemplo de instalación y configuración de un proyecto Vue



Nota: La figura muestra un ejemplo de instalación y configuración de un proyecto Vue, donde se observa todas las características seleccionables según las necesidades del desarrollador. Tomado de *Creación de una aplicación Vue*, por Vue, 2025.

Figura 32

Instalación de Dependencias (Frontend)



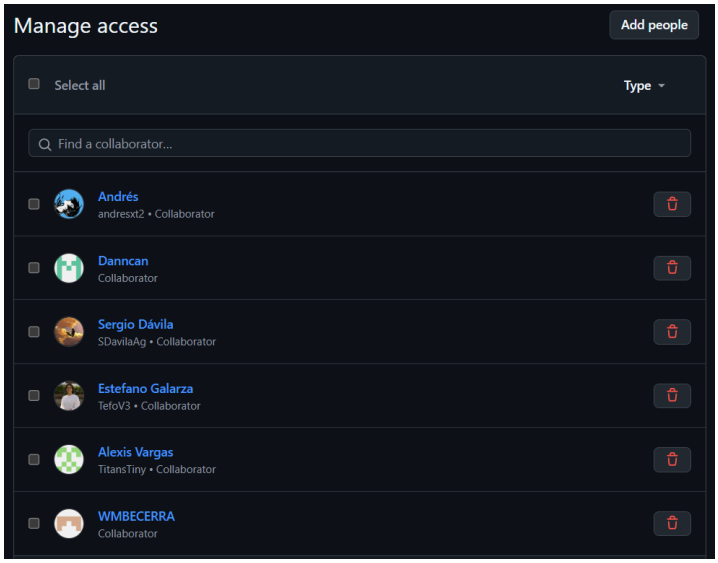
Nota: La figura muestra las principales dependencias que usa el frontend, junto con su comando de instalación. Fuente: Elaboración Propia.

4.1.3 Control de Versiones y Herramientas

En relación con el control de versiones, se utilizó Git y el código fuente se guardó en dos repositorios Github, el primero para la parte del Backend y otro para el Frontend, asignando los permisos respectivos solo a los miembros del equipo de manera que se pueda evitar cualquier cambio de terceros, facilitando el seguimiento de cambios y colaboración. Finalmente, todo el desarrollo se llevó a cabo con el editor de código gratuito Visual Studio Code.

Figura 33

Colaboradores del Repositorio Balanza Web (FrontEnd)



Nota: La figura muestra la lista de colaboradores que trabajaron en el repositorio del Frontend, excluyendo mi persona debido a la autoría del repositorio. Fuente: Elaboración Propia.

4.1.4 Entandares de codificación

De manera general, se usaron los lineamientos por parte de la Dirección de Informática – Departamento de Desarrollo (DI) de la Universidad, en consecuencia, el código y su nomenclatura están en el idioma inglés, a excepción del texto que se muestra en la aplicación web naturalmente. Cabe destacar también la reiteración de aplicar prácticas de seguridad, comentarios claros, formato (sangría) para mejor legibilidad, nomenclatura clara y manejo de errores.

4.2 Desarrollo del Backend

4.2.1 Estructura del Backend

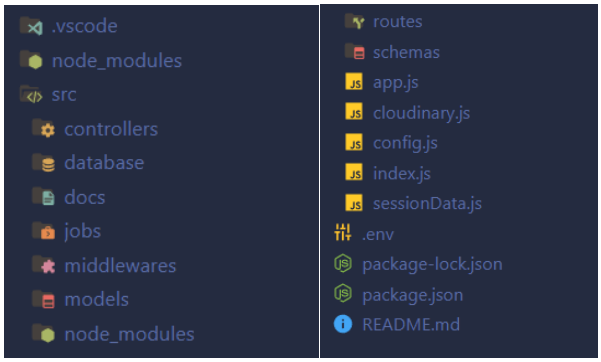
Como se ha venido mencionando, el diseño de la estructura del Backend sigue un patrón arquitectónico similar al Modelo-Vista-Controlador (MVC) con modificaciones como la

eliminación de la vista, y la incorporación de los esquemas de Sequalize. De esa forma, fue construido de forma conjunta con todos los colaboradores, iniciando con la creación de las carpetas que guardan la estructura principal mencionada (Esquemas, Base de Datos, Modelos, Controladores, Rutas, Middlewares y los archivos principales que contienen la configuración dependencias e inicialización el proyecto).

Visto de esta forma, los esquemas contienen la estructura de un modelo Sequalize que representan las tablas (campos PK, tipos de datos y relaciones FK) que se mapearan automáticamente a la base de datos. En segundo lugar, la carpeta de Base de Datos define cómo Sequalize se conecta a la base de datos (en mi caso no por localhost, sino la IP de la VM). Por otra parte, los Modelos contienen las clases que encapsulan la lógica de negocio, CRUD y operaciones principales de cada tabla (entidad) que se traducirán a operaciones SQL, siendo un intermediario con la base de datos y los controladores. Ahora bien, los Controladores se encargan de recibir y responder a las solicitudes HTTP del frontend, haciendo uso de los modelos para interactuar con la base de datos y manejando errores y validaciones. A continuación, la rutas que definen los endpoints de la API para ejecutar las operaciones respectivas (POST, GET, PUT, DELETE) y de esa forma están asociadas a un controlador. Finalmente, la carpeta de Middlewares que contienen la autenticación, manejo de cors y control de documentos (multer).

Figura 34

Estructura de Carpetas (Backend)



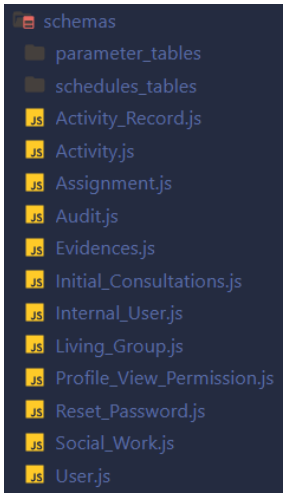
Nota: La figura muestra la estructura de carpetas del Backend, donde está la carpeta principal "src", la cual contiene los principales archivos del proyecto (app.js, esquemas, modelos, controladores, rutas, middlewares, base de datos) y archivos adicionales de configuración como el package.json que contienen las dependencias del proyecto. Fuente: Elaboración Colaborativa.

4.2.2 Implementación de Esquemas y Conexión con la B.D

Debido a que a este punto ya se tuvo la lógica de negocio y el levantamiento de información respectivo (separando las tablas del módulo de parámetros y horarios), se definieron los modelos de datos para cada entidad (tabla), definiendo sus columnas, tipos de datos, relaciones y validaciones. Naturalmente, el ORM nos permite mapear estos esquemas de forma automática en la base de datos MySQL, estableciendo un archivo de conexión en donde se definió la IP de la máquina que contiene la base de datos (AnduinOS) y estableciendo el nombre la base de datos junto con su contraseña. Una vez hecho esto, la función main permite utilizar esa conexión para levantar el servidor en el puerto 3000, creando automáticamente todas las tablas respectivas en caso de no existir.

Figura 35

Estructura de Esquemas (Backend)



Nota: La figura muestra la estructura de carpetas de los esquemas, donde se encuentran todas las entidades que se usarán en la aplicación web, separando en una carpeta a todas las tablas de parámetros y otra a la gestión de horarios. Fuente: Elaboración Colaborativa.

4.2.3 Desarrollo de Modelos y Controladores

En forma general, aquí está la lógica de negocio de la aplicación web que ya se ha explicado anteriormente, donde los modelos son el puente que se comunica con la Base de Datos y el controlador, y de ese modo todas las operaciones principales se encuentran definidas en los modelos ya que serán traducidas a SQL por medio de Sequalize. Por otra parte, los Controladores utilizan los métodos de los Modelos respectivamente para poder comunicarse con el Frontend, recibiendo y enviando peticiones con el protocolo HTTP/HTTPS. Debido al alcance de este módulo en concreto y lo explicado en el Diagrama (E/R), se trabajó con las siguientes entidades: InternalUser (Usuario del sistema), ResetPassword (Reiniciar Contraseña), Profile (Perfiles/Roles), ProfileViewPermission (Permisos de vista por perfil o rol), InitialConsultations (Primeras Consultas), User (Usuario Externo) y Evidences (Evidencia).

Dentro de este orden de ideas, cada modelo tiene su propósito dentro del desarrollo de este módulo, siendo en primer lugar la Gestión de Usuarios, en el cual se desarrolló la lógica principal CRUD en el modelo de InternalUser para gestionar los usuarios, como el método “create” básico dentro del modelo, y dentro del controlador se verifican que los datos sean correctos, se encripta la contraseña con Bcrypt y se le asigna manualmente el ID del rol que se haya elegido para poder gestionar sus permisos. Adicionalmente, están otros métodos principales que contienen la lógica para el inicio de sesión como el método “authenticate” que le permite a un usuario activo y con credenciales válidas ingresar al sistema, generando un token JWT de sesión o devolviendo null si hay algún error. Ahora bien, dentro del controlador el método “login” utiliza el “authenticate” del modelo para poder guardar el token en una cookie que tendrá la sesión activa del usuario (acess_token), con una duración de seis horas y en caso de haber algún error o que el usuario esté inactivo, el token no se creará bajo estas circunstancias. Por otra parte, los métodos para controlar cuando un usuario haya olvidado su contraseña fueron añadidos en el modelo de InternalUser, por lo que contienen la lógica para guardar el código en la tabla respectiva y comprobar si esos dígitos enviados coinciden con lo registrado en Base de Datos; de esa forma en el método “requestResetPassword” del controlador se genera el código de 6 dígitos, y se le envía un correo electrónico al usuario registrado con esa información, garantizando su identidad de forma segura. Finalmente, el método “logout” o cerrar sesión simplemente se lo aplica desde el controlador para poder eliminar la cookie que contiene la sesión.

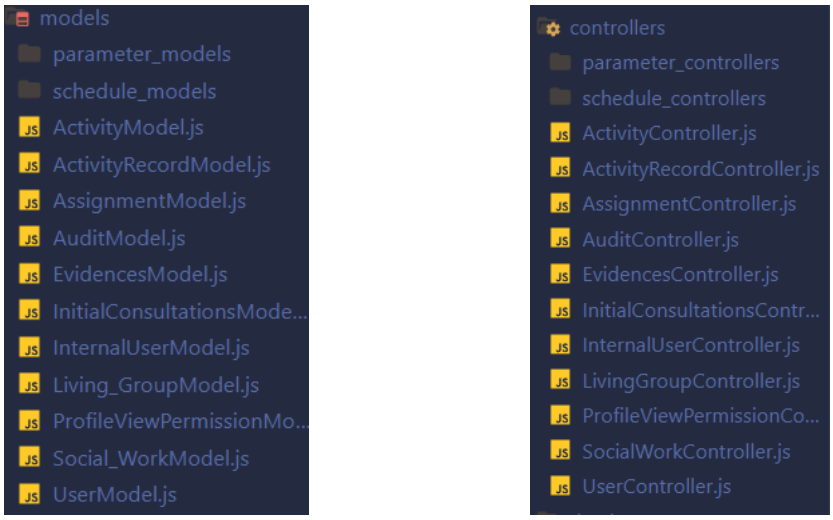
En cuanto a la Gestión de Primeras consultas, se maneja directamente con sus reportes y se tienen los métodos CRUD y adicionales para poder gestionar las fichas de atención de forma eficiente, comenzando con el “Create” que contiene la creación mixta del Usuario Externo, la primera consulta, un registro a la tabla de evidencias en caso de que existan junto a la auditoría

respectiva. En segundo lugar, un método para crear nuevas consultas cuando el usuario ya existe, modificación, rechazo del patrocinio a través del envío de un correo electrónico y la construcción de alertas de viabilidad, que se adjuntan en la creación de las fichas de atención, siempre y cuando se detecta en los datos que cierto usuario cuenta con cierto poder económico o reside fuera del DMQ.

Por último, la Gestión de Roles y Permisos que se maneja directamente con la relación entre la tabla de Usuarios Internos con Roles y la tabla que contiene los permisos para cada rol, gestionándose a través del nombre de cada Ruta del Frontend y el ID de los roles, para poder otorgar o revocar los permisos a cada rol, y de ese modo, estos permisos afectan directamente al Usuario del sistema, debido a que tiene la clave foránea, es decir el ID del rol.

Figura 36

Estructura de Modelos y Controladores (Backend)



Nota: La figura muestra la estructura de carpetas de los modelos y controladores, relacionando los esquemas con modelos y los modelos con sus controladores respectivamente. Fuente: Elaboración Colaborativa.

4.2.4 Creación de Rutas y Endpoints

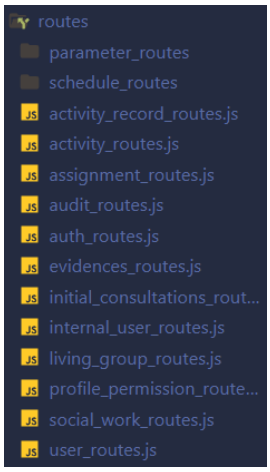
Cabe considerar, por otra parte, la de creación de rutas y endpoints para la construcción del módulo, donde los endpoints se dividieron en públicas y privadas. En ese sentido, dentro de las rutas de Gestión de Usuarios, la autenticación y el control general no requiere que el usuario esté autenticado (deben ser de acceso público), pues son todas las rutas que contienen operaciones POST para el control de ingreso de los usuarios al sistema, entre las que se incluye: al registro de usuarios, inicio de sesión, olvidé mi contraseña, verificar código, reiniciar contraseña (usando el código de reinicio enviado al mail), cambiar contraseña (usando la contraseña actual y verificando la cuenta), el número de intentos que tiene el sistema máximo para poder solicitar un reinicio de contraseña y la ruta más importante (/api/me), que permite verificar la sesión del usuario decodificando su token para poder usar desde el frontend y saber si el usuario tiene una sesión activa. A partir de este punto, no existen más rutas de acceso público, pues contienen operaciones esenciales de la lógica de negocio que no pueden ser expuestas sin una autenticación respectiva, de esa forma, dentro de la Gestión de Usuarios están adicionalmente sus rutas privadas en (internal_user_routes), usando el controlador de "Internal_User" con sus métodos (GET, POST, PUT, DELETE), siendo los más importantes los GET que devuelven la lista de usuarios del sistema, la creación de un usuario del sistema (por el administrador), actualización y eliminación (cambiando el estado de activo a inactivo).

Desde una perspectiva más general, el resto de las rutas para Gestión de Primeras Consultas y Gestión de Roles y Permisos, siguen un proceso de creación similar a lo explicado anteriormente, con la diferencia que existen operaciones adicionales definidas en los controladores claramente y que se utiliza su respectivo método del controlador, como la subida de documentos, generación de

la ficha técnica, recibir los casos según el área asignada del usuario (Civil, Penal, …), obtener y actualizar los permisos para cada rol, entre otros.

Figura 37

Estructura de Rutas (Backend)



Nota: La figura muestra la estructura de carpetas de las rutas, relacionado con sus controladores y el tipo de operación respectivamente. Fuente: Elaboración Colaborativa.

4.2.5 Implementación de Middleware de Seguridad

Si bien se han descrito algunos middlewares y su respectivo uso, en este módulo se trabajó con la creación del middleware de seguridad en concordancia con la Gestión de Usuarios y su respectivo mecanismo de control para evitar que intrusos o un usuario no autenticado accedan a un recurso protegido en la parte de rutas cuando no tiene token válido, es decir una sesión activa que le permita acceder a la aplicación web. De esa forma, el archivo (auth) permite proteger las rutas, verificando que el usuario haya iniciado sesión y tenga un token válido (almacenado en las cookies); la forma en que lo hace es leyendo la cookie de la sesión (access_token), verifica que sea válido (decodificándolo con la clave secreta) y guardando el contenido del token dentro de la sesión para poder usar los datos del usuario autenticado en el Frontend, en caso de no tener una

sesión válida simplemente el usuario no puede acceder a la aplicación web. Este mismo middleware es el que se utiliza para poder dividir las rutas públicas y privadas, donde se lo llama directamente después de definir las rutas públicas, haciendo que el resto de las rutas tenga un acceso restringido hasta que el usuario esté autenticado.

4.3 Desarrollo del Frontend

4.3.1 Estructura del Frontend

Uno de los componentes más importantes de Vue.js junto con Vite es la configuración del proyecto tal como se había mostrado antes, influyendo en la estructura del proyecto como tal. En este caso, la estructura consta de una estructura de carpetas dividida en los módulos de node, el src (fuente) y los principales archivos del proyecto como lo es la página principal (index.html), donde se renderizará todo el contenido de forma dinámica en concordancia con la tecnología SPA, los archivos que contienen las dependencias principales, la configuración de TypeScript para la aplicación y la configuración de Vite (definiendo los plugins necesarios) para construir la aplicación web.

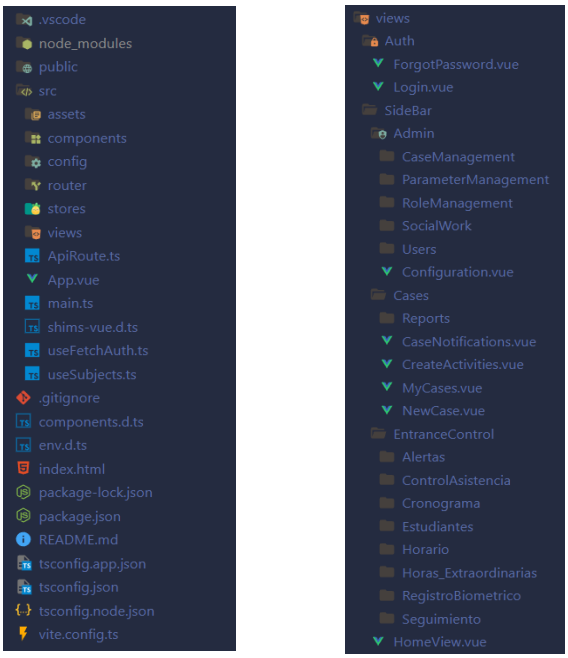
No obstante, la carpeta principal será el “src”, pues contiene todo el núcleo del Frontend ya que se definen los estilos (assets), los componentes de la aplicación (components), la configuración de vistas para gestionar los permisos de cada perfil de usuario o rol (config), la definición de vistas dentro de la aplicación web (router), control de estados (stores), las interfaces de toda la aplicación (views) y archivos de configuración como “ApiRoute”, “App” y “main”.

Ahora bien, de forma global las interfaces de la aplicación se dividen en dos grupos principales “Auth” carpeta la cual contiene las dos únicas vistas públicas que puede ver un usuario sin estar autenticado y adicionalmente, “SideBar” el cual hace referencia al menú lateral de sistema y que está dividido por vistas que podría ver el administrador “Admin” y que contiene vistas de

gestión de usuarios, roles/permisos, trabajo social, configuración de usuario. En otro apartado, la carpeta de “Cases” que contiene las vistas de primeras consultas y las actividades del caso de manera general. Finalmente, la carpeta de “Entrance Control” siendo las vistas que se manejan en el módulo de control de ingreso desarrollado para la Secretaria de Consultorios Jurídicos.

Figura 38

Estructura de Carpetas (Frontend)



Nota: La figura muestra la estructura de carpetas del Frontend, donde se puede observar por un lado los archivos principales que construyen la aplicación y el contenido de la carpeta “src”, donde está el núcleo de la aplicación; en el otro lado, se pude ver la carpeta “views” que contiene todas las interfaces y que está divida por vistas de acceso público y privado (Auth y SideBar respectivamente). Fuente: Elaboración Colaborativa.

4.3.2 Gestión de Estado Global (Pinia)

El estado principal que se maneja dentro de la carpeta “stores” es la sesión de usuario en la aplicación web (en el archivo auth.ts), permitiendo guardar la información del usuario y que permisos tiene, además de hacer que cualquier componente de la aplicación pueda acceder a dichos componentes de forma ordenada y reactiva. Dicho estado funciona por medio de 3 partes principales, comenzando con el Estado (state), el cual permite definir las variables que contienen la información de la sesión y que en este caso es el objeto usuario (que ha iniciado sesión), guardando su información que llega del token (ID, nombre, rol, ...); adicionalmente, se almacena un diccionario de permisos que indica las interfaces que puede ver el usuario dependiendo el perfil o rol que tenga.

En segundo lugar se tiene a las propiedades computadas (getters), siendo funciones que leen el “state” y devolver un valor derivado, por lo que se tienen funciones como “isAuthenticated” para saber rápidamente si hay una sesión activa, “canView” para ver si un usuario actual tiene el permiso de acceder a una determinada vista, destacando que si el usuario es de tipo Administrador, siempre podrá ver todo, caso contrario se consulta el “state” del diccionario de permisos y se verifica si la vista tiene un valor “true” para el rol actual del usuario, si esto no es así entonces el usuario verá un mensaje indicando que no tiene el permiso respectivo.

En tercer lugar, se definen las acciones (actions) que consisten en métodos pueden modificar al estado, donde se incluye una función para verificar la sesión “verifySession()” ejecutada al cargar la aplicación y que llama al endpoint (/api/me) explicado anteriormente, haciendo que, si el usuario tiene una sesión válida, se guarden sus datos en el estado. Además, hay otras acciones como cargar los permisos para el usuario, donde se usa el id del perfil del usuario para llamar a la función y verificar desde el backend (/api/profile/auth/:profileId) los permisos que

tiene ese rol en particular; el método para cerrar sesión (logout) y la función “setUser()” la cual recibe los datos del usuario que llegan del backend y los guarda en el estado cuando hay algún cambio en sus datos, permisos de rol o cuando se reestablece su sesión.

4.3.3 Comunicación con el Backend

La comunicación con el Backend se da a través de dos partes principales que trabajan de forma conjunta, en primer lugar está el archivo “ApiRoute.ts”, el cual su función es definir en un solo sitio la dirección del servidor (backend), exportando una constante llamada API; esto permite que en cada operación que se necesite utilizar los endpoints no sea necesario escribir en todas las funciones, sino solo cambiar en ese archivo, facilitando completamente el manejo de las APIs, más aún cuando se ponga la aplicación web en un entorno de producción debido a que la dirección cambiará.

De este modo, cuando se necesita utilizar una operación de las API definidas en las rutas, se utilizó “axios”, que básicamente permite enviar y recibir información entre el frontend y el backend de forma segura y eficiente. Si bien ya existe una herramienta para esto (fetch), axios tiene la ventaja de mejorarlo y simplificarlo, como en el envío de credenciales (sesión) por ejemplo, siendo una característica esencial ya que recordemos que las rutas tienen permisos y esto lo maneja la librería de forma automática; por el contrario, con axios se debe especificar como parámetro que se desea enviar utilizando credenciales. De forma general, el flujo consiste en que la constante “API” le agrega la dirección al endpoint para completar la URL, enviando la petición por medio de axios al backend y siguiendo el flujo en concordancia al tipo de operación, pero siempre recibiendo y devolviendo los datos en JSON.

4.3.4 Desarrollo de la Interfaz de Usuario (UI/UX)

Primeramente, se eligió la tecnología principal de Frontend (Vue.js) porque fue un lineamiento por parte de la Dirección de informática de la PUCE (DI - Desarrollo), siendo una tecnología utilizada por el equipo. No obstante, más allá de este aspecto y el concepto ya analizado de esta tecnología, se puede ver la importancia de cómo el framework permite un desarrollo a nivel profesional, robusto y que contiene las mejores prácticas que permite a la aplicación web ser mantenible durante mucho tiempo. Por otra parte, Typescript que permite un tipado estático, mayor escalabilidad/mantenibilidad, y la transformación a JavaScript en navegadores que no lo admiten nativamente.

4.3.4.1 Desarrollo del Diseño Inicial del Proyecto (U.I)

Ahora bien, luego de analizar y construir el entorno de desarrollo se comenzó con la construcción de dos tipos de disposición (layout) de diseño para la aplicación, ya que recordemos que existe una parte pública donde se necesita que el usuario pueda ver todo el contenido de la página y en las privadas es necesario desplegar el contenido unos cuantos pixeles hacia la derecha, debido a que es el espacio designado para el menú lateral de navegación. De esa forma, se definieron dos archivos, el primero (AuthLayout) el cual es el privado y no contiene ninguna alteración en la disposición de elementos y el segundo (DefaultLayout) donde se integra directamente con el componente del menú principal (SideBar) y se despliega un poco el contenido hacia la derecha, renderizando las vistas en ese espacio.

Luego de esto, la construcción de las rutas de navegación que usará la aplicación es necesario porque sin ella no hay navegabilidad, por lo que se define en la carpeta “router” haciendo uso de la biblioteca (vue-router) para gestionar esto para la aplicación

(SPA) siendo necesario definir las rutas de cada vista, el componente del layout que usará la vista (público o privado), su nombre, ruta y si requiere autenticación, esto por medio de un guardia de navegación que da un acceso libre a la página de inicio de sesión, donde si no está autenticado redirige al usuario siempre y en el caso de que el usuario ya este autenticado, pero coloca una ruta a la que no tiene permisos desde el navegador, se lo redirige al inicio (home), con un mensaje de error. Esto se logra gracias al estado de Pinia y sus acciones, ya que aquí es una de las partes donde se utiliza la lógica para verificar si el usuario puede ver esa ruta para bloquear el acceso como ya se mencionó.

El menú de la aplicación web es un pilar fundamental en el diseño inicial (U.I), porque es la herramienta que permite la navegación entre interfaces al usuario, por lo que debe ser intuitivo, atractivo y fácil de usar. Este es un componente que consta de tres partes fundamentales, siendo en primer lugar la navegación por roles que permite usar el estado de Pinia (definiéndose como authStore) para poder renderizar las opciones del menú de forma condicional, utilizando las acciones que ya se hablaron anteriormente (si el rol tiene acceso a una vista determinada, se procede a renderizarla), esto mediante el getter (canView) que permite mostrar u ocultar cada enlace según los permisos, los cuales son controlados en la Gestión de Roles/Permisos que evita que un usuario vea opciones que no debería. En segundo lugar, el uso de los componentes de Vue y PrimeVue para la construcción del menú, definiendo listas desordenadas de elementos, iconos, secciones de contenido y estilos. En tercer lugar, el apartado donde aparece el nombre del usuario que ha iniciado sesión y su foto de perfil siendo necesario el uso de los componentes (Menu y Avatar), siendo el primero las opciones que aparecen cuando el usuario hace clic en su perfil (Cerrar sesión, Configuración) y el avatar que es el circulo donde se renderiza la foto

a través del atributo (Internal_Picture); por lo tanto, lo principal es que el estado de Pinia es el que brinda la información del usuario actual para poder cargar esa información en esa sección del menú. Finalmente, cabe destacar que está un ícono de Sol/Luna, siendo el modo claro u oscuro de la aplicación, lográndose gracias al uso de PrimeVue ya que cuenta con la funcionalidad al alternar entre sus temas, siendo una característica que se debe definir dentro de la construcción de la aplicación como una opción de la biblioteca y un archivo que lo gestione (ThemeSwitcher).

Una vez que estos primeros componentes fueron construidos como parte del diseño inicial (UI) de la aplicación web, se definió el estilo de diseño que deben seguir los otros miembros del equipo en el desarrollo de sus módulos para la aplicación, con el fin de tener consistencia profesional de las interfases y una buena experiencia de usuario (colores, diferentes componentes de PrimeVue, notificaciones Toast, etc.). De este modo, se procedió a desarrollar todas las interfaces del módulo propuesto (Gestión de Usuarios, Gestión de Roles/Permisos y Primeras Consultas).

4.3.4.2 Desarrollo de las Interfaces: Gestión de Usuarios

Comenzando por la Gestión de Usuarios, está la vista de inicio de sesión (login), compuesta por el correo y contraseña del usuario, y en cuanto a diseño, se utilizó el logo oficial del Consultorio Jurídico, junto con una imagen difuminada de la Universidad. También está la vista de “olvidé mi contraseña”, que es accesible mediante la anterior y está compuesta por cuatro secciones (petición de correo, ingreso del código de verificación, ingreso de nueva contraseña junto con su confirmación y una sección de regreso en caso de un cambio exitoso). En cuanto a las vistas que serán privadas a partir de este punto, está la interfaz de inicio (home) siendo el comienzo que ve cada usuario que ha iniciado sesión

correctamente y que contiene la misión, visión y objetivos del CJGP, además de un teléfono de contacto.

En cuanto a la interfaz de gestión de usuarios, es una vista compuesta por la lista de usuarios actuales del sistema que muestra su foto de perfil, nombres, correo electrónico, su rol y las acciones que se pueden realizar sobre los registros (ver detalles, editar, eliminar); también se puede crear usuarios desde un botón que redirige a la interfaz designada para ese propósito, destacando las validaciones que se hacen sobre los datos para su creación (ID, correo electrónico, contraseña, rol, etc.), principalmente asegurándose que no haya un usuario con el mismo ID, ni utilizar correos ya registrados o no válidos. Finalmente, está la configuración del usuario que se accede desde el menú, el cual despliega la interfaz que contiene los datos sobre el usuario, la sección de cambio de contraseña y foto de perfil.

4.3.4.3 Desarrollo de las Interfaces: Primeras Consultas

Las interfaces de Primeras Consultas siendo la principal y más compleja “Nuevo Caso”, que sigue el respectivo flujo del proceso de manera automatizada. En este caso, está compuesta de cinco partes principales, comenzando por introducir la cédula o pasaporte del usuario que acude a la consulta y de esa forma la aplicación cargará la información del usuario externo junto con sus fichas de atención, por otro lado, cuando no existe el usuario se habilita los campos, pero no se despliega la barra de navegación de fichas de atención, ni las opciones de actualización, exportar ficha técnica y alertas.

En segundo lugar está la lógica de los campos, donde los que son de tipo lista seleccionable son cargados dinámicamente a través de su endpoint URI, por lo que si se llega a hacer alguna operación CRUD sobre esos campos afectará directamente a lo que se

le despliegue como opciones al usuario, haciendo que sea un proceso dinámico y modular para la gestión de las opciones que requiera los Consultorios Jurídicos, sin una intervención técnica; adicionalmente, está la validación de campos obligatorios que van en concordancia con la lógica de negocio, y de ese modo, al no ser llenado un campo obligatorio (*), se despliega un mensaje indicando el requerimiento para continuar.

En tercer lugar, está lógica para almacenar los documentos, siendo ya mencionada en el backend y en el lado del cliente, simplemente se hace uso de las APIs para poder ser enviados usando una verificación del espacio (2MB).

En cuarto lugar, la forma en la funciona la gestión de fichas de atención y actividades del caso, recordando la relación de base de datos, donde un usuario puede tener varios casos (fichas de atención) y de ese modo, cuando se crea su primera consulta es desplegado en la parte inferior una paginación que permite navegar entre todas las atenciones que haya recibido el usuario, además de las opciones que permiten dicho control (Crear nueva ficha de atención, editar la ficha, exportar la ficha a PDF y las alertas). Es importante mencionar que en la lógica de negocio general de esta interfaz si usuario del sistema es un estudiante o abogado, no podrá editar la ficha de atención o rechazar los patrocinios, por otra parte, los abogados si pueden ver las actividades del caso, que van en concordancia con lo que hace cada estudiante para ir resolviendo su gestión con el caso que le haya tocado y que son simplemente consumidas a través de una API de ese modulo en concreto. Finalmente, las alertas de viabilidad que son ciertos datos donde el usuario no cumple con el requerimiento necesario para que el consultorio pueda dar su patrocinio, en otras palabras, le indican al usuario que no es viable llevar su caso por una determinada razón (ya sea que resida fuera del DMQ, tenga los recursos económicos necesarios, etc.)

Dentro de las interfaces de Primeras Consultas, también está la revisión de casos, donde este es un proceso que actualmente se lo hace manualmente, es decir, se revisa la ficha técnica que se genera del caso llenado por un estudiante, y debe ser aprobada por el coordinador. De esa forma, esta interfaz funciona como una bandeja de casos en forma de tabla (incluyendo varios filtros de búsqueda por parámetro o general), donde se van incorporando a la lista de forma automática una vez han sido llenados, haciendo que el coordinador tome la decisión de dar efectivamente el patrocinio, rechazarlo o dejarlo como una asesoría (luego de revisar toda la información usando el botón de redirección al caso). Por otra parte, una interfaz que permite ver todos los casos (cualquier ficha de atención) de todos los usuarios externos, con la incorporación de filtros de búsqueda, la opción para ver la ficha técnica correspondiente al caso y una redirección al caso completo con los datos del usuario y su ficha de atención. Además, está la interfaz de reportes que permite seleccionar el tipo (Primeras Consultas) y que es generado automáticamente un documento en formato Excel que contiene toda la información de los usuarios externos junto con sus fichas de atención en un rango seleccionado de tiempo, con las respectivas validaciones.

4.3.4.4 Desarrollo de las Interfaces: Gestión de Roles/Permisos

En último lugar, la Gestión de Roles/Permisos que es una interfaz localizada en el apartado de administración y que contiene una tabla de todos los roles definidos en el sistema, junto con la opción que abre un cuadro de dialogo y permite habilitar o desactivar el acceso a una determinada interfaz del sistema, tal como ya se ha venido explicando. Por otra parte, hay un botón que permite redireccionar a la gestión del rol (crear, ver, editar, eliminar), asegurando principalmente la integridad de los datos por medio de las

validaciones, es decir, si el administrador quiere eliminar un rol el cual está asignado a varios usuarios, naturalmente no podrá hacerlo.

4.4 Implementación del Sistema de Seguridad

4.4.1 Protección de Credenciales (Bcrypt)

Uno de los componentes más importantes es nunca almacenar contraseñas del usuario en texto plano, porque sencillamente no tendrán ningún tipo de protección y cualquier persona que tenga el conocimiento para acceder a la Base de Datos puede hacer uso de esa información, trayendo graves consecuencias, especialmente en un ambiente que maneja datos diarios de carácter delicado como lo es el Consultorio Jurídico de la PUCE. Es así como se añadió una capa de protección "hashing de contraseña", lo que permite transformar la contraseña de texto plano en hash a través de un algoritmo matemático; no obstante, este proceso no es suficiente pues aún es vulnerable a ciertos ataques como la tabla arcoíris, por ende, la mejor forma es utilizar el "SALT" en el proceso hash, lo que implica agregar una cadena de caracteres al azar a la contraseña antes de aplicarse la función. Dentro de este orden de ideas, se garantiza que si dos o más contraseñas son iguales nunca tendrán el mismo hash porque serán únicas, agregando una capa extra de seguridad contra los ataques de fuerza bruta.

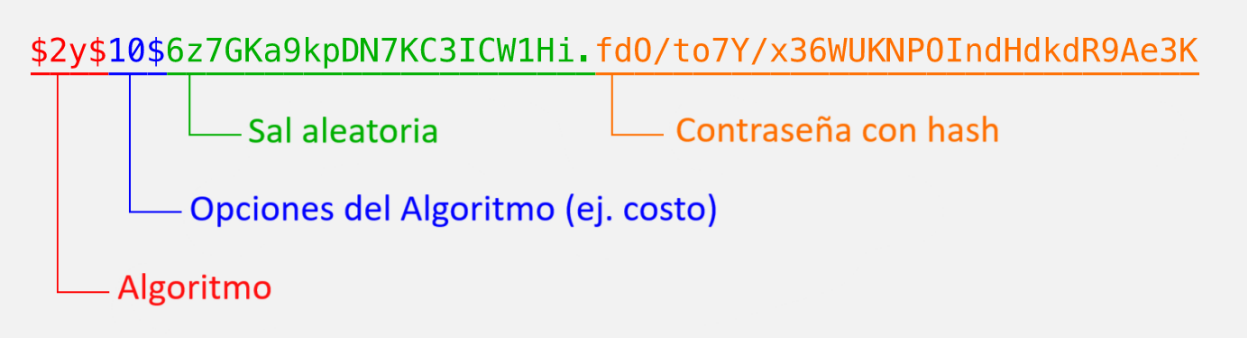
Existen funciones criptográficas que se pudieron haber utilizado, como por ejemplo la familia SHA, sin embargo, el problema de estas funciones es que son computacionalmente rápidas, es decir que, mientras más rápido calcula el hash la función, el ataque de fuerza bruta también será veloz (como para probar millones de hashes por segundo), afectando en la seguridad directamente. Considerando esta vulnerabilidad, se optó por usar la función de hash "Bcrypt" que está basado en el cifrado blowfish (pez globo), usada como ese medio para la construcción de contraseñas seguras y protegidas entre los usuarios del sistema, porque en primer lugar utiliza el "SALT" ya

mencionado, anulando la efectividad de los ataques de tabla de arcoíris (rainbow tables). En segundo lugar, es un algoritmo lento siendo configurable para incrementar a medida que las computadoras se vuelvan más rápidas (escalando a futuro), y esta característica hace que los ataques de fuerza bruta tomen mucho más tiempo en poder obtener la contraseña, porque fue diseñado para ser computacionalmente caro y lento, a diferencia de SHA-256.

Ahora bien, ya se había mencionado sobre cómo era el proceso de creación de usuarios, involucrando a Bcrypt en el cifrado de la contraseña, donde primero se utiliza el parámetro de "SALT" para poder generar la cadena aleatoria de caracteres, una vez hecho esto, la función concatena la cadena generada con la contraseña antes de aplicar el hash, luego el algoritmo es ejecutado con dicha cadena donde realiza múltiples rondas de hashing y el valor hash es almacenado en la base de datos. Con respecto a la verificación en el inicio de sesión, lo que hace el sistema es extraer la cadena guardada de la Base de Datos, y ahí Bcrypt extrae la "SALT" que se había generado originalmente, y el algoritmo vuelve a generar un hash de esa contraseña ingresada (usando la misma SALT), comparándolo con lo almacenado y si coinciden el usuario tiene acceso, porque la contraseña es la correcta. Así es como se garantiza la seguridad de las cuentas de los usuarios en la aplicación web, utilizando una función robusta diseñada para este propósito.

Figura 39

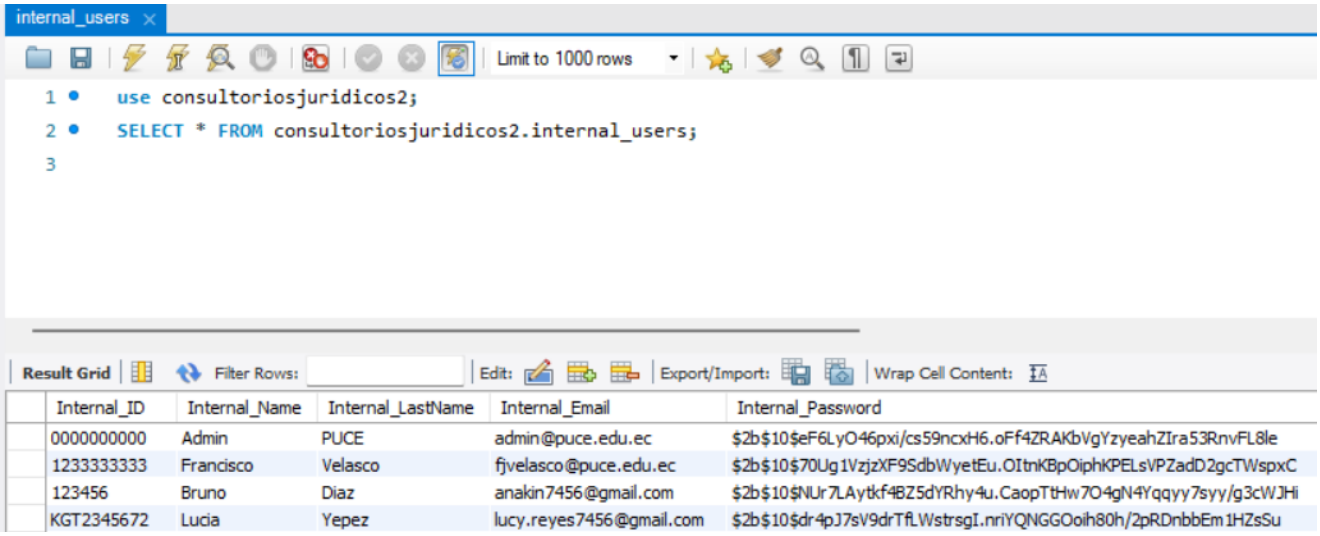
Estructura de Bcrypt



Nota: La figura muestra la estructura del valor hash elaborado por la función Bcrypt, transformando la contraseña de texto plano con su SALT a dicho valor. Fuente: Elaboración Propia.

Figura 40

Funcionamiento de Bcrypt (Base de Datos)



Internal_ID	Internal_Name	Internal_LastName	Internal_Email	Internal_Password
0000000000	Admin	PUCE	admin@puce.edu.ec	$2b$10$eF6LyO46pxi/cs59ncxH6.oFf4ZRAKbVgYzyeahZIra53RnvFL8le
1233333333	Francisco	Velasco	fjvelasco@puce.edu.ec	$2b$10$70Ug1VzjzXF9SdbWyetEu.OItnKBpOiphKPELsVPZadD2gcTWspxC
123456	Bruno	Diaz	anakin7456@gmail.com	$2b$10$NUr7LAytkf4BZ5dYRhy4u.CaopTtHw7O4gN4Yqqyy7syy/g3cWJHi
KGT2345672	Lucia	Yepez	lucy.reyes7456@gmail.com	$2b$10$dr4pJ7sV9drTfLWstrsgI.nriYQNGGOoih80h/2pRDnbbEm1HZsSu

Nota: La figura muestra el funcionamiento de la función Bcrypt aplicado a las contraseñas de los usuarios del sistema, protegiéndolas contra los ataques de fuerza bruta y tabla de arcoiris. Fuente: Elaboración Propia.

4.4.2 Gestión de Sesiones y Protección de Rutas (JWT)

En función de lo planteado anteriormente, luego de encriptar la contraseña del usuario es necesario una forma de indicar si el usuario está autenticado, es decir, la forma de saber si ha iniciado sesión con sus credenciales correctamente (manejar dicha sesión en la navegabilidad de la aplicación web), es quien dice ser y tiene autorización para ejecutar las operaciones privadas de la API REST, en concordancia con lo descrito en el desarrollo del Backend. Para lograr esto, se utilizó en la aplicación la autenticación basada en tokens con JWT (Json Web Token), siendo la herramienta que en este contexto nos permite intercambiar información de forma segura entre dos partes, lo que verifica que los usuarios sean quienes dicen ser y tienen la autorización para usar las rutas API protegidas (punto ya descrito a profundidad anteriormente). En ese sentido, su estructura consta de tres partes (Cabecera, Carga útil y Firma), siendo uno de los puntos más importantes la clave secreta dentro de la firma, ya que con ella es posible la decodificación del JWT.

Su uso dentro de la aplicación web radica en su uso, siendo la autenticación sin tener que guardar los estados de la sesión en el servidor, ya que toda la información necesaria se encuentra directamente en el token, siendo escalables e interoperables y aseguran la seguridad tanto en la sesión como en las rutas API REST.

Ahora bien, para implementarlo es necesario la instalación de la dependencia y crear el token, siendo un método creado dentro de Modelo de Internal_User, con el nombre de “authenticate” el cual recibe como parámetro el correo y contraseña del usuario, se hace la verificación respectiva sobre sus credenciales, donde si el usuario ha verificado su identidad correctamente (además de estar Activo), se crea un token JWT (estableciendo la clave secreta en las variables de entorno), guardando en la carga útil (payload) con los datos principales del usuario que serán recuperados en el Frontend y su tiempo de vida (6 horas). Se plantea entonces el

problema sobre donde guardar el token, ya sea local storage o cookies, y en este caso se optó por evitar local storage porque las cookies son menos vulnerables a los ataques Cross Site Script (XSS), siendo una capa extra de seguridad, además de que tienen un tiempo de expiración de forma nativa y son configurables para ser enviadas solo por HTTPS; si bien estas son medidas extras de seguridad en lo que respecta a utilizar la cookie sobre el local storage, no es una garantía de 100% seguridad, pero si asegura en gran medida contra los ataques mencionados, más aún si se configura en el entorno de producción para que se envíen solo por el dominio de la PUCE.

Por consiguiente, dentro del controlador, una vez se llama al método (authenticate), se procede a retornar como respuesta la cookie, la cual tiene el token guardado como "access_token" y configurando los parámetros de seguridad como "httpOnly" (la cookie solo se puede acceder en el servidor), "secure" (la cookie solo se puede acceder en https), "sameSite" (la cookie sole se puede acceder sobre el mismo dominio) y "maxAge" (el tiempo de validez de la cookie).

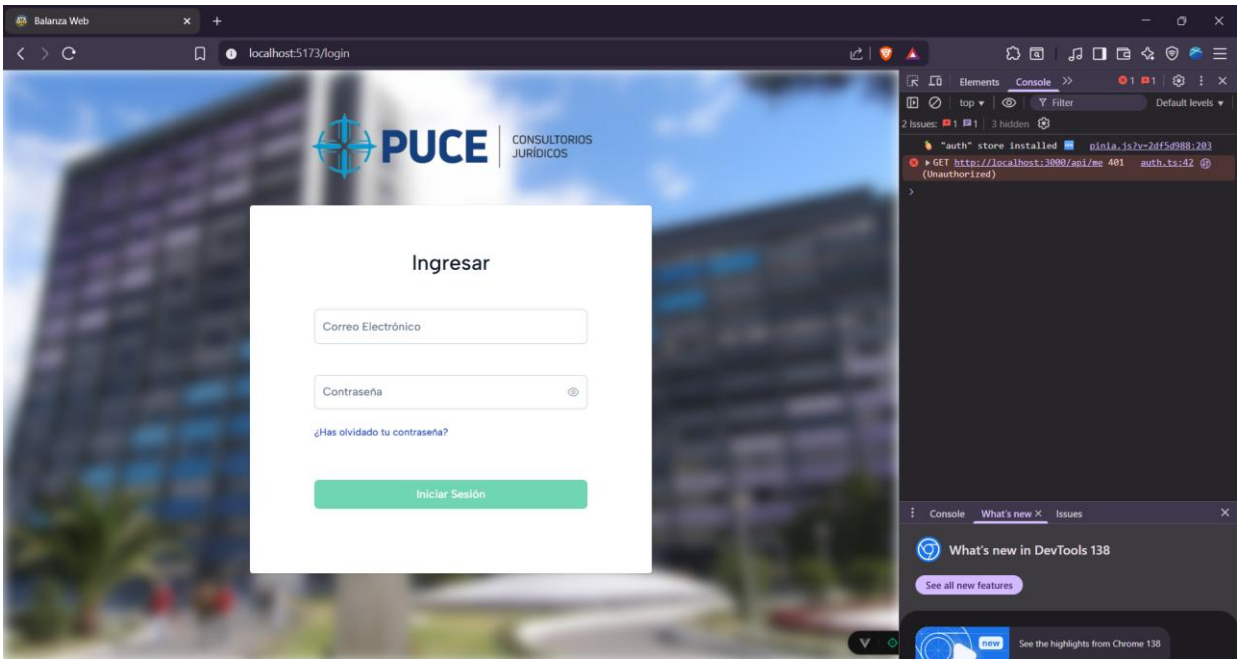
En relación con este tema, una vez ya implementado el mecanismo para gestionar las sesiones, el middleware para la seguridad de protección de rutas entra en juego, el cual ya se ha explicado brevemente, haciendo uso de la URI (/api/me), el cual brevemente llama al middleware de autenticación (authMiddleware), que recibe la cookie de la sesión desde el cliente y se hace un proceso de verificación, si el usuario no tiene un token válido hay un error que se maneja como mecanismo para redirigir a la interfaz de inicio de sesión siempre, haciendo imposible ir a otra ruta que no sea esa hasta obtener un token válido. Por el contrario, si el token es validado correctamente con su clave secreta "jwt.verify(token, KEY)", entonces el usuario esta correctamente autenticado y tiene una sesión válida.

Finalmente, dentro de la capa de aplicación (Backend) se define primero las rutas públicas (sin protección) y luego se llama al middleware de autenticación (authMiddleware), antes de

definir las rutas privadas, logrando que se protejan hasta verificar que el usuario es quien dice ser y tenga el permiso para consumir la información de la URI con respecto a las rutas del servidor, no obstante, en el lado del cliente se utiliza un manejador de estados llamado pinia (para saber que permisos y accesos puede tener el usuario en la aplicación web), con un archivo que controla las sesiones y rutas de acceso (authStore y router index), donde también se utiliza la URI (/api/me) y su funcionamiento es detallado en el siguiente punto.

Figura 41

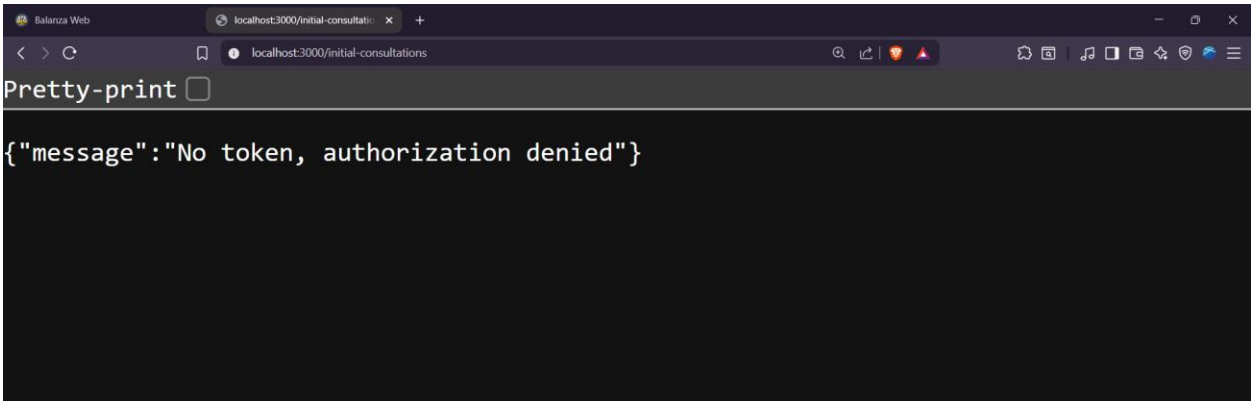
Acceso a una vista sin sesión válida (Frontend)



Nota: La figura muestra el funcionamiento de cómo se comporta la aplicación al ingresar rutas no autorizadas, donde el usuario no tiene una sesión válida y se visualiza claramente el error 401, que indica la no autorización del acceso a ese contenido. Fuente: Elaboración Propia.

Figura 42

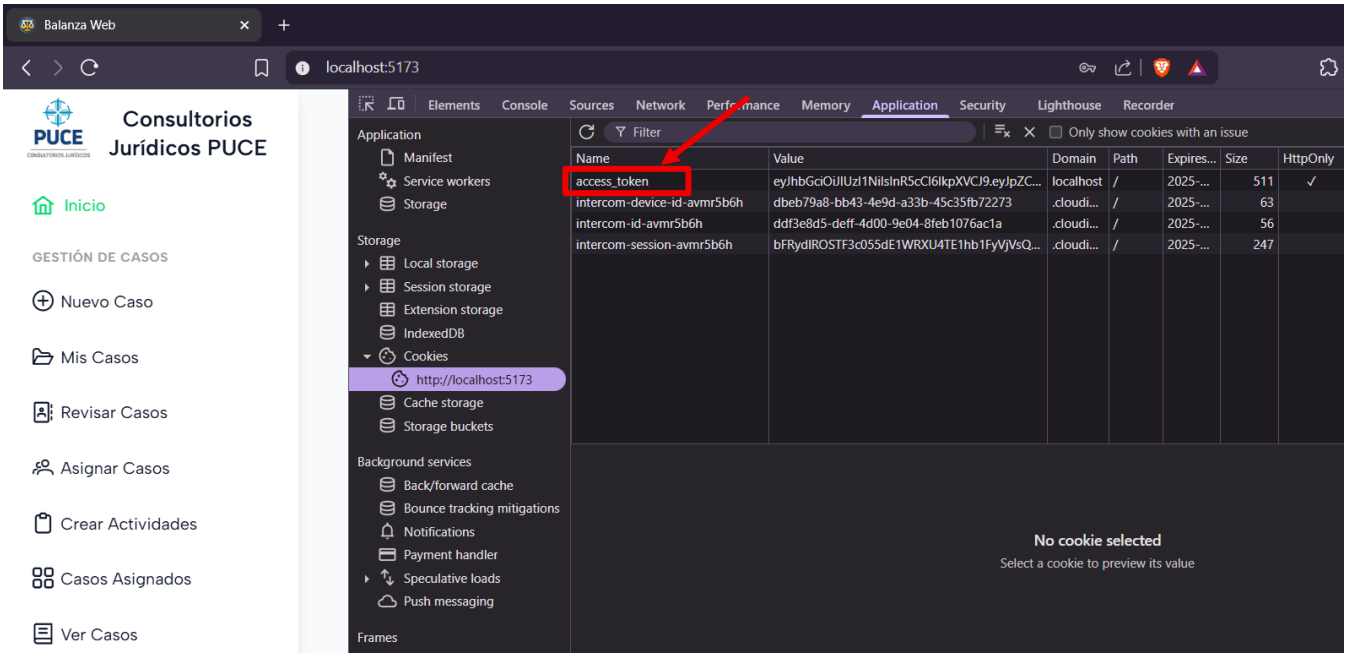
Acceso a una ruta API REST sin sesión válida



Nota: La figura muestra el funcionamiento de cómo se comporta el servidor backend y se visualiza claramente el error, que indica la no autorización del acceso a ese contenido debido a que no hay un token. Fuente: Elaboración Propia.

Figura 43

Creación del token JWT y almacenamiento en Cookies (access_token)



Nota: La figura muestra el funcionamiento del proceso de creación y almacenamiento del token JWT en cookies, que se ve finalmente reflejado en el navegador del cliente, siendo creado porque el usuario fue autenticado correctamente. Fuente: Elaboración Propia.

4.4.3 Control de Acceso en el Frontend (Pinia)

Una vez el usuario ha sido autenticado, es necesaria la forma de saber en el lado del cliente como gestionar la información de la carga útil que llega del token almacenado en la cookie, para ello se utiliza el manejador de estados "Pinia", cuya función permite controlar la sesión del usuario en el frontend y darle la autorización para ver las vistas generales de un usuario que ha sido autenticado. No obstante, la seguridad no se limita solo a proteger las contraseñas de los usuarios y su sesión, también es necesario poner ciertos permisos a las vistas de la aplicación web, ya que el hecho de estar autenticado no significa que ese usuario automáticamente debe poder hacer "todo" dentro de la aplicación web.

Visto de esta forma, se controla las vistas que pueden acceder los usuarios con una respectiva autorización que se gestiona por medio de los permisos de cada rol que tiene implícitamente el usuario, es decir, un rol de tipo "Estudiante" no puede tener los mismos que un "Administrador" por ejemplo. De esa forma, para implementar esta capa final de seguridad, primero es necesaria una tabla donde se almacene los permisos que cada rol puede tener sobre las vistas (punto ya mencionado a detalle en el desarrollo) y de ese modo, dentro de la gestión de estados (auth.ts), se mantiene el estado de la sesión del usuario en toda la aplicación web, permitiendo guardar de forma segura los datos de la sesión, y mapeando los permisos de visualización que fueron asignados a cada rol por medio de la URI (/profile/auth/:ID).

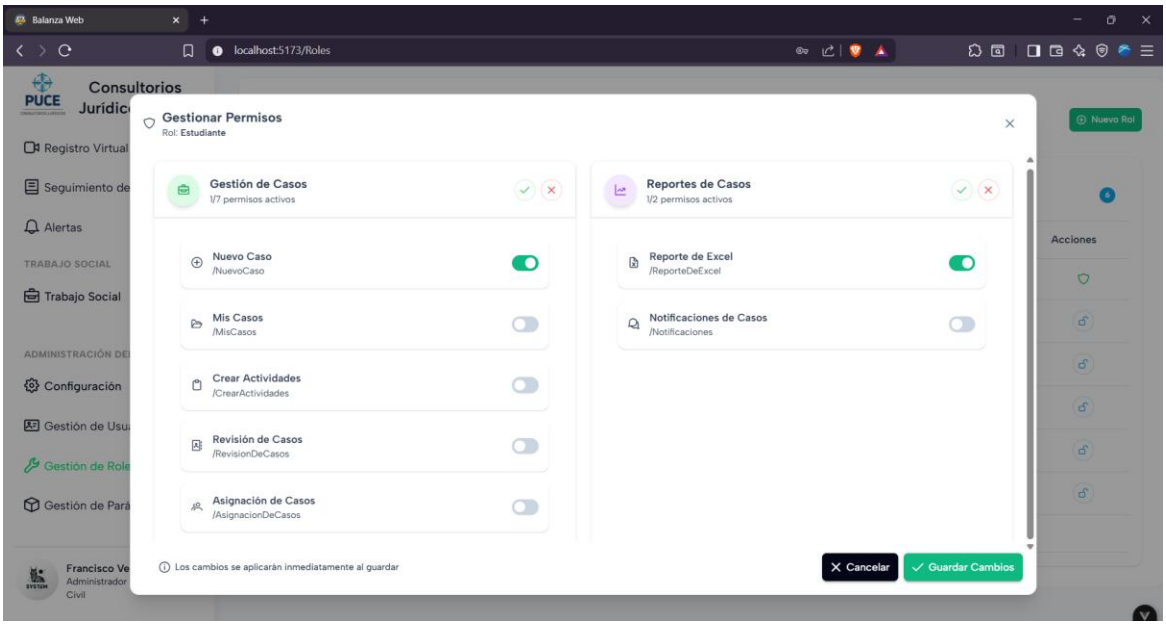
Por lo tanto, dentro de la definición del estado de la sesión "authStore", se tiene un getter llamado (canView), que permite controlar si el usuario puede o no ver la vista en concordancia con

sus permisos; cabe destacar que ya se ha detallado su funcionamiento con mayor profundidad en la sección "*4.3.2*", además de que su uso se ve reflejado en el menú de la aplicación web, la cual hace uso de dicho getter para ocultar las vistas a las que el usuario no tiene acceso por defecto.

Finalmente, utilizando el componente que permite manejar la definición de las rutas con (vue-router), se utiliza un mecanismo que permite la protección de rutas con autenticación y permisos, llamado "Router Guard", donde se protegen las rutas no autorizadas verificando la sesión (si no la tiene, se redirige a la interfaz de login), por otra parte, se verifica los permisos del rol al que pertenece el usuario, y en caso de que la ruta requiera un permiso para ser accedida y si el usuario no tiene dicho permiso, se le redirige a la página de inicio.

Figura 44

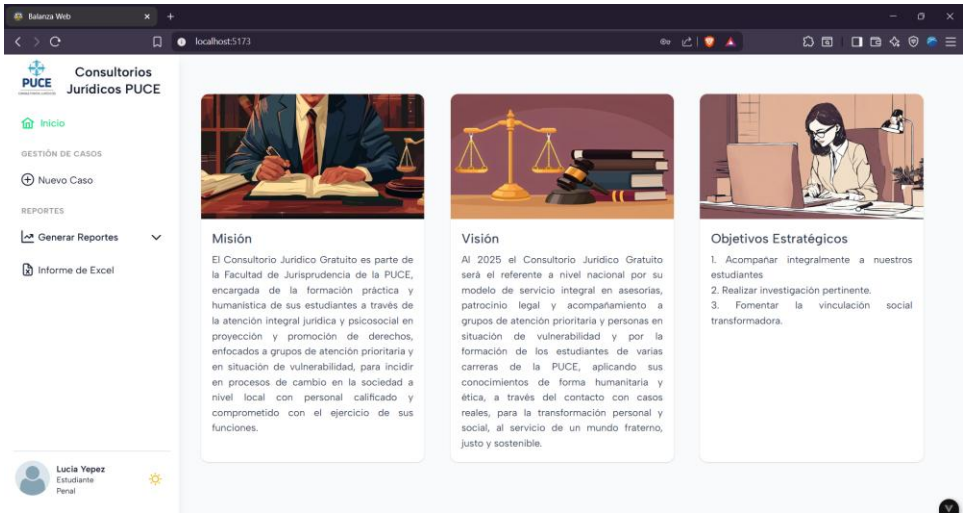
Prueba de permisos al rol Estudiante (Vista de Gestión)



Nota: La figura muestra la gestión de permisos para el rol "Estudiante", donde en el ejemplo se visualiza que el rol tendrá dos permisos de vista asignados (Nuevo Caso y Reporte de Excel). Fuente: Elaboración Propia.

Figura 45

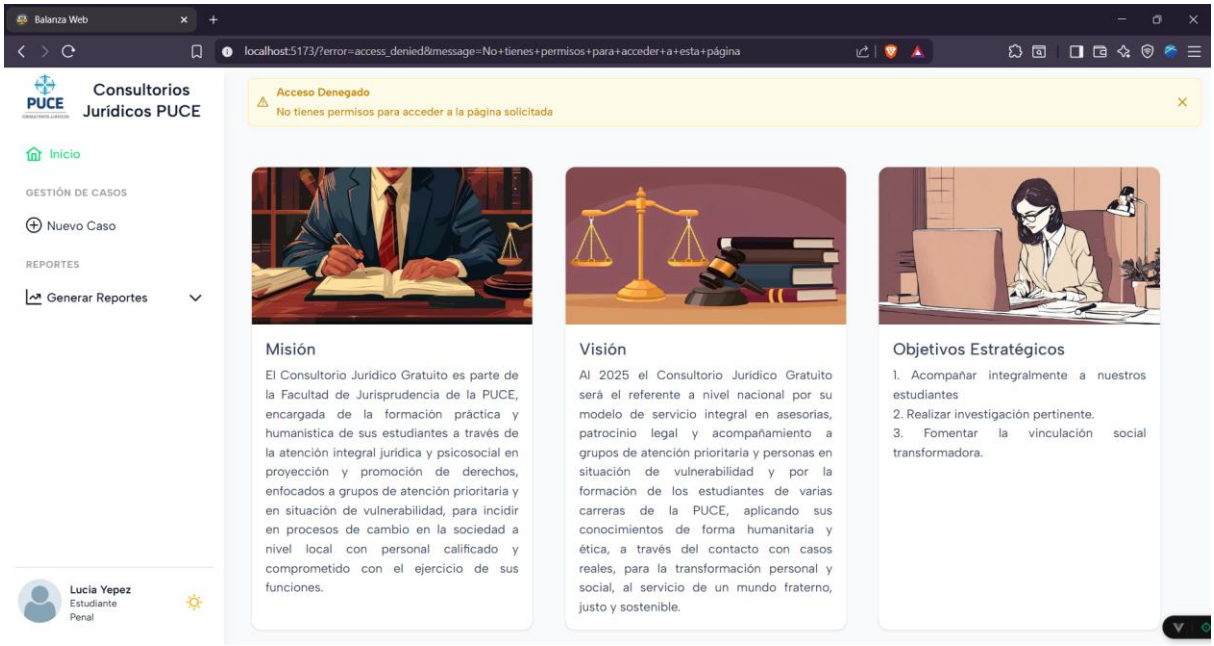
Prueba de permisos al rol Estudiante (Modificación del Menú)



Nota: La figura muestra la gestión de permisos para el rol "Estudiante", donde en el ejemplo se visualiza un usuario que ha iniciado sesión y se han cargado únicamente las dos vistas las cuales tiene acceso el rol. Elaboración Propia.

Figura 46

Prueba de permisos al rol Estudiante (Acceso Denegado)



Nota: La figura muestra la gestión de permisos para el rol “Estudiante”, donde en el ejemplo se había escrito en la barra de navegación una ruta la cual el rol no tiene permisos, redirigiendo al usuario a la interfaz de Inicio e indicando el respectivo error. Elaboración Propia.

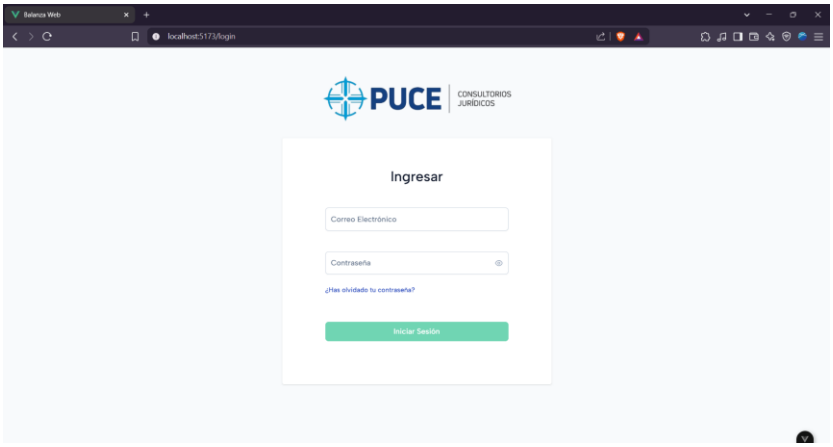
4.5 Aplicación de la Metodología (Prototipado Evolutivo)

4.5.1 Construcción del Prototipo Inicial

En concordancia con la metodología del prototipado evolutivo, se desarrolló una primera versión funcional de la Gestión de Usuarios y Primeras Consultas, destacando al inicio de sesión, olvidé mi contraseña, la interfaz de control de usuarios en donde se puede realizar las operaciones de crear, ver y editar la información del usuario del sistema. Por otra parte, en primeras consultas está el registro de un nuevo usuario externo, junto con su evidencia (en caso de ser discapacitado) y los datos de la consulta inicial. Al ser un primer prototipo, no se contempla todavía el resto de las funcionalidades descritas, incluyendo a la Gestión de Roles/Permisos; cabe señalar también que, al ser un trabajo colaborativo, también se observa en las figuras a continuación las vistas de los otros módulos del sistema.

Figura 47

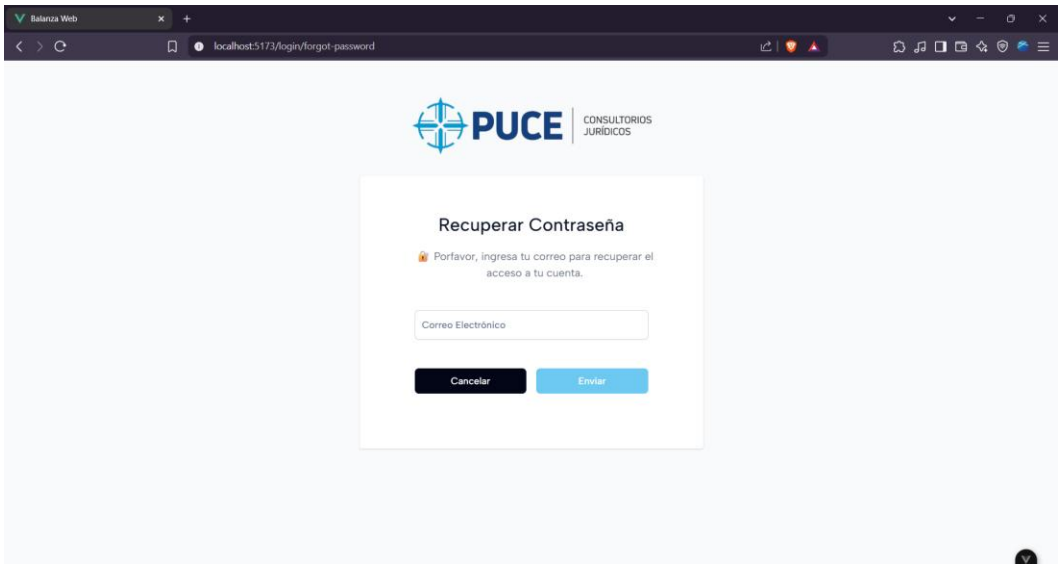
Prototipo Inicial: Inicio de sesión



Nota: La figura muestra el prototipo inicial del inicio de sesión de los usuarios. Fuente: Elaboración Propia.

Figura 48

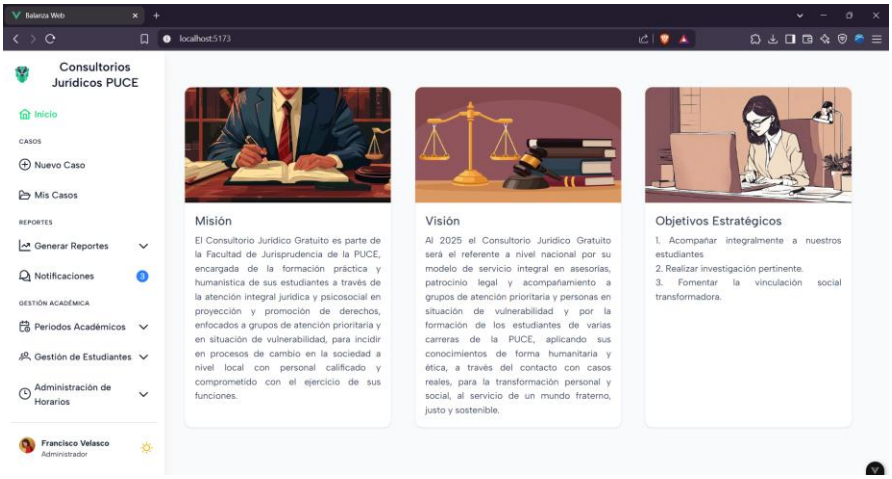
Prototipo Inicial: Olvidé mi contraseña



Nota: La figura muestra el prototipo inicial de la página recuperación de contraseña. Fuente: Elaboración Propia.

Figura 49

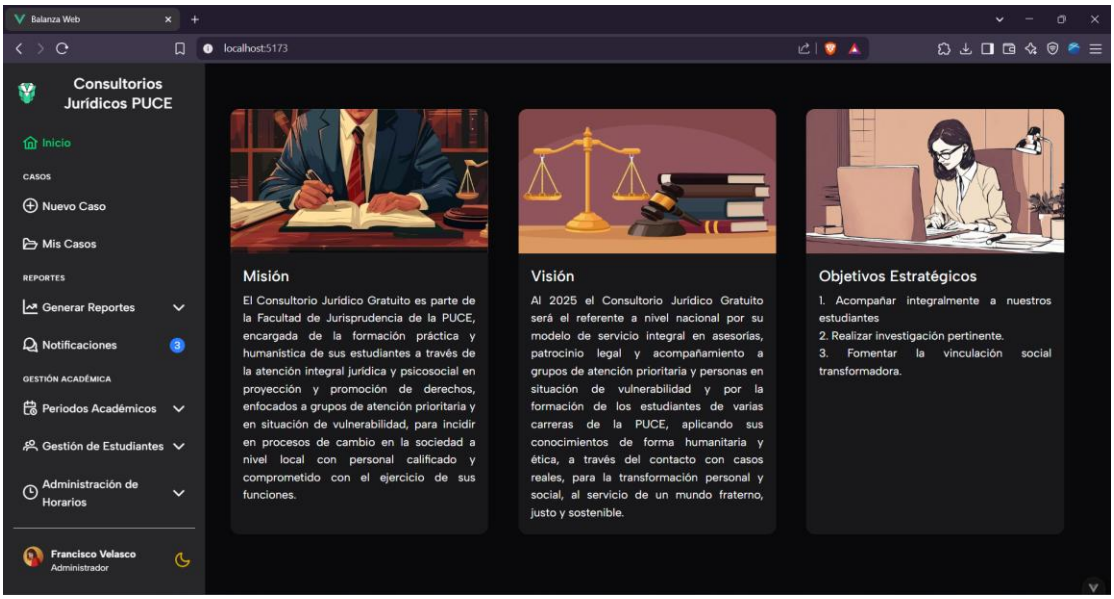
Prototipo Inicial: Página de inicio



Nota: La figura muestra el prototipo inicial de la página de inicio de la aplicación web. Fuente: Elaboración Propia.

Figura 50

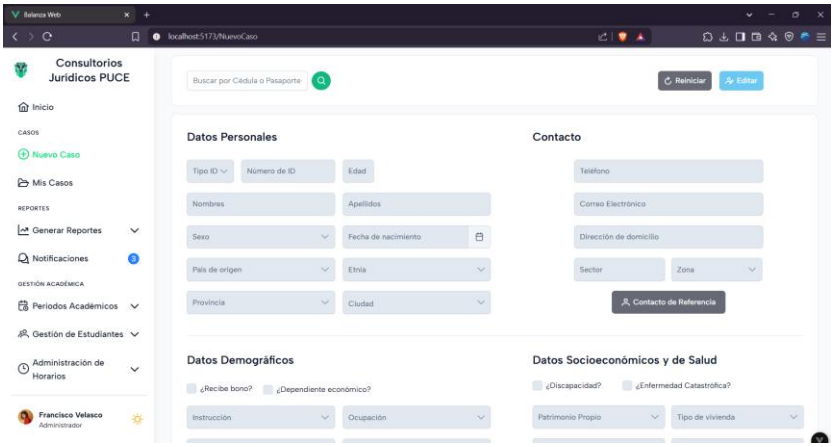
Prototipo Inicial: Página de inicio (Modo oscuro)

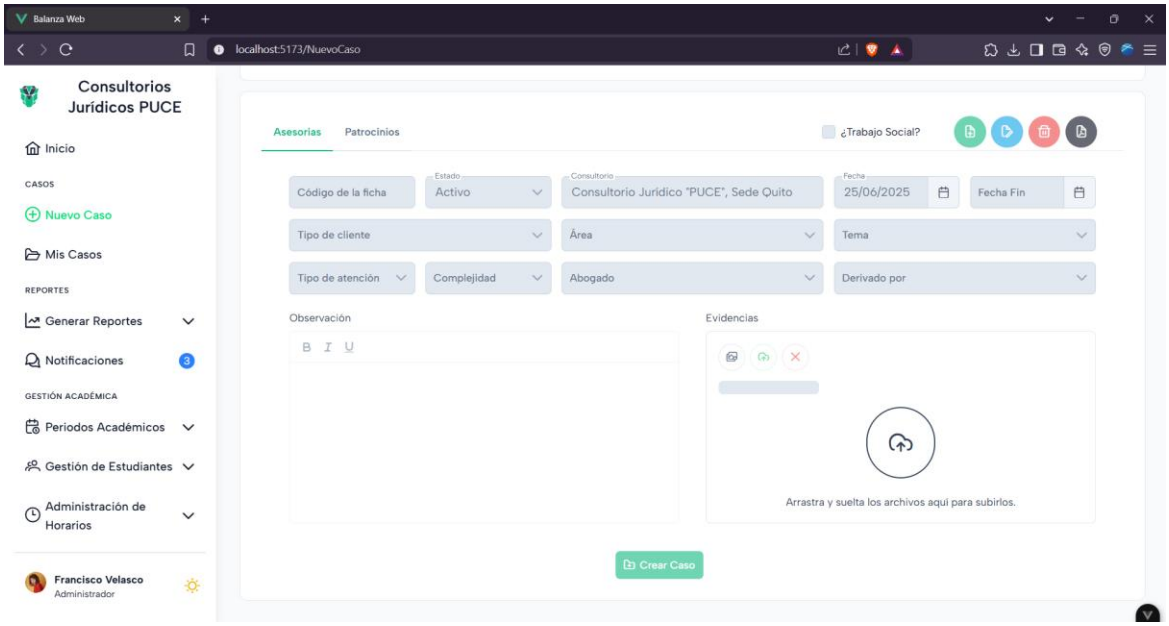


Nota: La figura muestra el prototipo inicial de la página de inicio de la aplicación web en modo oscuro. Fuente: Elaboración Propia.

Figura 51

Prototipo Inicial: Nuevo Caso (Primeras Consultas)

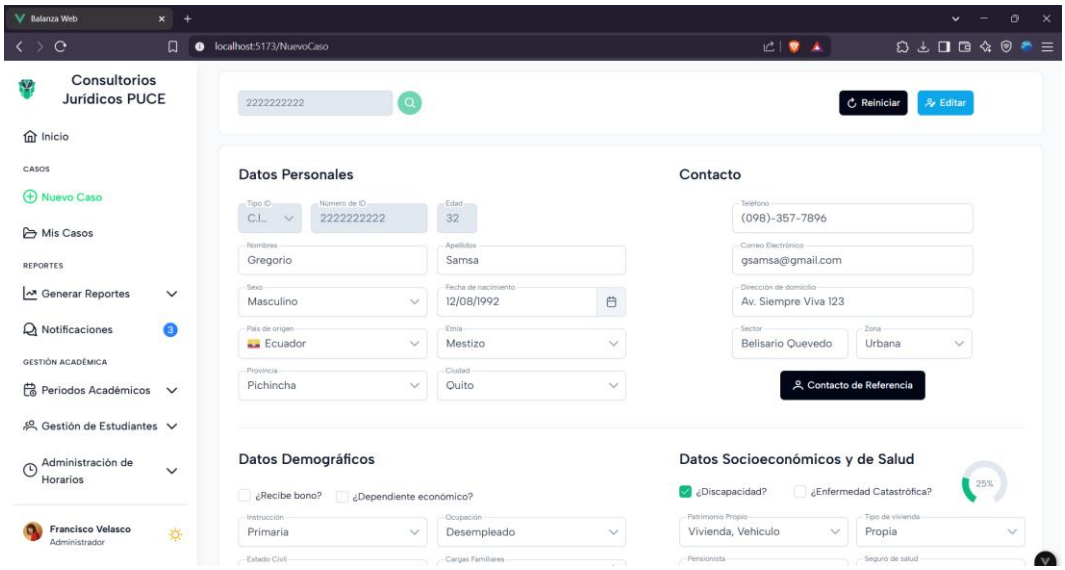




Nota: La figura muestra el prototipo inicial de la página de nuevo caso, donde se hace la gestión de primeras consultas. Fuente: Elaboración Propia.

Figura 52

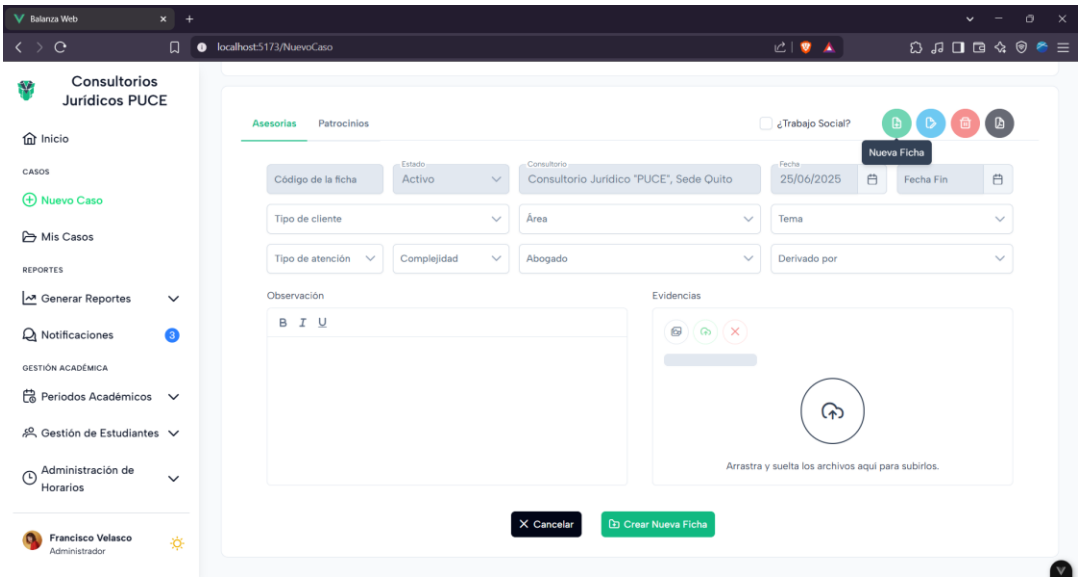
Prototipo Inicial: Usuario ya registrado (Primeras Consultas)



Nota: La figura muestra el prototipo inicial de un usuario externo que ya está registrado en primeras consultas. Fuente: Elaboración Propia.

Figura 53

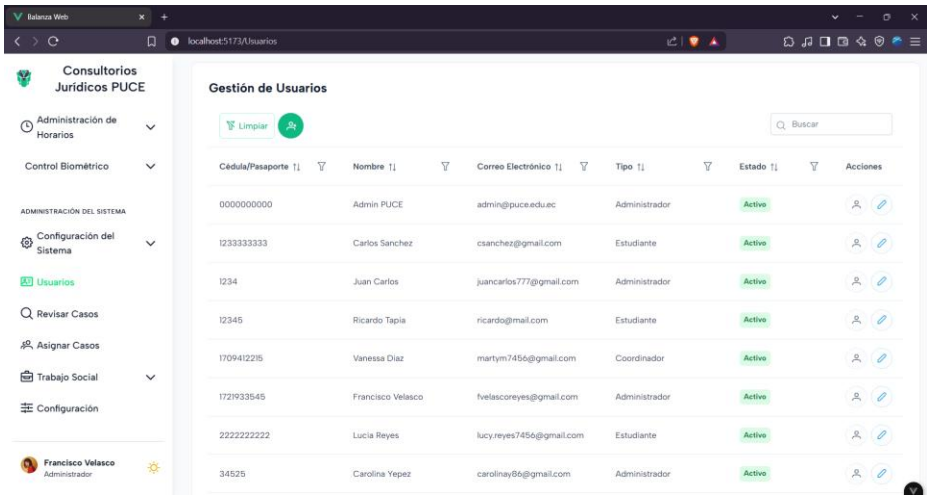
Prototipo Inicial: Nuevo Ficha de Atención (Primeras Consultas)



Nota: La figura muestra el prototipo inicial de la creación de una nueva ficha de atención para los usuarios que ya existen. Fuente: Elaboración Propia.

Figura 54

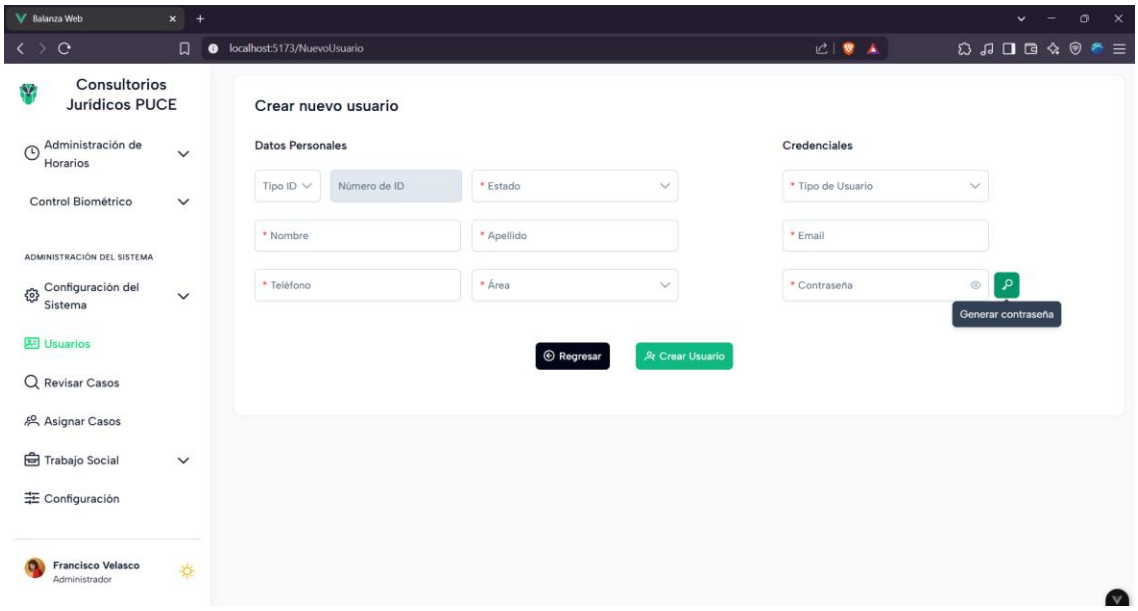
Prototipo Inicial: Gestión de Usuarios



Nota: La figura muestra el prototipo inicial de la página de gestión de usuarios, donde se puede ver la lista de todos los usuarios registrados en el sistema junto con sus datos principales y la búsqueda general con sus parámetros individuales. Fuente: Elaboración Propia.

Figura 55

Prototipo Inicial: Gestión de Usuarios (Nuevo Usuario)



Nota: La figura muestra el prototipo inicial de la página de creación de un nuevo usuario, donde se puede destacar la selección del tipo de ID que se desea elegir (cédula o pasaporte), además de un botón que permite generar una contraseña automática al nuevo usuario. Fuente: Elaboración Propia.

4.5.2 Validación y Retroalimentación con el Usuario

Como un proceso esencial dentro de la metodología, se procedió a llevar a cabo cinco reuniones con la Dra. Camila Cedeño (Coordinadora General) en calidad de usuario. Durante sesiones, se presentó el primer prototipo, recopilando retroalimentación esencial que permitía obtener una corrección de las funcionalidades y generar nuevos requerimientos que guiaron el

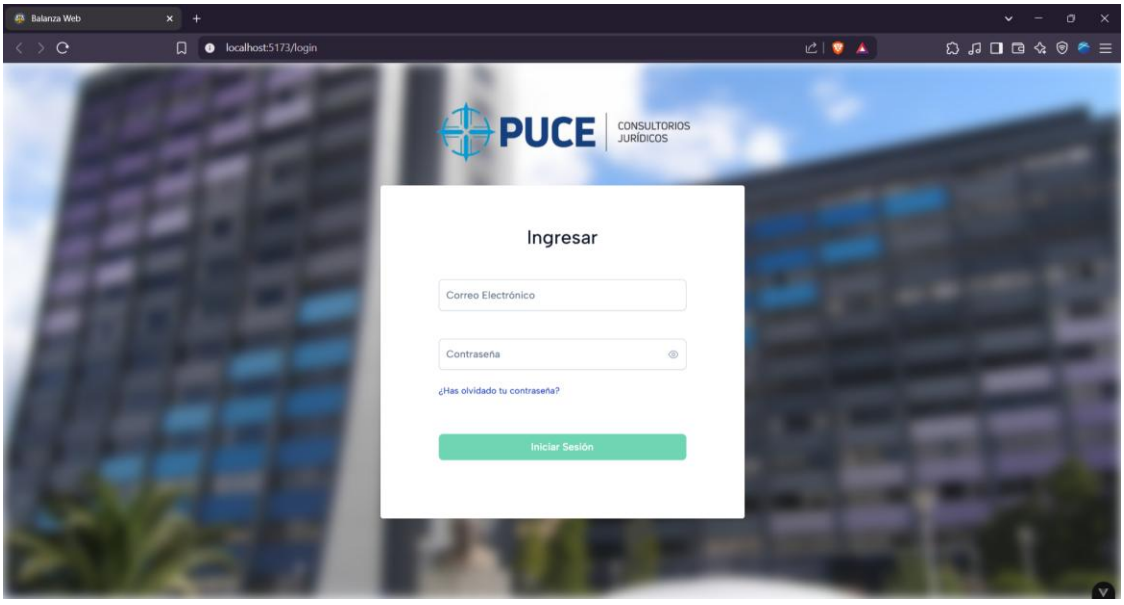
desarrollo el proyecto (haciendo iteraciones). En esencia, lo más destacable fue las correcciones que eran necesarias en primeras consultas, eliminando varios campos que ya no se consideraban necesarios y reestructurando bastante contenido, cambio en nombres del menú y, de manera importante, la implementación de notificaciones por correo electrónico sobre la gestión de usuarios (cuando el usuario es creado) y en primeras consultas (cuando se rechaza el patrocinio).

4.5.3 Prototipo Final de la Aplicación

Luego de varias reuniones con el usuario y la Dirección de Informática de la PUCE, se procedió a mejorar continuamente el prototipo gracias a la retroalimentación, haciendo que se tengan varias iteraciones como se había mencionado en el diseño y desarrollo. El resultado final, fue un prototipo aprobado por la Coordinadora General, el cual cumplía con todos los requerimientos funcionales y de usabilidad solicitados del módulo propuesto en este trabajo.

Figura 56

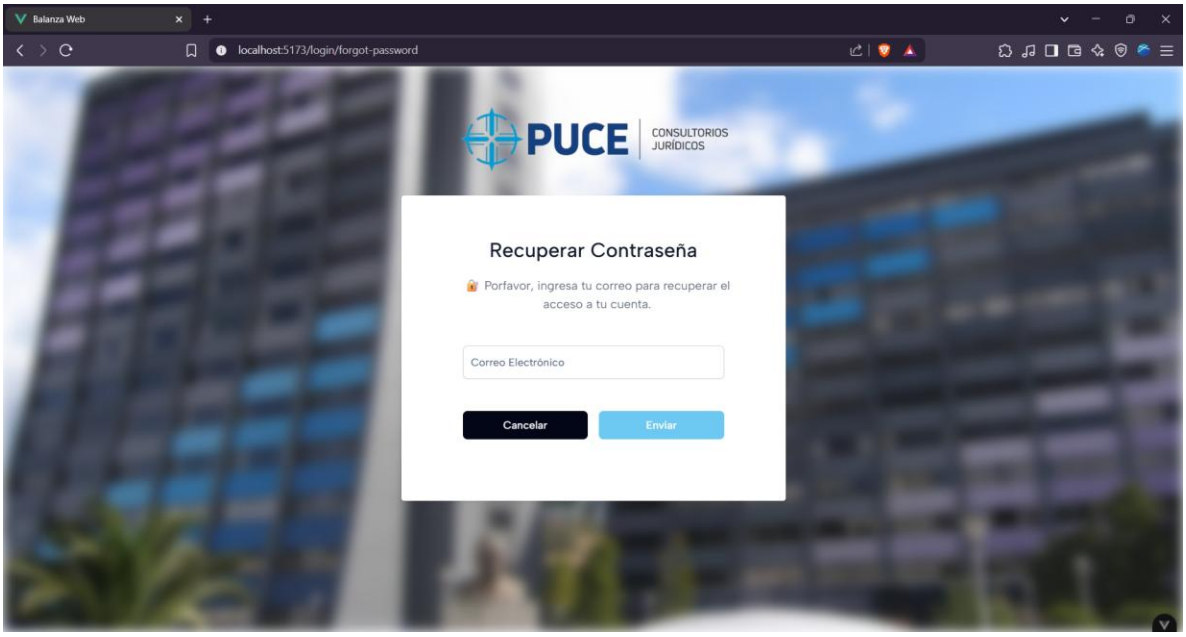
Prototipo Final: Inicio de sesión



Nota: La figura muestra el prototipo final del inicio de sesión de los usuarios. Fuente: Elaboración Propia.

Figura 57

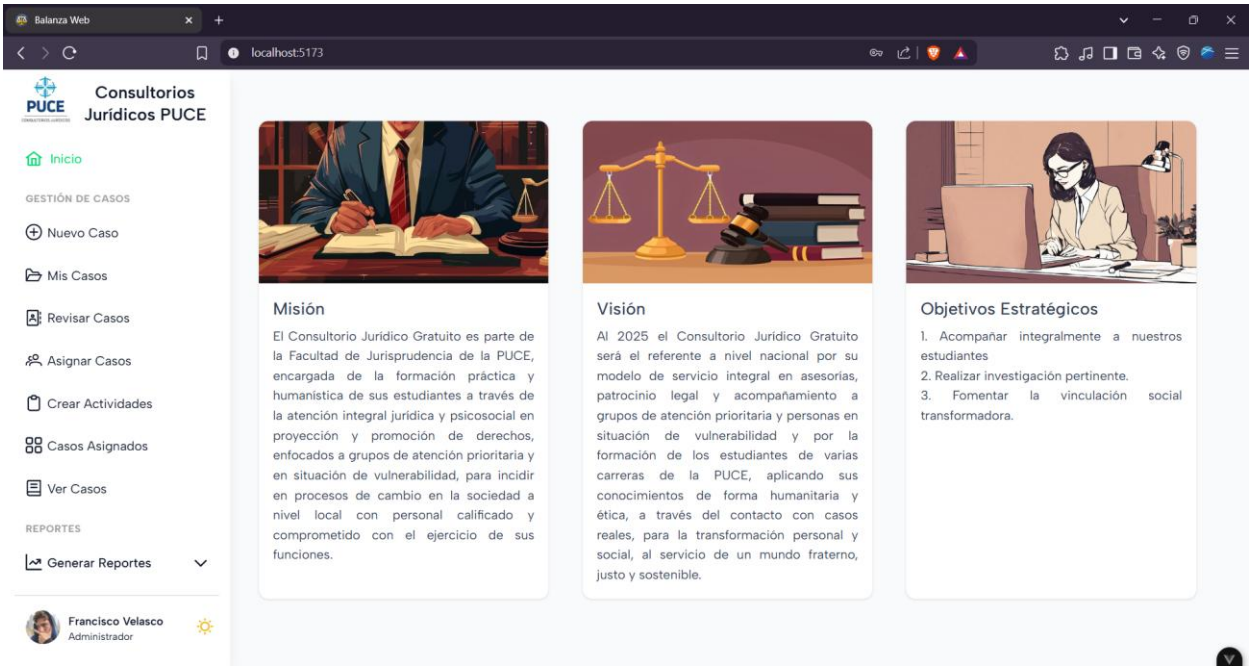
Prototipo Final: Olvidé mi contraseña



Nota: La figura muestra el prototipo final de olvidé mi contraseña. Fuente: Elaboración Propia.

Figura 58

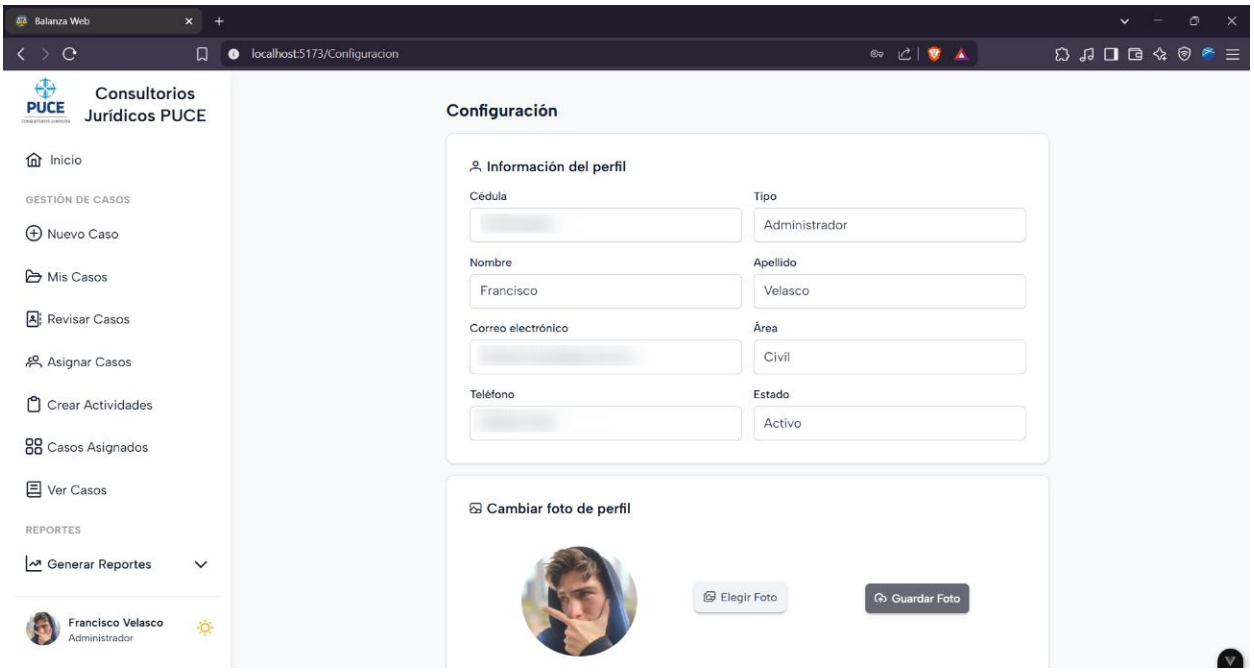
Prototipo Final: Página de inicio y Menú lateral



Nota: La figura muestra el prototipo final de la página de inicio, junto con el menú de la aplicación web, dividido por secciones de contenido más claras. Fuente: Elaboración Propia.

Figura 59

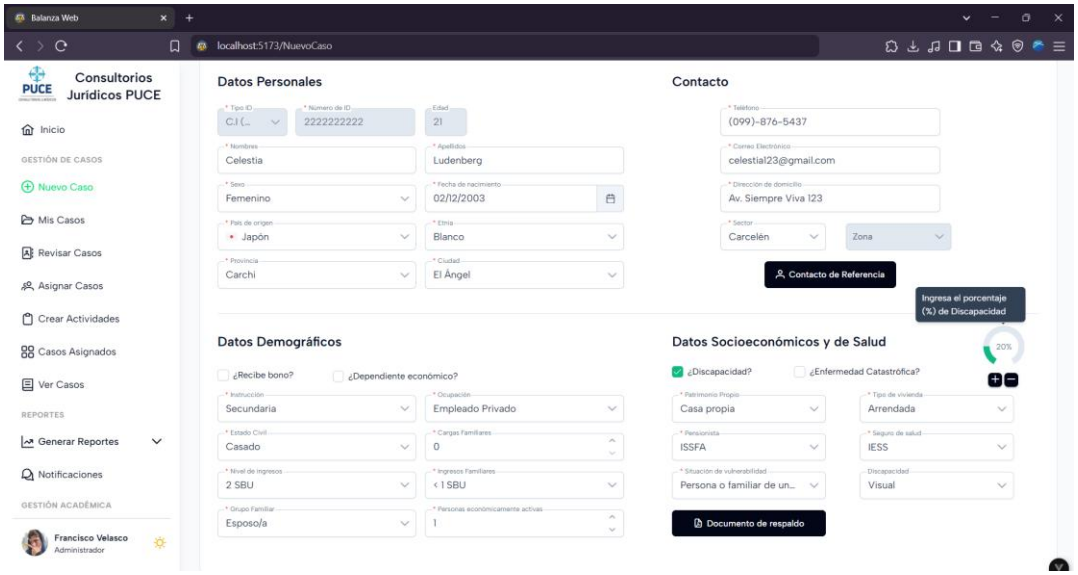
Prototipo Final: Configuración del Usuario



Nota: La figura muestra el prototipo final de la configuración del usuario. Fuente: Elaboración Propia.

Figura 60

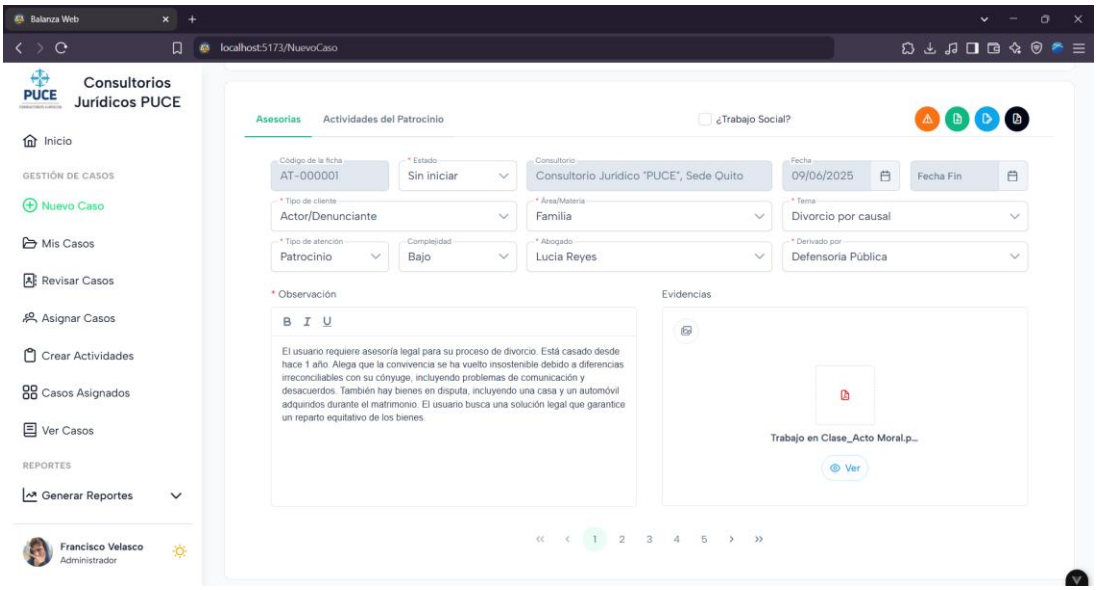
Prototipo Final: Usuario ya registrado (Primeras Consultas)



Nota: La figura muestra el prototipo final de un usuario externo que ya está registrado en primeras consultas, donde se observa la reestructuración de varios campos y la adición de validaciones. Fuente: Elaboración Propia.

Figura 61

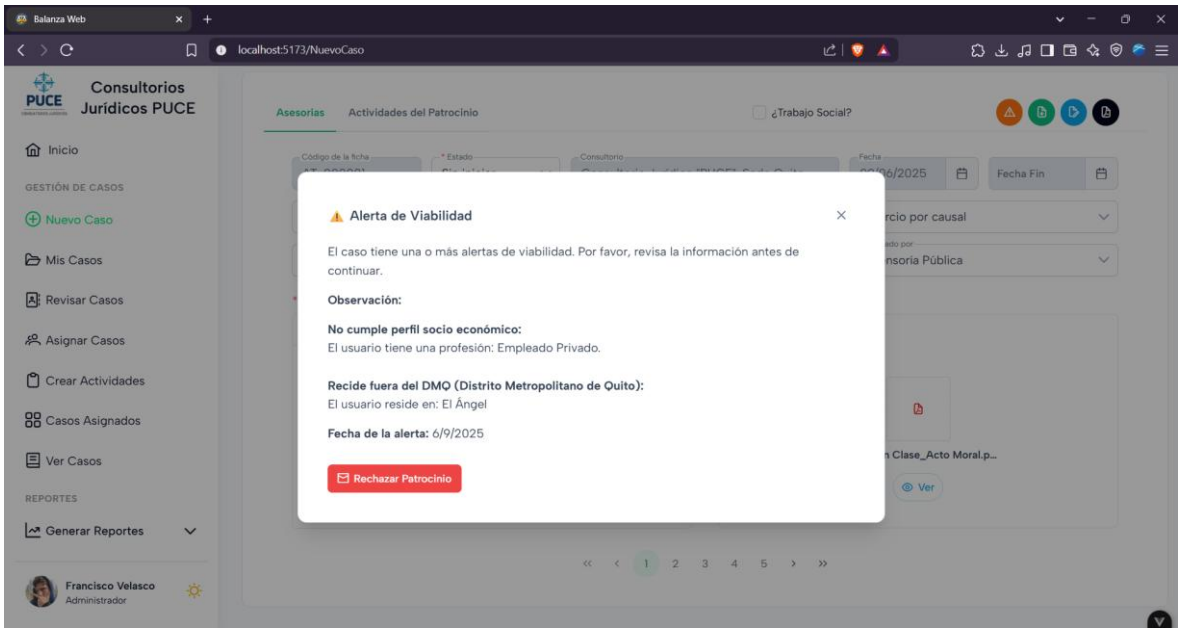
Prototipo Final: Fichas de Atención (Primeras Consultas)



Nota: La figura muestra el prototipo final de un usuario externo que ya está registrado en primeras consultas, donde se observa la paginación que permite ver todas sus consultas de atención en los consultorios jurídicos. Fuente: Elaboración Propia.

Figura 62

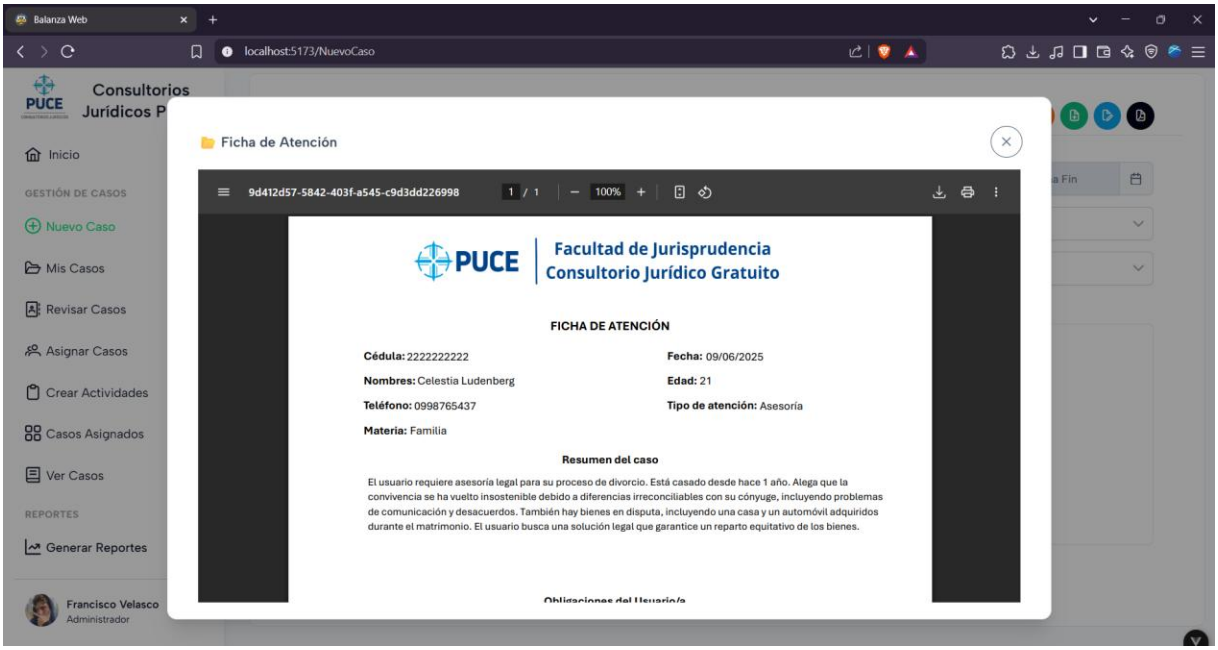
Prototipo Final: Alertas de viabilidad (Primeras Consultas)



Nota: La figura muestra el prototipo final de la alerta de viabilidad que aparecerá cada vez que se crea un caso y se detecta algún valor que la activa, en este ejemplo el usuario reside fuera del DMQ. Fuente: Elaboración Propia.

Figura 63

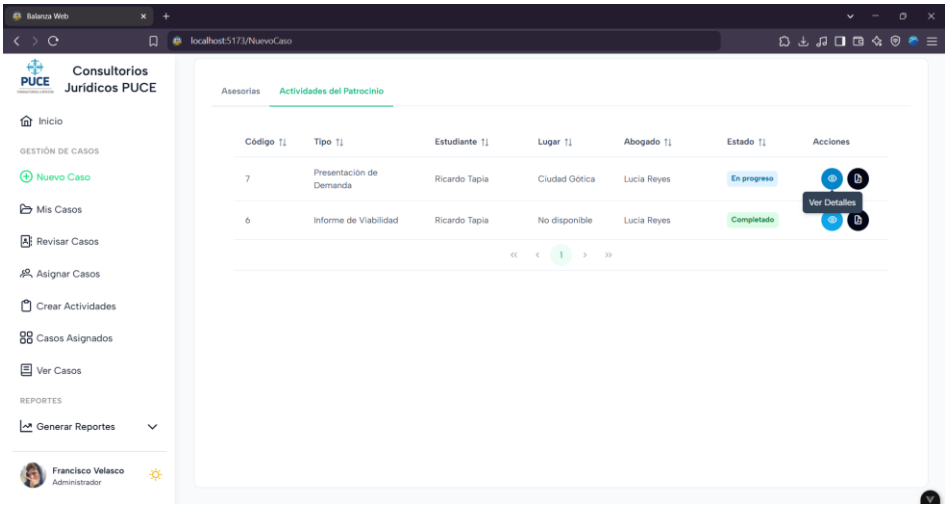
Prototipo Final: Ficha de Atención (Primeras Consultas)



Nota: La figura muestra el prototipo final de la generación de la ficha de atención que se puede exportar una vez se haya registrado la primera consulta o una nueva atención. Fuente: Elaboración Propia.

Figura 64

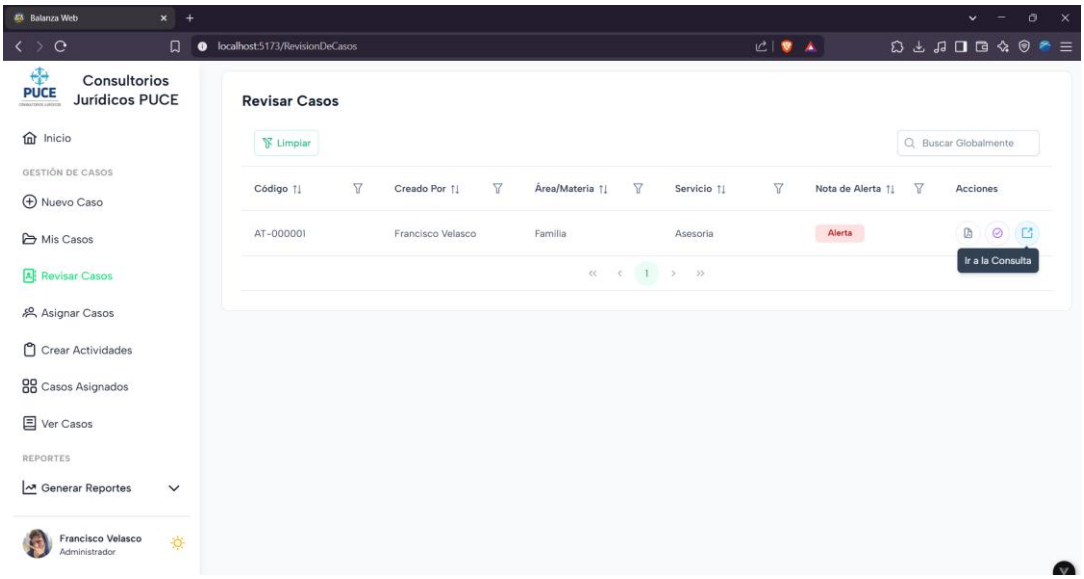
Prototipo Final: Actividades del Patrocinio



Nota: La figura muestra el prototipo final de las actividades de un patrocinio de un caso en particular, donde se puede ver los detalles y la evidencia de la actividad realizada por el estudiante. Fuente: Elaboración Propia.

Figura 65

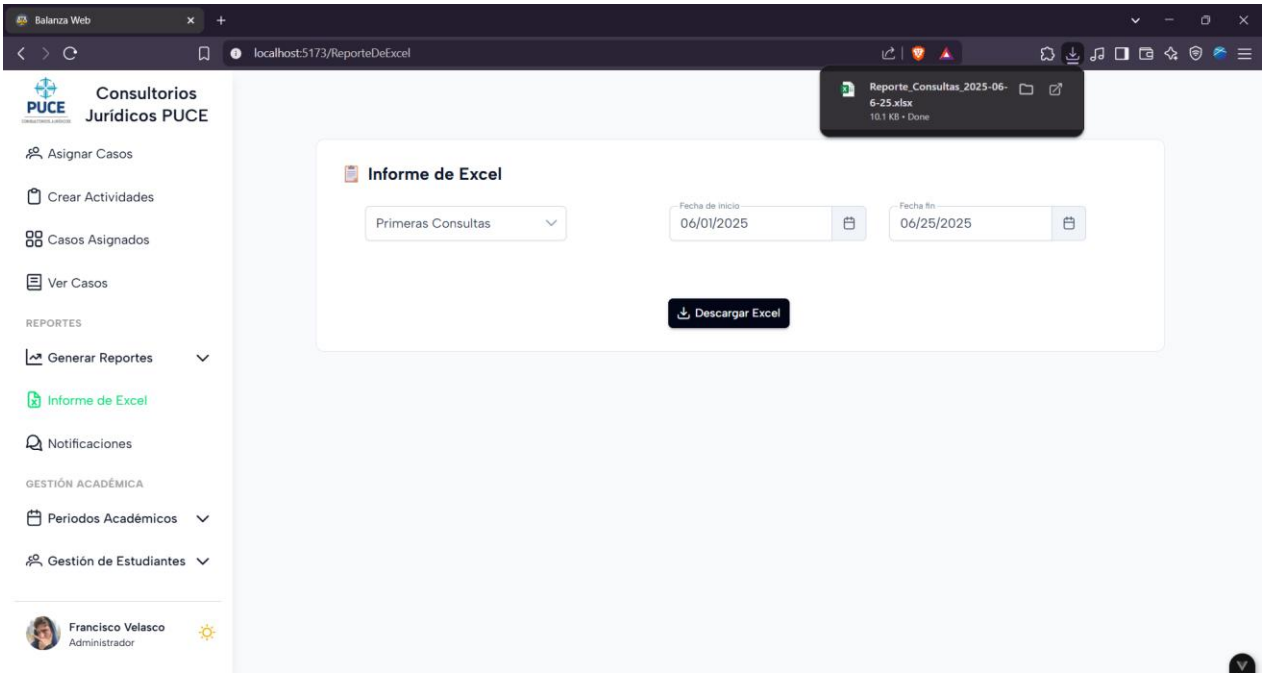
Prototipo Final: Revisión de Casos



Nota: La figura muestra el prototipo final de la interfaz que usa el coordinador para poder revisar los casos que lleguen de los estudiantes que los han registrado desde primeras consultas. Fuente: Elaboración Propia.

Figura 66

Prototipo Final: Reporte de Excel (Primeras Consultas)



Nota: La figura muestra el prototipo final de los reportes en Excel de Primeras Consultas Fuente: Elaboración Propia.

Figura 67

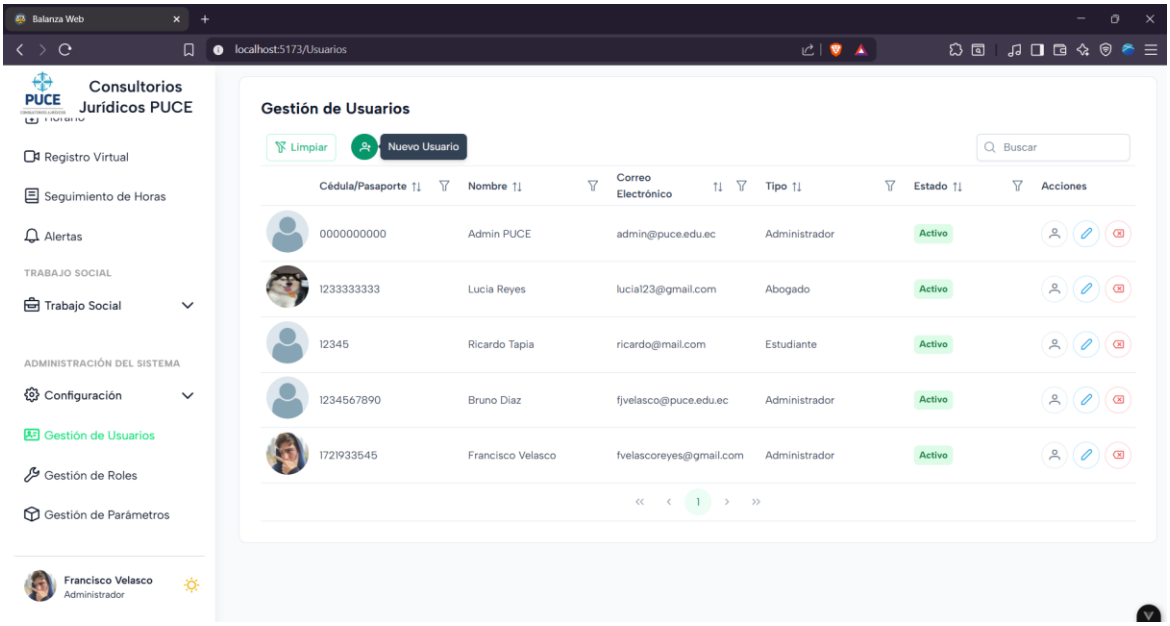
Prototipo Final: Reporte de Excel (Ejemplo de Prueba)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Tipo ID	Número ID	Nombres	Apellidos	Edad	Género	Fecha Nacimien	Nacionalidad	Etnia	Provincia	Ciudad	Teléfono	Correo Electrónico
2	DATOS PERSONALES DEL USUARIO												
3	cedula	2222222222	Vanessa	Diaz	25	Femenino	1/2/2000	Ecuador	Mestiza/o	Pichincha	Sangolquí	0998765432	vanessadiaz@gmail.com
4	cedula	1233333333	Ricardo	Rodriguez	32	Masculino	5/3/1992	Ecuador	Mestiza/o	Pichincha	Quito	0983655784	carlossanchez23@gmail.com
5													
6													

Nota: La figura muestra el encabezado y datos de prueba de la generación de reportes de Primeras Consultas. Fuente: Elaboración Propia.

Figura 68

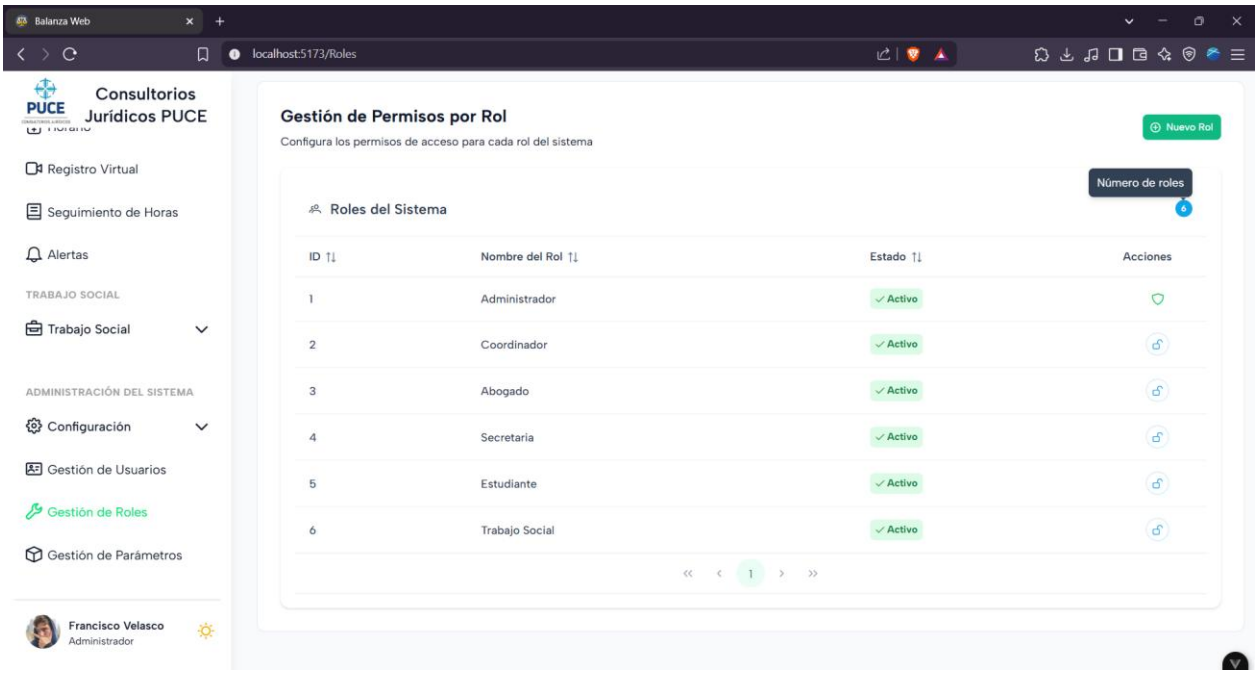
Prototipo Final: Gestión de Usuarios



Nota: La figura muestra el prototipo final de la Gestión de Usuarios Fuente: Elaboración Propia.

Figura 69

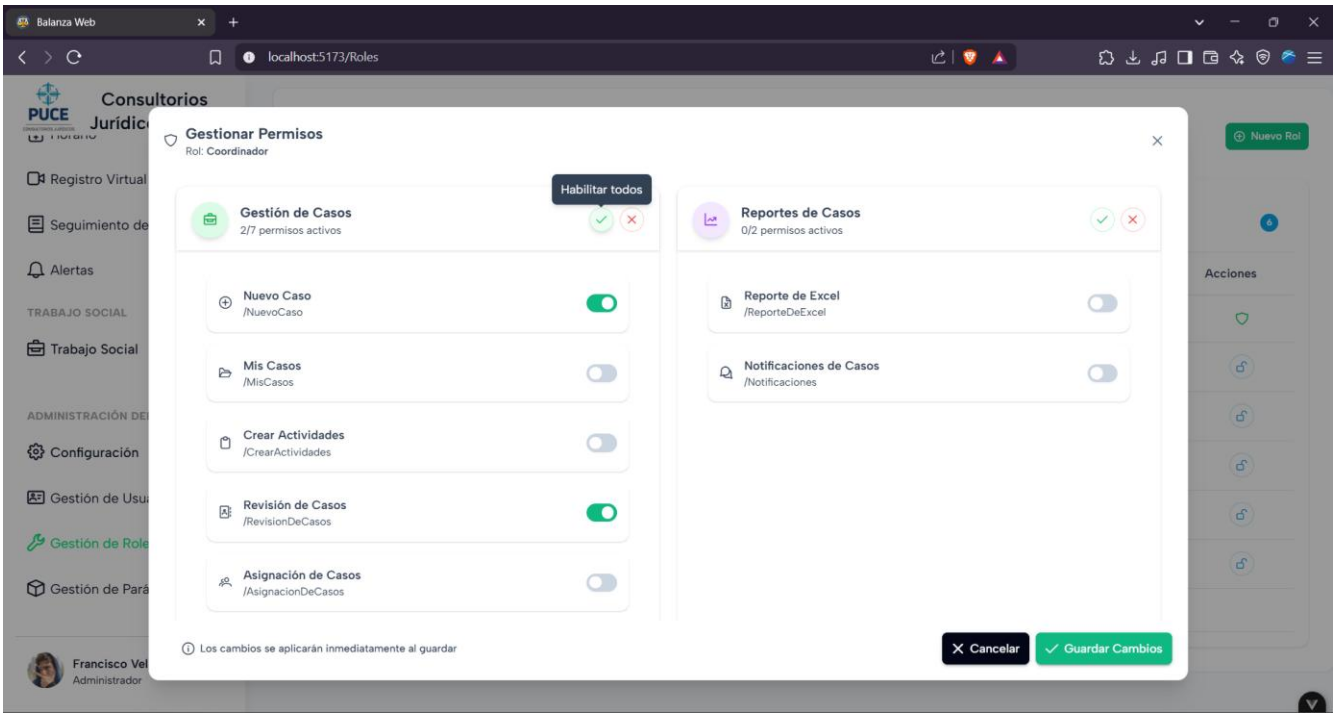
Prototipo Final: Gestión de Roles/Permisos



Nota: La figura muestra el prototipo final de la Gestión de Roles y Permisos. Fuente: Elaboración Propia.

Figura 70

Prototipo Final: Gestión de Roles/Permisos (Activar o Desactivar)



Nota: La figura muestra el prototipo final de los permisos que puede tener un rol, siendo divididos por categorías en concordancia con el menú de la aplicación web. Fuente: Elaboración Propia.

CAPÍTULO V: PRUEBAS DE ACEPTACIÓN DEL USUARIO (UAT)

De acuerdo con el seguimiento de la metodología, del Prototipado Evolutivo, se llevaron a cabo tres sesiones de pruebas de aceptación de usuario con la coordinadora del área, lo que permitió refinar el producto pasando por iteraciones en las fases hasta obtener su validación y aprobación final del módulo de Gestión de Usuarios y Primeras Consultas desarrollado en este trabajo.

Tabla 10

Pruebas de Aceptación del Módulo

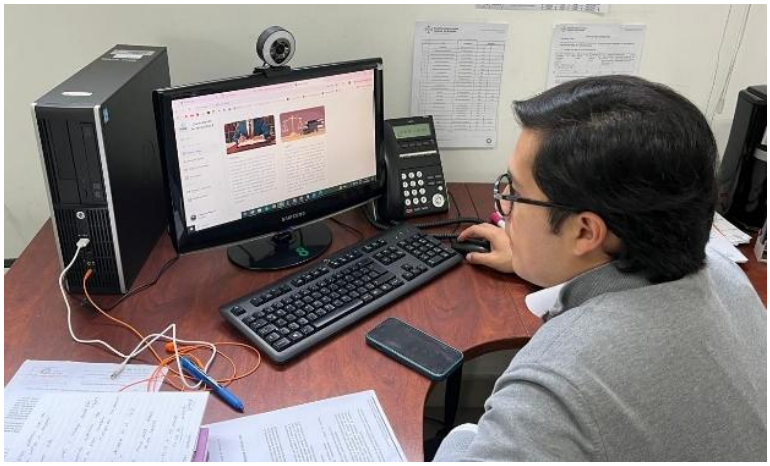
N	Funcionalidad Evaluada	Caso de Uso Asociado	Descripción de la Prueba	Resultado Esperado	Resultado Obtenido / Observaciones
1	Inicio de Sesión	Iniciar Sesión (Login)	El usuario ingresa su correo y contraseña válidos e ingresa a la página de inicio en la aplicación.	Ingreso exitoso a la aplicación.	Aprobado
2	Cambiar Foto de perfil	Configurar Perfil	El usuario ingresa a la configuración y cambia su foto de perfil.	Foto de perfil del usuario actualizada correctamente.	Aprobado
3	Nuevo Usuario del Sistema	F1.1 Insertar Usuario	El usuario ingresa los datos requeridos del registro y guarda la información.	Nuevo usuario creado exitosamente.	Aprobado
4	Modificar Usuario del Sistema	F1.2 Modificar Usuario	El usuario actualiza un campo de información y guarda los cambios.	Usuario actualizado exitosamente.	Aprobado
5	Eliminar Usuario del Sistema	F1.3 Eliminar Usuario	El usuario acepta el cuadro de dialogo de eliminación del usuario.	Usuario desactivado exitosamente.	Aprobado
6	Consultar Usuario del Sistema	F1.4.1 Consulta General	El usuario escribe en el cuadro de búsqueda cualquier información que desee encontrar.	Usuario filtrado según lo escrito en la búsqueda.	Aprobado

7	Registro de la Primera Consulta	F1.3.1.1 Registrar Usuario (Externo)	El usuario llena el formulario de registro del cliente y los datos de 8su primera ficha de atención.	Usuario externo y primera ficha de atención creados exitosamente.	Aprobado
8	Registro de Nueva Ficha de Atención	F3.3.1 Registrar Ficha de Atención	El usuario llena el formulario de la nueva ficha de atención y guarda la información.	Nueva ficha de atención creada exitosamente.	Aprobado
9	Modificar Ficha de Atención	F3.3.2 Modificar Ficha de Atención	El usuario modifica los campos requeridos de la ficha y guarda los cambios.	Ficha de atención actualizada exitosamente.	Aprobado
10	Generar Ficha de Atención (PDF)	F3.3.3 Generar Ficha de Atención (PDF)	El usuario hace clic en la opción y visualiza el cuadro de dialogo de la ficha de atención.	Ficha de atención (PDF) cargada exitosamente.	Aprobado
11	Aprobar el Patrocinio del Caso	F3.4.1 Aprobar Patrocinio	El usuario edita la ficha de atención, selecciona en el servicio “Patrocinio” y el caso se elimina de la tabla de Revisión de Casos.	Caso aprobado a patrocinio exitosamente.	Aprobado
12	Consulta de Casos General	F3.5.1 Consulta General	El usuario escribe en el campo de “Búsqueda” y se filtrará cualquier caso que coincida con lo anterior.	Caso/s filtrados en la tabla general.	Aprobado
13	Consulta de Actividades del Caso	F3.6 Consultar Actividades del Caso	El usuario selecciona el apartado de “Actividades del caso” y visualiza la tabla con toda la información de cada actividad realizada.	Visualización de la tabla de actividades del caso.	Aprobado

Nota: Esta tabla describe las pruebas de aceptación del módulo propuesto. Fuente: Elaboración propia.

Figura 71

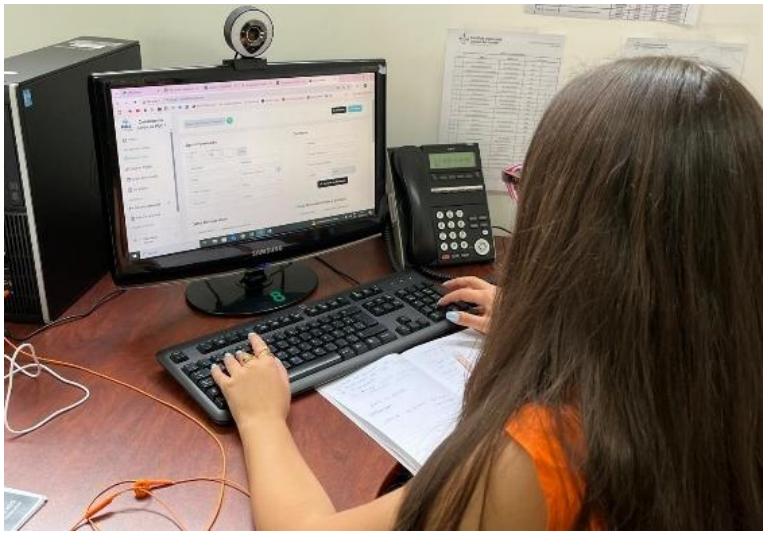
Pruebas de aceptación: Dr. Isaías



Nota: Esta figura muestra las pruebas realizadas por el Doctor Isaías en Consultorios Jurídicos PUCE. Fuente: Elaboración propia.

Figura 72

Pruebas de aceptación: Dra. Camila Cedeño



Nota: Esta figura muestra las pruebas de aceptación del módulo realizadas por la Doctora Camila Cedeño en Consultorios Jurídicos PUCE. Fuente: Elaboración propia.

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

- El módulo desarrollado mejora los procesos del área de Primeras Consultas, pues la aplicación brinda una gestión eficiente por medio de nuevas funcionalidades identificadas a través de reuniones en concordancia con la metodología (alertas de viabilidad, revisión de casos eficiente, ficha técnica PDF, reportes Excel). Esto ayuda a tener un control rápido e intuitivo respecto a las consultas por las que acuden los usuarios, mejorando significativamente la calidad del servicio y la coordinación entre las demás áreas del Consultorio Jurídico al ser el punto de partida clave de su proceso.
- El diseño y construcción de las interfaces de la aplicación web mejora considerablemente la experiencia del usuario (UX) llevándola a ser minimalista y fácil de usar.
- La información manejada del Consultorio Jurídicos requiere de un sistema de seguridades robusto, lo cual se implementó usando Json Web Tokens (JWT) para controlar la sesión de cada usuario, protegiendo efectivamente las sesiones y rutas del Backend, además de encriptar las contraseñas de los usuarios de forma irreversible con Bcrypt que es capaz de proteger contra ataques cibernéticos, y el control de permisos sobre cada vista utilizando Pinia, el cual bloquea los accesos en caso de no tener la autorización respectiva.
- Aplicar la metodología del prototipado evolutivo no solo es eficiente, sino también adaptable porque es un proceso iterativo de mejora continua. Sin duda alguna, la interacción con el usuario final brinda una retroalimentación que permite refinar cada aspecto de funcionalidad, ajuste en la lógica de negocio y nuevos requerimientos, asegurando que el producto efectivamente satisface las necesidades reales del usuario final.

- Los resultados obtenidos en las pruebas de aceptación de usuario muestran la satisfacción de una verdadera necesidad de cambio y mejora continua, siendo este el fin principal del desarrollo del módulo. Para lograr alcanzar esto, se simplifico la interfaz de usuario, al mismo tiempo que se incluyeron los nuevos requerimientos y necesidades del consultorio.

6.2 Recomendaciones

- Es prudente que la Dirección de Informática de la PUCE diseñe un plan de mantenimiento y actualización del módulo, haciendo además pruebas de carga que aseguren una estabilidad en los tiempos de respuesta en un entorno real operativo.
- Si se considera necesario aumentar las seguridades, se puede implementar capas extra como la autenticación de dos factores (2FA) que permita robustecer mucho más a la aplicación, validación obligatoria de contraseña segura y también usar tokens de actualización que ayudaran mantener la sesión activa si la cookie de la sesión expira.
- La construcción de un manual de usuario o videos explicativos será de gran ayuda para poder capacitar a las personas que utilizarán la aplicación web, haciendo que se familiaricen con la nueva herramienta de trabajo, reduciendo la resistencia al cambio del sistema actual y facilitando la transición general.
- Sería conveniente añadir la funcionalidad responsive a la aplicación web, de manera que se pueda adaptar el contenido de forma automática y también utilizar la aplicación en otros dispositivos de ser necesario.
- Se recomienda desarrollar un módulo de Inteligencia de Negocios, el cual permita mejorar la toma de decisiones de la coordinación del Consultorio Jurídico, identificando las tendencias de temas legales actuales y proponiendo soluciones rápidas a casos similares.

REFERENCIAS

Amorin, D. (2024). *10 principios de usabilidad de Jakob Nielsen (con ejemplos)*. Diego Amorin. https://diegoamorin.com/10-principios-usabilidad/#sobre-jakob

Arias, D. (2021). *Hashing in Action: Understanding bcrypt*. auth0. https://auth0.com/blog/hashing-in-action-understanding-bcrypt/

Cilsa. (2023). ¿Qué es un lenguaje de programación?. https://desarrollarinclusion.cilsa.org/tecnologia-inclusiva/que-es-un-lenguaje-de-programacion/

De Luca, S. (2022). Buenas prácticas para el desarrollo web. *UNLP. https://sedici.unlp.edu.ar/bitstream/handle/10915/145469/Documento_completo.pdf?sequence=1*

De Roy, S. (2023). ¿Qué es Tailwind CSS? Guía para principiantes. freeCodeCamp. https://www.freecodecamp.org/espanol/news/que-es-tailwind-css-guia-para-principiantes/

Delgado, L., & Díaz, L. (2021). Modelos de desarrollo de software. *Revista Cubana de Ciencias Informáticas*, *15*(1), (37-51). http://scielo.sld.cu/scielo.php?pid=s2227-18992021000100037&script=sci_arttext

Deyimar, A. (2023). ¿Qué es JSON?. Hostinger. https://www.hostinger.com/es/tutoriales/que-es-json

Erickson, J. (2024). *MySQL: qué es y cómo se usa*. Oracle. https://www.oracle.com/ar/mysql/what-is-mysql/

García, F. (2023). *Axios Javascript: analizamos las características de este ligero cliente HTTP*. arsys. https://www.arsys.es/blog/axios

García, F. (2024). *¿Qué es Visual Studio Code y cuáles son sus ventajas?*. arsys. https://www.arsys.es/blog/que-es-visual-studio-code-y-cuales-son-sus-ventajas

GitHub. (2025). *Acerca de GitHub y Git*. Documentación de GitHub. https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

Kumar, A. (2024). *A Guide to Web Development Frameworks*. Builtin. https://builtin.com/articles/web-development-frameworks

López, L. (2020). *Qué es Json Web Token y cómo funciona*. OpenWebinars. https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/

López, Y. (2023). *JavaScript vs. TypeScript: ¿Cuál es la diferencia?*. EDteam. https://ed.team/blog/javascript-vs-typescript-cual-es-la-diferencia

Marqués, M. (2011). *Bases de datos*. Castelló de la Plana: Publicacions de la Universitat Jaume I. https://repositori.uji.es/items/cf0051e4-1d42-4923-a2a4-1c056e52c8d1

MDN Web Docs. (2025). *The web standards model - HTML, CSS and JavaScript*. Mozilla. https://developer.mozilla.org/en-US/docs/Learn_web_development/Getting_started/Web_standards/The_web_standards_model

Muñoz, J. (2023). Introducción a las aplicaciones web. *IES Gonzalo Nazareno.* https://fp.josedomingo.org/iaw/pdf/introduccion_aplicaciones_web.pdf

Pinia. (2025). Introduction. https://pinia.vuejs.org/introduction.html

Pontificia Universidad Católica del Ecuador. (2023). *Consultorios Jurídicos Gratuitos*. https://www.puce.edu.ec/servicios/consultorios-juridicos-gratuitos/

PrimeVue. (2025). The Next-Gen UI Suite for Vue.js. https://primevue.org/

Red Hat. (2023). *¿Qué es y para qué sirve un IDE?*. https://www.redhat.com/es/topics/middleware/what-is-ide

Rich, A. (2023). *JWT claims*. Stytch. https://stytch.com/blog/jwt-claims/

Saad, A., & Shaharin, S. (2016). The methodology for ontology development in lesson plan domain. *International Journal of Advanced Computer Science and Applications, 7*(4). https://dx.doi.org/10.14569/IJACSA.2016.070472

Santander Universidades. (2020). *Metodologías de desarrollo de software*. Santander Open Academy. https://www.santanderopenacademy.com/es/blog/metodologias-desarrollo-software.html

Unity. (2025). ¿Qué es el control de versiones?. https://unity.com/es/topics/what-is-version-control

Universidad Tecnologica Nacional (UTN). (2025). Introducción a la Programación. *Seminario de Ingreso Tecnicatura Universitaria en Programación.* https://sanfrancisco.utn.edu.ar/documentos/archivos/ingreso/Intro_%20a%20la%20Programaci%C3%B3n%20-%20S_%20Ingreso%20TUP.pdf

Vite. (2025). Introducción. https://es.vite.dev/guide/

Zelaya, C. (2020). Nuevas tendencias en desarrollo Web. *Instituto Tecnológico de Chalatenango*. https://www.itcha.edu.sv/investigacion/1167

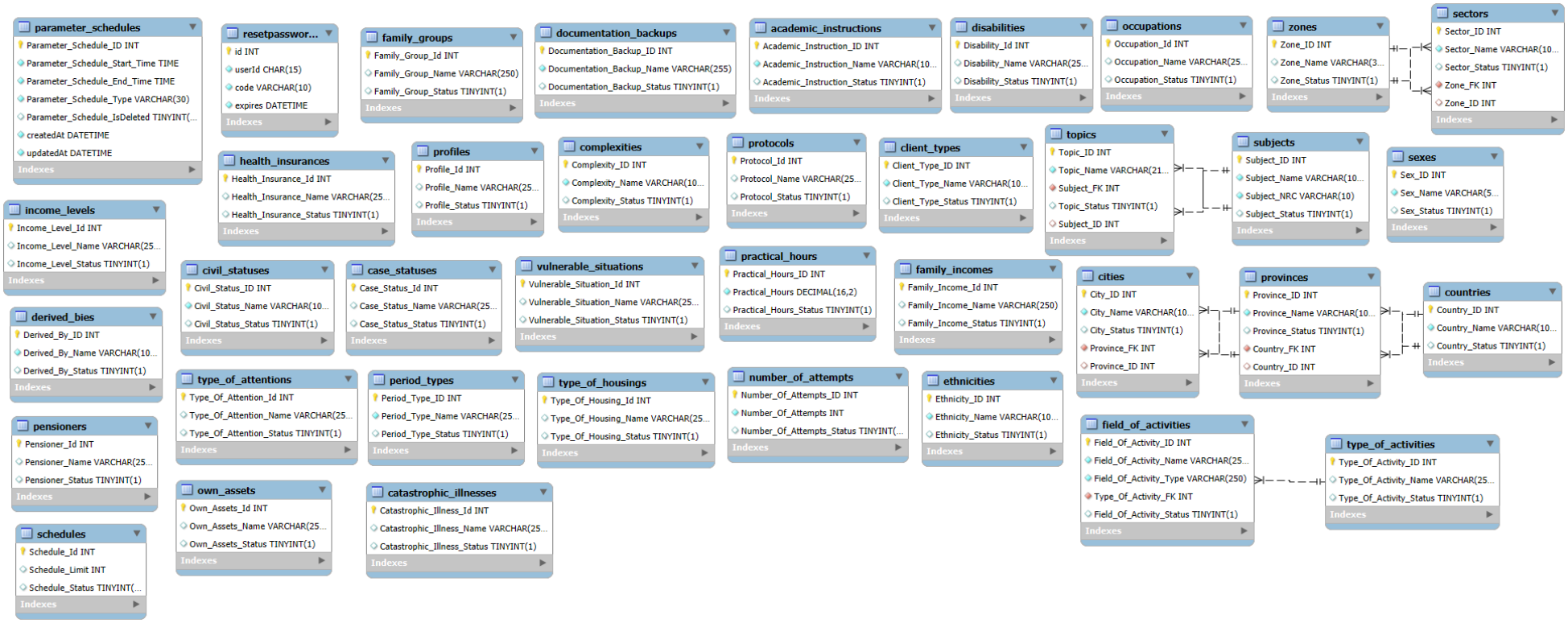
ANEXOS

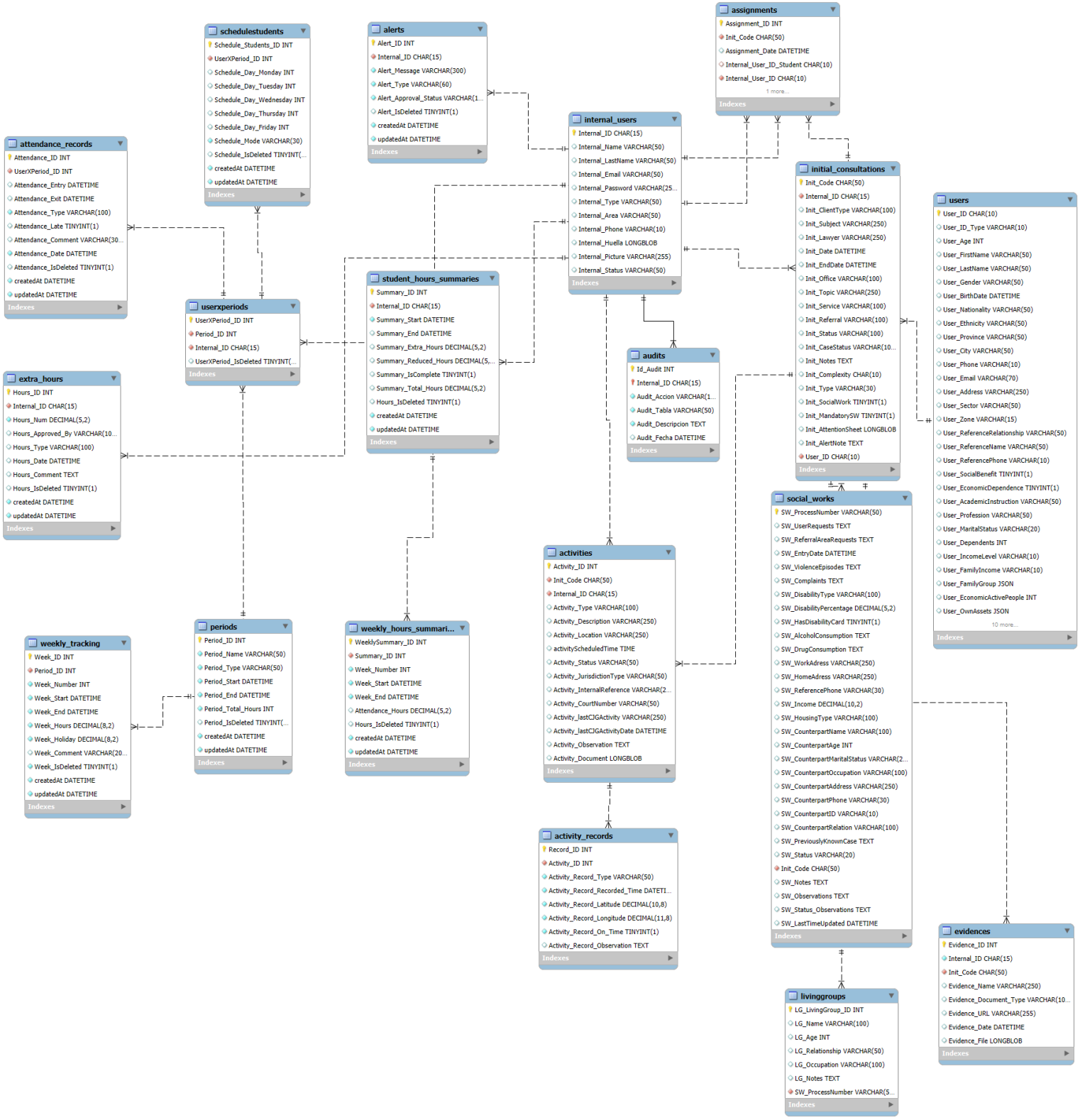
Video de Prueba de la aplicación:

https://drive.google.com/file/d/15RgJmXd1Hos2nwzVugPqU6kaGKooFSFI/view?usp=sharing

Diagrama Entidad-Relación de la aplicación:

https://drive.google.com/file/d/1SWFDi0xZFg7QJLAOKY41DjVaTBo1C0CA/view?usp=sharing




schedulestudents
Schedule_Students_ID INT
UserXPeriod_ID INT
Schedule_Day_Monday INT
Schedule_Day_Tuesday INT
Schedule_Day_Wednesday INT
Schedule_Day_Thursday INT
Schedule_Day_Friday INT
Schedule_Mode VARCHAR(30)
Schedule_IsDeleted TINYINT(...
createdAt DATETIME
updatedAt DATETIME
Indexes
alerts
Alert_ID INT
Internal_ID CHAR(15)
Alert_Message VARCHAR(300)
Alert_Type VARCHAR(60)
Alert_Approval_Status VARCHAR(1...
Alert_IsDeleted TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
assignments
Assignment_ID INT
Init_Code CHAR(50)
Assignment_Date DATETIME
Internal_User_ID_Student CHAR(10)
Internal_User_ID CHAR(10)
1 more...
Indexes
attendance_records
Attendance_ID INT
UserXPeriod_ID INT
Attendance_Entry DATETIME
Attendance_Exit DATETIME
Attendance_Type VARCHAR(100)
Attendance_Late TINYINT(1)
Attendance_Comment VARCHAR(30...
Attendance_Date DATETIME
Attendance_IsDeleted TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
internal_users
Internal_ID CHAR(15)
Internal_Name VARCHAR(50)
Internal_LastName VARCHAR(50)
Internal_Email VARCHAR(50)
Internal_Password VARCHAR(25...
Internal_Type VARCHAR(50)
Internal_Area VARCHAR(50)
Internal_Phone VARCHAR(10)
Internal_Huella LONGBLOB
Internal_Picture VARCHAR(255)
Internal_Status VARCHAR(50)
Indexes
initial_consultations
Init_Code CHAR(50)
Internal_ID CHAR(15)
Init_ClientType VARCHAR(100)
Init_Subject VARCHAR(250)
Init_Lawyer VARCHAR(250)
Init_Date DATETIME
Init_EndDate DATETIME
Init_Office VARCHAR(100)
Init_Topic VARCHAR(250)
Init_Service VARCHAR(100)
Init_Referral VARCHAR(100)
Init_Status VARCHAR(100)
Init_CaseStatus VARCHAR(10...
Init_Notes TEXT
Init_Complexity CHAR(10)
Init_Type VARCHAR(30)
Init_SocialWork TINYINT(1)
Init_MandatorySW TINYINT(1)
Init_AttentionSheet LONGBLOB
Init_AlertNote TEXT
User_ID CHAR(10)
Indexes
users
User_ID CHAR(10)
User_ID_Type VARCHAR(10)
User_Age INT
User_FirstName VARCHAR(50)
User_LastName VARCHAR(50)
User_Gender VARCHAR(50)
User_BirthDate DATETIME
User_Nationality VARCHAR(50)
User_Ethnicity VARCHAR(50)
User_Province VARCHAR(50)
User_City VARCHAR(50)
User_Phone VARCHAR(10)
User_Email VARCHAR(70)
User_Address VARCHAR(250)
User_Sector VARCHAR(50)
User_Zone VARCHAR(15)
User_ReferenceRelationship VARCHAR(50)
User_ReferenceName VARCHAR(50)
User_ReferencePhone VARCHAR(10)
User_SocialBenefit TINYINT(1)
User_EconomicDependence TINYINT(1)
User_AcademicInstruction VARCHAR(50)
User_Profession VARCHAR(50)
User_MaritalStatus VARCHAR(20)
User_Dependents INT
User_IncomeLevel VARCHAR(10)
User_FamilyIncome VARCHAR(10)
User_FamilyGroup JSON
User_EconomicActivePeople INT
User_OwnAssets JSON
10 more...
Indexes
student_hours_summaries
Summary_ID INT
Internal_ID CHAR(15)
Summary_Start DATETIME
Summary_End DATETIME
Summary_Extra_Hours DECIMAL(5,2)
Summary_Reduced_Hours DECIMAL(5,...
Summary_IsComplete TINYINT(1)
Summary_Total_Hours DECIMAL(5,2)
Hours_IsDeleted TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
audits
Id_Audit INT
Internal_ID CHAR(15)
Audit_Accion VARCHAR(1...
Audit_Tabla VARCHAR(50)
Audit_Descripcion TEXT
Audit_Fecha DATETIME
Indexes
userxperiods
UserXPeriod_ID INT
Period_ID INT
Internal_ID CHAR(15)
UserXPeriod_IsDeleted TINYINT(...
Indexes
extra_hours
Hours_ID INT
Internal_ID CHAR(15)
Hours_Num DECIMAL(5,2)
Hours_Approved_By VARCHAR(10...
Hours_Type VARCHAR(100)
Hours_Date DATETIME
Hours_Comment TEXT
Hours_IsDeleted TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
activities
Activity_ID INT
Init_Code CHAR(50)
Internal_ID CHAR(15)
Activity_Type VARCHAR(100)
Activity_Description VARCHAR(250)
Activity_Location VARCHAR(250)
activityScheduledTime TIME
Activity_Status VARCHAR(50)
Activity_JurisdictionType VARCHAR(50)
Activity_InternalReference VARCHAR(2...
Activity_CourtNumber VARCHAR(50)
Activity_lastCJGActivity VARCHAR(250)
Activity_lastCJGActivityDate DATETIME
Activity_Observation TEXT
Activity_Document LONGBLOB
Indexes
social_works
SW_ProcessNumber VARCHAR(50)
SW_UserRequests TEXT
SW_ReferralAreaRequests TEXT
SW_EntryDate DATETIME
SW_ViolenceEpisodes TEXT
SW_Complaints TEXT
SW_DisabilityType VARCHAR(100)
SW_DisabilityPercentage DECIMAL(5,2)
SW_HasDisabilityCard TINYINT(1)
SW_AlcoholConsumption TEXT
SW_DrugConsumption TEXT
SW_WorkAdress VARCHAR(250)
SW_HomeAdress VARCHAR(250)
SW_ReferencePhone VARCHAR(30)
SW_Income DECIMAL(10,2)
SW_HousingType VARCHAR(100)
SW_CounterpartName VARCHAR(100)
SW_CounterpartAge INT
SW_CounterpartMaritalStatus VARCHAR(2...
SW_CounterpartOccupation VARCHAR(100)
SW_CounterpartAddress VARCHAR(250)
SW_CounterpartPhone VARCHAR(30)
SW_CounterpartID VARCHAR(10)
SW_CounterpartRelation VARCHAR(100)
SW_PreviouslyKnownCase TEXT
SW_Status VARCHAR(20)
Init_Code CHAR(50)
SW_Notes TEXT
SW_Observations TEXT
SW_Status_Observations TEXT
SW_LastTimeUpdated DATETIME
Indexes
periods
Period_ID INT
Period_Name VARCHAR(50)
Period_Type VARCHAR(50)
Period_Start DATETIME
Period_End DATETIME
Period_Total_Hours INT
Period_IsDeleted TINYINT(...
createdAt DATETIME
updatedAt DATETIME
Indexes
weekly_hours_summari...
WeeklySummary_ID INT
Summary_ID INT
Week_Number INT
Week_Start DATETIME
Week_End DATETIME
Attendance_Hours DECIMAL(5,2)
Hours_IsDeleted TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
weekly_tracking
Week_ID INT
Period_ID INT
Week_Number INT
Week_Start DATETIME
Week_End DATETIME
Week_Hours DECIMAL(8,2)
Week_Holiday DECIMAL(8,2)
Week_Comment VARCHAR(20...
Week_IsDeleted TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
activity_records
Record_ID INT
Activity_ID INT
Activity_Record_Type VARCHAR(50)
Activity_Record_Recorded_Time DATETI...
Activity_Record_Latitude DECIMAL(10,8)
Activity_Record_Longitude DECIMAL(11,8)
Activity_Record_On_Time TINYINT(1)
Activity_Record_Observation TEXT
Indexes
livinggroups
LG_LivingGroup_ID INT
LG_Name VARCHAR(100)
LG_Age INT
LG_Relationship VARCHAR(50)
LG_Occupation VARCHAR(100)
LG_Notes TEXT
SW_ProcessNumber VARCHAR(5...
Indexes
evidences
Evidence_ID INT
Internal_ID CHAR(15)
Init_Code CHAR(50)
Evidence_Name VARCHAR(250)
Evidence_Document_Type VARCHAR(10...
Evidence_URL VARCHAR(255)
Evidence_Date DATETIME
Evidence_File LONGBLOB
Indexes