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 – Módulo de control de ingreso biométrico para los estudiantes que realizan prácticas preprofesionales

AUTOR:

ANDRÉS XAVIER TAYUPANTA ANDRANGO

DIRECTOR:

ING. GUIDO OCHOA

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO

EN SISTEMAS DE INFORMACIÓN

QUITO DM, SEPTIEMBRE 2025

DEDICATORIA

Dedico este trabajo con profunda gratitud hacia mis padres, cuyo esfuerzo constante me permitió estudiar lo que amo y su apoyo siempre quedará guardado en mi corazón. Ellos han sido los principales testigos de mi crecimiento personal y académico, guiándome en el camino que me ha convertido en un profesional capaz de alcanzar las metas que me proponga. De igual manera, extiendo esta dedicatorio a mis profesores, por todas las enseñanzas compartidas, sobre todo, por los valores que representan y le dan sentido a esta carrera.

AGRADECIMIENTO

Expreso mi profundo agradecimiento a mis padres, por su apoyo incondicional, esfuerzo constante y motivación a lo largo de mi carrera universitaria. Su acompañamiento me ha permitido alcanzar este logro académico. De igual manera, agradezco a mis amigos, quienes me acompañaron durante este proceso, brindándome apoyo y ánimo en cada etapa de la carrera.

De manera especial, agradezco al Ing. Guido Ochoa, por su guía durante el desarrollo del presente trabajo de titulación, cuyas observaciones y conocimientos fueron fundamentales para la culminación de este.

Asimismo, agradezco al personal del Consultorio Jurídico de la PUCE por el tiempo y recursos destinados a esta propuesta, los cuales permitieron el desarrollo y validación del proyecto realizado.

Finalmente, expreso mi sincero agradecimiento al Ing. Nelson Salgado por brindarnos la oportunidad, tanto a mis compañeros como a mí, de desarrollar el presente proyecto.

RESUMEN

El consultorio jurídico de la PUCE es un espacio donde los estudiantes de derecho realizan sus prácticas preprofesionales aplicando sus conocimientos en beneficio de la comunidad. Sin embargo, con el tiempo han surgido nuevas necesidades y procesos que el sistema actual no es capaz de cumplir. Frente a esta problemática, se contempla la necesidad de desarrollar una aplicación web que se adapte a las demandas actuales del consultorio jurídico. El trabajo por realizar es un módulo que permita modernizar el proceso de registro y control de asistencia de los practicantes, centralizando la información para su fácil acceso y manipulación. De manera que el sistema pueda automatizar el registro de ingreso y salida de los estudiantes, la gestión de horarios y el monitoreo de las horas establecidas. Finalmente, con ello se busca reemplazar los métodos manuales, optimizar la gestión académica y garantizar un mejor seguimiento del cumplimiento de las prácticas en el consultorio.

PALABRAS CLAVE: Consultorio jurídico, prácticas preprofesionales, asistencia, aplicación web, automatización, gestión académica, horari

ÍNDICE

Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I. INTRODUCCIÓN

1.1. Planteamiento del problema

Actualmente, el control de asistencia de las prácticas preprofesionales en el consultorio jurídico de la PUCE depende de métodos manuales que son demorados y propensos a errores humanos. El hecho de que los estudiantes tengan que firmar hojas físicas para llevar el registro de la asistencia, además de usar una aplicación externa para llevar el tiempo, es caótico para la persona encargada de gestionar esa información, debido a que implica la revisión de múltiples registros. Esta situación afecta la calidad del seguimiento de las horas llevadas por los estudiantes y genera dificultades para garantizar que cumplan con los requisitos de sus prácticas.

Con la presente problemática, se plantea la siguiente pregunta:

¿Cómo se puede automatizar el registro de asistencia de los estudiantes en sus prácticas preprofesionales por medio de una solución digital que elimine los métodos manuales y mejore la eficiencia del control de horas?

Este cuestionamiento tiene el objetivo de encontrar una solución digitalizada que permitirá optimizar el proceso de control de asistencia en el consultorio jurídico de la PUCE al otorgar un sistema web que, a través del control biométrico, automatice el registro de las horas de entrada y salida de los estudiantes. Este sistema mejorará significativamente el control de las horas que llevan los estudiantes en sus prácticas, y a la vez, proporcionará a los supervisores una herramienta confiable para verificar el cumplimiento de los requisitos solicitados a los estudiantes

1.2. Justificación

El crecimiento constante de la digitalización en las universidades ha hecho que muchos procesos administrativos y académicos migren hacia plataformas tecnológicas. Sin embargo, aún existen ciertas áreas que dependen de métodos más tradicionales, como es el caso de registro de horas de prácticas preprofesionales de los estudiantes de derecho en el consultorio jurídico de la PUCE. En la actualidad, los estudiantes deben registrar manualmente sus horas presenciales por medio de firmas en hojas físicas, además de utilizar una aplicación externa a la institución para registrar sus horas virtuales.

Por lo tanto, se ha propuesto el desarrollo de una aplicación web que automatice el registro de las horas de los estudiantes al ingresar y salir del consultorio con su huella dactilar. Para ello, se busca brindar a los supervisores de las prácticas un acceso inmediato y organizado a los registros de asistencia a través de un sistema.

Esta iniciativa busca responder a la necesidad de automatizar procesos en el consultorio jurídico, eliminando la dependencia de métodos manuales y externos que resultan ineficientes. De esta forma, se garantiza que los supervisores de las prácticas puedan monitorear el avance de sus estudiantes y evitar errores al momento de validar sus horas cumplidas.

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar una aplicación web que permita el control de ingreso biométrico para los estudiantes que realizan prácticas en el Consultorio Jurídico de la PUCE.

1.3.2. Objetivos Específicos

- Diseñar e implementar una aplicación web de control biométrico que permita a los estudiantes registrar su entrada y salida mediante huella dactilar.
- Establecer un sistema de asignación de horarios para mejorar la planificación de las prácticas preprofesionales en la aplicación web.
- Realizar el seguimiento de horas con el fin de consolidar los registros de asistencia de los estudiantes.

1.4. Alcance

El alcance de este trabajo de titulación consiste en el desarrollo de una aplicación web que permita el control de ingreso biométrico para los estudiantes que realizan prácticas en el consultorio jurídico de la PUCE. El sistema se limita a un entorno de pruebas y validación académica, por lo cual no se contempla su implementación en un ambiente de producción.

II. CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA

2.1. Consultorio Jurídico

Es una entidad vinculada a una facultad de derecho, la cual ofrece servicios legales gratuitos a personas en situación de vulnerabilidad o de escasos recursos. Aquí se les permite a los estudiantes de derecho de los últimos niveles aplicar sus conocimientos en casos reales bajo la guía de un profesional con experiencia. De acuerdo con Correa et al. (2023), "los consultorios jurídicos gratuitos están sujetos a procesos de registro y solicitud de permisos para su funcionamiento anual, y requieren, además, una prestación de servicios de manera eficiente u organizada".

2.2. Biometría

"La biometría es una ciencia que analiza las distancias y posiciones entre las partes del cuerpo para poder identificar o clasificar a las personas" (Serratosa, 2008). Este enfoque utiliza métricas y el análisis estadístico para analizar características físicas que sean únicas de cada individuo, como por ejemplo reconocimiento facial, patrones del iris, huellas dactilares, etc.

En la actualidad, la biometría es una de las herramientas más confiables para identificar y autenticar personas en distintos ámbitos. Esto se debe a que, a diferencia de los métodos tradicionales como el uso de contraseñas o tarjetas de identificación, los rasgos biométricos no se pueden compartir o perder debido a que forman parte intrínseca de las características físicas de cada persona.

Por ello, los sistemas biométricos se han convertido en una solución tecnológica clave para el control de accesos. El sistema utiliza el rasgo biométrico del individuo para realizar una comparación con el rasgo que existe dentro de una base de datos. Normalmente, este dato está encriptado y el proceso se lleva a cabo en tres etapas principales: captura, almacenamiento y comparación.

2.2.1. Biometría dactilar para la verificación de identidad

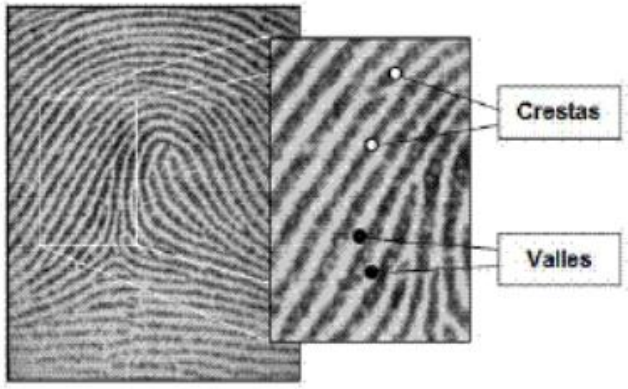
En la actualidad, la biometría dactilar es uno de los mecanismos más eficientes para autenticar a una persona, debido a que la huella de cada individuo es un identificador único que se compone de patrones únicos formados por las crestas y valles de los dedos, los cuales se crean a partir de factores ambientales y genéticos aleatorios (Albor, 2021).

El proceso de autenticación consta de algunas etapas que permiten validar la huella. En primer lugar, se realiza la captura en un dispositivo biométrico. Después se realiza la extracción de la imagen de la huella, en donde se identifican puntos singulares como bifurcaciones y terminaciones de líneas. Y finalmente se compara el patrón de valles y crestas con los patrones almacenados, dando un resultado sobre si la huella coincidió o no.

Los lectores ópticos de huella dactilar capturan la imagen mediante sensores de luz, comúnmente de tipo LED para iluminar las crestas de la huella digital. Albor menciona que "El lector mediante un led fotosensible convierte la imagen en un patrón de combinación binaria, donde la entrada del algoritmo es una imagen de 8/16 o 32-bits en escala de grises". A partir de esta conversión, se crea un hash de las imágenes, el cual se guarda como patrón para las próximas comparaciones de la huella capturada. Cabe recalcar que el lector asegura que se tome una imagen clara para evitar errores en el proceso de reconocimiento.

Figura 1

Huella Biométrica



Nota: Representación de una huella digital, donde se observan las crestas y valles que forman parte de los rasgos característicos de la identificación dactilar. Tomado de: *Algoritmo de Clasificación de Huellas Dactilares Basado en Redes Neuronales Función Base Radial* (p.104), por V.S. Flores, 2014, PGI.

2.2.2. Ventajas frente a métodos tradicionales

Como se explicó anteriormente, la huella digital es un rasgo único e intransferible que tiene cada persona. Por lo tanto, en comparación con métodos antiguos como el uso de contraseñas, tarjetas o registros manuales, este método de autenticación siempre está disponible y es difícil de falsificar. En este sentido, la autenticación dactilar minimiza los riesgos de suplantación de identidad y reduce la vulnerabilidad frente a robos de credenciales, siendo una de las alternativas más confiables para la protección de datos.

A nivel de experiencia de usuario este método de autenticación resulta mucho más ágil e intuitivo. El proceso es muy simple desde este punto de vista, pues únicamente requiere colocar el dedo sobre el lector para que el sistema verifique la identidad y otorgue el acceso

correspondiente. Esta mejora es significativa ya que hace que la interacción con el sistema se vuelva mucho más accesible, incrementando así la eficiencia en lugares donde se maneje un gran volumen de personas, como en empresas o instituciones educativas.

2.2.3. Consideraciones éticas

El uso de sistemas que cuentan con autenticación biométrica impone un reto en cuanto al manejo responsable de la información personal que procesan. Como se vio, a diferencia de métodos tradicionales, los datos biométricos no pueden ser modificados en caso de vulnerabilidad, lo que aumenta la responsabilidad del uso de estos. Las instituciones que implementen este método de autenticación están obligadas a pedir permiso a los usuarios a través de un consentimiento informado, de manera que les hagan conocer de forma clara cómo serán capturados, almacenados y utilizados sus datos biométricos.

2.3. Metodología de Desarrollo de Software

Es un marco de trabajo que se compone por un conjunto de procesos, técnicas y herramientas que permiten el desarrollo de sistemas de información. Su propósito es organizar y controlar cada una de las etapas del desarrollo, asegurando que el producto final cumpla con los requisitos deseados. Maida (2015) menciona que “cada una de las metodologías disponibles es más adecuada para tipos específicos de proyectos, basados en consideraciones técnicas, organizacionales, de proyecto y de equipo”. Esta afirmación es correcta, debido a que en la práctica no existe un modelo único que funcione siempre. Al momento de elegir una metodología se debe tener en cuenta aspectos como la complejidad del sistema, disponibilidad de recursos y el nivel de participación de los usuarios. En este contexto, la decisión de utilizar la metodología prototipada en el sistema no es fortuita, sino estratégica, ya que permite recibir

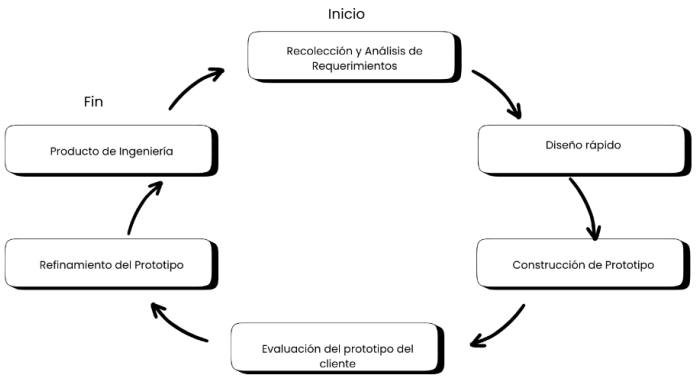
retroalimentación temprana por parte de los usuarios al ir desarrollando versiones preliminares del sistema.

2.3.1. Metodología Prototipada Evolutiva

Un prototipo se va transformando en la aplicación final a través de un proceso iterativo de ajustes y mejoras. Frecuentemente, hacer un prototipo es considerado una herramienta para entender los requerimientos reales del usuario, ya que permite visualizar de forma temprana las funcionalidades del sistema. Si las interfaces se diseñan de forma apropiada, permiten que los usuarios interactúen directamente con el prototipo y den su retroalimentación para comprobar tanto el diseño como la funcionalidad del sistema. Adaptar este enfoque proporciona una idea clara sobre el comportamiento esperado del software, reduciendo la incertidumbre y fortaleciendo la comunicación entre usuarios y desarrolladores.

Figura 2

Ciclo de vida prototipado evolutivo



Nota: Ciclo de vida de la metodología de desarrollo de software prototipado evolutivo. El diagrama muestra las fases iterativas. Fuente: Elaboración propia.

2.4. Arquitectura de Software

La arquitectura de software se comprende como la estructura fundamental de un sistema de información. Según Huet (2022), se refiere a una planificación que utiliza modelos, patrones y abstracciones conceptuales al desarrollar software, y funciona como un paso previo a la implementación. Constituye un modelo de alto nivel que permite establecer como interactúan los distintos módulos del sistema, garantizando un producto final de alta calidad.

En este aspecto, la arquitectura organiza los componentes principales de un sistema, como la base de datos, la lógica de negocio y la capa de presentación. Al separar las responsabilidades, cada módulo cumple con funciones específicas que no afectan a los demás, lo que facilita la compresión del sistema, su mantenimiento y escalabilidad.

2.4.1. Patrón Arquitectónico MVC

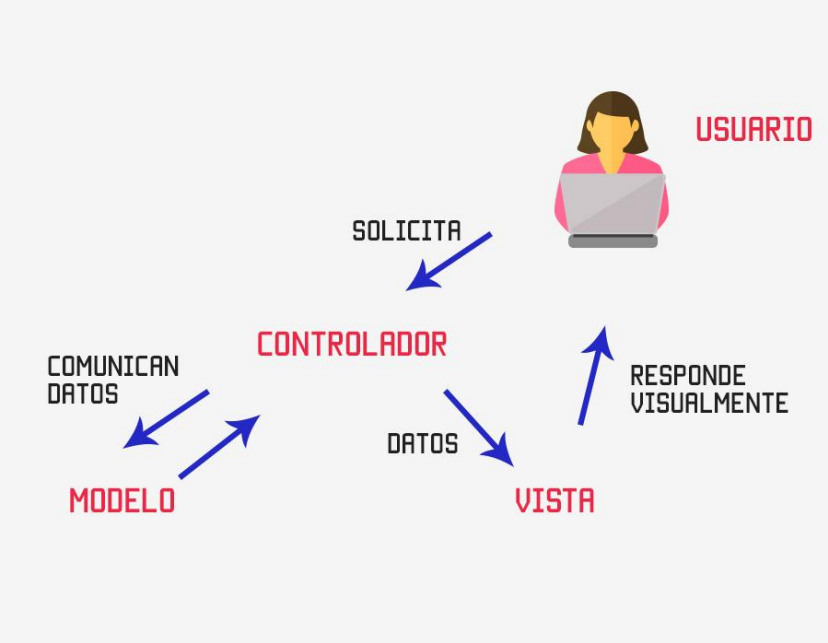
En el desarrollo de software un patrón se refiere a una solución probada y reutilizable que permite organizar y estructurar los componentes de un sistema. Dentro de estos patrones se encuentra el Modelo-Vista-Controlador (MVC), el cual es uno de los más utilizados debido a su capacidad para separar claramente las responsabilidades del software en 3 capas.

- Modelo: Se encarga de la capa de datos, siendo el encargado de interactuar con la base de datos y aplicar reglas definidas.
- Vista: Muestra la información al usuario y recibe sus interacciones.
- Controlador: Funciona como un intermediario entre el modelo y la vista, recibiendo las solicitudes de los usuarios a través de la interfaz y determinando las acciones que deben ejecutarse.

Este patrón ofrece varias ventajas, entre ellas la colaboración entre varios miembros del equipo de desarrollo, ya que es posible crear nuevas características sin afectar el trabajo de los demás. Al separar las capas cada equipo de trabajo puede centrarse en su responsabilidad específica. Otro beneficio relevante del uso de este patrón es que un mismo modelo puede ser utilizado en múltiples vistas, lo que facilita la creación de aplicaciones que compartan la misma lógica de negocio en diferentes plataformas como entornos web y móviles.

Figura 3

Modelo Vista Controlador



Nota: Representación de cómo funciona el patrón arquitectónico MVC con sus 3 capas. Tomado de *MVC (Model, View, Controller) explicado*, por U. Hernández, 2015.

2.5. Lenguaje de Programación

"Es una herramienta fundamental que permite a los seres humanos comunicarse con las computadoras mediante instrucciones claras y estructuradas" (Sebastiany, 2025). Por medio de los lenguajes de programación, es posible crear soluciones tecnológicas que resuelvan problemas específicos y automaticen procesos. A lo largo de los años, su evolución ha permitido el surgimiento de distintos paradigmas como la programación estructurada, orientada a objetos y funcional, cada uno con un enfoque que se adapta a diferentes situaciones y contextos.

2.5.1. JavaScript

En el desarrollo web, las páginas dejaron de ser simplemente estáticas para convertirse en plataformas interactivas capaces de responder ante las acciones de los usuarios. Justamente, JavaScript fue diseñado para crear páginas web dinámicas, incorporando efectos, animaciones y acciones ante los eventos ocasionados en pantalla. Eguíluz (2008) menciona que es un lenguaje interpretado que no necesita compilar para ejecutarse, teniendo la posibilidad de probar código directamente en cualquier navegador.

2.5.2. TypeScript

Por otro lado, tenemos a TypeScript. Un lenguaje pre-compilado desarrollado por Microsoft que mejora notoriamente JavaScript al incorporar tipado estático y características propias de la programación orientada a objetos. En esencia, funciona como un transpilador, ya que el código escrito en este lenguaje será compilado finalmente a JavaScript junto con otras configuraciones esenciales. Su mayor ventaja radica en la robustez que ofrece durante

el proceso de desarrollo, ya que no permitirá ejecutar la aplicación si se tiene algún error de codificación, mejorando así la escalabilidad y mantenimiento del código.

2.6. Sistema de Control de Versiones

Actualmente, el desarrollo de sistemas de información requiere de una gestión organizada y controlada de los cambios que se realizan en el código fuente. González et al. (2010) lo define como "un software que controla y organiza las distintas revisiones que se realicen sobre uno o varios documentos. Una revisión es un cambio realizado sobre un documento". En este sentido, los sistemas de control de versiones son mecanismos que garantizan que cada modificación sea registrada y pueda ser identificada posteriormente, lo que garantiza la integridad del código y su seguimiento durante la evolución del proyecto.

2.6.1. Git

Es un sistema de control de versiones distribuido que permite llevar un registro detallado de los cambios realizados en los archivos de un repositorio. Cada cambio se almacena junto con la información del autor, la fecha y descripción correspondiente, lo que facilita el seguimiento del proyecto. Al ser distribuido, Git le da a cada desarrollador una copia completa del repositorio con todo su historial, incrementando la seguridad y facilitando el trabajo en equipo sin depender de una conexión permanente. Existen 3 estados principales en los que se pueden encontrar los archivos dentro de git: modificado (modified), preparado (staged) y confirmado (committed). Gómez (2015) explica que un archivo en estado modificado significa que se han hecho cambios en el directorio de trabajo, pero aún no se han agregado al área de preparación. El estado preparado corresponde a los archivos que han sido marcados en su versión actual para que sean confirmados. Por último, un archivo pasa al

estado de confirmado cuando los cambios se han guardado de forma permanente en el repositorio local.

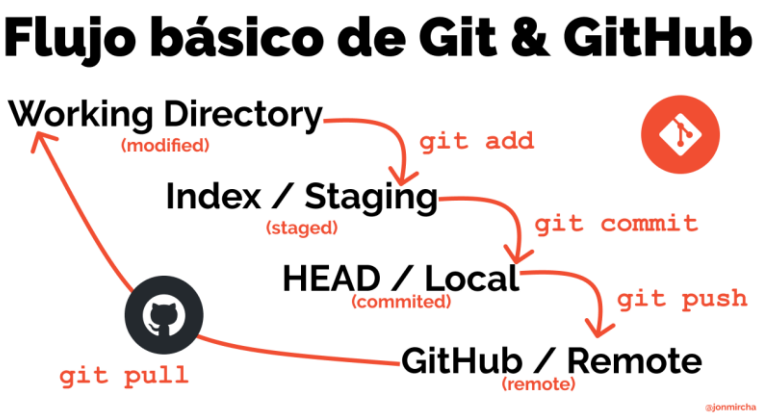
2.6.2. GitHub

Es una plataforma de desarrollo colaborativo en línea que permite alojar repositorios de Git. Proporciona un espacio centralizado en la nube donde los desarrolladores pueden gestionar sus proyectos de software de manera remota. A diferencia de Git, que solo se usa de forma local, GitHub otorga más funcionalidades al ofrecer herramientas que permiten revisar el código, coordinar tareas y la integración de nuevas funcionales dentro de los repositorios.

Fomenta la colaboración al permitir que los usuarios puedan contribuir a mejorar el software de otros mediante propuestas de cambio, revisiones y comentarios sobre el código. Entre las funcionalidades más conocidas para este propósito se encuentran el fork y el pull. Castillo (2017) indica que hacer un fork consiste en generar una copia de un repositorio existente dentro de la cuenta propia del usuario para corregir errores o hacer modificaciones sobre él. Una vez que se apliquen los cambios, el usuario puede enviar un pull request al dueño del proyecto, quien revisará las modificaciones y decidirá su incorporación en función de la calidad de la contribución.

Figura 4

Flujo básico de Git y GithHub



Nota: Flujo básico de trabajo con Git y GitHub. El diagrama representa las etapas principales del proceso de control de versiones. Tomado de Git, por J. Mircha, 2022.

2.7. Base de Datos

Según Pérez Mora et al. (2005), "Una base de datos de un SI es la representación integrada de los conjuntos de entidades instancia correspondientes a las diferentes entidades tipo del SI y de sus interrelaciones". Esto significa que organiza y relaciona los datos para que puedan ser utilizados de forma eficiente y coherente dentro de un sistema de información. Una base de datos garantiza integridad, seguridad y consistencia en los datos, facilitando su manipulación para responder ante necesidades de consulta, actualización y almacenamiento de información.

2.7.1. MySQL

Es un sistema gestor de base de datos relacional de código abierto que utiliza el lenguaje de consulta SQL para la manipulación de datos. Su modelo cliente-servidor permite centralizar la información y facilitar el acceso simultáneo a múltiples usuarios. Gracias a esta arquitectura MySQL es capaz de gestionar grandes volúmenes de datos y ofrecer un alto rendimiento. "Es un servidor multiprocesos, es decir, que cada vez que alguien establece una

conexión con el servidor, el programa servidor crea un subproceso para manejar la solicitud del cliente" (Gómez, R., 2016). Con este enfoque, MySQL asegura eficiencia en la gestión de recursos y asegura la estabilidad del sistema.

2.8. Framework para Desarrollo Web

A lo largo de los años en el desarrollo de aplicaciones, la necesidad de crear sistemas que cumplan con las necesidades requeridas ha llevado a los desarrolladores a utilizar herramientas que simplifiquen y estructuren la programación dentro de un proyecto. De acuerdo con Villalobos et al (2010), los frameworks tienen el propósito de "Normalizar y estructurar el código del sistema, facilitando un esquema (un patrón, un esqueleto) para el desarrollo y/o la implementación de aplicaciones". Esto ha permitido que los desarrolladores puedan enfocarse plenamente en la lógica de negocio de la aplicación en lugar de crear cada componente desde cero.

En el desarrollo web, los frameworks se han vuelto esenciales para agilizar la creación de aplicaciones modernas, debido a que proporcionan estructuras prediseñadas y componentes reutilizables que reducen la complejidad del trabajo. Estas herramientas se encargan de aspectos fundamentales como enrutamiento, comunicación entre cliente-servidor, seguridad, comunicación con la base de datos, etc.

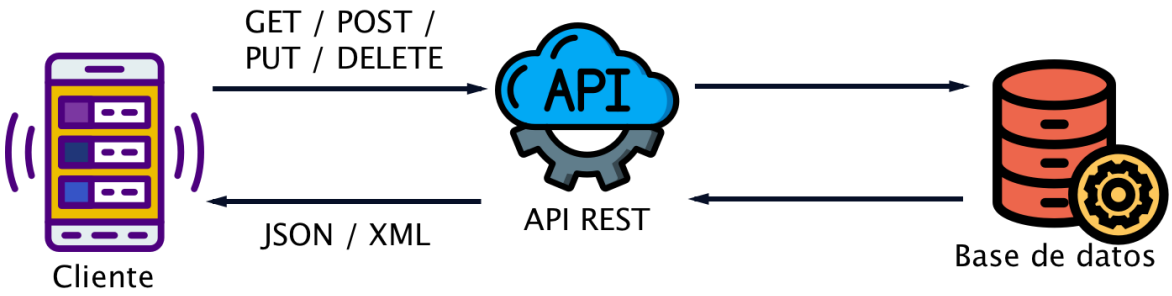
2.9. Rest

Según Alamilla (2021), REST "Define unos principios de arquitectura a seguir para implementar aplicaciones o servicios web. No obstante, se basa en estándares para su implementación: HTTP, XML". Se basa en algunos principios que facilitan la interacción entre distintos sistemas a partir del uso de recursos identificados por medio de URLs.

REST utiliza los métodos del protocolo HTTP para realizar las operaciones básicas de consulta (GET), creación (POST), actualización (PUT) y eliminación de recursos (DELETE), lo que facilita la comunicación entre cliente y servidor. Esto permite la interoperabilidad entre aplicaciones desarrolladas en distintos lenguajes o plataformas, ya que pueden interactuar mediante el uso de estos estándares.

Figura 5

Arquitectura REST



Nota: Esquema de la arquitectura REST, donde la aplicación del cliente se comunica con la base de datos a través de una API que actúa como intermediaria. Tomado de *¿Qué es una api rest?*, por S. Gutierrez, 2024.

2.10. Tecnologías de desarrollo

2.10.1. ORM - Sequelize

El mapeo objeto-relacional es una técnica utilizada en el desarrollo de software que permite establecer una correspondencia entre los objetos de un programa y las tablas de una base de datos relacional. Baño (2016) afirma que el ORM se encarga de forma automática de transformar los objetos de un programa en registros dentro de la base de datos y viceversa, lo que permite simular una base de datos orientada a objetos. Gracias a esto, se simplifica la

interacción con la base de datos al abstraer las operaciones de tipo CRUD, evitando que el programador deba escribir sentencias SQL. En su lugar, puede manipular objetos dentro del lenguaje de programación, mientras que el ORM se encarga de traducir dichas acciones para la base de datos. En este contexto, aparece Sequelize, una biblioteca que permite definir modelos, relaciones y consultas de forma declarativa con soporte para distintos motores de base de datos.

2.10.2. Vue.js

"Es un framework open source de JavaScript, que nos permite la creación de interfaces de usuario y aplicaciones de una sola página (single-page application o SPA, en inglés), de una forma muy sencilla" (Barragán, 2021). Vue.js facilita el desarrollo de aplicaciones dinámicas que cargan una sola vez en el navegador y actualizan sus contenidos de manera interactiva. Tiene una arquitectura basada en componentes, la cual permite que cada sección de la interfaz se construya de forma independiente y luego se integre modularmente.

Los componentes de este framework encapsulan fragmentos de código HTML, CSS y JavaScript en un único bloque reutilizable. Esto hace que los proyectos de desarrollo sean muy escalables y fáciles de mantener. Por otro lado, su sistema de reactividad asegura que cualquier cambio en los datos del modelo se refleje directamente en la interfaz de usuario, manteniendo el estado de la aplicación sincronizado con la vista. Al igual que otros frameworks, Vue.js proporciona un ciclo de vida para sus componentes, el cual establece las etapas por las que atraviesan desde su creación hasta eliminación dentro del DOM (Modelo de Objetos del Documento).

2.10.3. Express

Es un framework flexible para el desarrollo de aplicaciones web que funcionen sobre el entorno de Node.js. Node.js es un entorno de ejecución de Javascript que funciona en el lado del servidor. Flores (2019) menciona que "utiliza un modelo de entrada y salida sin bloqueo controlado por eventos que lo hace ligero y eficiente. Puede referirse a cualquier operación, desde leer o escribir archivos de cualquier tipo hasta hacer una solicitud HTTP". Esto quiere decir que está diseñado para realizar múltiples operaciones simultáneas, lo cual es ideal para aplicaciones que requieren de alta concurrencia y rapidez para resolver peticiones.

En este contexto, Express facilita el desarrollo de aplicaciones web y servicios. Logra abstraer gran parte de la complejidad al ofrecer un sistema de enrutamiento sencillo y el uso de middlewares que permiten capturar y procesar solicitudes antes de enviar una respuesta. Es una gran herramienta para aplicaciones web modernas bajo arquitecturas como REST y patrones como MVC. Su sintaxis, en conjunto con el ecosistema de Node.js permite un desarrollo ágil que no sacrifica escalabilidad ni rendimiento.

2.11. Experiencia de Usuario (UX)

Este término hace referencia a la experiencia que tiene un usuario al interactuar con un sistema de información. Finn y Downie (2025) añaden que "es un campo multidisciplinario en el que participan muchos stakeholders, que se basa en principios del diseño, la psicología, la ingeniería y los negocios para crear experiencias que sean a la vez intuitivas y agradables para el usuario". Este enfoque no depende únicamente de la interfaz visual, sino de distintos factores emocionales, técnicos y cognitivos que garantizan que la interacción con el sistema sea clara y satisfactoria.

Entender la UX implica entender las preferencias del usuario y la forma en que interpretan la información que se les presenta. Un sistema debe tener un estilo visual coherente que transmita seguridad y confianza para que el usuario pueda realizar acciones sin preocuparse de tiempos de carga lentos, errores en la interfaz o respuestas poco claras. Cuando estos elementos se ignoran al desarrollar un sistema se incrementa la probabilidad de fracaso, ya que tienden a generar frustración y no cumplen con su propósito fundamental, el cual es garantizar la usabilidad.

Además, la experiencia de usuario contempla la percepción global que se tiene del sistema a lo largo del tiempo. Esto quiere decir que debe ser fácil de aprender, permitiendo que los usuarios comprendan cómo utilizarlo sin necesidad de una curva de aprendizaje alta. Otro aspecto importante que debe cumplir es la eficiencia del uso, es decir, la capacidad del sistema para que las tareas se cumplan en el menor tiempo y con el mínimo esfuerzo posible, optimizando así los procesos e incrementado la productividad del usuario.

Finalmente, recopilar el feedback de los usuarios es fundamental para evaluar la calidad de la experiencia de uso y las futuras mejoras del sistema. Se pueden realizar entrevistas, encuestas o pruebas de usabilidad para identificar las dificultades y fortalezas percibidas durante la interacción. Recopilar esta información permite ajustar el diseño de acuerdo con las necesidades del usuario.

III. CAPÍTULO III: ANÁLISIS Y DISEÑO DE LA APLICACIÓN

3.1. Análisis de Requerimientos

Esta etapa del ciclo de desarrollo de software es de las más importante, ya que permite identificar las necesidades de los usuarios y convertirlos en requisitos funcionales para el sistema. Un análisis adecuado permite que el producto final cumpla con los objetivos que se plantearon y se adapte a los procesos reales.

En el presente proyecto, se realizaron varias reuniones y entrevistas con el personal de consultorios jurídicos para levantar la información necesaria y comprender los procesos administrativos. En particular, se trabajó directamente con la secretaria encargada de la gestión de prácticas preprofesionales de los estudiantes de la facultad de Derecho, quien proporcionó detalles sobre los procedimientos y necesidades que debía cubrir el sistema.

Finalmente, se revisaron actas en las que se establecieron los lineamientos y requerimientos, lo que permitió obtener una visión amplia de los flujos administrativos y lo que se debía automatizar.

3.1.1. Requerimientos Funcionales

3.1.1.1. Gestión de Periodos Académicos

- El aplicativo web permitirá administrar los periodos académicos.
- **Desglose Semanal:** El aplicativo web, a partir del periodo registrado, generará un desglose en semanas, indicando fecha de inicio y fin de cada una. Eso facilitará el control de las prácticas preprofesionales.

3.1.1.2. Gestión de Estudiantes

- El aplicativo web permitirá administrar a los estudiantes.
- **Asignación de Periodo Académico:** El aplicativo web permitirá que cada estudiante se vincule a uno o más periodos académicos.
- **Asignación de Área:** El aplicativo web permitirá que cada estudiante se vincule a un área de práctica (civil, penal, niñez, etc.)

3.1.1.3. Gestión de Horarios

- El aplicativo web permitirá administrar los horarios presenciales y virtuales de las prácticas de los estudiantes que realizan prácticas.

3.1.1.4. Gestión de Huella Biométrica

- El aplicativo web permitirá administrar las huellas digitales de los estudiantes mediante su captura inicial, recaptura en caso de fallos o cambios, y eliminación cuando ya no sea necesaria.

3.1.1.5. Gestión de Asistencia

- El aplicativo web permitirá administrar los registros de asistencia de los estudiantes. Puede realizarse mediante huella, ingreso manual o asignación de horas extraordinarias.
- **Control de registros:** El aplicativo web permitirá al administrador cerrar registros incompletos o modificar asistencias en casos excepcionales.

3.1.2. Requerimientos No Funcionales

3.1.2.1. Seguridad

- El aplicativo web protegerá los datos biométricos mediante encriptación en Base64 y almacenamiento en formato BLOB.
- Se implementará un control de acceso basado en roles.

3.1.2.2. Escalabilidad

- La arquitectura utilizada debe permitir la incorporación de nuevos módulos sin comprometer la funcionalidad o rendimiento del aplicativo web.

3.1.2.3. Mantenibilidad

- El aplicativo web deberá seguir los estándares establecidos por la Dirección de Informática de la PUCE.
- El código fuente deberá estructurarse en capas para facilitar su comprensión.

3.1.2.4. Usabilidad

- La interfaz debe ser intuitiva, clara y fácil de usar.
- Debe contar con una curva de aprendizaje baja.

3.2. Especificación de Requerimientos

En este punto se delimitarán los requerimientos que serán implementados en este módulo del aplicativo web, especificando los más esenciales para su funcionamiento.

F1. Gestión de Periodos Académicos

F2. Gestión de Estudiantes

F3. Gestión de Horarios

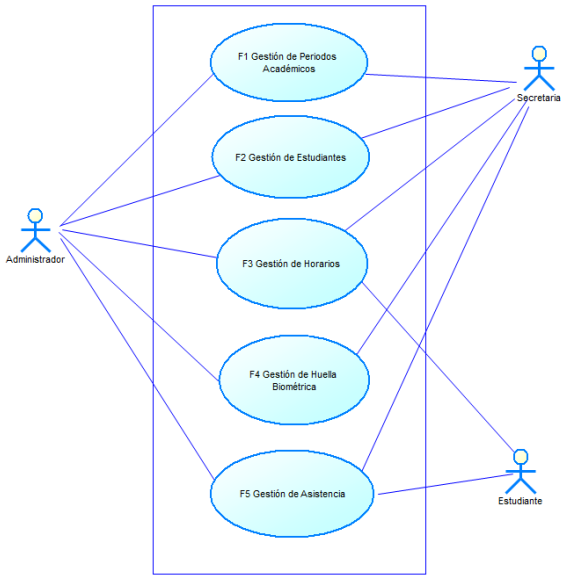
F4. Gestión de Huella Biométrica

F5. Gestión de Asistencia

3.2.1. Diagrama de Casos de Uso General

Figura 6

Caso de uso: Módulo Control de Estudiantes



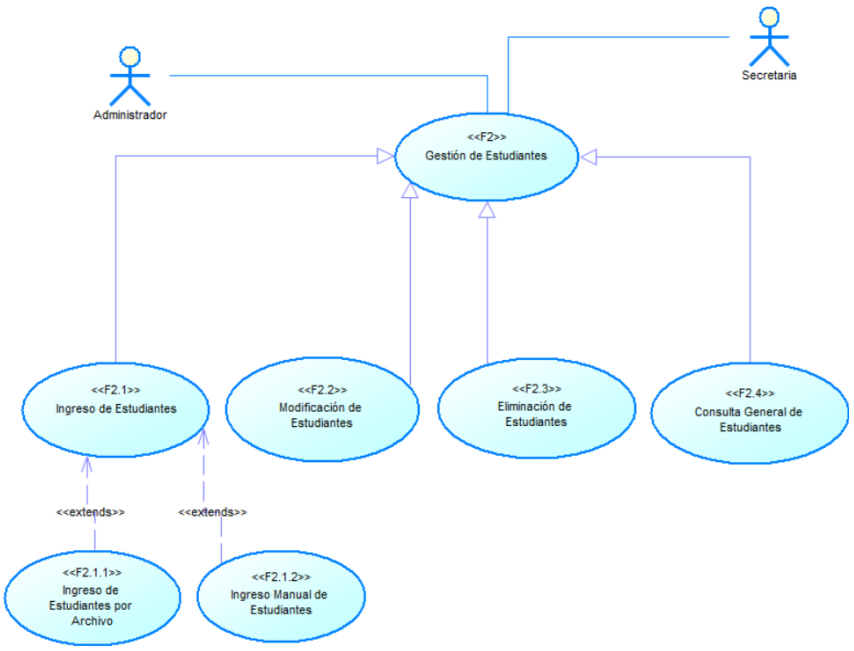
Nota: La figura representa el caso de uso general del módulo de control de estudiantes, en el cual participan 3 tipos de usuario. Fuente: Elaboración propia.

3.2.2. Diagrama de Casos de Uso: Siguiente Nivel

3.2.2.1.Siguiente Nivel F2: Gestión de Estudiantes

Figura 7

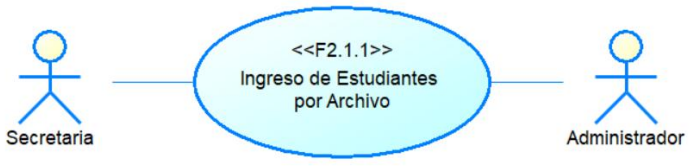
Caso de Uso siguiente nivel: Gestión de Estudiantes



Nota: Caso de uso siguiente nivel F2. Fuente: Elaboración propia.

Figura 8

Caso de uso: F2.1.1 Ingreso de Estudiantes por Archivo (A detalle)



Nota: Caso de uso F2.1.1 Ingreso de Estudiantes por Archivo. Fuente:

Elaboración propia.

Tabla 1

Caso de uso: F2.1.1 Ingreso de Estudiantes por Archivo

Nombre del caso de uso: F2.1.1 Ingreso de Estudiantes por Archivo	**ID Único:** PC-001
Área: Gestión Académica	
Actor(es): Administrador, secretaria	
Interesados: Coordinación Académica, Secretaría, Administración del sistema	
Nivel: -	
Descripción: Este caso de uso describe el proceso mediante el cual un usuario autorizado registra estudiantes en el sistema de manera masiva a través de la carga de un archivo Excel (.xlsx). La funcionalidad permite importar la información de los estudiantes a un período académico específico, validando la estructura del archivo y los datos ingresados antes de guardarlos en la base de datos.	
Evento desencadenador: El actor selecciona la opción "Ingreso por Archivo" dentro del submenú Gestión de Estudiantes.	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal)	**Información para los pasos**
1. El actor pulsa "Seleccionar archivo .xlsx".	
2. El actor selecciona el período académico donde se registrarán los estudiantes.	Lista desplegable de períodos activos.
3. El actor carga el archivo Excel que contiene la información de los estudiantes.	Archivo con columnas obligatorias.
4. El sistema procesa el archivo y valida que su estructura contenga las columnas requeridas: Cédula, Apellidos, Nombres, Correo y Área.	
5. El sistema verifica que no existan duplicados (por cédula) ni campos vacíos en los registros cargados.	
6. Si se detectan errores, el sistema muestra un mensaje indicando los campos incompletos o inválidos y permite al actor editar manualmente los datos directamente en la grilla o reemplazar el archivo	Corrección manual o recarga del archivo.
7. Una vez corregidos los errores, el actor presiona el botón "Guardar".	
8. El sistema almacena los registros válidos en la base de datos y asocia cada estudiante al período académico seleccionado.	

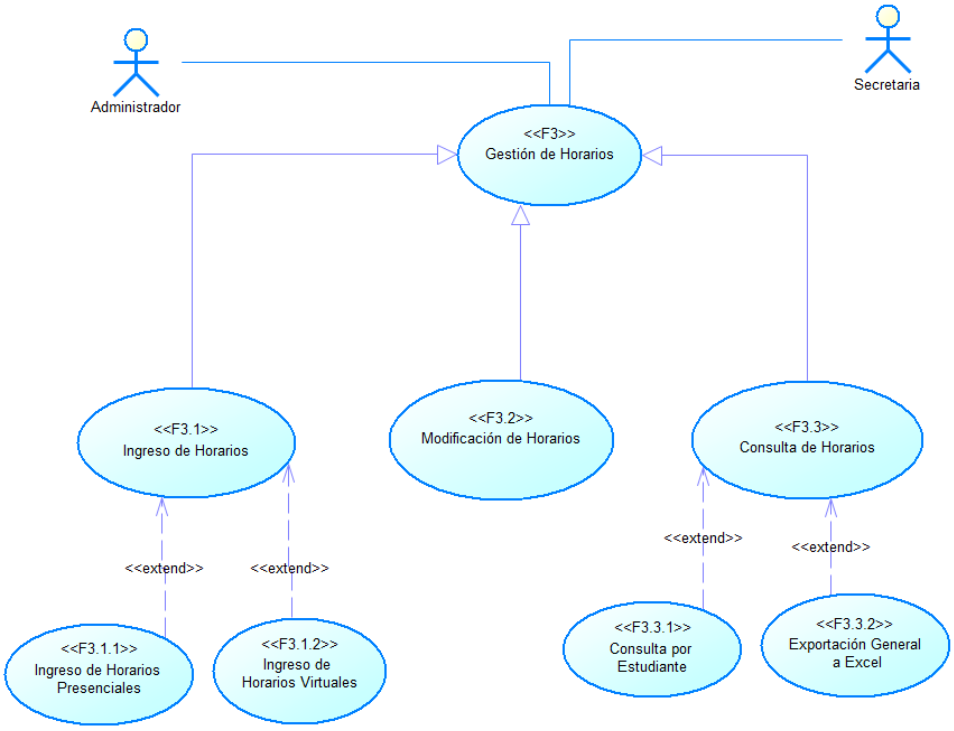
9. El sistema registra la acción en auditoría.	
Precondiciones: • El actor debe haber iniciado sesión con el perfil de Administrador o secretaria. • Debe existir al menos un período académico activo en el sistema. • El archivo Excel debe cumplir con la estructura obligatoria: Cédula, Apellidos, Nombres, Correo y Área.	
Postcondiciones: • Los estudiantes quedan registrados correctamente en la base de datos. • Cada registro queda asociado al período académico correspondiente.	
Suposiciones: El actor cuenta con el archivo Excel válido y la información completa para realizar la carga masiva.	
Garantía de éxito: El sistema valida, registra y confirma la carga masiva de estudiantes de forma exitosa.	
Garantía mínima: En caso de errores, el sistema notifica al actor e impide el registro parcial de datos.	
Requerimientos cumplidos: Ingreso de Estudiantes por Archivo	
Cuestiones pendientes: Ninguna.	
Prioridad: Alta	
Riesgo: Bajo	

Nota: Tabla que describe el caso de uso F2.1.1 “Ingreso de Estudiantes por Archivo” a detalle. Fuente: Elaboración propia.

3.2.2.2. Siguiente Nivel: F3. Gestión de Horarios

Figura 9

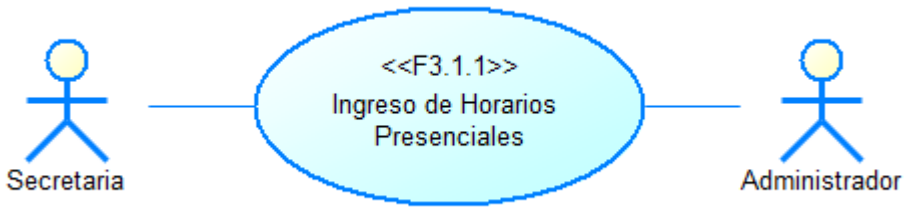
Caso de Uso siguiente nivel: Gestión de Horarios



Nota: Caso de uso siguiente nivel F3. Fuente: Elaboración propia

Figura 10

Caso de uso: F3.1.1 Ingreso de Horarios Presenciales (A detalle)



Nota: Caso de uso F3.1.1 Ingreso de Horarios Presenciales. Fuente: Elaboración propia.

Tabla 2

Caso de uso: F3.1.1 Ingreso de Horarios Presenciales

Nombre del caso de uso: F3.1.1 Ingreso de Horarios Presenciales	**ID Único:** PC-002
Área: Gestión Académica	
Actor(es): Administrador, secretaria	
Interesados: Coordinación Académica, Secretaría, Administración del sistema	
Nivel: -	
Descripción: Este caso de uso describe el proceso mediante el cual un usuario autorizado asigna el horario presencial de un estudiante para un período académico. La funcionalidad permite seleccionar a un estudiante, configurar sus tramos horarios de lunes a viernes, definir horarios especiales y ajustar el máximo de horas semanales permitido antes de guardar.	
Evento desencadenador: El actor selecciona la opción "Horario Presencial" dentro del módulo Gestión de Horarios, selecciona un periodo académico y un estudiante para colocarle su respectivo horario.	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal)	**Información para los pasos**
1. El actor configura los tramos horarios requeridos para los días necesarios usando los selectores disponibles y, de ser necesario, ajusta la cantidad de horas por día.	
2. (Opcional) El actor hace clic en "Agregar Horario Especial", define hora de inicio y fin y confirma.	Crea un tramo excepcional reutilizable
3. (Opcional) El actor hace clic en "Cambiar máximo de horas", ingresa el nuevo valor semanal permitido y guarda.	Nuevo tope de horas semanales
4. El actor presiona "Guardar Horario".	
5. El sistema valida que haya al menos un día asignado, que los rangos horarios sean válidos (inicio < fin), que no se exceda el máximo de horas semanales y que no existan solapes con horarios ya asignados en el mismo período y modalidad.	
6. Si la validación es exitosa, el sistema registra el horario presencial del estudiante.	
7. El sistema registra la acción en auditoría	
Precondiciones: • El actor ha iniciado sesión con perfil Administrador o secretaria. • Existen períodos y áreas configurados. • Los estudiantes del período/área están disponibles para asignación	

Postcondiciones: • El horario presencial del estudiante queda registrado por día y asociado al período seleccionado.
Suposiciones: • Los tramos predefinidos y los horarios especiales creados están dentro del horario operativo de la institución. • La contabilización de horas es automática a partir de los rangos seleccionados.
Garantía de éxito: El sistema confirma la asignación del horario presencial sin exceder el máximo semanal ni generar solapes.
Garantía mínima: Si ocurre un error de validación o de sistema, no se guardan cambios y se informa al actor el motivo.
Requerimientos cumplidos: Ingreso de Horarios Presenciales
Cuestiones pendientes: Ninguna.
Prioridad: Alta
Riesgo: Bajo

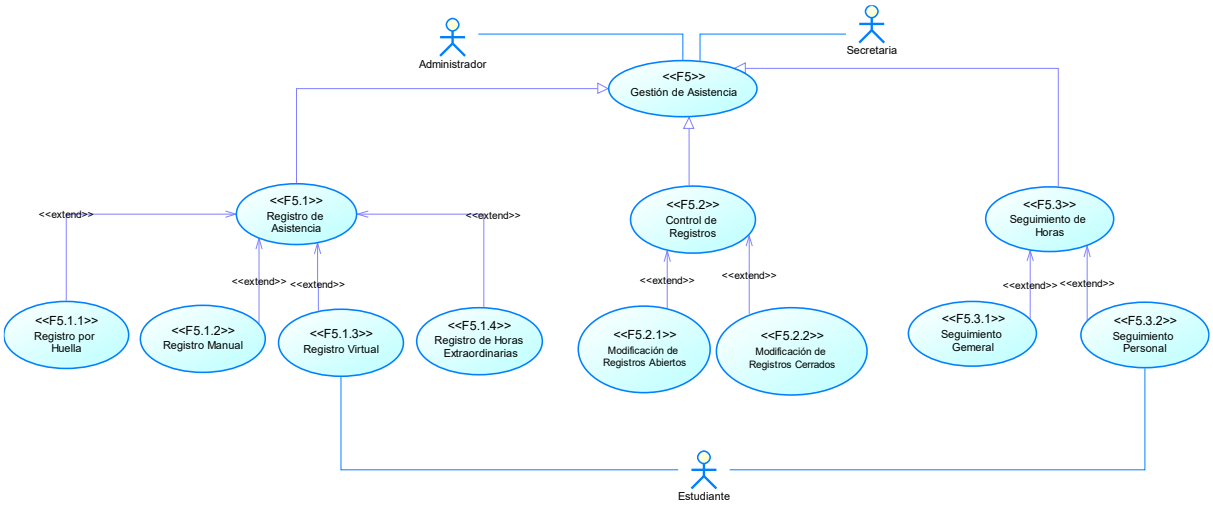
Nota: Tabla que describe el caso de uso F3.1.1 “Ingreso de Horarios Presenciales” a detalle.

Fuente: Elaboración propia.

3.2.2.3.Siguiente Nivel: F5. Gestión de Asistencia

Figura 11

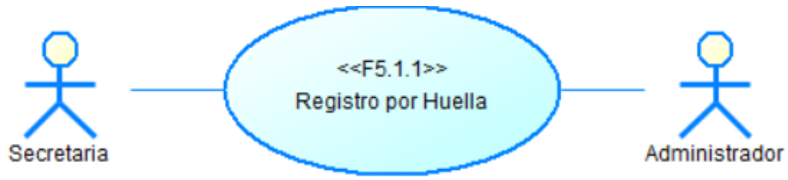
Caso de Uso siguiente nivel: Gestión de Asistencia



Nota: Caso de uso siguiente nivel F5. Fuente: Elaboración propia

Figura 12

Caso de uso: F5.1.1 Registro por Huella (A detalle)



Nota: Caso de uso F5.1.1 Registro por Huella. Fuente: Elaboración propia.

Tabla 3

Caso de uso: F5.1.1 Registro por Huella

Nombre del caso de uso: F5.1.1 Registro por Huella	**ID Único:** PC-003
Área: Gestión Académica	
Actor(es): Administrador, secretaria	
Interesados: Coordinación Académica, Secretaría, Administración académica	
Nivel: -	
Descripción: Este caso de uso describe el proceso mediante el cual un estudiante registra su asistencia de manera automática utilizando el lector biométrico. El sistema valida la identidad del usuario, su horario asignado dentro del período académico activo y determina si el ingreso corresponde a una entrada puntual o atrasada. La captura se realiza en tiempo real, almacenando tanto la huella como el registro de hora en la base de datos.	
Evento desencadenador: La secretaria accede a la sección "Registro Por Huella" en el submenú de Control Biométrico.	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal)	**Información para los pasos**
1. El actor ingresa el número de cédula del estudiante y presiona el botón "Buscar".	
2. El sistema busca al estudiante en la base de datos y verifica que esté activo y tenga un horario asignado en el período actual.	
3. Si el estudiante tiene horario válido, el sistema determina si el registro se encuentra dentro del rango permitido (±10 minutos del horario establecido).	Validación de hora del sistema vs. hora programada.

4. El sistema muestra la información del estudiante y su horario actual (hora de entrada/salida programada).	
5. Se activa la captura biométrica y el estudiante coloca su dedo en el lector.	
6. Si la huella es reconocida, el sistema registra la hora exacta de entrada/salida, y almacena la información en la base de datos.	
7. El sistema registra la acción en auditoría.	
Precondiciones: • El estudiante debe estar previamente registrado en el sistema y tener una huella biométrica asignada. • Debe existir un horario activo en el período académico actual. • El lector biométrico debe estar conectado y operativo.	
Postcondiciones: • Se almacena el registro de asistencia en la base de datos con el estado correspondiente (Normal o Atrasado). • El estudiante queda habilitado para registrar su salida posterior dentro del mismo día.	
Suposiciones: • El estudiante realiza el registro dentro de un entorno físico autorizado y con el dispositivo configurado correctamente.	
Garantía de éxito: El sistema valida la identidad y horario del estudiante, captura la huella y registra su asistencia correctamente en el sistema, confirmando el proceso mediante mensaje de éxito.	
Garantía mínima: En caso de error en la lectura o validación, el sistema informa el problema y no almacena registros incompletos o incorrectos.	
Requerimientos cumplidos: • Registro por Huella • Captura de Huella	
Cuestiones pendientes: Ninguna.	
Prioridad: Alta	
Riesgo: Bajo	

Nota: Tabla que describe el caso de uso F5.1.1 “Registro por Huella” a detalle. Fuente:

Elaboración propia.

3.3.Diagrama Entidad-Relación (ERD)

El análisis del Diagrama-Entidad-Relación (ERD) se basará principalmente en el módulo de Control de Ingreso y Gestión de Estudiantes, que constituye el núcleo del proceso académico. El modelo describe la estructura en que la información fluye y se organiza dentro del sistema, representando las relaciones entre las distintas entidades involucradas. Su objetivo es garantizar la integridad y persistencia de los datos en la administración estudiantil.

La entidad principal del modelo es “Internal_User”, la cual representa a los usuarios internos del sistema. En este caso se hará énfasis en el tipo de usuario “Estudiante” ya que es el punto de partida para la mayoría de las relaciones dentro del módulo, y en general dentro del sistema. Cada estudiante cuenta con un registro único que agrupa su información personal, académica y de contacto, lo que permite mantener un control sobre su participación en los diferentes periodos académicos registrados en la entidad “Period”.

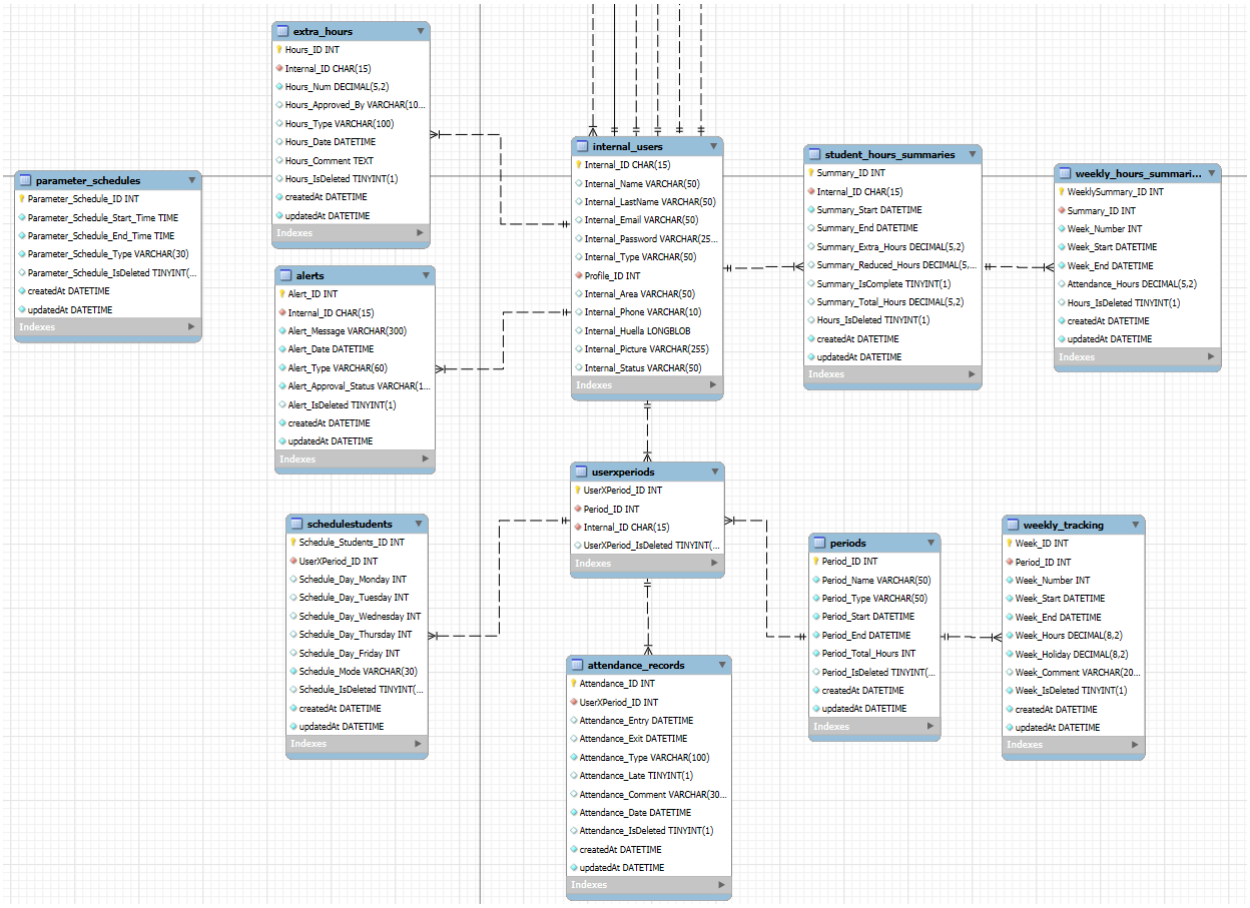
A partir de esta relación central, el modelo se expande hacia otras entidades funcionales que permiten manejar la trazabilidad de las prácticas. Por ejemplo, la entidad “Schedule_Students” almacena los horarios asignados a cada estudiante dentro de un periodo académico, definiendo los días y horas en los cuales debe realizar sus prácticas. Por otro lado, la entidad “Attendance_Record” registra el ingreso y salida del estudiante, controlando su puntualidad y modalidad de registro (presencial o virtual).

Para el seguimiento de horas de los estudiantes, el modelo incorpora 2 entidades que permiten resumir la información registrada a lo largo de los periodos en los que participan. La entidad “Student_Hours_Summary” consolida el número total de horas de cada estudiante en sus prácticas, incluso cuando estas se extienden por más de un periodo académico. Este

registro considera las asistencias registradas de manera regular y las horas extraordinarias (Extra_Hours) que hayan sido agregadas o descontadas por actividades externas. Su función es controlar el avance general de cada estudiante y verificar que esté cumpliendo con las expectativas establecidas en el plan de prácticas y requisitos institucionales. En cambio, la entidad “Weekly_Hours_Summary” desglosa esta información semanalmente, permitiendo realizar un seguimiento más detallado. Gracias a esta estructura, los coordinadores y administrativos pueden analizar de forma más precisa el rendimiento del estudiante a lo largo del periodo, identificando inasistencias o retrasos frecuentes. Esta información es esencial para que los estudiantes puedan finalizar sus prácticas dentro del plazo establecido y consultar transparentemente su progreso, en caso de que necesiten verificar o aclaras dudas sobre el cumplimiento de sus horas.

Figura 13

Diagrama Entidad-Relación del Módulo Biométrico



Nota: Modelo Entidad-Relación del módulo de control de acceso biométrico con las entidades y relaciones correspondientes. Fuente: Elaboración propia.

3.4. Diseño de la Arquitectura del Sistema

El diseño de la arquitectura del sistema se fundamenta en un modelo de 3 capas, que busca mantener una separación clara entre responsabilidades de presentación, lógica de negocio y datos. Esta estructura facilita la escalabilidad, mantenimiento y organización interna del proyecto.

Al optar por el estilo arquitectónico REST, se garantiza una comunicación clara entre el cliente y el servidor por medio de peticiones HTTP. Gracias a este enfoque, el sistema puede intercambiar información en formato JSON, asegurando interoperabilidad y una integración sencilla con otros sistemas que utilice la universidad.

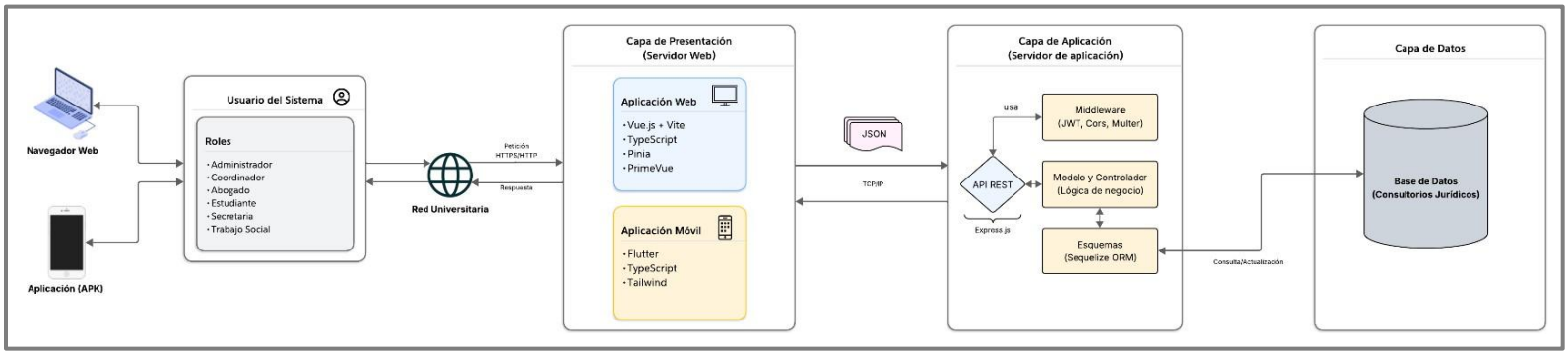
La primera capa corresponde a la Capa de Presentación, compuesta por la aplicación web y la aplicación móvil. La versión web está desarrollada con Vue.js para la gestión de componentes, junto a TypeScript y Pinia para el control de estado global. Para el diseño se utiliza la biblioteca PrimeVue, la cual contiene componentes modernos y personalizables que se usan en la interfaz.

La capa intermedia, denominada como Capa de Aplicación implementa la lógica de negocio a través de una API REST desarrollada en Express.js, utilizando el patrón Modelo-Vista-Controlador (MVC). En esta capa se definen los controladores que gestionan las peticiones HTTP y los esquemas de datos que representan las entidades del sistema. El ORM Sequelize facilita la creación y administración de modelos relacionales, bajo el enfoque code first, es decir, la estructura de la base de datos se genera a partir del código definido en los esquemas del sistema. Esto no implica que el modelo entidad–relación (ERD) no haya sido diseñado previamente; por el contrario, el modelo conceptual y lógico se estableció antes del desarrollo como se vio en el anterior apartado.

Por último, la capa de Datos se encuentra en una base de datos MySQL, en donde se almacena la información de los distintos módulos del sistema. Esta capa garantiza la persistencia e integridad de los datos, permitiendo consultas y actualizaciones ágiles que se realizan desde la capa de aplicación.

Figura 14

Arquitectura del Sistema



Nota: La figura representa la arquitectura del sistema utilizada para realizar el sistema "Balanza Web" y la aplicación móvil. Fuente: Elaborada por el equipo de desarrollo.

IV. CAPÍTULO IV: DESARROLLO DE LA APLICACIÓN

4.1. Configuración del Entorno de Desarrollo

4.1.1. Estándares de Codificación

Los estándares para el desarrollo de este sistema fueron establecidos por la Dirección de Informática (DI), por lo que el código y su nomenclatura están hechos en inglés. Esta decisión permitió mantener coherencia con el resto de los proyectos institucionales y facilitó la colaboración entre el equipo de desarrollo. De igual manera, se aplicaron las prácticas de seguridad que utilizan los sistemas de la universidad, garantizando la protección de datos personales y académicos.

4.1.2. Control de Versiones

Para el control de versiones se utilizó Git como herramienta principal, y GitHub como repositorio en línea para el desarrollo en equipo. El proyecto se dividió en dos repositorios independientes, uno para el frontend y otro para el backend, asignando los permisos de acceso y edición únicamente a los miembros del equipo y la DI. Esto permitió que cada miembro trabaje en su propia rama sin causar conflictos en la rama principal.

Gracias al uso de estas herramientas se pudieron aplicar buenas prácticas de desarrollo colaborativo, garantizando un flujo de trabajo ordenado. El uso de pull requests facilitó la revisión del código, ayudando a detectar errores o inconsistencias dentro del código. Los commits descriptivos mostraban un historial claro del progreso del sistema y las decisiones tomadas durante el desarrollo.

4.2. Implementación de la Funcionalidad

4.2.1. Enfoque de desarrollo y metodología aplicada

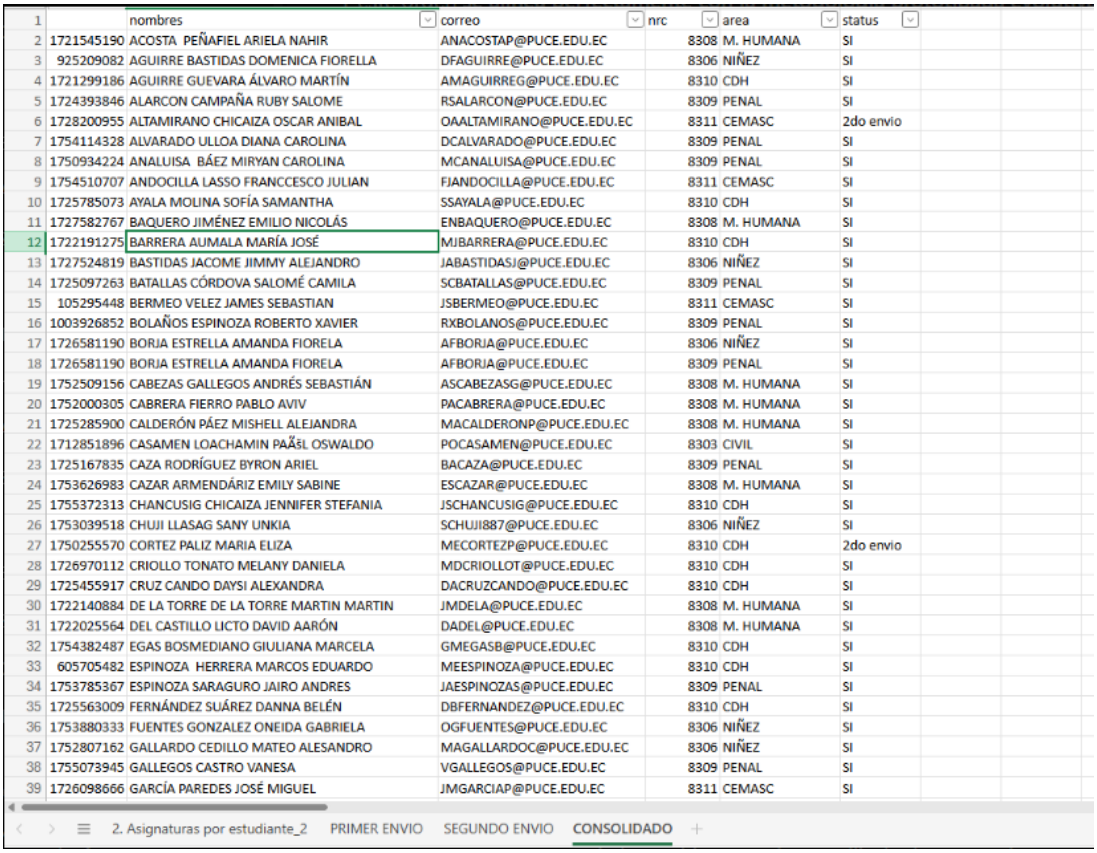
El desarrollo de este módulo tomo como punto de partida los procesos administrativos que la secretaría del Consultorio Jurídico realizaba de forma manual mediante hojas de cálculo en Excel. Esta herramienta se utilizaba para registrar la información de los estudiantes, administrar sus horarios de práctica, calcular las horas semanales y llevar un seguimiento del avance total de cada periodo académico.

Se aplicó la metodología de desarrollo Prototipada Evolutiva por ser iterativa y adaptable a la mayoría de los proyectos de software. Este enfoque permitió construir el sistema en distintas fases sucesivas, partiendo de un prototipo inicial que fue refinado continuamente a partir de la retroalimentación del personal administrativo del lugar. En cada iteración se realizaron mejoras funcionales y visuales, que garantizaron que el prototipo final se ajuste a las necesidades de los usuarios.

El proceso empezó con el levantamiento de información, realizado con la secretaria encargada del control académico. En esta etapa se analizaron los documentos utilizados durante la gestión diaria, identificando los puntos críticos y oportunidades de automatización. La información que se obtuvo fue formalizada mediante un acta de requerimientos funcionales, que sirvió como base para crear la estructura y flujo de este módulo del sistema.

Figura 15

Archivo de consolidado de estudiantes



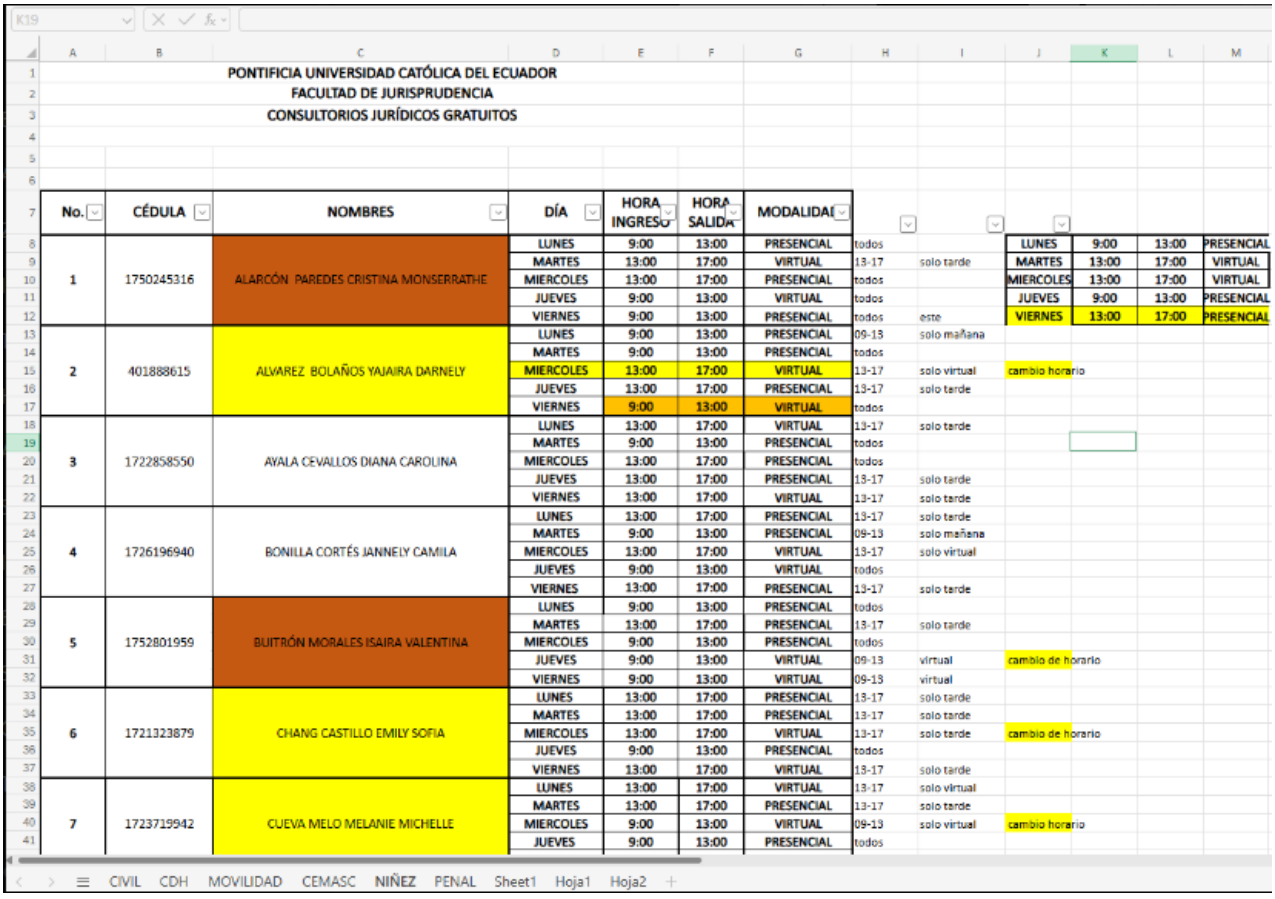
		nombres	correo	nrc	area	status
2	1721545190	ACOSTA PEÑAFIEL ARIELA NAHIR	ANACOSTAP@PUCE.EDU.EC	8308	M. HUMANA	SI
3	925209082	AGUIRRE BASTIDAS DOMENICA FIORELLA	DFAGUIRRE@PUCE.EDU.EC	8306	NIÑEZ	SI
4	1721299186	AGUIRRE GUEVARA ÁLVARO MARTÍN	AMAGUIRREG@PUCE.EDU.EC	8310	CDH	SI
5	1724393846	ALARCON CAMPAÑA RUBY SALOME	RSALARCON@PUCE.EDU.EC	8309	PENAL	SI
6	1728200955	ALTAMIRANO CHICAIZA OSCAR ANIBAL	OAALTAMIRANO@PUCE.EDU.EC	8311	CEMASC	2do envio
7	1754114328	ALVARADO ULLOA DIANA CAROLINA	DCALVARADO@PUCE.EDU.EC	8309	PENAL	SI
8	1750934224	ANALUISA BÁEZ MIRYAN CAROLINA	MCANALUISA@PUCE.EDU.EC	8309	PENAL	SI
9	1754510707	ANDOCILLA LASSO FRANCCESCO JULIAN	FJANDOCILLA@PUCE.EDU.EC	8311	CEMASC	SI
10	1725785073	AYALA MOLINA SOFÍA SAMANTHA	SSAYALA@PUCE.EDU.EC	8310	CDH	SI
11	1727582767	BAQUERO JIMÉNEZ EMILIO NICOLÁS	ENBAQUERO@PUCE.EDU.EC	8308	M. HUMANA	SI
12	1722191275	BARRERA AUMALA MARÍA JOSÉ	MJBARRERA@PUCE.EDU.EC	8310	CDH	SI
13	1727524819	BASTIDAS JACOME JIMMY ALEJANDRO	JABASTIDASJ@PUCE.EDU.EC	8306	NIÑEZ	SI
14	1725097263	BATALLAS CÓRDOVA SALOMÉ CAMILA	SCBATALLAS@PUCE.EDU.EC	8309	PENAL	SI
15	105295448	BERMEO VELEZ JAMES SEBASTIAN	JSBERMEO@PUCE.EDU.EC	8311	CEMASC	SI
16	1003926852	BOLAÑOS ESPINOZA ROBERTO XAVIER	RXBOLANOS@PUCE.EDU.EC	8309	PENAL	SI
17	1726581190	BORJA ESTRELLA AMANDA FIORELA	AFBORJA@PUCE.EDU.EC	8306	NIÑEZ	SI
18	1726581190	BORJA ESTRELLA AMANDA FIORELA	AFBORJA@PUCE.EDU.EC	8309	PENAL	SI
19	1752509156	CABEZAS GALLEGOS ANDRÉS SEBASTIÁN	ASCABEZASG@PUCE.EDU.EC	8308	M. HUMANA	SI
20	1752000305	CABRERA FIERRO PABLO AVIV	PACABRERA@PUCE.EDU.EC	8308	M. HUMANA	SI
21	1725285900	CALDERÓN PÁEZ MISHELL ALEJANDRA	MACALDERONP@PUCE.EDU.EC	8308	M. HUMANA	SI
22	1712851896	CASAMEN LOACHAMIN PAÃšL OSWALDO	POCASAMEN@PUCE.EDU.EC	8303	CIVIL	SI
23	1725167835	CAZA RODRÍGUEZ BYRON ARIEL	BACAZA@PUCE.EDU.EC	8309	PENAL	SI
24	1753626983	CAZAR ARMENDÁRIZ EMILY SABINE	ESCAZAR@PUCE.EDU.EC	8308	M. HUMANA	SI
25	1755372313	CHANCUSIG CHICAIZA JENNIFER STEFANIA	JSCHANCUSIG@PUCE.EDU.EC	8310	CDH	SI
26	1753039518	CHUJI LLASAG SANY UNKIA	SCHUJI887@PUCE.EDU.EC	8306	NIÑEZ	SI
27	1750255570	CORTEZ PALIZ MARIA ELIZA	MECORTEZP@PUCE.EDU.EC	8310	CDH	2do envio
28	1726970112	CRIOLLO TONATO MELANY DANIELA	MDCRIOLLOT@PUCE.EDU.EC	8310	CDH	SI
29	1725455917	CRUZ CANDO DAYSI ALEXANDRA	DACRUZCANDO@PUCE.EDU.EC	8310	CDH	SI
30	1722140884	DE LA TORRE DE LA TORRE MARTIN MARTIN	JMDELA@PUCE.EDU.EC	8308	M. HUMANA	SI
31	1722025564	DEL CASTILLO LICTO DAVID AARÓN	DADEL@PUCE.EDU.EC	8308	M. HUMANA	SI
32	1754382487	EGAS BOSMEDIANO GIULIANA MARCELA	GMEGASB@PUCE.EDU.EC	8310	CDH	SI
33	605705482	ESPINOZA HERRERA MARCOS EDUARDO	MEESPINOZA@PUCE.EDU.EC	8310	CDH	SI
34	1753785367	ESPINOZA SARAGURO JAIRO ANDRES	JAESPINOZAS@PUCE.EDU.EC	8309	PENAL	SI
35	1725563009	FERNÁNDEZ SUÁREZ DANNA BELÉN	DBFERNANDEZ@PUCE.EDU.EC	8310	CDH	SI
36	1753880333	FUENTES GONZALEZ ONEIDA GABRIELA	OGFUENTES@PUCE.EDU.EC	8306	NIÑEZ	SI
37	1752807162	GALLARDO CEDILLO MATEO ALESANDRO	MAGALLARDOC@PUCE.EDU.EC	8306	NIÑEZ	SI
38	1755073945	GALLEGOS CASTRO VANESA	VGALLEGOS@PUCE.EDU.EC	8309	PENAL	SI
39	1726098666	GARCÍA PAREDES JOSÉ MIGUEL	JMGARCIAP@PUCE.EDU.EC	8311	CEMASC	SI

2. Asignaturas por estudiante_2 | PRIMER ENVIO | SEGUNDO ENVIO | CONSOLIDADO

Nota: Figura que muestra el archivo que contiene la información de los estudiantes de derecho matriculados a Prácticas Preprofesionales en los Consultorios Jurídicos. Fuente: Elaborado por la secretaria de los consultorios.

Figura 16

Archivo de horarios de prácticas



PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
FACULTAD DE JURISPRUDENCIA
CONSULTORIOS JURÍDICOS GRATUITOS

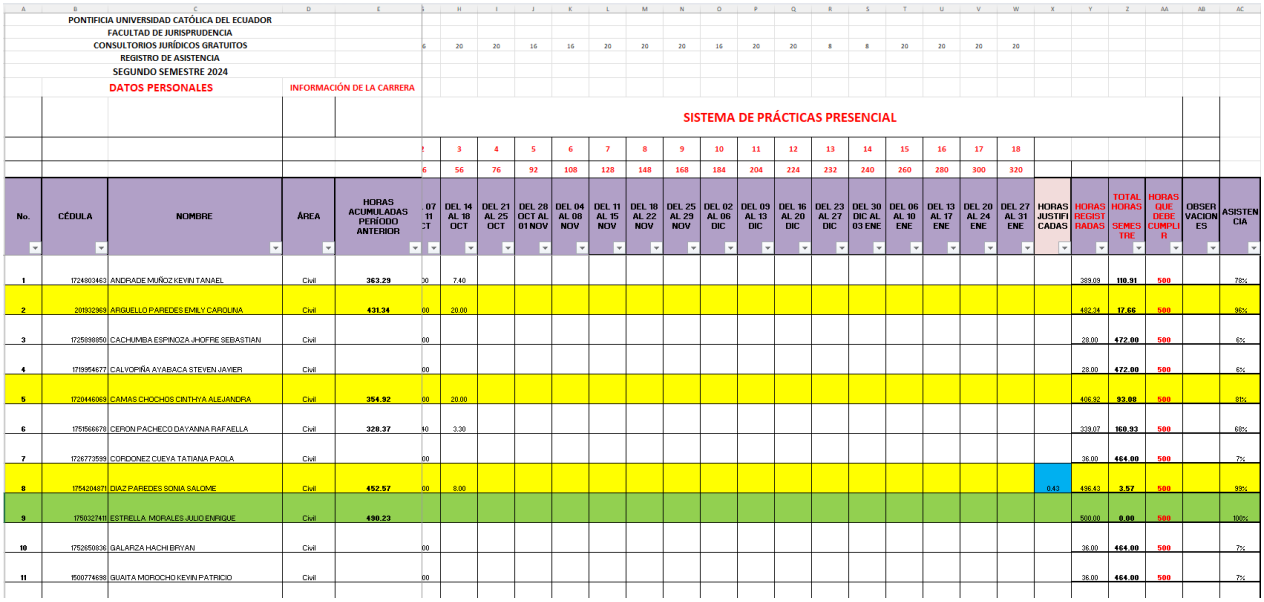
No.	CÉDULA	NOMBRES	DÍA	HORA INGRESO	HORA SALIDA	MODALIDAD						
1	1750245316	ALARCÓN PAREDES CRISTINA MONSERRATHE	LUNES	9:00	13:00	PRESENCIAL	todos		LUNES	9:00	13:00	PRESENCIAL
			MARTES	13:00	17:00	VIRTUAL	13-17	solo tarde	MARTES	13:00	17:00	VIRTUAL
			MIERCOLES	13:00	17:00	PRESENCIAL	todos		MIERCOLES	13:00	17:00	VIRTUAL
			JUEVES	9:00	13:00	VIRTUAL	todos		JUEVES	9:00	13:00	PRESENCIAL
			VIERNES	9:00	13:00	PRESENCIAL	todos	este	VIERNES	13:00	17:00	PRESENCIAL
2	401888615	ALVAREZ BOLAÑOS YAJAIRA DARNELY	LUNES	9:00	13:00	PRESENCIAL	09-13	solo mañana				
			MARTES	9:00	13:00	PRESENCIAL	todos					
			MIERCOLES	13:00	17:00	VIRTUAL	13-17	solo virtual	cambio horario			
			JUEVES	13:00	17:00	PRESENCIAL	13-17	solo tarde				
			VIERNES	9:00	13:00	VIRTUAL	todos					
3	1722858550	AYALA CEVALLOS DIANA CAROLINA	LUNES	13:00	17:00	VIRTUAL	13-17	solo tarde				
			MARTES	9:00	13:00	PRESENCIAL	todos					
			MIERCOLES	13:00	17:00	PRESENCIAL	todos					
			JUEVES	13:00	17:00	PRESENCIAL	13-17	solo tarde				
			VIERNES	13:00	17:00	VIRTUAL	13-17	solo tarde				
4	1726196940	BONILLA CORTÉS JANNELY CAMILA	LUNES	13:00	17:00	PRESENCIAL	13-17	solo tarde				
			MARTES	9:00	13:00	PRESENCIAL	09-13	solo mañana				
			MIERCOLES	13:00	17:00	VIRTUAL	13-17	solo virtual				
			JUEVES	9:00	13:00	VIRTUAL	todos					
			VIERNES	13:00	17:00	PRESENCIAL	13-17	solo tarde				
5	1752801959	BUITRÓN MORALES ISAIRA VALENTINA	LUNES	9:00	13:00	PRESENCIAL	todos					
			MARTES	13:00	17:00	PRESENCIAL	13-17	solo tarde				
			MIERCOLES	9:00	13:00	PRESENCIAL	todos					
			JUEVES	9:00	13:00	VIRTUAL	09-13	virtual	cambio de horario			
			VIERNES	9:00	13:00	VIRTUAL	09-13	virtual				
6	1721323879	CHANG CASTILLO EMILY SOFIA	LUNES	13:00	17:00	PRESENCIAL	13-17	solo tarde				
			MARTES	13:00	17:00	PRESENCIAL	13-17	solo tarde				
			MIERCOLES	13:00	17:00	VIRTUAL	13-17	solo tarde	cambio de horario			
			JUEVES	9:00	13:00	PRESENCIAL	todos					
			VIERNES	13:00	17:00	VIRTUAL	13-17	solo tarde				
7	1723719942	CUEVA MELO MELANIE MICHELLE	LUNES	13:00	17:00	VIRTUAL	13-17	solo virtual				
			MARTES	13:00	17:00	PRESENCIAL	13-17	solo tarde				
			MIERCOLES	9:00	13:00	VIRTUAL	09-13	solo virtual	cambio horario			
			JUEVES	9:00	13:00	PRESENCIAL	todos					

CIVIL CDH MOVILIDAD CEMASC NIÑEZ PENAL Sheet1 Hoja1 Hoja2

Nota: Figura que muestra el archivo que contiene la información de los estudiantes de derecho matriculados a Prácticas Preprofesionales en los Consultorios Jurídicos. Fuente: Elaborado por la secretaria de los consultorios.

Figura 17

Archivo de Seguimiento General



PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
FACULTAD DE JURISPRUDENCIA
CONSULTORIOS JURÍDICOS GRATUITOS
REGISTRO DE ASISTENCIA
SEGUNDO SEMESTRE 2024

DATOS PERSONALES | INFORMACIÓN DE LA CARRERA | SISTEMA DE PRÁCTICAS PRESENCIAL

					6	20	20	16	16	20	20	20	16	20	20	8	8	20	20	20	20							
					6	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18							
					6	56	76	92	108	128	148	168	184	204	224	232	240	260	280	300	320							
No.	CÉDULA	NOMBRE	ÁREA	HORAS ACUMULADAS PERÍODO ANTERIOR	07 11 CT	DEL 14 AL 18 OCT	DEL 21 AL 25 OCT	DEL 28 OCT AL 01 NOV	DEL 04 AL 08 NOV	DEL 11 AL 15 NOV	DEL 18 AL 22 NOV	DEL 25 AL 29 NOV	DEL 02 AL 06 DIC	DEL 09 AL 13 DIC	DEL 16 AL 20 DIC	DEL 23 AL 27 DIC	DEL 30 DIC AL 03 ENE	DEL 06 AL 10 ENE	DEL 13 AL 17 ENE	DEL 20 AL 24 ENE	DEL 27 AL 31 ENE	HORAS JUSTIFICADAS	HORAS REGISTRADAS	TOTAL HORAS SEMESTRE	HORAS QUE DEBE CUMPLIR	OBSERVACIONES	ASISTENCIA
1	1724803463	ANDRADE MUÑOZ KEVIN TANAEL	Civil	363.29	00	7.40																	389.09	110.91	500		78%
2	2019322969	ARGUELLO PAREDES EMILY CAROLINA	Civil	431.34	00	20.00																	482.34	17.66	500		96%
3	1725898850	CACHUMBA ESPINOZA JHOFRE SEBASTIAN	Civil		00																		28.00	472.00	500		6%
4	1719954677	CALVOPIÑA AYABACA STEVEN JAVIER	Civil		00																		28.00	472.00	500		6%
5	1720446069	CAMAS CHOCHOS CINTHYA ALEJANDRA	Civil	354.92	00	20.00																	406.92	93.08	500		81%
6	1751566678	CERON PACHECO DAYANNA RAFAELLA	Civil	328.37	00	3.30																	339.67	160.33	500		68%
7	1726773599	CORDONEZ CUEVA TATIANA PAOLA	Civil		00																		36.00	464.00	500		7%
8	1754204871	DIAZ PAREDES SONIA SALOME	Civil	452.57	00	8.00																0.43	496.43	3.57	500		99%
9	1750327411	ESTRELLA MORALES JULIO ENRIQUE	Civil	490.23																			500.00	0.00	500		100%
10	1752650936	GALARZA HACHI BRYAN	Civil		00																		36.00	464.00	500		7%
11	1500774638	GUAITA MOROCHO KEVIN PATRICIO	Civil		00																		36.00	464.00	500		7%

Nota: Figura que muestra el archivo que contiene el seguimiento general de los estudiantes en sus prácticas. Fuente: Elaborado por la secretaria de los consultorios.

4.2.2. Desarrollo del Backend

4.2.2.1. Arquitectura

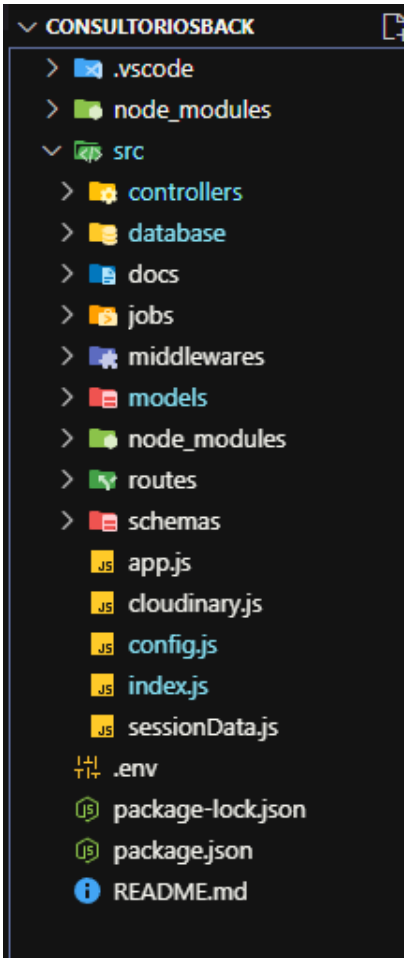
El desarrollo del backend se basó en una adaptación del patrón arquitectónico MVC. En este caso se agregó una subestructura con esquemas definidos mediante Sequelize ORM, que permiten modelar las entidades de la base de datos y establecer sus relaciones sin crearlas manualmente en SQL. Adaptar este enfoque otorgó mayor flexibilidad para la programación, facilitando el trabajo con datos estructurados y asegurando coherencia en la base de datos.

El directorio principal del proyecto (src) está organizado en módulos. En la carpeta schemas, se encuentran las entidades con sus atributos y restricciones correspondientes. Sirven como base para que Sequelize genere y sincronice automáticamente las tablas de la base de datos. Por otra parte, los modelos son una capa intermedia entre los esquemas y controladores. Aquí se manejan las operaciones CRUD que contienen la lógica de negocio y validación de datos. Dentro de la misma estructura, se encuentra la carpeta controllers, la cual agrupa los controladores que procesan las solicitudes HTTP que entran desde las rutas. Cada controlador se encarga de coordinar las acciones necesarias para responder a las peticiones del cliente, ejecutando las funciones que se definen en los modelos y devolviendo respuestas en formato JSON. Las rutas actúan como el punto de entrada al backend, permitiendo organizar los distintos endpoints de la API según su funcionalidad o módulo correspondiente.

Finalmente, se crearon carpetas adicionales para el manejo del servidor. Entre ellas, middlewares, para los procesos de autenticación a través de JWT, validaciones intermedias, control de accesos, y configuraciones de la base de datos.

Figura 18

Estructura Completa del BackEnd



Nota: Estructura Completa de Carpetas del BackEnd desde el directorio del proyecto. Fuente: Elaboración propia.

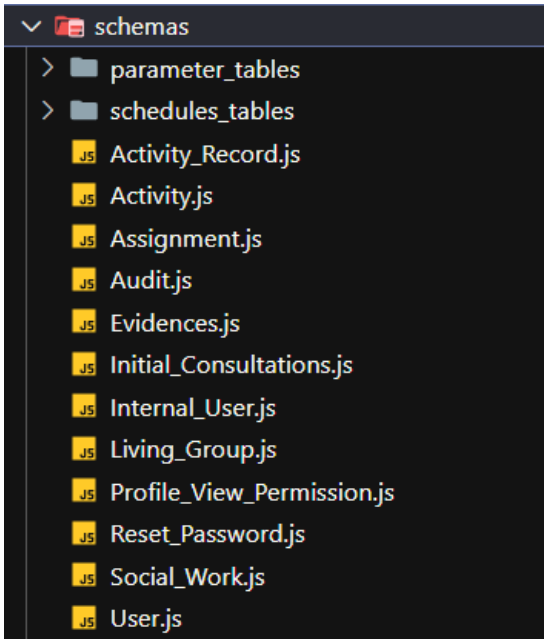
4.2.2.2. Conexión con la Base de Datos

La conexión a la base de datos se realizó a través del ORM Sequelize, el cual vincula automáticamente el servidor con la base de datos de forma eficiente. Durante la configuración del entorno, se definieron los parámetros para que el servidor pudiera conectarse a la base de datos local utilizada durante el desarrollo. La configuración incluye el nombre de la base, credenciales de accesos y zona horaria.

Al iniciar el backend, Sequelize crea la conexión y valida la estructura de las tablas definidas en los esquemas, garantizando que no existan inconsistencias y los modelos estén sincronizados. Esta conexión es persistente y estable, permitiendo realizar operaciones de lectura, escritura y actualización de datos de forma instantánea. Además, el ORM incorpora mecanismos que gestionan errores de conexión automáticamente durante la ejecución del servidor.

Figura 19

Estructura de esquemas del proyecto



Nota: Estructura de carpetas de los esquemas que contienen las entidades de la base de datos. Fuente: Elaboración propia.

4.2.2.3. Implementación de Modelos y Controladores en el Servidor

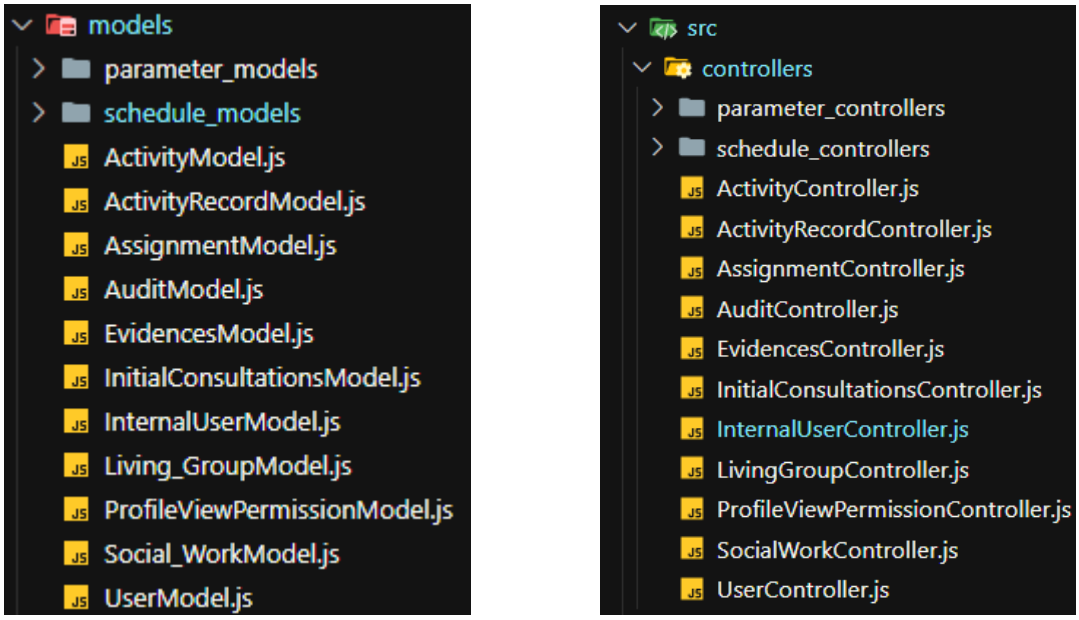
En el backend, los modelos y controladores son los encargados de hacer que los esquemas tengan funcionalidad al administrar la lógica de negocio de todo el sistema. Están encargados de gestionar la información y el flujo de interacción entre la base de datos y solicitudes que llegan desde el lado del cliente.

Los modelos gestionan las operaciones CRUD de las entidades definidas en los esquemas. En este módulo los modelos manejan procesos esenciales como el control de periodos académicos, registro de estudiantes, asignación de horarios y seguimiento de horas realizadas. Estas operaciones realizadas sobre las entidades son el resultado directo de los requerimientos funcionales anteriormente definidos, que fueron traducidos en modelos dinámicos capaces de interactuar con la base de datos.

Por otra parte, los controladores funcionan como conectores entre estas funcionalidades y las solicitudes que realiza el cliente. Se encargan de recibir las peticiones HTTP, procesar la información enviada y devolver una respuesta estructurada al frontend. Esta capa asegura una comunicación fluida dentro del sistema, manteniendo coherencia entre las transacciones y permitiendo que los procesos que se realizan en el modelo se ejecuten de manera segura.

Figura 20

Estructura de Modelos y Controladores del Proyecto



Nota: Estructura de carpetas de los modelos y controladores existentes dentro del proyecto. Fuente: Elaboración propia.

4.2.2.4. Definición de rutas y servicios REST

Las rutas son el punto de acceso a los servicios que administra el backend, actuando como la interfaz de comunicación entre el cliente y servidor. Estas rutas dirigen las peticiones recibidas hacia el controlador correspondiente, el cual tiene la responsabilidad de procesar la información, aplicar reglas de negocio en su respectivo modelo y devolver una respuesta en un formato estructurado hacia el frontend.

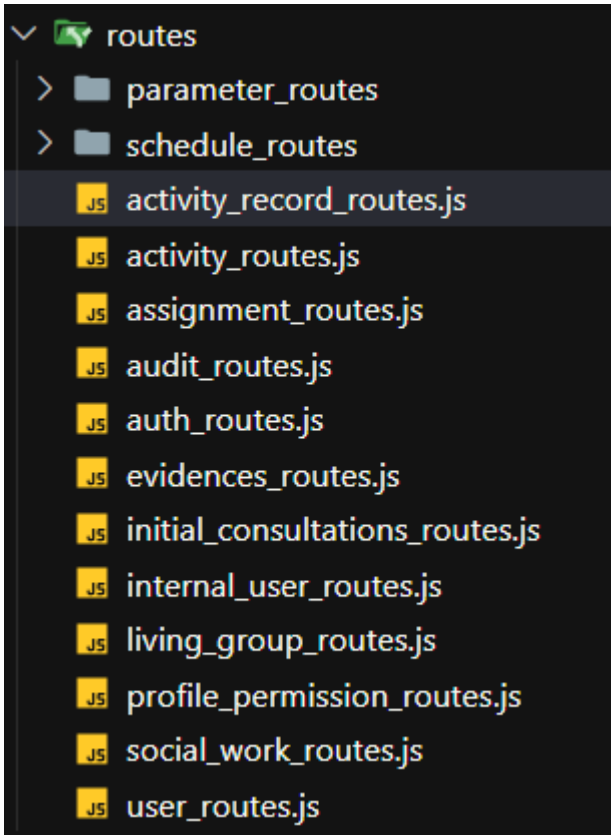
En todos los módulos, las rutas fueron organizadas acorde a los servicios que ofrecen, agrupando funcionalidades con su respectiva entidad del sistema. Gracias a esta estructura fue sencillo mantener trazabilidad en las operaciones realizadas, ya que cada conjunto de rutas corresponde a un modelo y controlador específico.

Adicionalmente, los endpoints fueron clasificados en públicos y privados, de acuerdo con el nivel de acceso requerido. Las rutas públicas están destinadas a procesos de autenticación y recuperación de credenciales, mientras que las rutas privadas contienen la funcionalidad principal del sistema, abarcando la mayoría de los módulos existentes.

Todas las rutas están protegidas por medio de middlewares de autenticación que verifican la validez del token de sesión JWT antes de permitir el acceso. Esta medida permite que solo los usuarios autorizados pueden acceder a los servicios del sistema, garantizando la seguridad y los lineamientos establecidos por la DI.

Figura 21

Estructura de Rutas del Proyecto



Nota: Estructura de carpetas de las rutas existentes dentro del proyecto. Fuente: Elaboración propia.

4.2.3. Desarrollo del FrontEnd

4.2.3.1. Estructura del FrontEnd

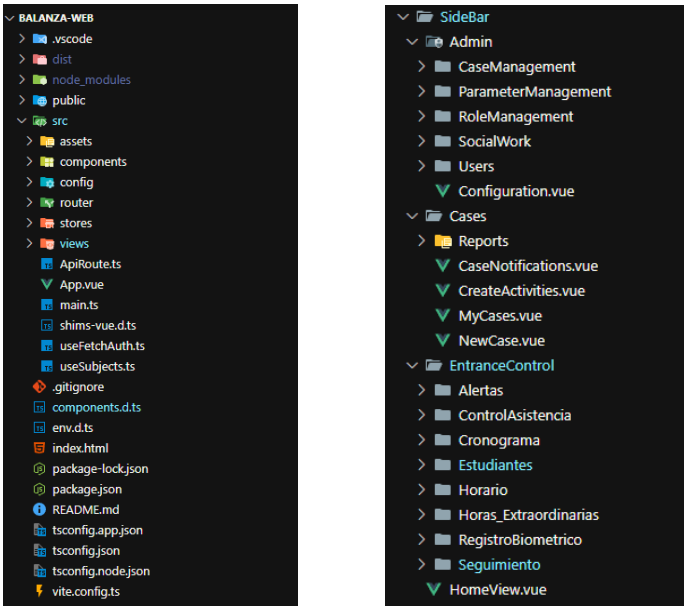
Vue es un framework que permite un enfoque modular basado en componentes, donde cada parte de la interfaz es reutilizable. Dentro de este proyecto, la carpeta principal *src* maneja toda la lógica visual y estructural de la capa de presentación. En ella se encuentran las carpetas *assets* para el uso de recursos estáticos como íconos, imágenes o estilos globales, *components* para los elementos reutilizables de la interfaz, *router* donde se definen las rutas de navegación y permisos de accesos, y *stores*, destinada al manejo de estados globales con Pinia.

El archivo *ApiRoute.ts* contiene la configuración base para conectarse a la API y comunicarse con el backend, definiendo la dirección principal del servidor. Para su implementación se aplicaron buenas prácticas de TypeScript, incluyendo interfaces y tipado estático que aseguran consistencia en los datos que se envían y reciben entre cliente y servidor.

Finalmente, en la carpeta *views,* se organizan las vistas de la aplicación con base en los módulos que la componen. La subcarpeta *Auth* contiene las dos únicas pantallas accesibles sin autenticación las cuales son el inicio de sesión y recuperación de contraseña. Por otro lado, *SideBar* agrupa las secciones internas visibles únicamente tras realizar la autenticación. Dentro de esta se encuentran los módulos *Admin,* orientado a la administración de usuarios y configuración general, *Cases,* encargado de la gestión de primeras consultas y casos legales por área de derecho, y *EntranceControl,* donde se hallan las vistas pertenecientes al módulo de control de estudiantes y registro de asistencia.

Figura 22

Estructura de Carpetas del FrontEnd



Nota: Figura que contiene la estructura de carpetas del FrontEnd, donde se pueden ver la organización modular del directorio principal src. Fuente: Elaboración propia.

4.2.3.2. Comunicación del FrontEnd con el BackEnd

Como se mencionó anteriormente, el acceso a las rutas del backend se gestiona desde el archivo “ApiRoute.ts”, el cual contiene la dirección principal del servidor y define la constante API, encargada de almacenar la URL que permite acceder a las rutas del servidor. Gracias a esta estructura se puede mantener un control único sobre la conexión con la API, facilitando la modificación de la URL del servidor en caso de migración o despliegue en producción, sin cambiar las rutas en cada vista o componente. En la práctica, todas las operaciones del frontend apuntan a esta constante, concatenando únicamente la parte correspondiente a cada endpoint.

Para la comunicación entre frontend y backend, se utilizó la librería Axios, debido a su claridad y facilidad en el control de errores frente a la API nativa Fetch. Axios permite realizar peticiones HTTP de manera más intuitiva al permitir la configuración de encabezados, tokens de autenticación y parámetros sin repetir código innecesario. En adición, su soporte nativo para interceptores ayudó a automatizar tareas como el envío de credenciales.

Todas las solicitudes realizadas hacia la API se realizaron en formato JSON, con el objetivo de garantizar un intercambio estructurado de datos entre el cliente y servidor. De esta forma, se garantiza un flujo de comunicación coherente que reduce la posibilidad de errores de interpretación entre ambas partes del sistema.

Ejemplo de solicitud hacia API:

Método HTTP: POST

URI del servicio: /api/registros

Funcionalidad: Registrar asistencia

Parámetros enviados (Request Body-JSON)

```
{
  "UserXPeriod_ID": "UP_1023",
  "Attendance_Type": "Presencial",
  "Attendance_Entry": "2025-11-24T08:58:00",
  "Attendance_Late": false,
  "Attendance_Comment": null,
  "Attendance_IsDeleted": false
}
```

Formato de respuesta exitosa:


```
{
  "message": "Registro de asistencia creado correctamente",
  "Attendance_ID": 4587,
  "Attendance_Date": "2025-11-24T08:58:00",
  "Attendance_Status": "Activo"
}
```


4.2.4. Diseño e implementación de la interfaz de usuario (UI)

Para comprender la disposición de elementos dentro de la interfaz es necesario entender la estructura base sobre la que se diseñó el sistema. Se utilizó un contenedor (layout) principal llamado “Default Layout”, el cual define la organización general de la interfaz y permanece visible en la mayoría de las vistas del sistema. Este layout incluye un menú lateral (sidebar) persistente que se ubica al lado izquierdo, el cual permite acceder a las diferentes secciones sin necesidad de refrescar la página. De esta forma, solo cambia el contenido central según la opción del menú seleccionada, manteniendo una navegación fluida.

Por el contrario, el AuthLayout está destinado únicamente a las vistas de autenticación, como el inicio de sesión o recuperación de contraseña. A diferencia del layout principal, este diseño no utiliza el sidebar y otros elementos secundarios, ofreciendo una vista más limpia y enfocada en los procesos de validación de credenciales. Esta separación logró que las secciones privadas y públicas del sistema sean consistentes y estén enfocadas en lo que desean transmitir.

La navegación y control de acceso se gestionaron por medio del enrutador de Vue.js junto al manejo de estado global a través de Pinia, definido como authStore. Esto permitió definir reglas dinámicas según el rol del usuario, asegurando que cada perfil visualice

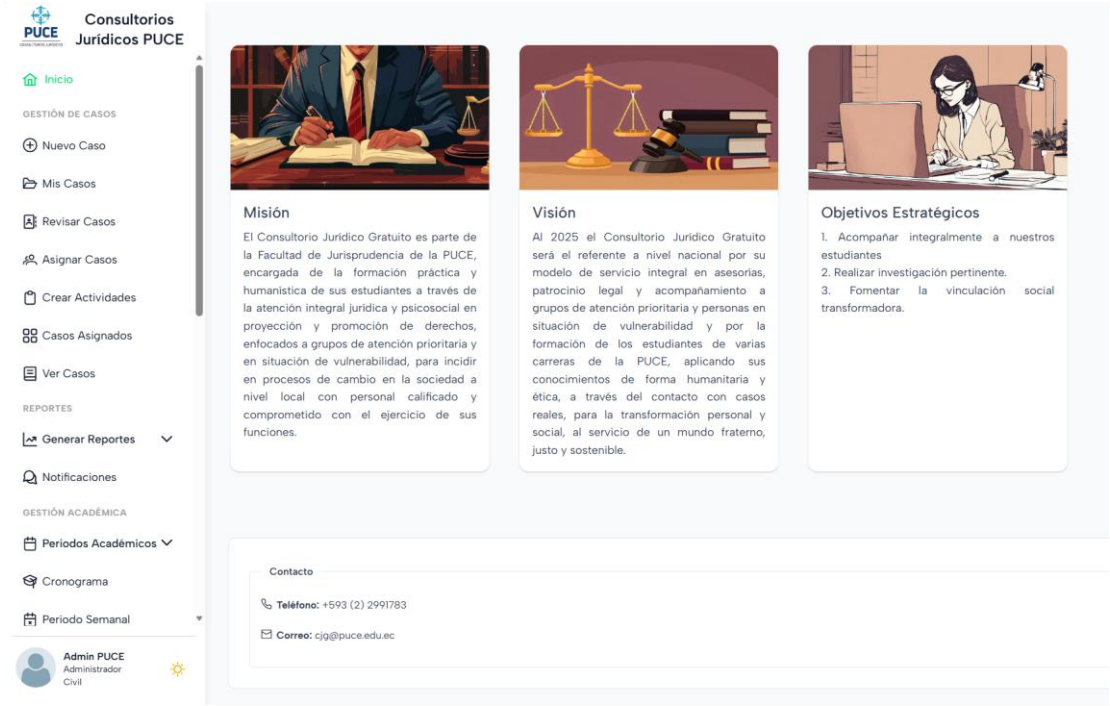
únicamente las secciones que le corresponden a sus permisos. Como resultado de esto se mejoró la seguridad y usabilidad al presentar una interfaz con secciones personalizadas para cada tipo de usuario.

En cuanto al diseño visual, se respetaron los estándares gráficos establecidos por la DI, manteniendo uniformidad con los demás sistemas institucionales, tipografías legibles y un estilo visual minimalista, lo que permitió manejar una estética profesional y funcional. Además, la interfaz cuenta con un modo oscuro, el cual brinda un entorno visual más cómodo para los usuarios que tienen largas jornadas laborales utilizando un computador.

La página de inicio muestra los elementos que reflejan la identidad de los Consultorios Jurídicos de la PUCE, incluyendo su misión, visión y objetivos estratégicos. Este enfoque de diseño unificado hizo que todos los módulos mantuvieran una apariencia coherente y una navegación intuitiva.

Figura 23

Página de inicio del sistema



Nota: Figura que contiene la página inicial del sistema Balanza Web y muestra la misión, visión, objetivos estratégicos y datos de contacto para los Consultorios Jurídicos de la PUCE. Fuente: Elaborado por el equipo de desarrollo del sistema.

4.2.5. Configuración del lector biométrico Secugen Hamster Plus IV

4.2.5.1. Instalación y Configuración

Para habilitar la captura de huellas dactilares dentro del sistema, fue necesario configurar el dispositivo biométrico instalando el controlador oficial proporcionado por Secugen. Este componente permite que el sistema operativo reconozca correctamente el dispositivo.

En adición, se instaló la *Secugen Web Api*, un servicio que se ejecuta en segundo plano y habilita la lectura de huellas del lector. Gracias a esta instalación previa, el sistema web puede solicitar al equipo la captura de una huella de forma segura y estandarizada.

Figura 24

Características del Lector Biométrico

Nota: Figura que muestra el lector de huellas dactilares SecuGen Hamster Plus IV, un dispositivo óptico certificado por el FBI para uso biométrico. Destaca por su alta resolución (500 DPI), capacidad de lectura en condiciones difíciles (dedos secos, húmedos o dañados) y su diseño compacto y resistente. Fuente: Elaboración propia.

4.2.5.2. Integración del lector biométrico en la aplicación web

Una vez instalada, la WebAPI permite que el sistema se comunique con el lector usando peticiones locales. El manual explica que la WebAPI expone servicios específicos para capturar huellas y obtener la información en formato JSON, lo que facilita su uso en cualquier sistema moderno.

Esta integración permite que, al seleccionar un estudiante, el sistema active la captura biométrica. Si la lectura es satisfactoria, se guarda la huella, caso contrario se puede realizar una recaptura y obtener los datos biométricos correspondientes. La huella nunca se envía ni almacena como una imagen, sino como un código biométrico cifrado generado por el mismo servicio. Esto evita que terceros puedan reconstruir u obtener la huella real. Por lo que, la captura se realiza dentro de un entorno totalmente controlado y autorizado por el fabricante, minimizando cualquier riesgo de manipulación de datos sensibles.

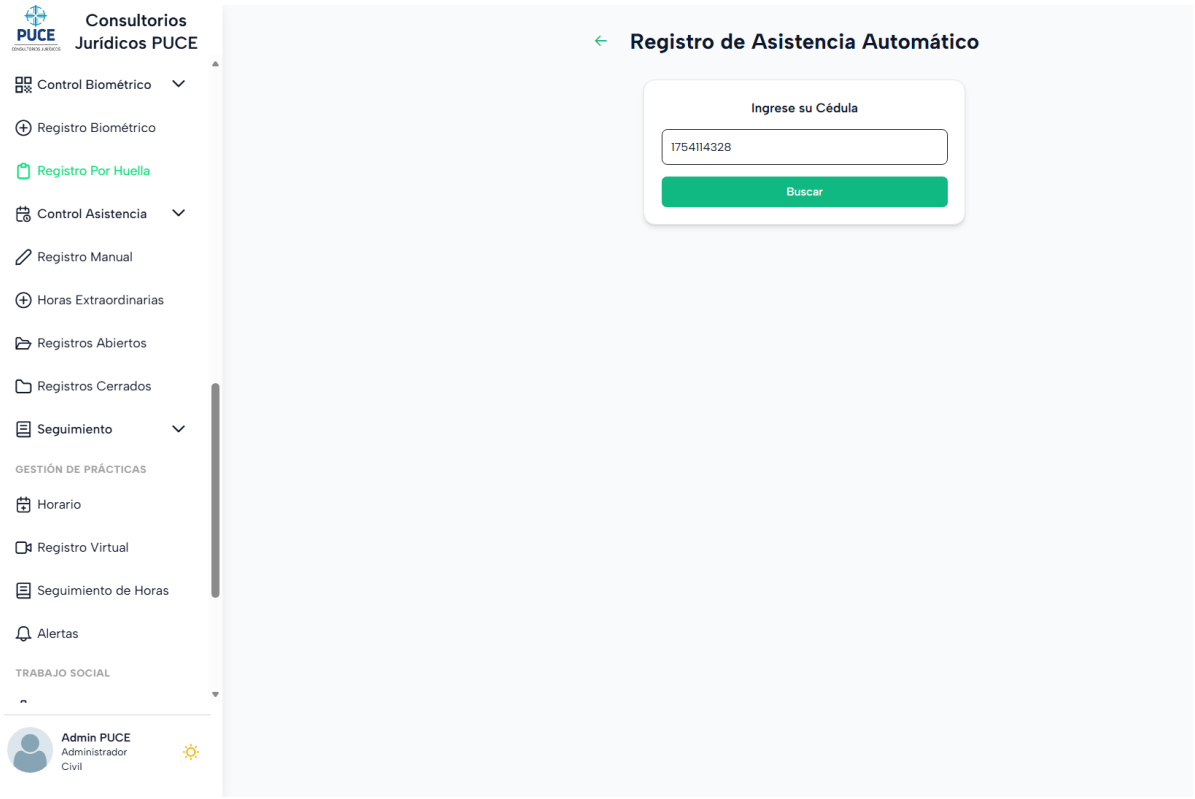
V. Capítulo V: Pruebas del Sistema

5.1. Validación del caso de uso F5.1.1: Registro de Asistencia por Huella

Una vez registrada la huella, en la sección de Registro Biométrico, el administrador puede registrar la asistencia ingresando la cédula del estudiante, donde se valida su huella y horario para determinar si el registro corresponde a una entrada o salida. El sistema asegura que cada registro sea válido y esté acorde a las reglas académicas del proceso de prácticas.

Figura 25

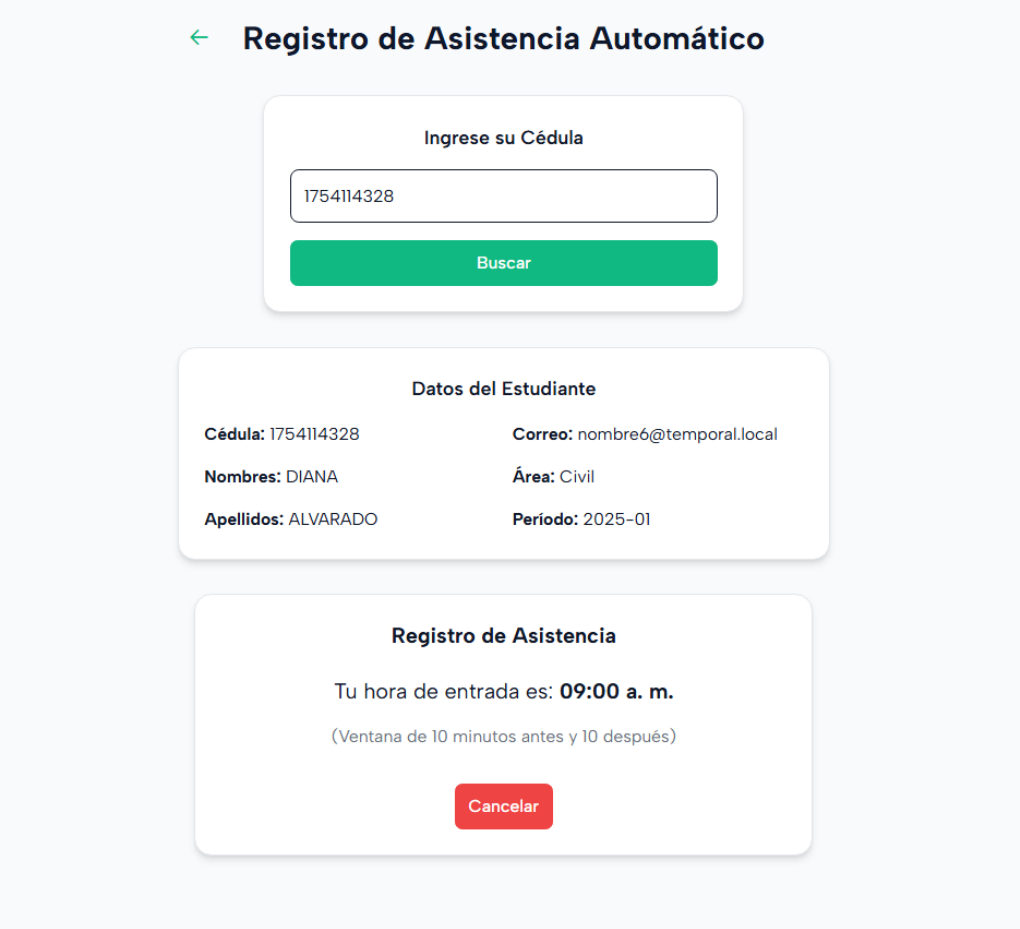
Pantalla de búsqueda del estudiante por número de cédula



Nota: La figura muestra el inicio del proceso de registro, donde la secretaria ingresa el número de cédula del estudiante. El sistema valida la existencia del usuario, su estado activo y la asignación de un horario dentro del período académico vigente. Fuente: Elaboración propia.

Figura 26

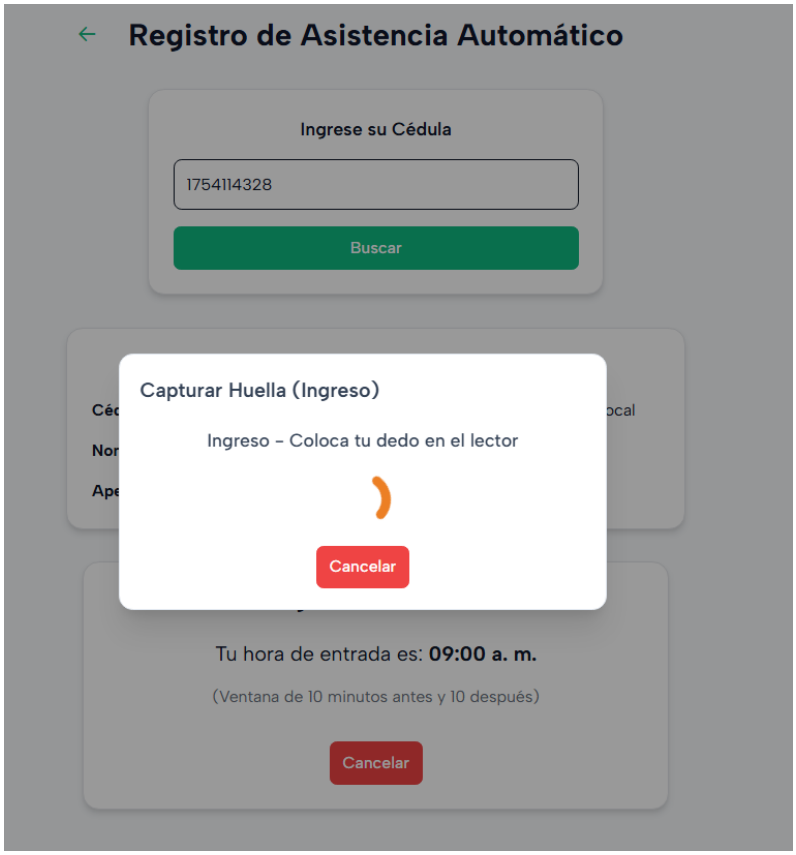
Visualización de datos del estudiante y horario asignado



Nota: Una vez validado el estudiante, el sistema presenta la información personal y el horario correspondiente al día actual, confirmando que el registro se encuentra dentro del rango permitido para su ejecución. Fuente: Elaboración propia.

Figura 27

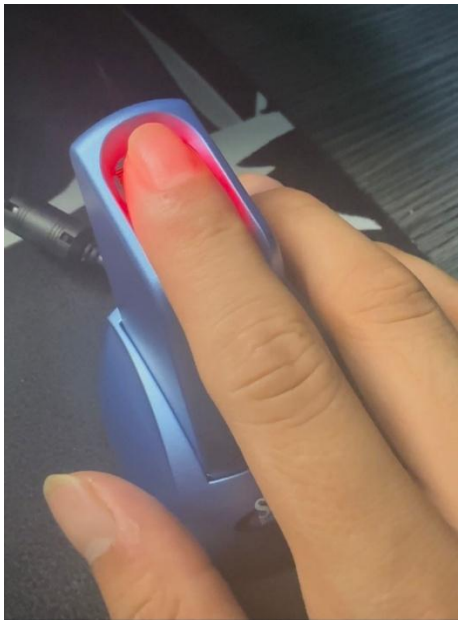
Activación del módulo de captura biométrica



Nota: El sistema activa el lector biométrico y muestra el mensaje de captura de huella, cumpliendo con el paso de autenticación mediante huella dactilar. Fuente: Elaboración propia.

Figura 28

Captura de huella mediante lector biométrico

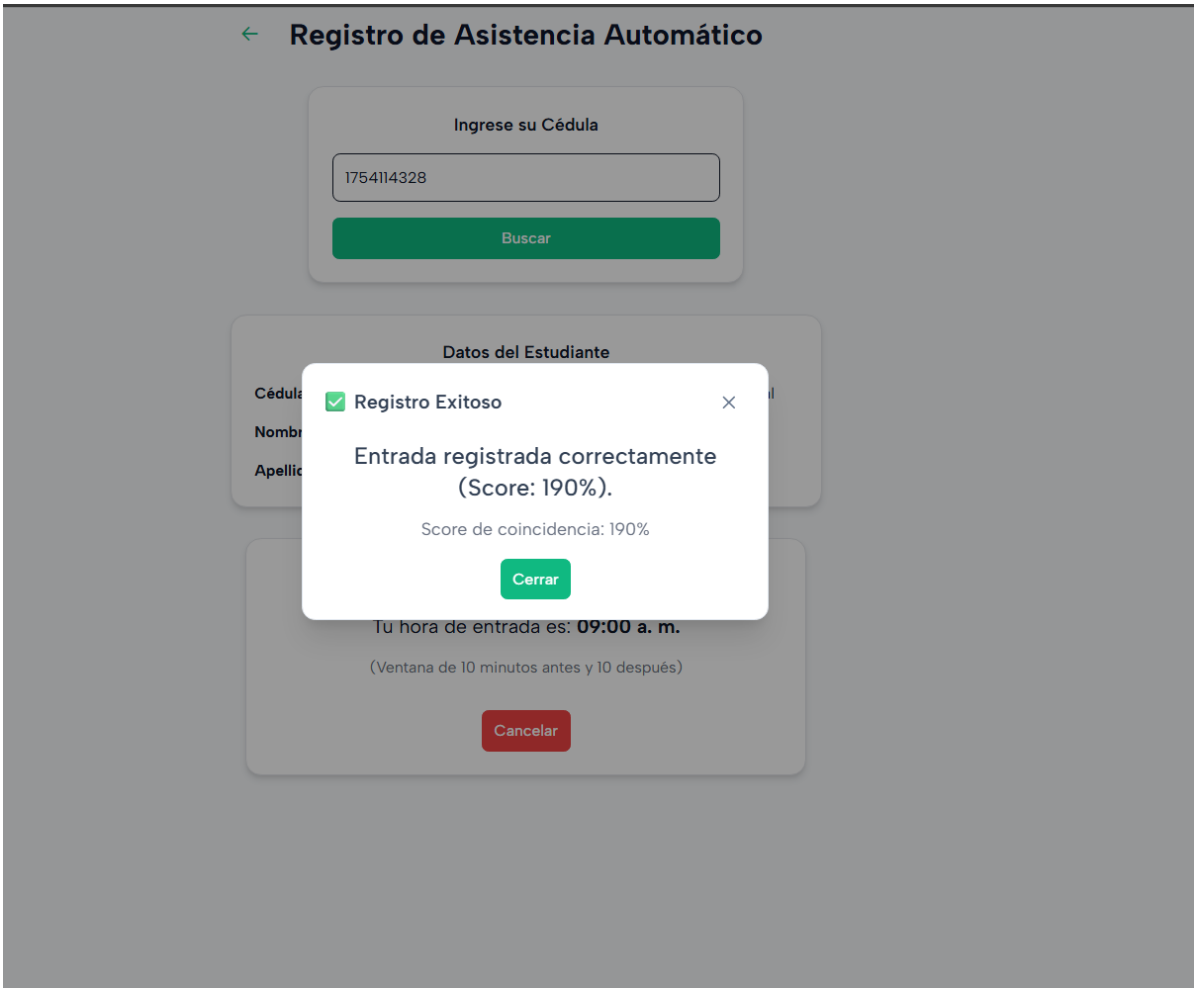


Nota: Figura que muestra cómo el lector biométrico captura la huella. El color rojo visible corresponde al infrarrojo del sensor, y se activa al detectar el contacto con el dedo.

Fuente: Elaboración propia.

Figura 29

Confirmación del registro de Asistencia



Nota: Figura que muestra un mensaje de confirmación, garantizando que la asistencia ha sido almacenada correctamente. La verificación biométrica se comprueba mediante un umbral mínimo del 70 % de coincidencia para aceptar la autenticidad de la huella. Fuente: Elaboración propia.

Las evidencias presentadas demuestran que el sistema cumple con lo descrito en el caso de uso F5.1.1, guardando correctamente la asistencia del estudiante a través de la verificación de sus datos y la captura de su huella dactilar.

VI. CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

- El módulo desarrollado ha logrado optimizar la gestión de prácticas en el Consultorio Jurídico de la PUCE al reemplazar los procesos manuales por un sistema automatizado, lo que garantiza mayor seguridad e integridad en la información. Esta nueva transición hace que el proceso sea más eficiente, reduciendo tiempos administrativos y asegurando un control total sobre la información.
- La implementación del control biométrico mediante huella dactilar hizo que el control de acceso de los estudiantes sea mucho más ágil. Al asegurar que la asistencia sea registrada únicamente por el practicante presente, el sistema elimina errores humanos y registros fraudulentos.
- La gestión de horarios estudiantiles ha permitido mejorar la planificación de las prácticas en cada periodo, permitiendo validar con exactitud el cumplimiento de las horas asignadas a los estudiantes. El sistema confirma que el practicante desempeñe su trabajo en los tiempos establecidos.
- Este módulo permite generar estadísticas sobre la asistencia de los estudiantes, facilitando la evaluación de su desempeño y el control de cumplimiento de las actividades realizadas en los consultorios.

6.2. Recomendaciones

- Se recomienda adaptar el sistema a formatos responsive, permitiendo su uso en cualquier tipo de dispositivo portátil, lo cual facilitaría el acceso en cualquier momento y lugar.
- Se sugiere que la DI evalúe a futuro la apertura del sistema en entornos externos a la red interna de la universidad, especialmente para el control de prácticas virtuales. Para garantizar un acceso seguro desde el exterior, esta implementación debería incorporar mecanismos de autenticación de doble factor o el uso de códigos temporales de un solo uso, asegurando que la plataforma sea accesible únicamente para el personal administrativo y estudiantil.
- Se recomienda que la DI planifique un mantenimiento periódico del módulo, considerando las tecnologías utilizadas en su desarrollo. Esto con el fin de asegurar que el sistema funcione correctamente y pueda escalar a futuro.

BIBLIOGRAFÍA

Albor, S. (2021). Autenticación de usuarios utilizando Biometría. http://bibliotecadigital.econ.uba.ar/download/tpos/1502-2210_AlborSE.pdf

Baño Freddy, Chingo, W., & Viscaino, F. (2016). Herramienta de Mapeo Objeto Relacional y la Productividad de la Empresa Interfases Software Group. https://revistas.utb.edu.ec/index.php/sr/article/view/88

Barragán, A. (2021). Qué es Vue JS y qué lo diferencia de otros frameworks | OpenWebinars. https://openwebinars.net/blog/que-es-vue-js-y-que-lo-diferencia-de-otros-frameworks/

Castillo, L. (2017). Conociendo GitHub Documentation Release 0.1. https://conociendogithub.readthedocs.io/_/downloads/en/latest/pdf/

Durante, R., Quijano, P., González, L., & Montes, N. (2010). Sistemas para el Control de Versiones. https://rodin.uca.es/handle/10498/9785

Eguíluz Pérez, J. (2008). Introducción a JavaScript. www.librosweb.es

Elizabeth Correa Manzano, D., Anarcaly Zambrano Olvera, M., & Clemencia Triviño Vera, K. (2025). Consultorios jurídicos gratuitos: instrumento clave para garantizar la justicia en sectores vulnerables y prioritarios. https://doi.org/10.29018/issn.2588

Finn, T., & Downie, A. (n.d.). ¿Qué es la experiencia de usuario (UX)? | IBM. 2021. Retrieved September 25, 2025, from https://www.ibm.com/es-es/think/topics/user-experience

Flores, L. (2019). Qué es NodeJS y para qué sirve | OpenWebinars. https://openwebinars.net/blog/que-es-nodejs/

Gómez Bachiller, S. (2015). INTRODUCCIÓN A GIT Y GITHUB-DÍA 1. https://www.uco.es/aulasoftwarelibre/wp-content/uploads/2015/11/git-cosfera-dia-1.pdf

Gómez, R. (2016). Estudio Teórico de MySQL.

https://ereding.etsi.us.es/bibing/proyectos/use/abreproy/11226/

Huet, P. (2022, August 24). Arquitectura de software: Qué es y qué tipos existen | OpenWebinars. https://openwebinars.net/blog/arquitectura-de-software-que-es-y-que-tipos-existen/

Maida, E. G. (2015). FACULTAD DE QUÍMICA E INGENIERIA “FRAY ROGELIO BACON” PONTIFICIA UNIVERSIDAD CATÓLICA ARGENTINA SANTA MARIA DE LOS BUENOS AIRES Cátedra Seminario de Sistemas. *Metodologías de Desarrollo de Software*. https://repositorio.uca.edu.ar/handle/123456789/522

Miguel, L., Hernández, A., Alonso, V., Romero, P., Sosa González, S. A., Adrián, J., & Rodríguez, V. (n.d.). Arquitectura REST para el desarrollo de aplicaciones web empresariales (Vol. 8). https://ctes.org.mx/index.php/ctes/article/download/748/909

Pérez Mora, O ., et al. (2005). Bases de datos 71Z799014MO. www.glo.org.mx

Sebastiany, F. (2025). Qué es un lenguaje de programación y sus aplicaciones | Salesforce. https://www.salesforce.com/mx/blog/lenguaje-de-programacion/

Serratosa, F. (2008). La biometría para la identificación de las personas. https://openaccess.uoc.edu/server/api/core/bitstreams/ffcfd70d-0c19-4a13-a2fa-891c573fb7d3/content

Villalobos, G. M., Darío, G., Sánchez, C., Alberto, D., & Gutiérrez, B. (2010). DISEÑO DE FRAMEWORK WEB PARA EL DESARROLLO DINÁMICO DE APLICACIONES. Scientia et Technica Año XVI, 44. https://www.redalyc.org/pdf/849/84917316032.pdf