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- APLICACIÓN MÓVIL PARA EL CONTROL DE PRODUCTIVIDAD Y ACTIVIDADES

AUTOR:

DANNY SEBASTIÁN CANO YÉPEZ

DIRECTOR:

ING. GUIDO OCHOA

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

QUITO DM, ABRIL DE 2026

DEDICATORA

Este trabajo lo dedico con mucho cariño y nostalgia a mi abuelito Eduardo Yépez, que donde sea que este estoy seguro de que va está orgulloso y sobre todo feliz que, aunque en vida no logró ver lo de seguro lo va a leer, gracias, por tanto. Agradezco también a mi abuelita Martha Suarez, que a sido una segunda madre para mi desde que tengo uso de memoria, y se la dedico por toda su entrega y esfuerzo que a puesto en mí. Dedico a mi madre Martha Yépez, que siempre le repito lo importante que es para mí, como se lo dije un día es mi motor para hacer las cosas y me siento feliz y orgulloso de la madre que tengo. A mi enamorada Ninel Rodríguez que me a enseñado muchas cosas importantes en la vida, a la que estimo y amo con todo mi ser. Dedico a mis mejores amigos este proyecto David, Joel, Leandro, Steve, Alejandro, Mateo, Jimmy, Andres, Sofi, Katy y Caro por ser una parte fundamental a lo largo de esta trayectoria, al resto de mis amigos tanto de la carrera como del Gym y entre otros. También dedico a mi compañero Elías, que aunque no lo conocí a profundidad el poco tiempo que compartimos fue en ser humano increíble que espero que donde este sepa que es uno más de nosotros. Además, dedico a toda mi familia que me apoya y esta conmigo en cada uno de los momentos importantes de mi vida. Y sobre todo a Dios que sin el sin la salud, el conocimiento y la fuerza que nos da no estaría aquí en este momento.

AGRADECIMIENTO

Expreso mi más sincero agradecimiento a todas las personas que, de manera directa o indirecta, contribuyeron al desarrollo del presente proyecto de titulación. En especial, agradezco a la institución académica que hizo posible este proceso formativo, brindándome herramientas, espacios y conocimientos para realizar este proyecto.

Así mismo agradezco a los docentes que a lo largo de mi formación universitaria, aportaron con sus grandes enseñanzas y orientación, sentando una base teóricas y prácticas que peritireron consolidar los conocimientos aplicados. En especial al Ing. Nelson Salgado por su orientación académica y acompañamiento y su gran aporte en mi etapa académica.

De manera especial, agradezco a quién dedica su tiempo a la lectura y revisión de este trabajo ya que su análisis y criterio constituyen un aporte fundamental para la reflexión académica y mejora continua.

Finalmente, agradezco a mi entorno personal por el apoyo brindado durante esta etapa, el cual fue clave para culminar este proyecto.

RESUMEN

Dentro de los últimos años, el Consultorio Jurídico Gratuito de la PUCE (CJGP) se ha enfrentado a distintos desafíos en la gestión y control de las actividades realizadas tanto por los estudiantes como por los abogados, lo que ha dejado en evidencia la necesidad de herramientas tecnológicas que permiten mejorar la eficiencia administrativa y el seguimiento de las prácticas profesionales. En este contexto, se desarrolló un módulo de Aplicación Móvil orientado a la productividad y registro de actividades, que van a complementar y fortalecer los sistemas existentes dentro del consultorio jurídico

El módulo permite la asignación y seguimiento de actividades, el registro de entrada y salida mediante geolocalización, la documentación de actividades extraordinarias y la generación de informes con métricas de productividad. Adicionalmente, incorpora notificaciones automáticas y control de acceso basado en roles, garantizando la seguridad de la información y la trazabilidad de las acciones. La implementación realizada contribuye a optimizar los procesos tanto internos como externos del CJGP, mejorar la coordinación tanto de estudiantes como administrativos asegurándonos un registro confiable de las prácticas estudiantiles.

PALABRAS CLAVE: Aplicación Móvil; Consultorio Jurídico Gratuito; Registro de Actividades; Geolocalización; Gestión de Prácticas; Informes de Productividad. Diseño de Interfaz (UI).

ABSTRACT

…

In recent years, the Free Legal Aid Clinic of the Pontificia Universidad Católica del Ecuador (CJGP) has faced several challenges in the management and control of activities carried out by both students and lawyers, highlighting the need for technological tools to improve administrative efficiency and the monitoring of professional practice. In this context, a mobile application module focused on productivity and activity tracking was developed to complement and strengthen the existing systems within the legal clinic.

The module enables the assignment and monitoring of activities, check-in and check-out registration through geolocation, documentation of extraordinary activities, and the generation of reports with productivity metrics. Additionally, it incorporates automatic notifications and role-based access control, ensuring information security and action traceability. The implemented solution contributes to the optimization of both internal and external processes of the CJGP, improves coordination between students and administrative staff, and guarantees a reliable record of student practice activities.

Keywords: Mobile Application; Free Legal Aid Clinic; Activity Tracking; Geolocation; Internship Management; Productivity Reports; User Interface Design (UI).

ÍNDICE

ÍNDICE DE FIGURAS Y GRÁFICOS

ÍNDICE DE TABLAS

CAPÍTULO I: INTRODUCCIÓN

1.1 Planteamiento del problema

El Consultorio Jurídico Gratuito de la PUCE (CJGP) enfrenta un problema significativo en la gestión y seguimiento de las actividades realizadas por estudiantes y abogados. Actualmente, no existe un sistema formal que permita registrar de manera estructurada los procedimientos ejecutados y las tareas diarias, lo que genera incertidumbre respecto al desempeño individual y colectivo dentro del consultorio.

La práctica habitual se basa en el uso de hojas de cálculo en Excel en donde se documenta las actividades semana tras semana. Sin embargo, este mecanismo resulta ser insuficiente, ya que no llega a garantizar un control adecuado ni facilita la generación de reportes detallados de productividad. Como consecuencia, se ha visto limitada la capacidad de los supervisores de cada área para evaluar con precisión el rendimiento que tienen los estudiantes, así como el impacto que tienen las actividades desarrolladas.

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

- ¿De qué manera una aplicación móvil puede optimizar el registro y seguimiento de actividades en el Consultorio Jurídico Gratuito de la PUCE?

Y las siguientes preguntas secundarias:

- ¿Cómo puede la aplicación móvil facilitar la generación de informes de productividad útiles para supervisores y coordinadores?
- ¿Qué beneficios aporta el uso de geolocalización en el control de actividades externas realizadas por estudiantes y abogados?
- ¿De qué forma la aplicación móvil contribuye a mejorar la transparencia y la trazabilidad de las prácticas preprofesionales en el consultorio?

1.2 Justificación

. El Consultorio Jurídico Gratuito (CJG) de la PUCE cumple un papel fundamental en la formación práctica de los estudiantes de Derecho, al brindarles la oportunidad de aplicar sus conocimientos en casos reales bajo la supervisión de profesionales experimentados. A su vez, constituye un espacio de apoyo social para personas de escasos recursos que requieren asesoría legal gratuita y de calidad.

No obstante, uno de los principales retos que enfrenta actualmente el consultorio es la ausencia de un sistema especializado que permita un control eficiente de las actividades y la productividad tanto de abogados como de estudiantes en el desarrollo de sus prácticas preprofesionales. La dependencia de herramientas básicas, como fichas en Excel, limita la capacidad de seguimiento y no garantiza la realización adecuada de las actividades.

Ante esta problemática, se justifica el desarrollo de un módulo de una aplicación móvil orientado al registro y seguimiento de actividades mediante geolocalización y control de tiempos. Esta solución permitirá optimizar la gestión interna del consultorio, fomentar la transparencia en las prácticas, fortalecer el control académico y mejorar el aprendizaje práctico de los estudiantes. Asimismo, contribuirá a incrementar la calidad del servicio brindado a la comunidad, al asegurar un monitoreo más preciso y una trazabilidad adecuada de las acciones realizadas.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar una aplicación móvil que permita el seguimiento, registro y control de actividades realizas por abogados y estudiantes dentro del Consultorio Jurídico Gratuito de la PUCE, mejorando la eficiencia operativa y optimizando su gestión interna.

1.3.2 Objetivos Específicos

- Realizar el levantamiento de la información, requerimientos funcionales y no funcionales

mediante reuniones con los actores del consultorio jurídico.

- Diseñar y codificar interfaces móviles intuitivas que faciliten a los usuarios el registro de actividades, detallando información como tipo de actividad, fecha, hora y ubicación geográfica.
- Mejorar la funcionalidad y la usabilidad de la aplicación móvil por medio de un proceso iterativo de prototipado, junto con la retroalimentación del usuario final hasta poder obtener un prototipo final y aprobado.

1.4 Alcance

El presente trabajo de titulación se enfoca en el desarrollo de un módulo de aplicación móvil para el Consultorio Jurídico Gratuito de la PUCE, el cual tiene como propósito registrar las actividades específicas realizadas por los estudiantes y abogados durante sus prácticas, incorporando el control de ingreso y salida mediante su geolocalización. La aplicación permitirá a los usuarios documentar las actividades asignadas y validad su ubicación en tiempo real, garantizando un registro confiable y seguro, 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.

El alcance que tiene este proyecto contempla el diseño e implementación del módulo móvil. Este trabajo no abarca aspectos relacionados con la gestión financiera, administrativa o legal de los casos, ni la puesta en producción en Google Store ni Apple Store, más bien se limita a la construcción del módulo móvil orientado exclusivamente al control de las actividades y productividad de quienes están involucrados en el consultorio jurídico.

CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA

2.1 Base teoría de Consultorios Jurídicos PUCE

El Consultorio Jurídico Gratuito de la Pontificia Universidad Católica del Ecuador (CJG) es una iniciativa social y académica, la cual se dedica a brindar servicios de asesorías y patrocinios legal gratuito en las áreas de derecho, tales como civil, penal, laboral, movilidad humana, niñez y adolescencia, familia, entre otras[1]. Su principal objetivo es atender a personas en situación de vulnerabilidad en conjunto de grupos de atención prioritaria, teniendo el fin de promover el acceso a la justicia y respeto de los derechos. Además, el CJG, es un espacio en el cual los estudiantes de la Facultad de Jurisprudencia realizan sus prácticas preprofesionales, bajo la supervisión de expertos multidisciplinarios conformados por abogados, trabajadores sociales y psicólogos, permitiéndoles así aplicar sus conocimientos con casos reales, además les permite fortalecer su trabajo práctico contribuyendo así a su cambio social. En síntesis, el consultorio opera con un enfoque integral, en donde combinan la formación académica con el servicio comunitario.

2.2 Metodología de desarrollo de software

Una metodología de desarrollo de software consiste en un conjunto de principios, fases y prácticas que nos ayudan a organizar de manera sistemática, partiendo en la ingeniería de requerimientos hasta la implementación. Su aplicación nos asegura que el desarrollo se vaya a realizar de forma estructurada, reduciendo errores, mejorando la comunicación de los participantes y garantizando un producto que vaya a responder a las necesidades que tienen los usuarios.[2]

Las metodologías se clasifican en tradicionales y ágiles. Un ejemplo de una metodología tradicional es el modelo en cascada, donde los requerimientos están totalmente definidos y constan de una documentación exhaustiva y poca posibilidad de cambios. Por otro lado, las metodologías

[1] (Pontificia Universidad Católica del Ecuador (PUCE), 2024)
[2] (Santander Universidades, 2020)

ágiles permiten variar los requerimientos durante el desarrollo, favoreciendo la adaptabilidad, la retroalimentación continua y la entrega temprana de versiones funcionales del producto.

Para el presente proyecto, se optó por un enfoque de prototipado evolutivo, propio de las metodologías ágiles. Este modelo nos permite construir una versión funcional de la aplicación móvil y a través de ella ir incorporando las mejoras de manera iterativa por medio de los comentarios de los usuarios del consultorio jurídico. Así, el desarrollo del módulo de actividades y geolocalización se realizó por medio de ciclos de diseño, implementación, prueba y ajuste, permitiéndome asegurar que el sistema final cumpla con los requerimientos identificados y se logre adaptar a las necesidades reales donde será utilizado.

2.2.1 Prototipado Evolutivo

El prototipado Evolutivo es una metodología de desarrollo de software que consiste en crear un prototipo inicial el cual se mejora de forma iterativa en base a la retroalimentación de los usuarios, hasta llegar a ser un producto final. Según Sommerville[3], este enfoque es apropiado cuando los requerimientos llegan a ser inciertos o son variables con frecuencia, ya que permite adaptar el sistema de una manera flexible.

Así mismo Pressman y Maxim [4] destacan que el prototipado facilita la comunicación con los usuarios, al momento de ofrecer versiones preliminares que ayudan a validar funcionalidades y reducir errores. No obstante, como nos advierte Boehm[5] una planificación deficiente en el número de iteraciones a realizar puede ocasionar sobrecostos y retrasos.

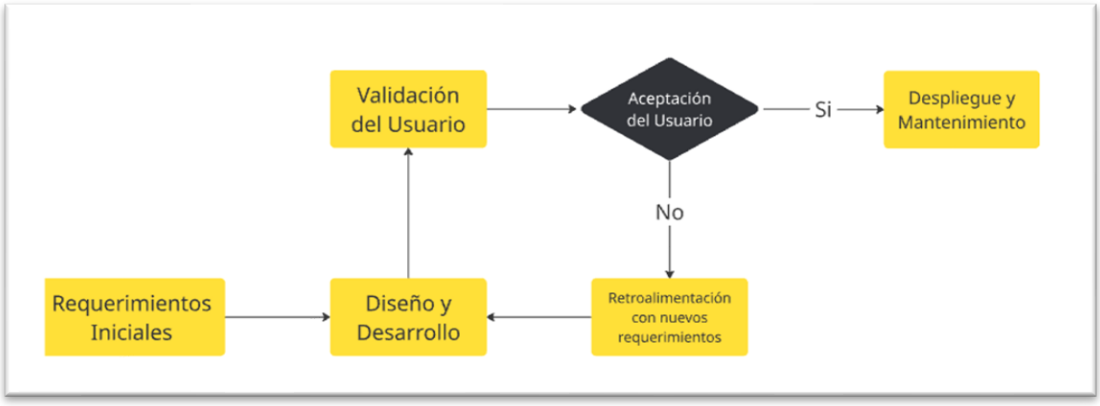
Durante este proyecto se aplicó el prototipad Evolutivo para el desarrollo del módulo de la aplicación móvil del CJG de la PUCE, iniciando con prototipos de interfaces (moockups) de registro de actividades, loggin y pantalla de geolocalización. Estos fueron revisados e iterados en

[3] (Sommerville, 2016)
[4] (Pressman, 2015)
[5] (Boehm, 2018)

varias ocasiones con los responsables del consultorio, lo que nos permitió refinar el diseño y asegurar que tanto la parte web como la parte móvil respondieran a las necesidades reales de los estudiantes y abogados.

Figura 1

Modelo del Prototipado Evolutivo



Nota: La figura es el diagrama que representa las fases del Prototipado Evolutivo. Fuente: Elaboración propia.

2.3 Lenguajes de programación

Un lenguaje de programación es un medio el cual nos permite expresar algoritmos a través de instrucciones donde la computadora los interpreta para poder realizar tareas específicas. Los lenguajes de programación pueden emplearse para desarrollar programas sencillos hasta complejos con múltiples funcionalidades, dependiendo de las necesidades del usuario final[6].

Cada lenguaje de programación tiene una sintaxis (regla de escritura) y una semántica (significado de las instrucciones), lo que va a determinar la forma de uso y su respectiva aplicación[7]. Estas se clasifican en dos tipos, los compilados e interpretados, un ejemplo de lenguaje compilado puede ser C++, Java ,C# y Flutter, mientras que el lenguaje interpretado puede ser

[6] (Cilsa, 2023)
[7] (Sommerville, 2016)

Python y JavaScript, donde sus principales diferencias radican en el rendimiento y la potabilidad del software [8].

En la actualidad, los lenguajes de programación se utilizan para diversos fines, desde el desarrollo web, aplicaciones móviles, videojuegos, bases de datos, entre otros. Su propósito fundamentas es lograr permitir la comunicación entre la máquina y el programador, ofreciendo una herramienta sistemática y estructurada para resolver problemas de maneras eficientes.[9]

2.3.1 Flutter

Flutter es un framework de desarrollo de aplicaciones móviles que fue creado por Google el cual permite construir aplicaciones nativas para IOS, Android, Escritorio y Web utilizando un único lenguaje de programación: Dart. La característica principal que tiene Flutter es dar un enfoque en el desarrollo multiplataforma donde una sola base de código ayuda a reducir tiempos y costos de implementación [10]. Además, este utiliza un motor de renderizado llamado Skia, el cual nos garantiza interfaces gráficas de gran flexibilidad en el diseño y alto rendimiento [11]

En el ámbito profesional y académico, Flutter ah sido fuertemente adoptado por su arquitectura basada en widgets, la cual nos permite diseñar interfaces personalizadas e intuitivas de manera declarativa[12]. Esta es una característica clave que convierte a Flutter en una herramienta perfecta para el desarrollo de aplicaciones móviles multiplataforma modernas.

2.3.2 Dart

Dart es un lenguaje de programación desarrollado por Google en 2011, enfocado al desarrollo de aplicaciones multiplataforma y utilizado principalmente por el framework de Flutter.

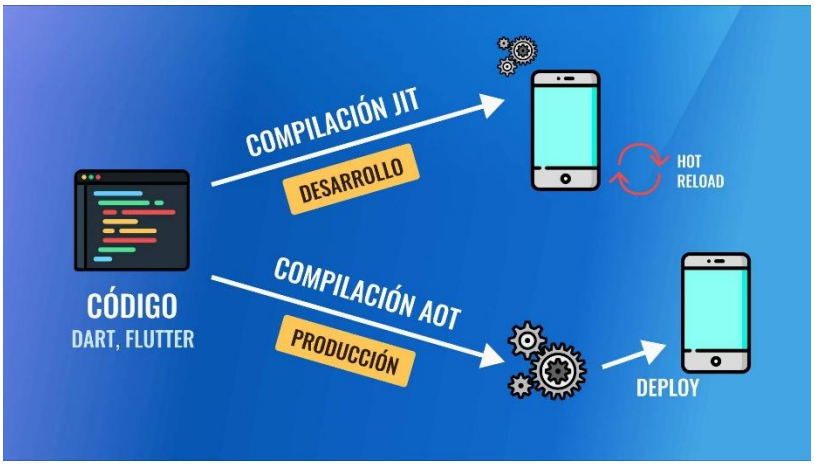
[8] (Sebesta, 2016)
[9] (Pressman & Maxim, 2015)
[10] (Google Developers., 2023)
[11] (Mahmud, 2021)
[12] (Carmona, 2020)

Se caracteriza por su tipado estático y seguro, su sintaxis moderna y su soporte para POO (Programación Orientada a Objetos) y programación funcional. [13]. Una de sus principales ventajas es la capacidad de compilación dual: Just-In-Time, usada comúnmente en el desarrollo para permitir la recarga rápida de cambios (hot reload) y AOT Ahead-of-Time, utilizada para generar código nativo y optimizado[14].

De acuerdo con Olanrewaju (2022), Dart ofrece un equilibrio entre productividad y rendimiento, que permite crear interfaces reactivas mediante un modelo declarativo con un simil al de otros frameworks modernos. Además, su compatibilidad con múltiples plataformas y su estrecha integración con Flutter hacen que sea una herramienta esencial para el desarrollo de aplicaciones modernas.

Figura 2

Proceso de compilación en Flutter: desarrollo y producción



Nota. La figura muestra el flujo de compilación del código fuente en Flutter, desarrollado en lenguaje Dart. En la etapa de desarrollo, Flutter utiliza la compilación Just-In-Time (JIT), que permite el uso de la función Hot Reload para aplicar cambios en tiempo real sin necesidad de

[13] (Google Developers, 2023)
[14] (Santos & Mendes, 2021)

reiniciar la aplicación. En cambio, en la etapa de producción, se emplea la compilación Ahead-Of-Time (AOT), que genera un código nativo optimizado para su despliegue final (deploy) en dispositivos móviles. Tomado de "EDteam Blog", por EDteam, 2023.

2.3.3 JavaScript

Es un lenguaje JavaScript es un lenguaje de programación interpretado, es un lenguaje de alto nivel y multiparadigma, que tiene definido un estándar ECMAScript, el cual define sintaxis, funciones, tipo de datos, módulos, etc.[15]. Su gran versatilidad se encuentra en su capacidad para poder integrarse de una manera nativa en los navegadores web para manipular el DOM (Document Object Model), facilitando así la creación de interfaces dinámicas e interactivas en tiempo real.[16]

Tiene varias características que lo destacan uno de ellas es el manejo de operaciones asíncronas por medio de callbacks, promises y su sintaxis de async/await, la cual permite procesar solicitudes sin necesidad de bloquear el resto del programa [17].Además, este incluye algunos entornos de ejecución como el de Node.js, que amplían su uso al lado del servidor, además de sus gran variedad de librerías y frameworks que ayudan a fortalecer su papel como una de las tecnologías que más se utilizan en el desarrollo de software [18]. Es por eso que Javascript se mantiene como una de las herramientas esenciales para el desarrollo de aplicaciones web y móviles híbridas, consolidándose, así como uno de los lenguajes de programación más utilizados en la industria del software actual.

2.4 Entorno de Desarrollo Integrado (IDE)

El Entorno de Desarrollo Integrado es un software que reúne en una misma interfaz de herramientas necesarias para construir aplicaciones, desde editores de código fuente,

[15] (Mozilla Developer Network, 2023)
[16] (Flanagan, 2020)
[17] (Eich, 2019)
[18] (W3Schools, 2023)

compiladores, depurador, soporte de versiones y gestor de proyectos [19]. Tiene como objetivo principal el agilizar el proceso de desarrollo, con el fin de mejorar la productividad del programador por medio de funciones de autocompletado, resaltado de sintaxis errónea y la detección temprana de errores[20].

2.4.1 Visual Studio Code

Es un IDE gratuito, ligero y multiplataforma creado por Microsoft, diseñado para la depuración y edición de código de varios lenguajes de programación Su gran uso junto con su gran flexibilidad radica en el uso de extensiones que permiten adaptarlo en varias tecnologías como : Flutter, JavaScript, Python o TypeScript. Además, este incluye herramientas como las de autocompletado, depuración, terminal incorporada y control de versiones de Git[21]. Debido a su arquitectura modular basada en Electron, es compatible con Linux, Windows y macOS. Por estos motivos y más Visual Studio Code se ha consolidado como uno de los IDE más utilizados a nivel mundial desde su facilidad de su, personalización y soporte para proyectos móviles como web.[22]

2.5 Emulador de Dispositivos

Un emulador de dispositivos es un software que reproduce el comportamiento de un dispositivo físico, permitiendo probar y ejecutar aplicaciones dentro de un entorno virtual, el cual simula las características del software y hardware del equipo real. Según Oralce (2023), los emuladores nos permiten validar el funcionamiento de un sistema sin necesidad de disponer físicamente del dispositivo, replicando aspectos como procesador, memoria, sistema operativo, la pantalla e incluso sensores específicos.

Este tipo de herramientas son fundamentales en el desarrollo de software, debido a que facilitan la depuración y análisis del rendimiento en distintos entornos, sin llegar a comprometer

[19] (Red Hat, 2023
[20] (JetBrains, 2023)
[21] (Microsoft, 2023
[22] (Stack Overflow, 2023; Visual Studio Magazine, 2023)

equipos reales [23]. Además, los emuladores pueden configurarse con diferentes versiones del sistema operativos, capacidad de red, resolución de pantalla, garantizando que la aplicación sea estable y funcionan al en una gran gama de dispositivos.

2.6 Android Studio

Android Studio es un entorno de desarrollo integrado (IDE) oficial creado por Google para la construcción de aplicaciones en el sistema operativo de Android. Android Studio está basado en IntelliJ IDEA, este entorno proporciona herramientas especializadas que facilitan la edición de código, la complicación, la depuración y la emulación de dispositivos con sistema operativo Android. Esto nos permite probar las aplicaciones en distintos entornos virtuales antes de implementarlas en hardware real [24]. Dentro de sus principales características están el uso del compilador Gradle, el soporte para múltiples lenguajes de programación como Java, Kotlin y Dart, su integración con Android Emulador para simular el comportamiento de diferentes dispositivos y versiones del sistema operativo[25].

2.7 Android Emulator

Android Emulator es una herramienta oficial proporcionado por Google, que permite simular el funcionamiento de dispositivos Android en un entorno virtual. Tiene como objetivo principal el facilitar el desarrollo, depuración y pruebas de aplicaciones sin necesidad de tener el dispositivo físico, reproduciendo las características del hardware, el sistema operativo y la interfaz de usuario [26]. Este emulador permite evaluar el rendimiento, compatibilidad y comportamiento de las aplicaciones bajo diferentes condiciones de red, tamaños de pantalla, resoluciones y capacidad de hardware. Además de esto ofrece funciones avanzadas como lo son la simulación de ubicación de GPS, llamadas, sensores y cámaras, lo que convierte a Android Emulator en una herramienta esencial dentro del proceso de

[23] (Google Developers, 2023)
[24] (Google Developers, 2023)
[25] (JetBrains, 2023)
[26] (Google Developers, 2023)

desarrollo y pruebas de aplicaciones móviles[27]

2.8 Base de Datos

Una base de datos es un conjunto estructurado de información que se almacena de forma electrónica y que puede ser gestionado mediante un sistema especializado denominado DBMS (Database Managment System), el cual nos permite crear modificar y realizar consultas de datos de manera eficiente. Estos sistemas nos ayudan a garantizar tanto la integridad, la seguridad y la disponibilidad de la información, siendo un punto clave para el desarrollo de aplicaciones modernas que requieren persistencia de datos. [28]. Existen dos modelos de bases de datos relacionales y las no relacionales. Las bases de datos relacionales son las que organizan la información en tablas que se vinculan entre sí por medio de claves primarias y foráneas, y que utilizan el lenguaje SQL (Structured Query Language) para su manipulación [29].

2.8.1 Structured Query Language (SQL)

El Lenguaje de Consulta Estructurada es un lenguaje estándar que se usa para gestionar y manipular datos almacenados en la base de datos relacionales. Su propósito es lograr realizar operaciones de modificación y consulta de la información por medio de sentencias estructuradas que interactúan directamente con el SGBD (Sistema Gestor de Base de Datos) [30]. Según Coronel y Morris (2019), SQL se compone de varias subcategorías de instrucciones, dentro de las cuales enfatizan el DDL (Data Definition Language) para poder definir estructuras, el DML (Data Manipulation Langguage) para gestionar registros y el DCL (Data Control Language) para controlar acceso y la seguridad. Cuenta con una Sintaxis declarativa, el cual permite ejecutar operaciones complejas con pocas líneas de código, logrando garantizar la integridad y consistencia de los datos.

[27] (Red Hat, 2023)
[28] (Coronel & Morris, 2019)
[29] (Date, 2021; Oracle, 2023)
[30] (Date, 2021)

2.8.2 MySQL

Es un SGBD (Sistema Gestor de Base de Datos) relacional de código abierto, desarrollando por MySQL AB y actualmente mantenido por Oracle Corporation. Utiliza un lenguaje de consulta estructurado SQL para le creación, administración de datos y modificación. Se caracteriza por su alto rendimiento junto con su fiabilidad y facilidad de uso[31]. Según Coronel y Morris (2019), MySQL destaca por su arquitectura cliente-servidor, la cual permite que múltiples usuarios accedan de manera simultánea a la información almacenada de forma segura

2.9 Control de Versiones

2.9.1 Importancia del control de versiones

El uso de un sistema de control de versiones resulta fundamental dentro del desarrollo de software, ya que su ausencia dificulta la gestión de los cambios realizados en el código y el seguimiento de cuándo y por quién fueron efectuados. En entornos profesionales, la falta de este tipo de herramientas puede generar conflictos entre diferentes versiones del proyecto y afectar la coherencia del desarrollo colaborativo[32].

2.9.2 Git

Git es un sistema de control de versiones (distribuido) que nos ayuda a gestionar los cambios realizados en el código fuente de un proyecto a través del tiempo. Fue desarrollado por Linus Torvalds en 2005, Git posibilita que múltiples desarolladores logren trabajar de una manera simultánea sobre los mismos archivos, manteniendo un historial completo de versiones del software al igual que su historial. [33]. A diferencia de los sistemas centralizados, Git almacena una copia completa del repositorio dentro de cada estación de trabajo, lo que ayuda a mejorar la velocidad, seguridad y disponibilidad de los datos [34]. También, Git nos ofrece

[31] (Oracle, 2023)
[32] (GitHub, 2025)
[33] (Chacon & Straub, 2014)
[34] (Git SCM, 2023)

funcionalidades como la creación de branches (ramas), merges (fusiones) y revisiones de cambios. Gracias a estas características,se ha convertido una herramienta esencial en la ingeniería de software permitiéndonos gestionar los cambios de código, mejorando la productividad y garantizando una construcción de software estructurada.

2.9.3 GitHub

Es una plataforma basada en la nube que permite almacenar y gestionar proyectos mediante repositorios vinculados al sistema de control de versiones Git. Su popularidad a nivel mundial se debe a las facilidades que ofrece para la colaboración entre desarrolladores, el seguimiento de modificaciones y la administración del código fuente en entornos de trabajo compartidos[35].

2.10 Aplicación Móvil

Una aplicación móvil es un tipo de software que está diseñado específicamente para ejecutarse en dispositivos portátiles como tabletas o teléfonos inteligentes, aprovechándote de las capacidades de hardware y sistemas operativos. A diferencia de una aplicación web, las aplicaciones móviles pueden funcionar de forma nativa al instalarlas directamente en el dispositivo, también pueden funcionar de manera híbrida al combinar tecnologías web y nativas, oh a su vez multiplataforma por medio de frameworks como Flutter o React Native, que nos permiten compilar un mismo código para diferentes sistemas operativos[36].

Según Sommerville (2016), destacan por su capacidad de interactuar con sensores del dispositivo, como GPS, micrófono, acelerómetro, etc. Ampliando así las funcionalidades y mejorando la experiencia del usuario. Además, estas aplicaciones pueden operar tanto en línea como sin conexión, sincronizando datos con el servidor cuando hay disponibilidad de red, lo que

[35] (GitHub, 2025)
[36] (Android e iOS) (Google Developers, 2023)

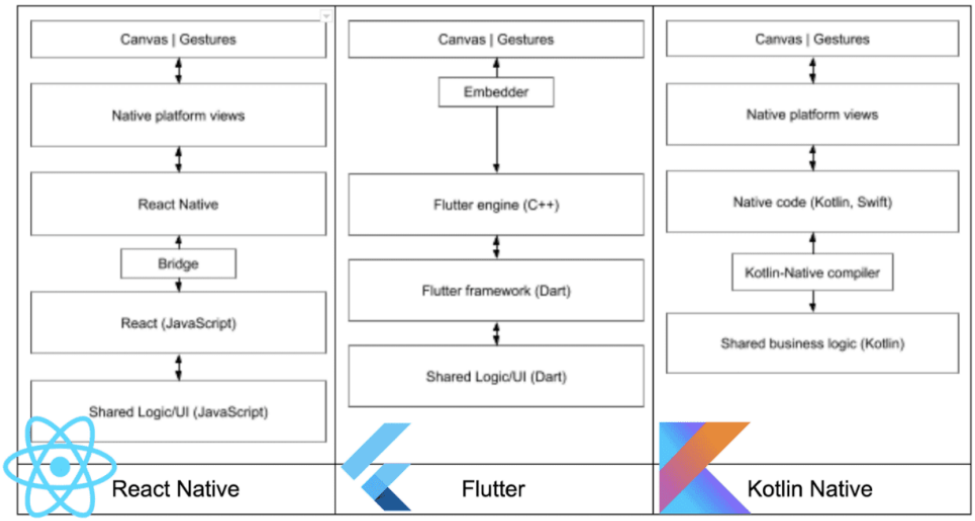
las convierte en herramientas versátiles, para la gestión de información en tiempo real.

2.10.1 Arquitectura de la Aplicación Móvil

La arquitectura de una aplicación móvil se organiza usualmente en capas, con el fin de separar responsabilidades. Existe la capa de presentación /UI, la cual incluye interacción y componentes visuales; lógica de negocio o gestión de estado, donde se recalcan reglas, validaciones, orquestación de flujo, patrones de diseños como el MVC,MVVM, BloC; y la capa de acceso a datos, donde se consumen servicios y persistencia. Para estas capas se comunican con servicios cliente-servidor por medio de APIS REST e integran diversos mecanismos de almacenamiento local, como lo son el uso de cache, SQLite, Key-Value, sincronización offline y online, manejo de permisos y sensores como los del GPS y cámara. La finalidad de separar las capas es mejorar la mantenibilidad, pruebas de software y escalabilidad al tiempo que reduce el acoplamiento y favorece la reutilización.[37]

Figura 3

Comparación de arquitecturas multiplataforma: React Native, Flutter y Kotlin Native



[37] (Pressman & Maxim, 2015; Sommerville, 2016; Bass, Clements, & Kazman, 2013).

Nota. La figura compara la estructura interna de tres frameworks de desarrollo multiplataforma. En React Native, el código JavaScript se comunica con las vistas nativas mediante un "bridge" o puente que puede afectar el rendimiento en procesos intensivos. En Flutter, el motor propio escrito en C++ y el framework en Dart permiten una renderización directa y de alto rendimiento sin necesidad de un puente intermedio. Por su parte, Kotlin Native compila el código de negocio directamente a código nativo mediante el compilador Kotlin-Native, integrándose de forma más cercana al sistema operativo. Esta comparación evidencia cómo Flutter ofrece una arquitectura más optimizada y consistente para el desarrollo de interfaces móviles. *Tomado de "Comparación entre frameworks multiplataforma: React Native, Flutter y Kotlin Native", por Nu Blog, 2021 muestra el flujo de compilación del código fuente en Flutter, desarrollado en lenguaje Dart.*

2.10.1.1 API REST

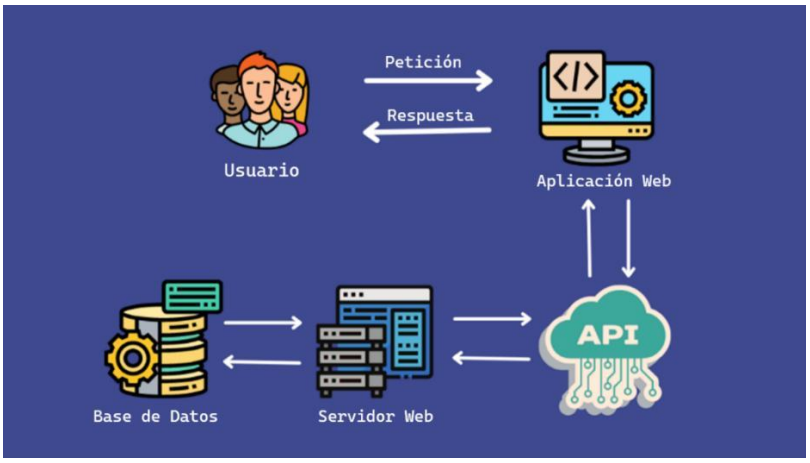
Una API REST (Applitaction Programming Interface Representational State Transfer) es un conjunto de reglas y convenciones que nos ayudan a la comunicación entre sistemas distribuidos por medio del protocolo HTTP. Su propósito es ofrecer una interfaz estandarizada la cual facilite el intercambio de información entre el servidor y el cliente, de manera ligera, eficiente y escalable [38].

Según Richardson y Ruby (2008), las APIs REST se basan en operaciones CRUD (Create,Read,Update,Delete) junto al uso de métodos como GET, POST, PUT Y DELETE, permitiendo así que diferentes aplicaciones o servicios interactúen sin depender de una tecnología especifica. Además, emplean formatos como XML o JSON, para estructurar las respuestas y solicitudes. Es así como este tipo de arquitectura se ha convertido en un estándar en desarrollo de aplicaciones por sur reutilización ,actualización y su flexibilidad.

[38] (Fielding, 2000).

Figura 4

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.

2.10.1.2 JSON (JavaScript Object Notation)

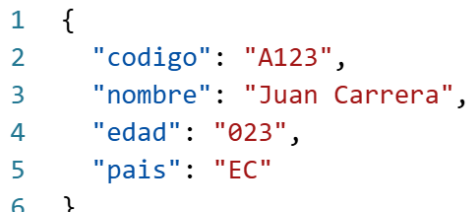
Conocido como JSON (JavaScript Object Notation) es un formato ligero, de intercambio de datos que permite estructurar información de manera legible tanto para el humano como para la máquina. Se basa en una estructura en pares clave-valor, además de estructura de datos como lista u objetos, lo que facilita su interpretación por parte de los distintos lenguajes de programación [39].

De acuerdo con Oracle (2023), JSON se ha convertido en el formato estándar de comunicación por medio de clientes y servidores en aplicaciones modernas, especialmente en las que utilizan APIs REST. Su sintaxis simple y compatibilidad con lenguajes de programación de tipado dinámico permiten el envío y recepción de datos de manera rápida y eficiente, favoreciendo la interoperabilidad entre sistemas.

Figura 5

Ejemplo de JSON

[39] (Crockford, 2013)



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

2.10.2 Interfaz de Usuario (Front-End Móvil)

La interfaz de usuario (UI) en una aplicación móvil es el medio principal entre el sistema y el usuario final. Su diseño tiene la finalidad de ofrecer una experiencia intuitiva, fluida y accesible, garantizando que las tareas se puedan ejecutar de manera agradable y eficiente. Según Nielsin (1994) una interfaz bien diseñada debe cumplir los principios de usabilidad, como la prevención de errores, consistencia, retroalimentación inmediata, los cuales son esenciales para mejorar la satisfacción del usuario.

En este sentido, la UX (User Experience) complementa la interfaz al centrarse en la percepción general del usuario respecto al uso de la aplicación. De acuerdo con Shneiderman y Plaisant (2010), la UX debe tener en consideración la accesibilidad, la coherencia entre pantallas, la respuesta visual y la optimización para distintos tamaños de dispositivos.

2.11 Biblioteca de Herramientas Flutter

En el desarrollo de aplicaciones móviles con Flutter, los packages o bibliotecas cumplen un rol fundamental al extender las funcionalidades del framework base, permitiendo implementar características avanzadas sin la necesidad de desarrollar desde cero. Según Google Developers (2023), estas bibliotecas se integran mediante el gestor de dependencias pub.dev, que proporciona paquetes oficiales y de terceros, ayudando a fomentar la reutilización de componentes

y estandarización de buenas prácticas en el desarrollo móvil.

2.12 Buenas Prácticas en el Desarrollo Móvil

Las buenas prácticas en desarrollo móvil son un conjunto de principios, estándares y guías que buscan asegurar calidad, usabilidad, accesibilidad, rendimiento y seguridad del software. Su aplicación sistemática mejora la experiencia del usuario, reduce defectos y facilita la mantenibilidad y escalabilidad de las aplicaciones[40].

2.12.1 Principios de diseño y usabilidad en aplicaciones móviles

El diseño de interfaces móviles tiene que priorizar la usabilidad según las heurísticas de Nielsen, consistencia con patrones nativos como Material Design oh, Humana Interface Guidelines , accesibilidad y rendimiento percibido como fluidez e interacción táctil. Una UI eficaz minimiza la carga cognitiva, ofrece retroalimentación inmediata y previene errores [41]

Tabla 1

Principios de UI/UX para aplicaciones móviles

Categoría	Buena práctica	Descripción
Accesibilidad (WCAG)	Contraste y tamaño táctil	Mantener contraste mínimo (≈4.5:1), objetivos táctiles ≥44–48 dp, soporte de lector de pantalla.
Jerarquía visual	Diseño claro y predecible	Priorizar información, espaciar y agrupar; evitar saturación visual.
Navegación y feedback	Estados y microinteracciones	Indicadores de carga, confirmaciones, estados vacíos y de error informativos.
Consistencia con guías nativas	Material Design / Apple HIG	Ayuda con la modularidad y evita conflictos de estilos.
Rendimiento percibido	Fluidez y tiempos de respuesta	60 fps cuando sea posible, lazy loading, diferir tareas en segundo plano.
Tipografía y color	Legibilidad	Escalas tipográficas responsivas, interlineado adecuado, uso moderado de color.

[40] (Pressman & Maxim, 2015; Sommerville, 2016).
[41] (Nielsen, 1994; Shneiderman & Plaisant, 2010)

Versionamiento y control	Usar un Sistema de control de versiones	Simplifica el trabajo en equipo y permite un control eficiente del historial de cambios.

Referencias clave: Nielsen (1994); Shneiderman & Plaisant (2010); Google Material Design (2023); Apple Human Interface Guidelines (2024); W3C WCAG 2.2 (2023). Esta tabla muestra un resumen de los estándares y buenas prácticas en el desarrollo móvil, dividido por categorías, la buena práctica/estándar y su descripción. Fuente: Elaboración propia, apoyado con herramientas de Inteligencia Artificial.

2.12.2 Seguridad en las aplicaciones móviles

La seguridad móvil abarca una arquitectura, almacenamiento, autenticación, comunicaciones y resiliencia del código. Existen estándares como OWASP,MASVS/MSTG y guías de plataformas definen controles mínimos para poder proteger datos y reducir superficies de ataque[42]. Desde el punto de vista arquitectónico, una aplicación móvil segura debe implementar buenas prácticas de diseño, tales como la separación de capas, el principio de mínimo privilegio y la validación de datos tanto en el cliente como en el servidor. En cuanto al almacenamiento, es fundamental proteger la información sensible que se guarda en el dispositivo, utilizando mecanismos de cifrado y evitando el almacenamiento innecesario de credenciales o datos críticos en texto plano.

[42] (OWASP, 2023)

CAPÍTULO III: ANÁLISIS Y DISEÑO DE LA APLICACIÓN

3.1 Análisis de Requerimientos

El análisis de requerimientos permite identificar las funcionalidades junto con las características que debe cumplir el sistema para lograr satisfacer las necesidades de los usuarios. En este caso, los requerimientos se enforcan exclusivamente en el módulo móvil del sistema de los Consultorios Jurídicos de la PUCE, cuya finalidad es registrar y controlar las actividades de los estudiantes y abogados por medio de geolocalización, control de tiempo y sincronización de datos.

3.1.1 Requerimientos Funcionales

3.1.1.1 Autenticación y Gestión de Sesión

- **Inicio de Sesión:** La aplicación permitirá iniciar sesión a la cuenta del usuario por medio de su correo y contraseña registrados previamente en el sistema de Balanza Web. Una vez autenticado correctamente por medio del token JWT generará acceso.
- **Recuperación de contraseña:** La aplicación deberá tener una sección para poder recuperar la contraseña a través de un código de verificación temporal vía correo
- **Cierre de sesión:** El usuario podrá cerrar su sesión en cualquier momento. Al hacerlo, se eliminarán credenciales y los datos temporales almacenados en caché para garantizar la seguridad.

3.1.1.2 Gestión de Actividades

- **Visualización de actividades:** La aplicación mostrará al usuario las actividades asignadas clasificadas en dos: “Actividades de Hoy” y “Actividades Futuras”, según la fecha programada.

- **Calendario y vista en lista:** La aplicación incluirá dos modos de visualización, además de la pantalla principal. Un calendario que muestre las actividades por fecha y una lista cronológica con los detalles de cada actividad.
- **Detalle de actividad:** Al seleccionar una actividad, el usuario podrá visualizar información relevante de la actividad a realizar incluyendo lugar, fecha, hora, nombre, tipo y observación.

3.1.1.3 Registro de Actividades

- **Registro de Entrada:** La aplicación permitirá al usuario registrar el inicio de una actividad, almacenando la fecha, la hora y coordenadas de geolocalización.
- **Registro de Salida:** Al finalizar la actividad, el usuario podrá registrar su salida, capturando nuevamente las coordenadas, el estado de la actividad: completada, suspendida, reprogramada o finalizada junto con una breve descripción o observación.

3.1.1.4 Modo Offline y Sincronización

- **Modo Offline:** La aplicación permitirá al usuario descargar todas las actividades asignadas y almacenarlas localmente mediante SharedPreferences para su uso en el modo sin conexión.
- **Registro sin conexión:** En caso de no contar con conectividad, la aplicación guardará localmente los registros de entrada y salida hasta que la conexión sea restablecida.
- **Sincronización automática:** Una vez detectada la conexión a internet, la aplicación sincronizará los datos almacenados en caché con el servidor y la base datos principal.

3.1.1.5 Interfaz de Usuario y Navegación

- **Pantalla Principal:** La aplicación contará con una pantalla de inicio que mostrará en resumen de las actividades y opciones de navegación.
- **Navegación por íconos:** El menú inferior permitirá al usuario acceder a tres vistas principales: inicio, calendario y lista de actividades.
- **Diseño Adaptable:** La interfaz se ajustará automáticamente a diferentes resoluciones de pantalla, manteniendo coherencia visual según los lineamientos de material Design.

3.1.2 Requerimientos No Funcionales

3.1.2.1 Rendimiento y Eficiencia

- **Tiempo de respuesta:** La aplicación deberá ejecutar las acciones principales (inicio de sesión, carga de actividades y registro de entradas/salidas) en un tiempo máximo de 5 segundos bajo una conexión estable.
- **Optimización de recursos:** El consumo de memoria y batería deberá mantenerse dentro de los límites recomendados para aplicaciones móviles ligeras, evitando procesos innecesarios en segundo plano.
- **Sincronización eficiente:** La transferencia de datos hacia y desde el servidor deberá realizarse de forma asíncrona, priorizando el envío en segundo plano cuando se detecte conectividad.

3.1.2.2 Seguridad y Privacidad

- **Cifrado de Comunicación:** Toda comunicación entre la aplicación y el servidor deberá realizarse por medio del protocolo HTTPS, asegurándonos de la confidencialidad de los datos transmitidos.

- **Gestión de Credenciales:** Las contraseñas y tokens de autenticación deberán almacenarse mediante mecanismos seguros como Flutter Secure Storage, evitando su exposición en el código o almacenamiento plano.
- **Protección de datos personales:** Los datos sensibles de los usuarios (ubicación, credenciales, registros de actividades) deberán tratarse conforme a los principios de protección de datos personales establecidos por la PUCE y las normativas vigentes.
- **Validación de acceso:** Solo los usuarios autenticados podrán acceder a la información almacenada localmente o sincronizada con el servidor.

3.1.2.3 Usabilidad y experiencia de Usuario

- **Interfaz Intuitiva:** La aplicación deberá cumplir con los lineamientos de Material Design, priorizando la claridad, legibilidad y accesibilidad en dispositivos móviles.
- **Lenguaje claro:** Todos los textos, mensajes y botones deberán redactarse en un lenguaje claro y comprensible para el usuario final.
- **Compatibilidad visual:** La interfaz deberá adaptarse correctamente a diferentes resoluciones y tamaños de pantalla.

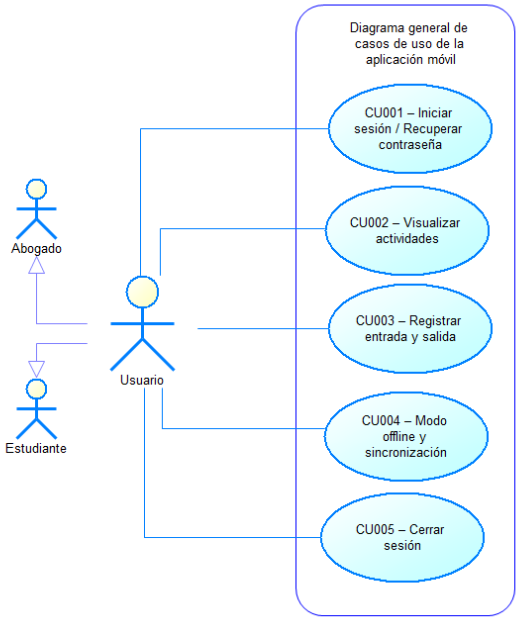
3.1.2.4 Compatibilidad y Portabilidad

- **Compatibilidad con Android:** La aplicación deberá ser funcional en dispositivos con sistema operativo Android 9.0 (Pie) o superior.
- **Conectividad variable:** El sistema deberá funcionar correctamente en contextos de conectividad limitada, mediante el uso del modo offline y sincronización automática.

3.1.3 Diagrama de Casos de uso general

Figura 6

Diagrama de Casos de Uso General



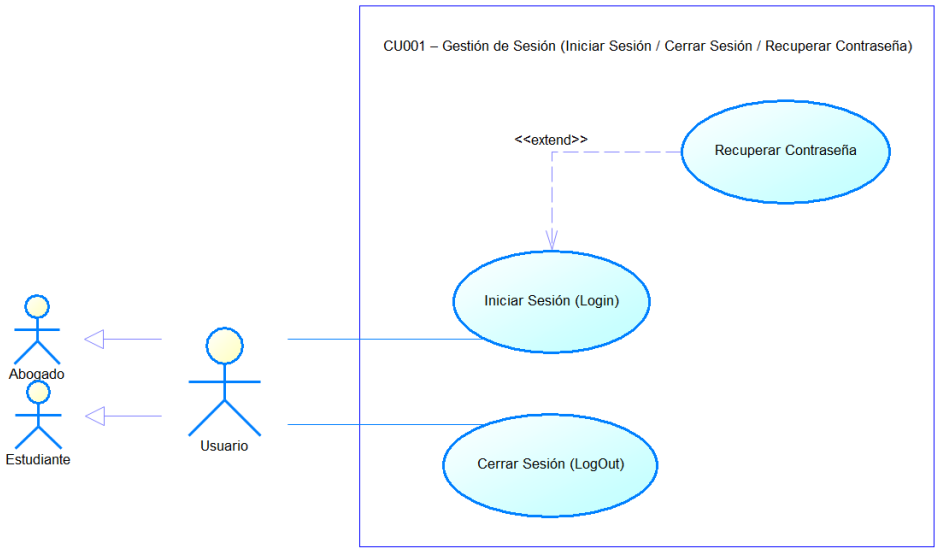
Nota: La figura representa el diagrama de casos de uso general del módulo propuesto, junto con los 2 roles que van a interactuar con las funcionalidades descritas, Fuente: Elaboración propia.

3.1.4 Diagrama de Casos de uso: Siguiente Nivel

3.1.4.1 CU001 y CU005 Gestión De sesiones:

Figura 7

CU001 – Gestión de Sesión (Iniciar Sesión / Cerrar Sesión / Recuperar Contraseña)



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

Tabla 2

Caso de uso: Gestión de Sesión (Iniciar Sesión / Cerrar Sesión / Recuperar Contraseña)

Nombre del caso de uso: Gestión de Sesión (Iniciar Sesión / Cerrar Sesión / Recuperar Contraseña)	**ID Único:** GU-001
Área: Aplicación Móvil CJG-PUCE	
Actor(es): usuarios del Sistema: Estudiante y/o Abogado	
Interesados: Usuario del Sistema	
Nivel: Usuario -Alto Nivel	
Descripción: Permite que el usuario con las credenciales válidas pueda lograr autenticarse en la aplicación móvil, gestionar su sesión activa y recuperar su contraseña en el caso de haberla olvidado. Incluye el proceso de incio de sesión (Login), cierre de sesión (Logout) y recuperación de contraseña por medio de código de verificación el cual se envía al correo registrado	
Evento desencadenador: El actor abre la aplicación móvil por primera vez y navega a la pantalla de inicio de sesión para ingresar las credenciales correspondientes o seleccionar la opción de recuperación de contraseña.	
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 inicio de sesión	Correo y contraseña del Sistema Web
2. El sistema valida el formato de los datos ingresados y envía las credenciales al servicio de autenticación	
3. El servidor verifica que el correo y contraseña correspondan a un usuario registrado y activo y genera un token de acceso	
4. El sistema valida el token, almacena el token y el usuario es redirigido a la pantalla principal de actividades.	El usuario recibe el Token de sesión para usar el sistema

Flujos Alternos:

A1- Credenciales incorrectas

1.- El sistema detecta que las credenciales no coinciden

2.- Se muestra un mensaje de error indicando que el usuario y/o contraseña son incorrectos

3.- El sistema permite reintentar el inicio de sesión.

A2- Recuperación de contraseña (<<extend>>)

1.- El usuario selecciona la opción de "¿Olvidó su contraseña?"

2- El sistema solicita el correo asociado a la cuenta

3.- Se envía un código temporal al correo del usuario

4.- Ingresa el código de restablecimiento temporal enviado

5.- El sistema valida el código de restablecimiento de contraseña

6.- El usuario ingresa la nueva contraseña

5.- Finaliza extensamente en Iniciar Sesión permitiendo el intento nuevamente con la nueva contraseña

Flujo Complementario: Cerrar Sesión (Logout)

1.- El usuario una vez iniciado sesión selecciona la opción de Cerrar Sesión

2.- El sistema invalida la sesión activa y elimina los datos temporales almacenados (tokens, cache de usuario)

3.- Se muestra nuevamente la pantalla de inicio de sesión.

Precondiciones:

1. El usuario debe tener una cuenta registrada en el sistema
2. Debe existir conexión a intenrt para validar credenciales.
3. La cuente debe estar activa

Postcondiciones:

1. La sesión del usuario queda regisrtada como activa dentro del sistema
2. Las opciones de la aplicación se habilitan según el rol del usuario
3. Si se cerró sesión, todos los datos temporales son eliminados
4. Si se realizó recuperación de contraseña, la nueva clave queda registrada en el sistema

Suposiciones:

- El usuario recuerda su correo registrado en el sistema web

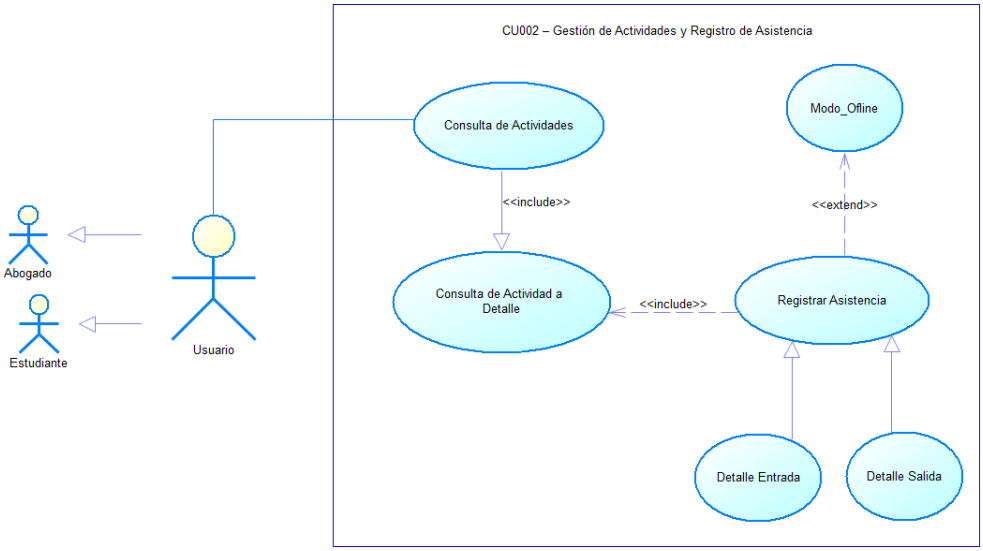
• El servidor de autenticación se encuentra disponible • El usuario dispone de acceso a su correo para recibir códigos de recuperación
Garantía de éxito: El usuario accede correctamente al sistema móvil mediante credenciales válidas y su sesión se mantiene activa mientras utiliza la aplicación
Garantía mínima: • Si ocurre un error en el inicio, los datos existentes no son alterados • El sistema informa al usuario mediante notificaciones claras del fallo (credenciales inválidas, código incorrecto, etc)
Requerimientos cumplidos: Autenticación de usuario , Gestión de sesiones activas, Recuperación de contraseña basada en correo electrónico
Cuestiones pendientes: Ninguna
Prioridad: Alta
Riesgo: Bajo

Nota: Esta tabla detalla el caso de uso "Gestión de Sesión (Iniciar Sesión / Cerrar Sesión / Recuperar Contraseña)", describiendo de forma estructurada sus actores, condiciones, flujos y restricciones. La información presentada corresponde al funcionamiento real del módulo de autenticación de la aplicación móvil del Consultorio Jurídico PUCE. Fuente: Elaboración propia.

3.1.4.2 CU002 y CU003 Gestión de Actividades y Registro de Asistencia:

Figura 8

CU002 – Gestión de Actividades y Registro de Asistencia Usuario



Nota: La figura representa el diagrama a detalle para el caso de uso "Gestión de Actividades y Registro de Asistencia", donde el Rol del usuario tanto el Abogado como el estudiante son los principales actores. Fuente: Elaboración propia.

Tabla 3

Gestión de Actividades y Registro de Asistencia (A detalle)

Nombre del caso de uso: Gestión de Actividades y Registro de Asistencia	**ID Único:** GU-002
Área: Aplicación Móvil CJG-PUCE	
Actor(es): usuarios del Sistema: Estudiante y/o Abogado	
Interesados: Usuario del Sistema, Secretaria, Coordinador de Área	
Nivel: Usuario -Medio	
Descripción: Permite al usuario consultar las actividades asignadas (del día y próximas) registras su asistencia, incluyendo los procesos de Registrar Entrada y Registrar Salida según la hora programada para cada actividad. El sistema muestra la información detallada de cada actividad (lugar, tipo, caso , juez asignado, etc). Además, permite ejecutar el registro en modo Offline, almacenando temporalmente los datos para sincronizarlos posteriormente con la base de datos cuando exista conexión a internet nuevamente.	
Evento desencadenador: El usuario accede a la aplicación móvil, visualiza la lista de actividades del día o próximas y selecciona una actividad para consultar su detalle o registrar entrada/salida.	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal):	**Información para los pasos**

1. El usuario selecciona una actividad desde la pantalla principal.	Datos de Usuario
2. El sistema ejecuta Consulta de Actividad a Detalle y muestra información completa (lugar, tipo, caso, juez, hora, observaciones, etc).	
3. Si la actividad corresponde al horario actual, el sistema habilita Registrar Entrada.	
4. El usuario confirma su entrada.	
5. Al finalizar, el usuario vuelve a abrir la actividad y selecciona Registrar Salida.	
6. El sistema solicita seleccionar resultado (Cancelada, Pospuesta, Finalizada y Suspendida) y una observación o resumen obligatoria.	Información complementaria de salida.
7. El usuario recibe la confirmación del registro de salida.	

Flujos Alternos:

A1- Actividad fuera de horario

1.- El sistema detecta que la hora registrada no esta dentro del margen de 30min de tiempo límite para registrar la entrada

2.- El sistema habilita el registro de la actividad

3.- El sistema marca como registro fuera de la hora establecida.

A2- Modo Offline (<<extend>>)

1.- El usuario selecciona el modo offline o el sistema detecta ausencia de internet.

2- El sistema carga actividades desde el almacenamiento local previamente descargado

3.- El sistema permite registrar entrada o salida normalmente

4.- Los registros se guardan en caché local.

5.- Al reconectarse, el sistema sincroniza automáticamente las entradas y salida pendientes con la base de datos.

Flujo Complementario:

Consulta de Actividades a Detalle (<<include>>)

- Siempre se ejecuta antes de registrar entrada o salida
- Muestra información completa para validar la diligencia.

Detalle Entrada y Detalle Salida (<<include>>)

- Subprocesos obligatorios para registrar datos adicaionales (ubicación, observación, resultado)

Precondiciones:

1. El usuario debe tener sesión activa en la aplicación móvil.
2. Las actividades deben haber sido previamente asignadas desde el sistema web.
3. El dispositivo debe tener almacenamiento local disponible para modo offline.
4. Si la actividad lo requiere, el GPS debe estar activo para registrar la ubicación.

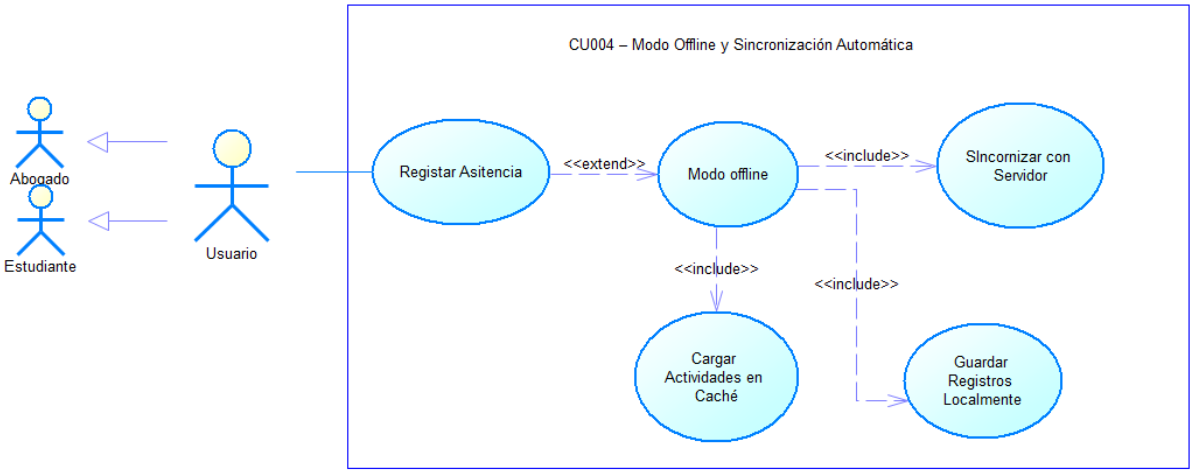
Postcondiciones: 1. Se registra correctamente la entrada o salida con hora y detalle. 2. La actividad cambia de estado según el registro realizado. 3. Si fue en modo offline, los datos quedan almacenados localmente y se sincronizan luego. 4. La salida queda registrada con resultado y observación obligatoria.
Suposiciones: • El usuario conoce el procedimiento de registro de actividades. • La información de las actividades se cargó correctamente en la aplicación. • El dispositivo móvil tiene la hora correcta.
Garantía de éxito: • Entrada y salida registradas correctamente y sincronizadas con la base de datos. • El usuario obtiene información actualizada y coherente sobre sus actividades.
Garantía mínima: • Si ocurre un error, los registros existentes no se alteran. • El sistema muestra mensajes claros y conserva los datos sin cambios.
Requerimientos cumplidos: Visualización de actividades asignadas, Registro de entrada y salida, Modo Offline y sincronización de datos,
Cuestiones pendientes: Ninguna
Prioridad: Alta
Riesgo: Medio-bajo (dependencia de conectividad y sincronización automática)

Nota: Esta tabla detalla el caso de uso "Gestión de Actividades y Registro de Asistencia", especificando sus actores, flujos y condiciones de operación dentro de la aplicación móvil del Consultorio Jurídico PUCE. La información refleja el funcionamiento real del módulo de actividades y asistencia. Fuente: Elaboración propia.

3.1.4.3 CU004 Modo Offline y Sincronización Automática:

Figura 9

Caso de Uso: Modo Offline y Sincronización Automática (A detalle)



Nota: La figura representa el diagrama a detalle para el caso de uso "Modo Offline y Sincronización Automática" , donde el Rol del usuario tanto el Abogado como el estudiante son los principales actores. Fuente: Elaboración propia.

Tabla 4

Modo Offline y Sincronización Automática (A detalle)

Nombre del caso de uso: Modo Offline y Sincronización Automática	**ID Único:** GU-004
Área: Aplicación Móvil CJG-PUCE	
Actor(es): usuarios del Sistema: Estudiante y/o Abogado	
Interesados: Usuario del Sistema, Secretaría, Coordinador de Área, Administración del Sistema Web	
Nivel: Usuario -Medio	
Descripción: Este caso de uso permite que el usuario continúe utilizando la aplicación móvil sin conexión a internet, pudiendo consultar actividades previamente cargadas y registrar asistencia (entrada y salida) en modo offline. Para ello, el sistema almacena en caché las actividades asignadas, guarda localmente los registros generados y sincroniza automáticamente los datos pendientes cuando la conexión vuelve a estar disponible. El Modo Offline extiende al caso "Registrar Asistencia", ya que solo ocurre cuando no hay conectividad durante el registro.	
Evento desencadenador: El sistema detecta que no existe conexión a internet mientras el usuario consulta actividades o intenta registrar entrada/salida. El usuario selecciona dentro del menú la opción de activar el modo offline	
Tipo de desencadenador: ☑ Externo ☐ Temporal	
Pasos realizados (ruta principal):	**Información para los pasos**
1. El usuario abre la aplicación móvil y selecciona una actividad.	Datos del usuario

2. El sistema detecta ausencia de conectividad y activa el Modo Offline.	
3. El sistema ejecuta el subcaso Cargar Actividades en Caché.	
4. El usuario registra entrada o salida normalmente.	Registro de datos sin conexión.
5. El sistema ejecuta Guardar Registros Localmente.	
6. Cuando la conexión vuelve, el sistema ejecuta Sincronizar con Servidor y se envía la confirmación al usuario	Recibe confirmación de datos sincronizados.

Flujos Alternos:

A1 – Caché vacío

1. El sistema detecta que no existen actividades almacenadas localmente.
2. Se muestra un mensaje indicando que se requiere conexión inicial para cargar actividades.
3. Se bloquea el registro de asistencia hasta que exista una sincronización inicial.

A2 – Error al sincronizar

1. El sistema intenta sincronizar los registros pendientes.
2. Si ocurre un error, mantiene los datos en memoria local.
3. Reintenta la sincronización automáticamente más adelante.

Flujo Complementario:

Subcasos Incluidos (<<include>>)

- **Cargar Actividades en Caché:** Obtiene actividades del almacenamiento local.
- **Guardar Registros Localmente:** Almacena entrada o salida hasta que exista conexión.
- **Sincronizar con Servidor:** Envia todos los registros pendientes a la base de datos del sistema web.

Precondiciones:

1. El usuario debe tener sesión activa en la aplicación móvil.
2. Debe existir al menos una sincronización previa (para que existan actividades cacheadas).
3. El dispositivo debe permitir almacenamiento local.
4. El usuario debe estar intentando registrar entrada o salida o consultar actividades.

Postcondiciones:

1. Las actividades pueden visualizarse sin conexión.
2. La entrada o salida queda registrada localmente si no existe conexión.
3. Los datos se sincronizan automáticamente cuando vuelve la conexión.
4. El estado de la actividad se actualiza correctamente en la base de datos.

Suposiciones:

- Existe un mecanismo interno de caché que guarda actividades y registros.
- El dispositivo tiene espacio suficiente para almacenar datos locales.
- La conexión regresará eventualmente.

Garantía de éxito: • Los datos creados en modo offline se sincronizan exitosamente en cuanto exista conexión. • El usuario puede continuar trabajando sin interrupciones.
Garantía mínima: • Si no se puede sincronizar, los registros permanecen almacenados localmente sin perderse. • El sistema informa al usuario mediante mensajes adecuados.
Requerimientos cumplidos: Registrar entrada y salida en la aplicación móvil, modo Offline y Sincronización Automática
Cuestiones pendientes: Ninguna
Prioridad: Alta
Riesgo: Medio (dependencia de conectividad para sincronización)

Nota: Esta tabla describe el caso de uso "Modo Offline y Sincronización Automática", detallando los flujos, condiciones y subprocesos requeridos para operar sin conexión y mantener la integridad de los datos del usuario. Fuente: Elaboración propia.

3.2 Diagrama Entidad-Relación (ERD)

En este caso, para el módulo de Control de Productividad y Actividades desarrollado para la aplicación móvil del Consultorio Jurrídico, el análisis se centra únicamente en las entidades que son necesarias para garantizar la ejecución, asignación y seguimiento de las actividades externas realizadas por los abogados y estudiantes. Aunque la base de datos general del sistema web contiene múltiples tablas complementarias, para el funcionamiento del módulo móvil haremos el uso de tres entidades principales, las cuales nos permiten manejar los procesos de autenticación, asignación de actividades y registro de evidencia en el tiempo real.

Partiendo desde este punto, la primera entidad es "Internal_Users", tabla que alacena la información de todos los usuarios internos que operan dentro de los Consultorios Jurídicos (Secretaria, Coordinado, Abogado, Estudiante, Administrador). Su clave primara es Internal_Id, que corresponde a la cédula o pasaporte del usuario, además de contar con atributos como nombre, correo, contraseña, tipo de usuario y área asignada. Esta es una entidad fundamental, debido a que

constituye de punto de origen para todas las actividades que se gestionan tanto en el sistema web como en la aplicación móvil. Por consiguiente, un usuario interno puede tener asignadas múltiples actividades, lo que estable una relación 1:N (uno a muchos) con la entidad Activities.

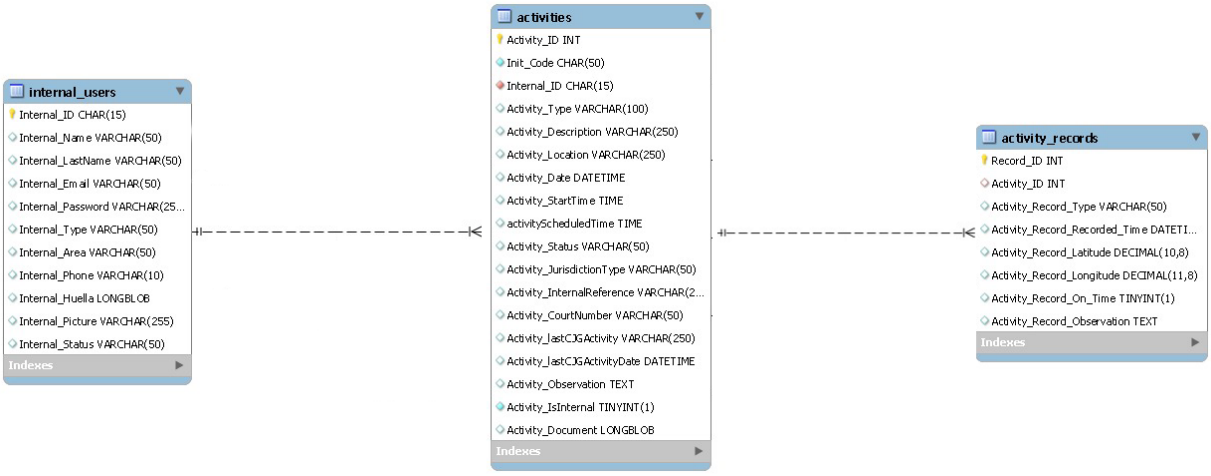
Por su parte, la tabla "Activities" contiene todos los registros de actividades tanto internas como externas, que incluye la información detallada como el tipo de actividad, fecha, hora, jurisdicción, ubicación asignada, etc. Dentro del módulo móvil, nos interesa trabajar con el tipo de actividad que sean externas, que son aquellas que requieren una validación de cumplimiento mediante geolocalización y registro de tiempo por parte del usuario. Cada actividad pertenece a un usuario interno responsable, por medio de Internal_ID como clave foránea, y a su vez puede registrar múltiples eventos vinculados, como entrada o salida, lo que establece una relación 1:N (Uno a muchos) con la entidad Activity_Records.

Finalmente, la entidad "Activity_Records" representa los registros generados por medio de la aplicación móvil durante la ejecución de una actividad asignada. Cada registro documenta información clave como el tipo de acción realizada (Entrada / Salida), la fecha y hora del registro, las coordenadas de geolocalización capturadas automáticamente por medio de la latitud y longitud, el indicador de puntualidad y cualquier observación relevante del usuario, Estos registros permiten validar la culminación adecuada de la actividad y constituyen la evidencia que posteriormente se resume en los informes de productividad y seguimiento. Cada registro pertenece a una sola actividad, consolidando así la relación (1:N) desde Activities hacia Activity_Records.

En conjunto, estas actividades permiten modelar correctamente el flujo operativo de la aplicación móvil: Autenticación del Usuario, Visualización de Actividades Asignadas, Registro Georreferenciado de entrada y salidas, y posteriormente consolidación de la información para la revisión por la parte de la secretaria y coordinadores. El diseño E/R completo se presenta en la sección de Anexos para facilitar su lectura e interpretación formal.

Figura 10

Diagrama entidad–relación del módulo de actividades



Nota: La figura muestra el diagrama entidad–relación correspondiente al módulo de gestión de actividades de la aplicación. En este esquema se representan las tablas internal_users, activities y activity_records, así como sus relaciones, las cuales permiten estructurar el registro de usuarios internos, la planificación de actividades judiciales y el almacenamiento de los registros de asistencia asociados a cada actividad. Fuente: Elaboración propia.

3.3 Diseño de la Arquitectura del Sistema

La arquitectura del módulo de actividades desarrollado para la aplicación móvil tiene el mismo enfoque estructural de tres capas utilizado en el sistema web (Balanza Web), garantizando coherencia, separación de responsabilidades y escalabilidad. Sin embargo, en este caso la capa de presentación corresponde a una aplicación móvil construida con Flutter, la cual interactúa directamente con los servicios expuestos por el backend mediante solicitudes HTTP al estilo REST. Esta capa se organiza siguiendo el patrón utilizado en Flutter basado en Widgets, Views y Providers, permitiendo una estructura clara entre interfaz, gestión de estado y lógica de interacción con los servicios. Asimismo, se incorpora un servicio especializado para el manejo de almacenamiento seguro (secure storage), donde se preserva la información sensible como tokens

de autenticación generado por el backend, además del uso de paquetes externos necesarios para la obtención de geolocalización, manejo de estado y comunicación con la API.

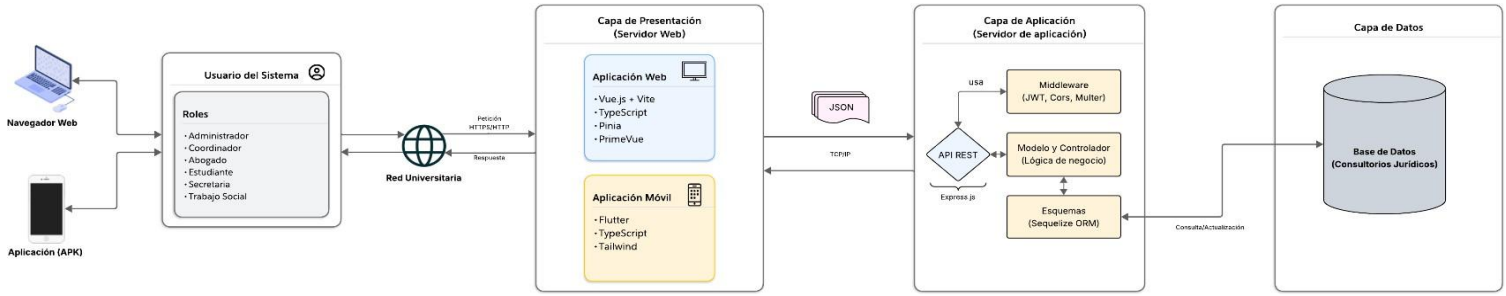
Por otra parte, la capa de aplicación o lógica de negocio reside en el servidor, compartido con el sistema web. Este backend, desarrollado con Node.js y Express.js, mantiene el patrón MVC(sin vista), donde los modelos definen el esquema de datos gestionado por el ORM Sequelize que es responsable de traducir los objetos de Javascript en estructuras y operaciones SQL sobre MySQL y los controladores procesan las solicitudes provenientes de las rutas, otorgando respuestas en formato JSON hacia el cliente móvil. La API expone funcionalidades como la autenticación con JWT, la consulta de actividades asignadas, el registro de entrada y salida con datos de geolocalización y la actualización del estado de una actividad todo mediante endpoints protegidos. También se emplean middlewares esenciales como validación de permisos, manejo de CORS para permitir la comunicación desde dispositivos móviles para la recepción adecuada de la información enviada desde la aplicación.

Finalmente, la capa de datos permanece encargada del almacenamiento persistente de la información relacionada con los usuarios internos, actividades y registros generados desde la aplicación móvil. Para ello se utiliza MySQL como motor de base de datos, comunicado directamente con el backend a través de Sequelize, que interpreta y transforma las operaciones lógicas en sentencias SQL. La inicialización de la conexión y sincronización de modelos se gestiona por medio de archivos de configuración definidos dentro del servidor, garantizando la integración adecuada con el resto del sistema.

Esta arquitectura permite mantener una comunicación eficiente entre la aplicación móvil y el sistema existente, permitiéndonos asegurar sobre todo la consistencia de datos, seguridad en la transmisión de información sensible y un diseño modular que facilita futuras ampliaciones del módulo.

Figura 11

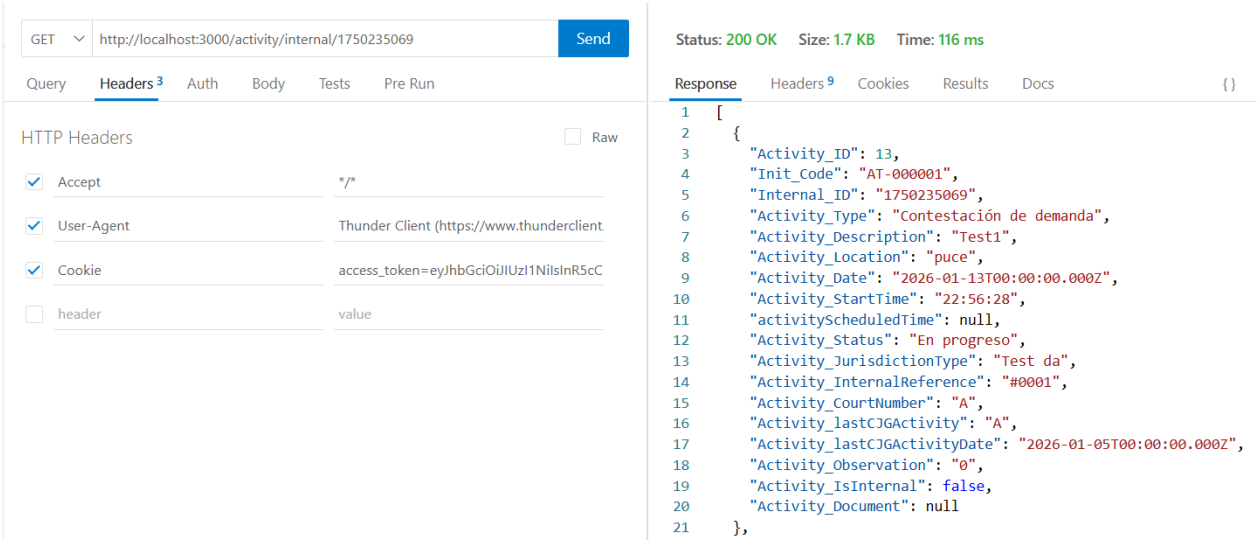
Arquitectura del sistema



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

Figura 12

Ejemplo de petición al backend para la consulta de actividades



Nota: La figura muestra una petición GET al backend para obtener las actividades de un usuario, enviando su identificador y la cookie de sesión. El sistema responde con un JSON que contiene la información de las actividades registradas. Fuente: Elaboración Propia

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

En cuanto al diseño de la interfaz de usuario U.I de la aplicación móvil, se centra

en el módulo de Control y Registro de Actividades, el cual es parte del flujo operativo del Consultorio Jurídico, permitiendo a los estudiantes y abogados visualizar las actividades asignadas y registrar su cumplimiento por medio de geolocalización y su respectivo control de tiempo.

En este sentido, se han diseñado las interfaces principales de la aplicación móvil, comprendiendo la pantalla de inicio de sesión (login) , la vista principal de visualización de actividades, y las pantallas de registro de entrada y registro de salida de una actividad. Estas interfaces permiten al usuario autenticarse en el sistema, consultar las actividades asignadas, registrar inicio y finalización de cada una de manera estructurada. El diseño de las pantallas se profundizará en el capítulo IV, en concordancia con el desarrollo del sistema siguiendo la metodología de prototipado evolutivo

Por otra parte, la idea general del diseño es ofrecer una aplicación móvil que sea fácil de usar para los usuarios finales, utilizando colores institucionales, íconos representativos y secciones de contenido bien definidas. Se adopta un diseño moderno y minimalista, orientado a la correcta visualización de la información relevante y a la ejecución rápida de acciones, considerando que el uso de la aplicación se realiza en contextos de movilidad y tiempo real.

Figura 13

Diseño mockup (bosquejo): Pantalla de inicio de sesión (Login)

Nota: La figura representa un bosquejo inicial (mockup) de la pantalla de inicio de sesión donde el usuario ingresa sus credenciales (correo y contraseña) para autenticarse. Fuente: Elaboración propia.

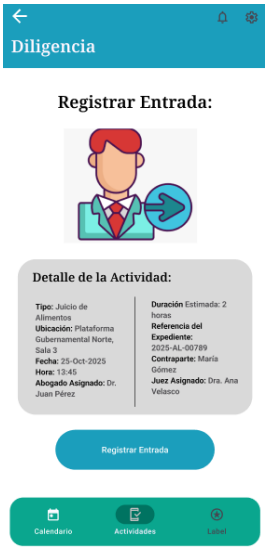
Figura 14

Diseño mockup (bosquejo): Pantalla de visualización de actividades

Nota: La figura representa un bosquejo inicial (mockup) de la pantalla de consulta de actividades asignadas, organizadas por secciones como actividades de hoy y actividades próximas. Se diseño con un formato de tarjeta que facilita la lectura rápida de información clave como tipo de actividad, fecha, hora y ubicación. Fuente: Elaboración propia.

Figura 15

Diseño mockup (bosquejo): Pantalla de registro de entrada



Nota: La figura representa un bosquejo inicial (mockup) de la pantalla de Registro de Entrada, donde muestra el detalle completo de una actividad seleccionada, incluyendo información relevante como tipo de diligencia, ubicación, fecha y responsable. Por medio de esta pantalla, el usuario puede registrar el inicio de actividad, dando paso al control de tiempo y ubicación. Fuente: Elaboración propia.

Figura 16

Diseño mockup (bosquejo): Pantalla de registro de salida

Nota: La figura representa un bosquejo inicial (mockup) de la pantalla de Registro de Salida, permite finalizar una actividad previene iniciada, el usuario selecciona el estado final de la actividad y registra un resumen u observación, completando el ciclo de registro de la actividad dentro del sistema. Fuente: Elaboración propia.

CAPÍTULO IV: DESARROLLO DE LA APLICACIÓN

4.1 Preparación del Entorno de Desarrollo

4.1.1 Entorno del Desarrollo (Cliente Móvil)

Para el desarrollo de la aplicación móvil del CJ de la PUCE, se decidió utilizar el framework Flutter, ya que la Dirección de Informática estableció como lineamiento institucional obligatorio del desarrollo de aplicaciones móviles multiplataforma sea realizado utilizando Fluter, garantizando de esta forma la eficiencia, mantenibilidad y compatibilidad con dispositivos Android.

Flutter emplea el lenguaje de programación Dart, el cual facilita el desarrollo de interfaces reactivas, un alto rendimiento gráfico y una estructura clara del código, que son aspectos fundamentales para aplicaciones móviles orientadas a la interacción constante del usuario.

El entorno de desarrollo fue configurado en un equipo local, utilizando como editor de código Visual Studio Code , por su ligereza extensibilidad y compatibilidad nativa con Flutter y Dart. Adicionalmente, se configuró el SDK de Android y un emulador Android, con el fin de ejecutar, depurar y validar el correcto funcionamiento de la aplicación durante todo el proceso de desarrollo. En ciertos escenarios, también se realizaron pruebas en dispositivos físicos para verificar el comportamiento real de la aplicación.

La instalación de Flutter se verificó por medio de la herramienta Flutter Docotr, la cual permite comprobar que todas las dependencias necesarias como: SDK, emuladores, plugins y variables de entorno; se encuentren correctamente configuradas antes de iniciar el desarrollo del proyecto.

Finalmente, se creó el proyecto base de Flutter utilizando la estructura estándar

recomendada por el framework, sobre la cual se implementaron progresivamente las funcionalidades definidas en los casos de uso del sistema, como lo son la gestión de sesión, consulta de actividades, registro de asistencia y funcionamiento en modo offline.

4.1.2 Dependencias y Librerías Utilizadas (Flutter)

Para el desarrollo de la aplicación móvil se emplearon diversas dependencias y librerías propias del ecosistema Flutter, las cuales permitieron implementar de manera eficiente las funcionalidades definidas en los requerimientos funcionales y casos de uso establecidos en el Capítulo III.

La selección de estas librerías fueron consideradas con criterios como estabilidad, soporte comunitario, compatibilidad multiplataforma, seguridad y adecuación al desarrollo de aplicaciones móviles con soporte offline.

Librerías para interfaz de Ususario y Experiencia del Usuario:

- Cupertino_icons

  Librería utilizada para incorporar íconos de distintos estilos, permitiendo una interfaz visual cohenrete y adaptada a los lineamientos de diseño móvil moderno

- smooth_page_indicator

  Se empleó para mejorar la experiencia de navegación entre pantallas con desplazamiento horizontal, facilitando la orientación del usuario dentro de flujos visuales específicos de la aplicación

- table_calendar

  Permite la visualización estructurada de actividades por medio de un calendario interactivo, facilitando la consulta de fechas asociadas a diligencias, audiencias u otras actividades asignadas a los estudiantes y abogados.

Librerías para Consumo de Servicios Web

- http

  Utilizada para el consumo de APIs REST del sistema backend, permitiendo realizar solicitudes HTTP (GET,POST) para funcionalidades de Autenticación de usuarios, Consulta de actividades asignadas, Registro de entrada y salida de actividades. Esta librería actúa como el principal medio de comunicación entre la aplicación móvil y el sistema web de soporte

Librerías para Gestión de Estado

- provider

  Implementada para la gestión del estado de la aplicación, permitiendo compartir información de sesión entre pantallas, Mantener sincronizados los datos de usuario y actividades. El uso de provider contribuye a una arquitectura más mantenible y escalable dentro del proyecto.

Librerías para Manejo de Conectividad y Modo Offline

- connectivity_plus

  Se utiliza para detectar el estado de conexión del dispositivo móvil, permitiendo identificar si el usuario se encuentra en línea u operando en modo offline, pese a que este no seleccione el modo offline.

- shared_preferences

  Permite el almacenamiento local de datos no sensibles, tales como configuraciones básicas y datos temporales necesarios para el funcionamiento offline de la aplicación.

Librerías para Seguridad y Gestión de Sesiones

- flutter_secure_storage

  Utilizada para el almacenamiento seguro de información sensible, principalmente el

token JWT generado durante el proceso de autenticación. Esta librería garantiza que los datos críticos no sean almacenados en texto plano dentro del dispositivo.

- jwt_decoder

  Permite decodificar y validar el token JWT, facilitando el control de la sesión del usuario y manejo adecuado del cierre de sesión.

Librerías de Soporte y Utilidades

- intl

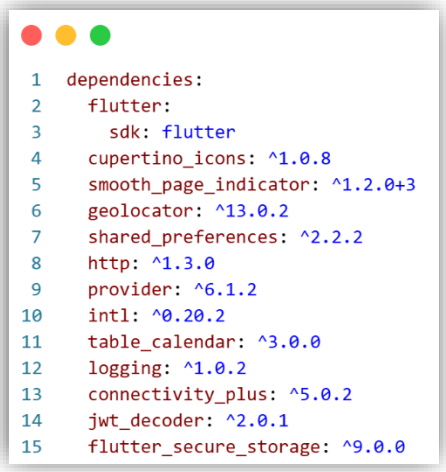
  Utilizado para el manejo y formateo de fechas y horas, especialmente relevante en el registro de entrada y salida de actividades y en la presentación de información temporal al usuario.

- Logging

  Implementada para el registro de eventos y mensajes de depuración, facilitando el monitoro del comportamiento de la aplicación durante s ejecución y el proceso de pruebas

Figura 17

Configuración de dependencias del proyecto Flutter (pubspec.yaml)



Nota: La figura muestra el archivo pubspec.yaml, donde se definen las dependencias utilizadas en la aplicación móvil para la gestión de estado, consumo de servicios REST, geolocalización, almacenamiento local, conectividad y seguridad de la información.

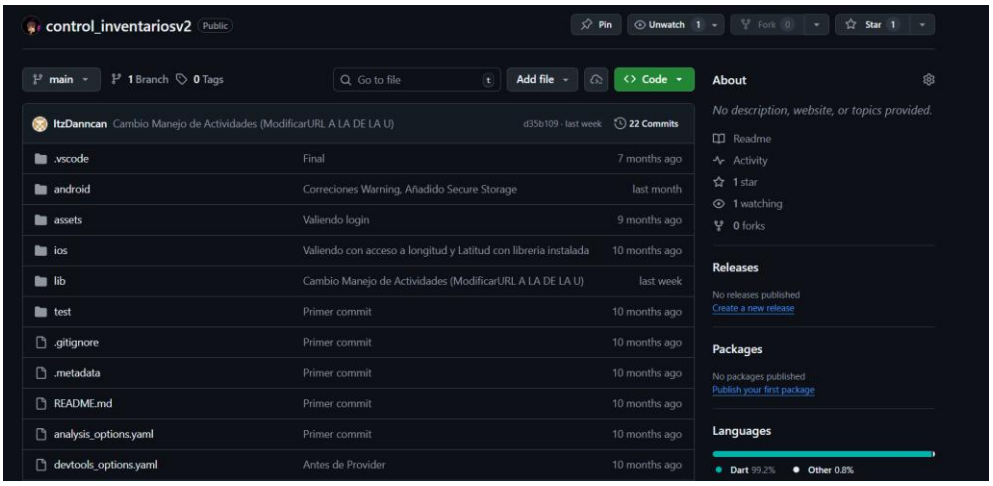
Fuente: Elaboración propia.

4.1.3 Control de Versiones

Durante el desarrollo de la aplicación móvil, se aplicaron prácticas formales de control de versiones, con el objetivo de garantizar la trazabilidad del código, la organización del proyecto y la integridad de los cambios realizados a lo largo del ciclo del desarrollo. Como herramienta en función de control de versiones, se utilizó Git y el código fuente se guardó en Github. El repositorio de la aplicación móvil fue gestionado de forma independiente del backend y del sistema web.

Figura 18

Figura referencial del proyecto en Github



Nota: La figura corresponde a una captura de pantalla donde se muestra la página principal del proyecto en GitHub. 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 móvil 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 Arquitectura de la Aplicación Móvil

4.2.1 Estructura del Proyecto Flutter

El proyecto Flutter se organizó mediante una estructura modular basada en la separación de responsabilidades, lo cual facilita el mantenimiento, la escalabilidad y la compresión del sistema.

La organización del proyecto responde a una arquitectura por capas, donde se distingue claramente la lógica de presentación, la gestión de estado, la lógica de negocio y los servicios de soporte, alineándose con principios de diseño similares a MVC y Clean Architecture, adaptado al entorno móvil.

4.2.1.1 *Estructura General del Proyecto*

El proyecto Flutter presenta la siguiente estructura general de directorios:

```
control_productividad/
├── lib/
│   ├── main.dart
│   ├── providers/
│   ├── services/
│   ├── views/
│   └── widgets/
├── android/
├── ios/
├── assets/
├── test/
└── pubspec.yaml
```

Esta organización permite separar claramente las responsabilidades de cada componente del sistema, mejorando la legibilidad del código y facilitando la

incorporación de nuevas funcionalidades.

4.2.1.2 *Directorio lib/ - Código Fuente Principal*

El directorio lib/ contiene la totalidad del código fuente de la aplicación móvil. En su interior, el proyecto se organiza en subdirectorios especializados según la función que cumplen dentro de la arquitectura.

4.2.1.3 *Archivo main.dart- Punto de Entrada de la Aplicación*

El archivo main.dart constituye el punto de entrada de la aplicación móvil. En este archivo se realiza la inicialización general de la aplicación móvil, la configuración global del tema visual, la inicialización de los proveedores de estado mediante multiprovider y la definición de la pantalla inicial del sistema. Este archivo actual como el núcleo de arranque de la aplicación y establece las bases para la gestión de estado y navegación inicial

Figura 19

Archivo main.dart como punto de entrada de la aplicación Flutter

```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'providers/activity_provider.dart';
import 'views/splash_screen.dart';

void main() {
  runApp(
    MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => ActivityProvider()),
      ],
      child: const MyApp(),
    ),
  );
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Gestión de Actividades',
      theme: ThemeData(primarySwatch: Colors.teal),
      home: const SplashScreen(),
    );
  }
}
```

Nota: La figura muestra el archivo main.dart, donde se configura la inicialización de la aplicación móvil, la gestión global del estado mediante providers y la definición de la pantalla inicial del sistema. Fuente: Elaboración Colaborativa.

4.2.1.4 *Capa de Gestión de Estado – providers/*

El directorio providers/ contiene las clases responsables de la gestión del estado de la aplicación, implementando el patrón Provider. En esta capa se centraliza la lógica que controla los datos compartidos entre las distintas pantallas tales como listado de actividades asignadas, estado de sincronización, actualización de información mostrada en la interfaz. El provider permite notificar automáticamente a la interfaz cuando ocurre un cambio en los datos, garantizando una experiencia reactiva y consistente para el usuario.

4.2.1.5 *Capa de Servicios – services/*

El directorio services/ encapsula la lógica relacionada con servicios externos y funcionalidades transversales de la aplicación. Esta capa se implementa servicios como comunicación con la API REST del sistema backend, Almacenamiento seguro de información sensible, manejo de autenticación y tokens de sesión. Esta separación permite desacoplar la lógica del resto de la aplicación, facilitando la reutilización de código y mejorando la seguridad.

4.2.1.6 *Capa de Presentación – views/*

El directorio views/ contiene las pantallas principales de la aplicación móvil, cada una asociada a una funcionalidad específica del sistema. Entre las vistas implementadas se encuentran la pantalla de inicio y validación de sesión, la pantalla de

autenticación de usuarios, listado de actividades asignadas, vista de calendario de actividades, pantallas de registro de entrada y salida, entre otras. Cada vista se comunica con los providers correspondientes para obtener y actualizar la información necesaria, manteniendo una separación clara entra la interfaz y la lógica de negocio.

4.2.1.7 *Componentes Reutilizables– widgets/*

El directorio widgets/ agrupa componentes reutilizables de la interfaz gráfica, diseñados para ser utilizados en múltiples pantallas. Estos widgets permiten mantener consistencia visual en la aplicación, reducir duplicados de código, facilitar modificaciones globales en la interfaz. Como ejemplos de componentes reutilizables tenemos las tarjetas de actividades y barra de navegación personalizadas.

Figura 20

Widget tarjeta de ejemplo de Audiencia

Nota: Este componente muestra información relevante de una audiencia judicial, como el nombre del evento, ubicación, fecha y hora, además de un botón de acción (“Ver más”) que permite acceder al detalle completo. El widget forma parte de los componentes reutilizables de la interfaz gráfica y es utilizado en diferentes pantallas

para mantener consistencia visual y facilitar la navegación del usuario. Fuente: Elaboración propia.

4.2.1.8 *Directorios de Plataforma – Android/ e ios/*

Los directorios Android/ e ios/ contienen la configuración específica para cada sistema operativo móvil. En Android/ se encuentran los archivos de configuración de Gradle, permisos de geolocalización y ajustes del manifiesto. En ios/ se aloja el proyecto Xcode, junto con la configuración de permisos y ajustes del sistema operativo.

4.2.1.9 *Recursos Estáticos – assets/*

El directorio assets/ almacena recursos gráficos utilizados por la aplicación, como imágenes, íconos y otros elementos visuales necesarios para la interfaz de usuario

4.2.1.10 *Archivo pubspec.yaml – Configuración del Proyecto*

El archivo pubspec.yaml define las dependencias del proyecto, los recursos estáticos y la configuración general de la aplicación Flutter. Este archivo es fundamental para la correcta compilación y ejecución del proyecto.

4.2.2 Comunicación con el Backend

La aplicación móvil se comunica con el sistema web del Consultorio Jurídico por medio de una API REST, la cual actúa como un sistema de soporte encargado de proveer información y recibir los registros generados desde la aplicación móvil.

Es importante recalcar que el backend no constituye el objetivo principal de este proyecto, sino que se describe únicamente como un componente externo que permite la interoperabilidad y correcto funcionamiento de la aplicación móvil.

4.2.2.1 *Modelo de Comunicación Cliente-Servidor*

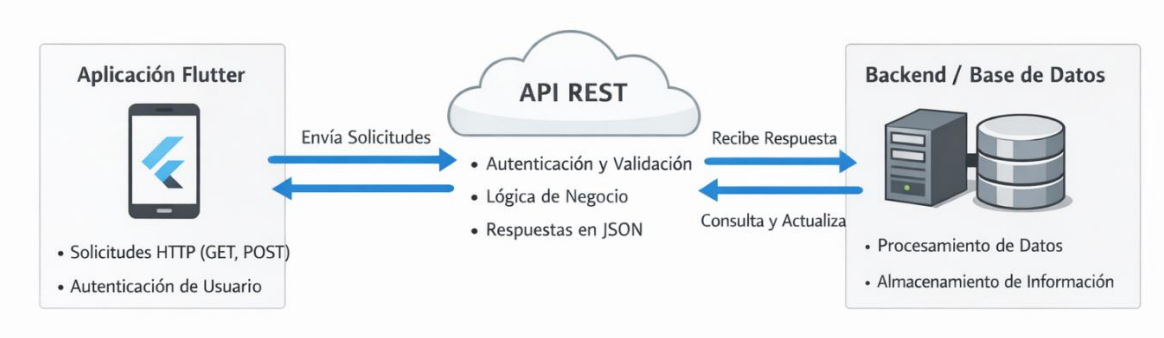
La comunicación entre la aplicación móvil y el backend sigue un modelo cliente-servidor donde la aplicación móvil actúa como cliente, el sistema web existene expone servicios mediante endpoints REST. El intercambio de información se realiza a través de solicitudes HTTP utilizando el formato JSON. Este enfoque permite una integración, flexible, escalabre y desacoplada entre los diferentes componentes del sistema

4.2.2.2 *Consumo de la API REST*

El consumo de los servicios web se implementa utilizando la librería http, la cual permite realizar solicitudes HTTP de tipo GET y POST desde la aplicación móvil. Cada solicitud incluye, la URL del endpoint correspondiente, encabezados HTTP para la autenticación, el cuerpo de la solicitud cuando es necesario, cookie de autenticación y manejo de códigos de respuesta HTTP. La lógica de comunicación se encuentra encapsulada dentro de la capa de servicios, lo que evita que la interfaz gráfica dependa directamente de los detalles técnicos del backend.

Figura 21

Comunicación entre la aplicación móvil Flutter y la API REST



Nota: La figura ilustra el modelo de comunicación cliente–servidor entre la aplicación

móvil desarrollada en Flutter y el sistema backend, mediante el consumo de servicios REST para autenticación, consulta de actividades y registro de asistencia. Fuente: Elaboración propia.

4.2.2.3 *Endpoints Utilizados por la Aplicación Móvil*

La aplicación móvil utiliza únicamente los endpoints necesarios para cumplir con el objetivo de la aplicación, entre los principales servicios se encuentran la autenticación de usuarios que permite validar las credenciales del usuario y obtener un token JWT para la gestión de sesión, Recuperación de contraseña que nos permite en caso de haber olvidado la contraseña recuperarla por medio de un token de verificación al correo registrado y permitiendo asignar nuevamente la contraseña correspondiente, Consulta de actividades asignadas permite obtener el listado de actividades asociadas al usuario autenticado según su rol, Registro de entrada y salida de actividades permite enviar al backend la información de inicio y finalización de una actividad incluyendo fecha, hora y geolocalización. Estos endpoints permiten cubrir las funcionalidades principales de la aplicación móvil sin exponer lógica innecesaria del sistema backend.

4.2.2.4 *Autenticación y Manejo de Seguridad*

La autenticación se realiza mediante tokens JWT, los cuales son generados por el backend tras una autenticación exitosa. Una vez recibido el token, se almacena de forma segura en el dispositivo utilizando fluter_secure_storage, se incluye como cookie en el encabezado Authorization de cada solicitud HTTP y se valida su vigencia antes de permitir el acceso de funcionalidades protegidas. Este mecanismo garantiza que únicamente usuarios identificados puedan acceder a la información y registrar actividades desde la aplicación móvil.

4.2.2.5 *Manejo de Respuestas y Errores*

La aplicación móvil gestiona las respuestas del backend de acuerdo con los códigos HTTP recibidos, permitiendo confirmar operaciones exitosas, detectar errores de autenticación, informar fallos de comunicación y activar el modo offline cuando no existe conectividad. Este manejo estructurado de errores contribuye a una experiencia de usuario más robusta y confiable.

4.2.2.6 *Integración con el Modo offline*

Cuando la aplicación detecta la ausencia de conexión a internet, las solicitudes al backend no se ejecutan inmediatamente, en su lugar la información se almacena localmente en el dispositivo, los registros pendientes quedan marcados para sincronización y una vez establecida la conectividad, los datos se envían automáticamente al backend.

4.3 Funcionalidades de la Aplicación Móvil

4.3.1 Gestión de Sesión

La gestión de sesión constituye una de las funcionalidades fundamentales de la aplicación móvil, debido a que permite controlar el acceso seguro de los usuarios al sistema y garantiza que únicamente estudiantes y abogados previamente registrados en el sistema web puedan utilizar la aplicación.

4.3.1.1 Inicio de Sesión de Usuarios

La aplicación móvil no permite el registro de nuevos usuarios, ya que las credenciales son gestionadas exclusivamente por el sistema web “Balanza Web”. En este contexto, la aplicación móvil actúa únicamente como cliente consumidor de los servicios

de autenticación expuestos por la API REST.

El proceso de inicio de sesión se realiza de la siguiente manera, el usuario ingresa sus credenciales (usuario y contraseña) desde la pantalla de inicio de sesión, la aplicación envía las credenciales al backend mediante una solicitud HTTP, el backend valida la información y en caso de ser correcta retorna un token JWT, la aplicación recibe el token y lo almacena de forma segura en el dispositivo. Este mecanismo garantiza que la autenticación se realice de manera centralizada y segura.

Figura 22

Pantalla de inicio de sesión de la aplicación móvil

Nota: La figura muestra la interfaz de inicio de sesión de la aplicación móvil Flutter, donde los usuarios ingresan sus credenciales para autenticarse mediante la API REST del sistema backend. Fuente: Elaboración propia.

4.3.1.2 Almacenamiento Seguro del Token de Sesión

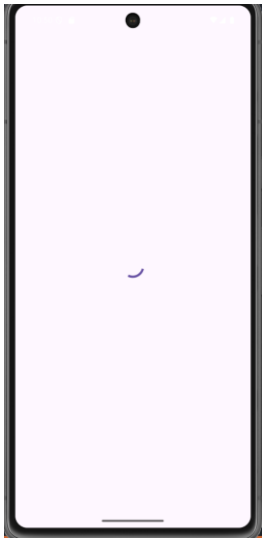
Una vez autenticado el usuario, el token JWT recibido es almacenado utilizando la librería flutter_secure_storage, lo que permite guardar información sensible de forma cifrada dentro del dispositivo móvil. Es importante destacar que la contraseña la contraseña del usuario no se almacena en el dispositivo, el token JWT representa la sesión activa del usuario, el almacenamiento seguro evita accesos no autorizados a la información de sesión. Este enfoque refuerza la seguridad de la aplicación y protege los datos sensibles del usuario.

4.3.1.3 Validación de Sesión mediante SplashScreen

Al iniciar la aplicación, se presenta una pantalla inicial (SplashScreen) cuya función es verificar el estado de la sesión del usuario antes de permitir el acceso a las funcionalidades del sistema. Durante esta etapa, la aplicación ejecuta un proceso de validación que consiste en leer el token almacenado en el dispositivo mediante el servicio de almacenamiento seguro, mostrar la pantalla de inicio durante un breve intervalo. Evalúa la existencia del token, en este caso existen dos casos, si el token existe es redirigido automáticamente a la pantalla principal, si el token no existe o el token expiró el usuario es redirigido a la pantalla de inicio de sesión. Este mecanismo permite mantener la sesión activa entre ejecuciones de la aplicación sin requerir que el usuario se autentique repetidamente.

Figura 23

Validación de sesión mediante SplashScreen



Nota: La figura ilustra el flujo de validación de sesión ejecutado en la pantalla SplashScreen, donde se verifica la existencia del token de autenticación para redirigir al usuario a la pantalla principal o al inicio de sesión. Fuente: Elaboración propia.

4.3.1.4 Persistencia de la Sesión

La aplicación implementa un mecanismo de persistencia de sesión que permite al usuario mantenerse autenticado mientras el token de sesión esté almacenado en el dispositivo y este no expiere. Este comportamiento mejora la experiencia de usuario, ya que evita autenticaciones repetitivas y permite un acceso rápido a las funcionalidades principales de la aplicación móvil.

4.3.1.5 Cierre de Sesión (Logout)

La aplicación permite al usuario cerrar sesión de manera manual desde la interfaz por medio del menú superior. Al ejecutar esta acción se elimina el token JWT almacenado localmente, se limpia el estado de la sesión en la aplicación, el usuario redirigido a la pantalla de inicio de sesión.

Figura 24

Opción de cierre de sesión en la aplicación móvil

Nota: La figura muestra la opción de cierre de sesión disponible en la aplicación móvil, la cual permite eliminar la información de sesión almacenada localmente y redirigir al usuario a la pantalla de autenticación. Fuente: Elaboración propia.

4.3.1.6 Recuperación de Contraseña

Desde la pantalla de inicio de sesión la aplicación ofrece la opción de recuperación de contraseña. El proceso de recuperación comienza cuando el usuario da click en la opción de recuperar contraseña, luego el usuario ingresa el mail, el cual es validado que exista dentro de los registros de usuarios, el backend envía un código de verificación al correo que válido por 15 minutos, el usuario ingresa el código que recibió por mail y si este es correcto avanza al siguiente paso caso contrario en donde el código haya expirado o sea incorrecto lo puede volver a intentar o a su vez reenviar (máximo 3 veces), el siguiente paso a realizar es ingresar la nueva contraseña para actualizarla y se redirige al loggin para ingreso de las credenciales con la nueva contraseña.

Figura 25

Opción de cierre de sesión en la aplicación móvil

Nota: La figura muestra la interfaz de recuperación de contraseña de la aplicación móvil. Esta pantalla permite al usuario iniciar el proceso de restablecimiento de credenciales mediante el ingreso de su correo electrónico institucional, el cual será validado por el sistema antes de continuar con el flujo de recuperación. Fuente: Elaboración propia.

4.3.2 Consulta de Actividades

La consulta de actividades constituye una funcionalidad central de la aplicación móvil, ya que permite a los estudiantes y abogados visualizar las actividades asignadas dentro del Consultorio Jurídico Gratuito, como diligencias, audiencias u otras actividades.

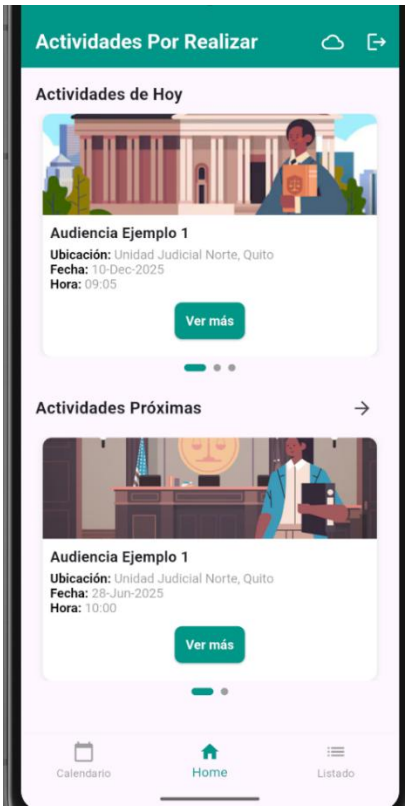
4.3.2.1 Obtención de Actividades desde el servidor

Cuando la aplicación dispone de conexión a internet, la consulta de actividades se realiza por medio del consumo de un servicio REST expuesto por el sistema backend. Es así como la aplicación obtiene el identificador del usuario previamente almacenado en el dispositivo, recupera el token de sesión almacenado de forma segura, envía una solicitud

HTTP al backend para solicitar las actividades asociadas al usuario autenticado, cabe recalcar que esta petición se encarga de filtra únicamente las actividades que requieren de la aplicación móvil (actividades externas), recibe información en formato JSON, correspondiente a las actividades asignadas.

Figura 26

Consulta de actividades desde la aplicación móvil



Nota: La figura muestra el consumo del servicio REST para la consulta de actividades asignadas al usuario, evidenciando la obtención de información desde el sistema backend mediante una solicitud HTTP autenticada. Fuente: Elaboración propia.

4.3.2.2 Procesamiento y Formateo de la Información

Una vez recibida la información desde el backend, la aplicación móvil procesa los datos con el objetivo de preséntalos de forma clara y comprensible para el usuario. Cada

actividad contiene información relevante como :

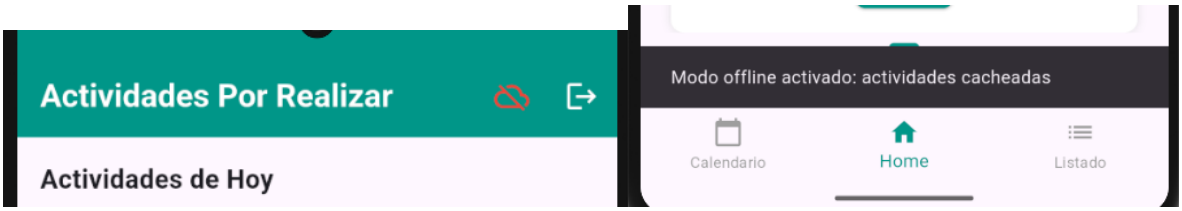
- Identificador de la actividad.
- Tipo o título de la actividad.
- Fecha programada.
- Hora de inicio.
- Ubicación asignada.
- Estado actual del registro.

4.3.2.3 Modes de Opreación: Online y Offline

La funcionalidad de consulta de actividades fue diseñada para operar tanto de modo online como en modo offline, garantizando la disponibilidad de la información incluso cuando no existe conexión a internet. En el caso del modo online, la aplicación descarga las actividades desde el servidor y almacena la información localmente como respaldo. Modo offline, en ausencia de conectividad, la aplicación utiliza los datos almacenados en el dispositivo, permitiendo al usuario consultar las actividades previamente descargadas sin realizar llamadas al servidor. Este comportamiento se integra con el mecanismo de almacenamiento loca, fortaleciendo la continuidad operativa de la aplicación

Figura 27

Consulta de actividades en modo offline



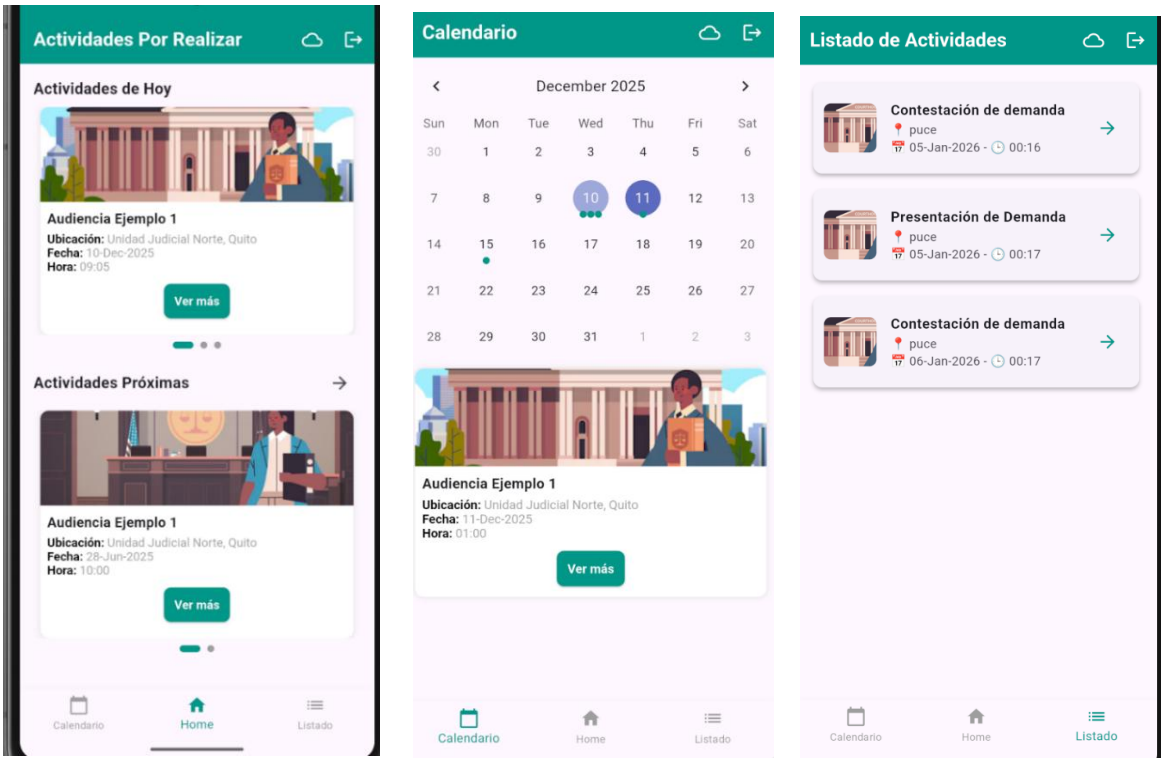
Nota: La figura ilustra la visualización de actividades almacenadas localmente en el dispositivo cuando la aplicación opera sin conexión a internet. Fuente: Elaboración propia.

4.3.2.4 Visualización

Las actividades son presentadas de tres formas distintas, la primera forma es mediante la pantalla principal (Home). La pantalla Home es la vista central de la aplicación y muestra al usuario un resumen organizado de sus actividades pendientes. Desde aquí puede visualizar las actividades del día, las próximas fechas y acceder rápidamente al detalle de cada una.

Figura 28

Visualización de Actividades - Pantalla Principal



Nota: La figura presenta las tres modalidades de visualización de actividades disponibles en el sistema. En primer lugar, la pantalla principal (Home) muestra las actividades organizadas en secciones como “Actividades de Hoy” y “Actividades Próximas”, permitiendo al usuario visualizar de forma resumida sus compromisos pendientes y acceder al detalle de cada actividad. En segundo lugar, la vista de calendario permite visualizar las

actividades programadas de manera mensual, facilitando la identificación de días con eventos registrados y apoyando la planificación de diligencias y audiencias futuras. Finalmente, la vista de listado presenta todas las actividades registradas en formato de lista cronológica, mostrando información relevante como tipo de actividad, ubicación, fecha y hora, y permitiendo el acceso directo al detalle de cada registro. Fuente: Elaboración propia.

4.3.2.5 Almacenamiento en Caché Local

Todas las actividades obtenidas desde el servidor son almacenadas localmente en el dispositivo como mecanismo de respaldo. Este almacenamiento permite acceder a las actividades sin conexión, reducir la dependencia constante del servidor y mejorar el rendimiento de la aplicación. Este enfoque refuerza el soporte al modo offline.

4.3.3 Consulta de Actividades

El registro de asistencia constituye una funcionalidad clave de la aplicación móvil, ya que permite a los estudiantes y abogados registrar el inicio y la finalización de las actividades asignadas, garantizando un control preciso del tiempo y la ubicación en la que se realizan las actividades

4.3.3.1 Registro de Entrada a una Actividad

El proceso de registro de entrada se habilita únicamente cuando la actividad cumple con las validaciones establecidas por el sistema. Al seleccionar una actividad disponible la aplicación verifica que la actividad corresponda a la fecha actual, el registro se realice dentro de un rango de tiempo permitido y que el usuario cuente con los permisos necesarios para acceder a la ubicación del dispositivo

Una vez superadas las validaciones, el proceso de registro de entrada se desarrolla de la siguiente manera:

1. La aplicación obtiene la ubicación geográfica del usuario mediante el GPS del dispositivo.
2. Se establece un tiempo máximo de espera para la obtención de la ubicación
3. Se detecta el estado de conectividad del dispositivo
4. Se construye el registro de entrada con la información correspondiente
5. El registro se envía al servidor o se almacena localmente según el modo de operación

La información registrada incluye datos como la hora de registro, la ubicación geográfica y la confirmación de puntualidad

Figura 29

Pantalla de registro de entrada de una actividad

Nota: La figura muestra la interfaz utilizada para el registro de entrada de una actividad, donde se valida la fecha, el horario permitido y se obtiene la ubicación geográfica del usuario antes de completar el registro. Fuente: Elaboración propia.

4.3.3.2 Registro de Salida de una Actividad

El registro de salida se realiza al finalizar una actividad y requiere información adicional que permite documentar el resultado de esta. Para ello el usuario debe seleccionar el estado final de la actividad (completada, suspendida, cancelada o pospuesta), ingresar una observación o descripción de la actividad realizada y permitir el acceso de la ubicación del dispositivo

Una vez superadas las validaciones al igual que en el registro de entrada, el proceso de registro de salida se desarrolla de la siguiente manera :

1. La aplicación valida los campos obligatorios (resultado y observación)
2. Obtiene la ubicación geográfica del usuario
3. Se detecta el estado de conectividad del dispositivo
4. Construye el registro de salida con estado y observaciones
5. Envía el registro al servidor o almacenamiento local según el modo de operación
6. Actualización del estado de la actividad en la aplicación

Una vez completado el registro, la actividad se elimina del listado activo del usuario, evitando registros duplicados

Figura 30

Pantalla de registro de salida de una actividad

Nota: La figura presenta la interfaz de registro de salida, donde el usuario selecciona el estado final de la actividad e ingresa una observación antes de completar el registro. Fuente: Elaboración propia.

4.3.4 Modo Offline y Sincronización Automática

La aplicación móvil implementa un mecanismo de funcionamiento en modo offline que permite a los usuarios consultar actividades previamente sincronizadas y registra información aun cuando no exista conexión a internet. Para ello, los datos obtenidos desde el servidor se almacenan localmente en el dispositivo, mientras que los registros generados en ausencia de conectividad se conservan como pendientes de sincronización. Una vez que la aplicación detecta la restauración de la conexión, estos registros son enviados automáticamente al sistema backend, garantizando la integridad y consistencia de la información. Este enfoque nos permite asegurar la continuidad operativa de la aplicación en entornos de conectividad limitada, además de cumplir con uno de los requisitos del Consultorio Jurídico de la PUCE.

4.4 Arquitectura de la Aplicación Móvil

La aplicación móvil desarrollada, incorpora mecanismos de seguridad orientada a proteger la información del usuario y garantizar un acceso controlado al sistema. La autenticación se gestiona por medio de tokens JWT emitidos por le backend, los cuales son almacenados de forma segura en el dispositivo utilizado mecanismo de almacenamiento cifrado, evitando el uso de texto plano y el resguardo de credenciales sensibles como contraseñas. Adicionalmente, la aplicación valida la sesión del usuario al iniciar su ejecución, controla el acceso a las funcionalidades según el estado de autenticación y gestiona de manera segura el cierre de sesión mediante la eliminación de la información local. Así mismo, el manejo de datos en modo offline y su posterior sincronización automática se realiza preservando la integridad de la información registrada. En conjunto, estas medidas permiten asegurar la confidencialidad, integridad y disponibilidad de los datos, alineándose con buenas prácticas de seguridad en aplicaciones móviles.

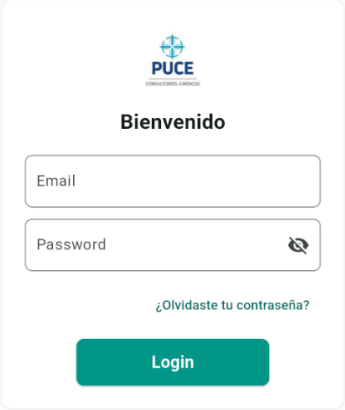
4.5 Aplicación de la Metodología (Prototipado Evolutivo)

4.5.1 Construcción del Prototipo Inicial

En concordancia con la metodología del prototipado evolutivo, se desarrolló una primera versión funcional de la aplicación móvil destacando el inicio de sesión, interfaz principal de actividades y registro de entrada de actividad. Cabe recalcar que si bien en la pantalla de inició de sesión se encontraba la opción de inicio de sesión, esta funcionalidad no estaba implementada en el prototipo inicial, de igual manera se tenía planeado realizar una única pantalla en donde se registre tanto la entrada como la salida, pero se terminó descartando esta opción, debido a que por el flujo de la aplicación era necesario realizarlo como dos pantallas individuales.

Figura 31

Prototipo Inicial: Inicio de sesión

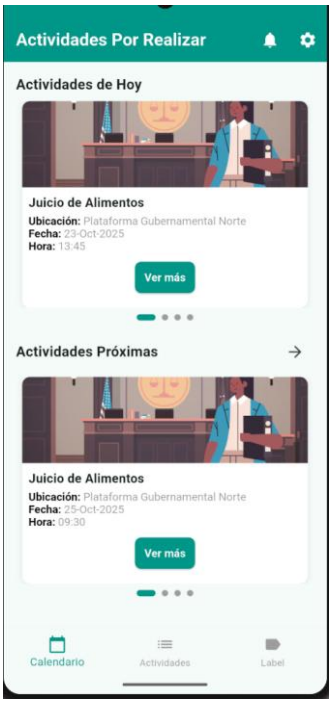


Nota: La figura muestra el prototipo inicial del inicio de sesión de los usuarios.

Fuente: Elaboración Propia.

Figura 32

Prototipo Inicial: Pantalla de inicio

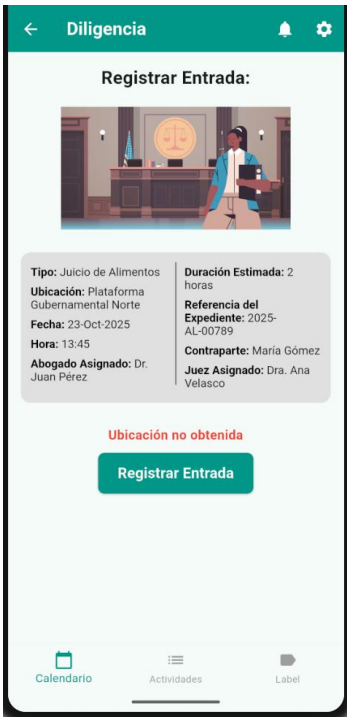


Nota: La figura muestra el prototipo inicial de la página de inicio de la aplicación móvil.

Fuente: Elaboración Propia.

Figura 33

Prototipo Inicial: Página de Registro de Actividad



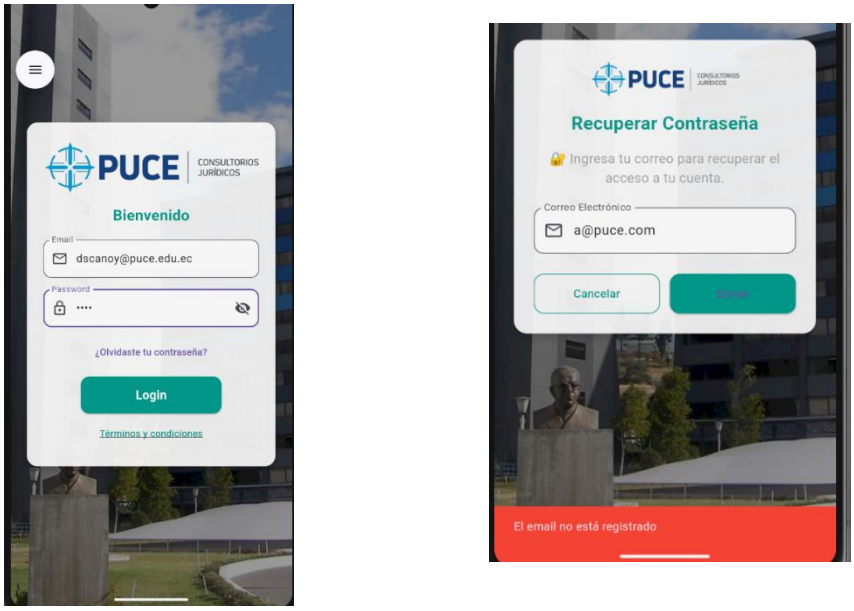
Nota: La figura muestra el prototipo inicial de la pantalla de registro de actividades. Fuente: Elaboración Propia.

4.5.2 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 con la retroalimentación recibida, haciendo que se tengan varias iteraciones. 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 34

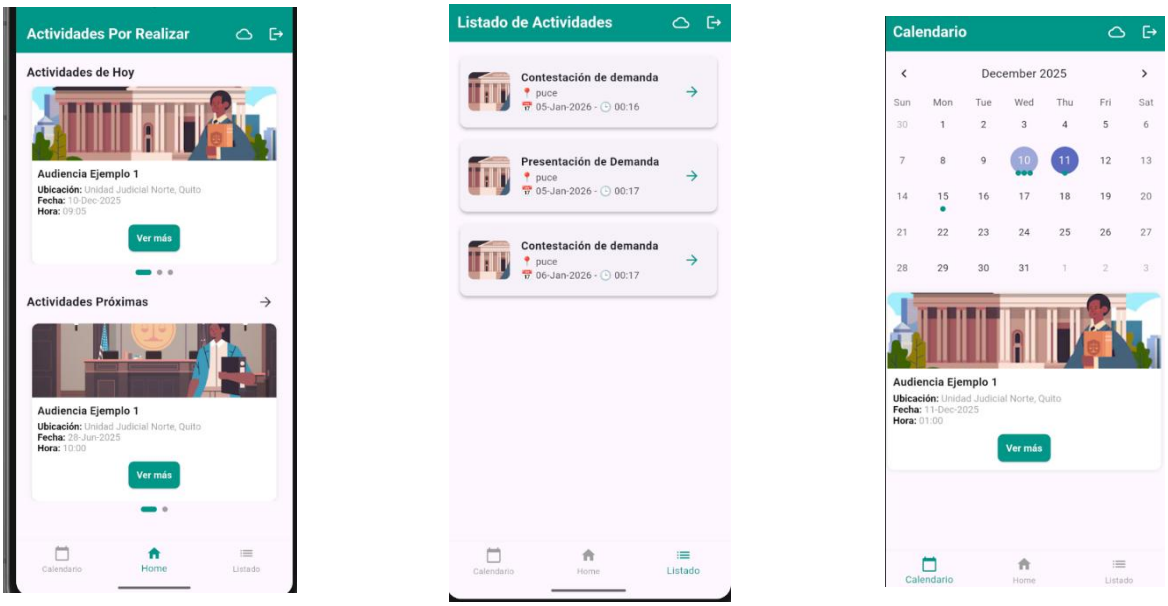
Prototipo Final: Página de Inicio de Sesión y Olvidé Contraseña



Nota: La figura muestra el prototipo final de la pantalla de inicio de sesión, así como el de recuperar contraseña. Fuente: Elaboración Propia.

Figura 35

Prototipo Final: Página de Principal, Página Vista Listado y Página Calendario



Nota: La figura muestra el prototipo final de las 3 formas de visualizar las actividades, siendo la primera la página principal donde se muestran las actividades de hoy y las próximas, la pantalla de listado donde se presentan todas las actividades en un formato de listado, y por último el formato de calendario donde se puede visualizar las fechas de próximas como un calendario resaltando con puntos la cantidad de actividades para ese día. Fuente: Elaboración Propia.

Figura 36

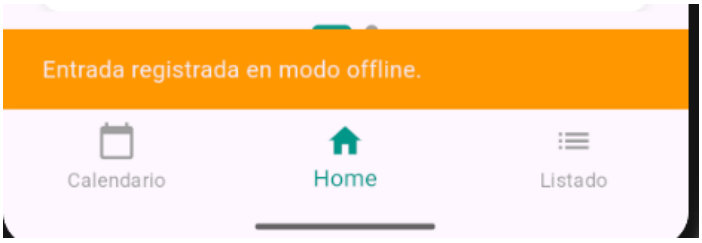
Prototipo Final: Activación del modo offline

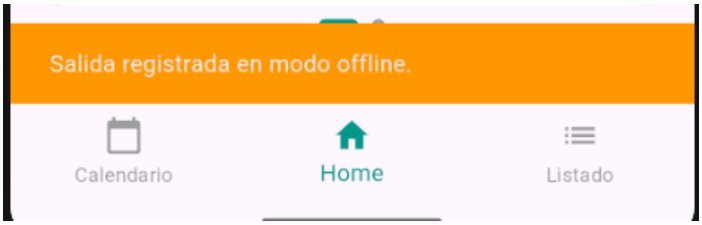
Nota: La figura muestra el prototipo final de la activación del modo offline por medio de la barra de navegación superior y la confirmación de la misma.

pantalla de inicio de sesión, así como el de recuperar contraseña. Fuente: Elaboración Propia.

Figura 37

Prototipo Final: Registro de Entrada y Salida en el modo offline





Nota: La figura muestra el prototipo final del registro de las actividades tanto de entrada como de salida del modo offline, evidenciando al usuario por medio de un mensaje confirmando su registro. Fuente: Elaboración Propia.

CAPÍTULO V: PRUEBAS

De acuerdo con el seguimiento de la metodología, y la prueba correspondiente se a procedido a realizar una prueba del Caso de Uso : Gestión de Actividades y Registro de Asistencia. Para realizar la prueba dentro de la aplicación Web se debe asignar una actividad al usuario correspondiente sea estudiante o abogado.

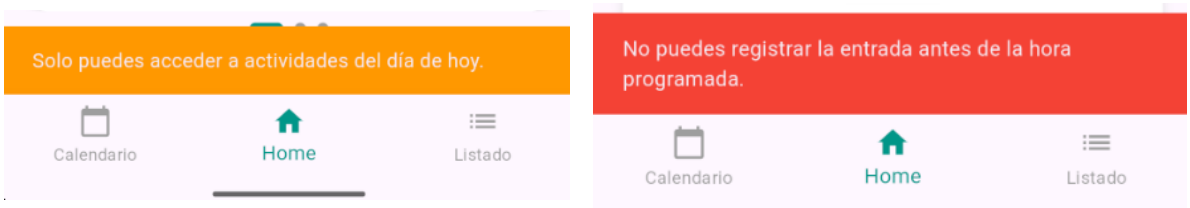
Figura 38

Visualización de actividades asignadas en la pantalla principal

Nota: La figura muestra la pantalla principal (*Home*) de la aplicación móvil, en la cual se presentan las actividades previamente registradas y asignadas desde el sistema web. Una vez que el usuario ha iniciado sesión con credenciales válidas, la aplicación despliega de forma organizada dos tipos de actividades: aquellas correspondientes al día actual y las actividades programadas para fechas futuras. Fuente: Elaboración Propia.

Figura 39

Validación de acceso y registro de entrada según fecha y horario de la actividad



Nota: La figura muestra los mensajes informativos presentados por la aplicación móvil cuando el usuario intenta interactuar con una actividad fuera de las condiciones establecidas. En primer lugar, el sistema restringe el acceso a actividades que no corresponden al día actual, mostrando un mensaje que indica que solo es posible acceder a actividades del día de hoy. Asimismo, cuando la actividad aún no corresponde al horario programado, la aplicación impide el registro de entrada y notifica al usuario que no puede registrar la asistencia antes de la hora establecida. Únicamente cuando la actividad corresponde al día y horario actual, el sistema habilita la opción Registrar Entrada. Fuente: Elaboración Propia.

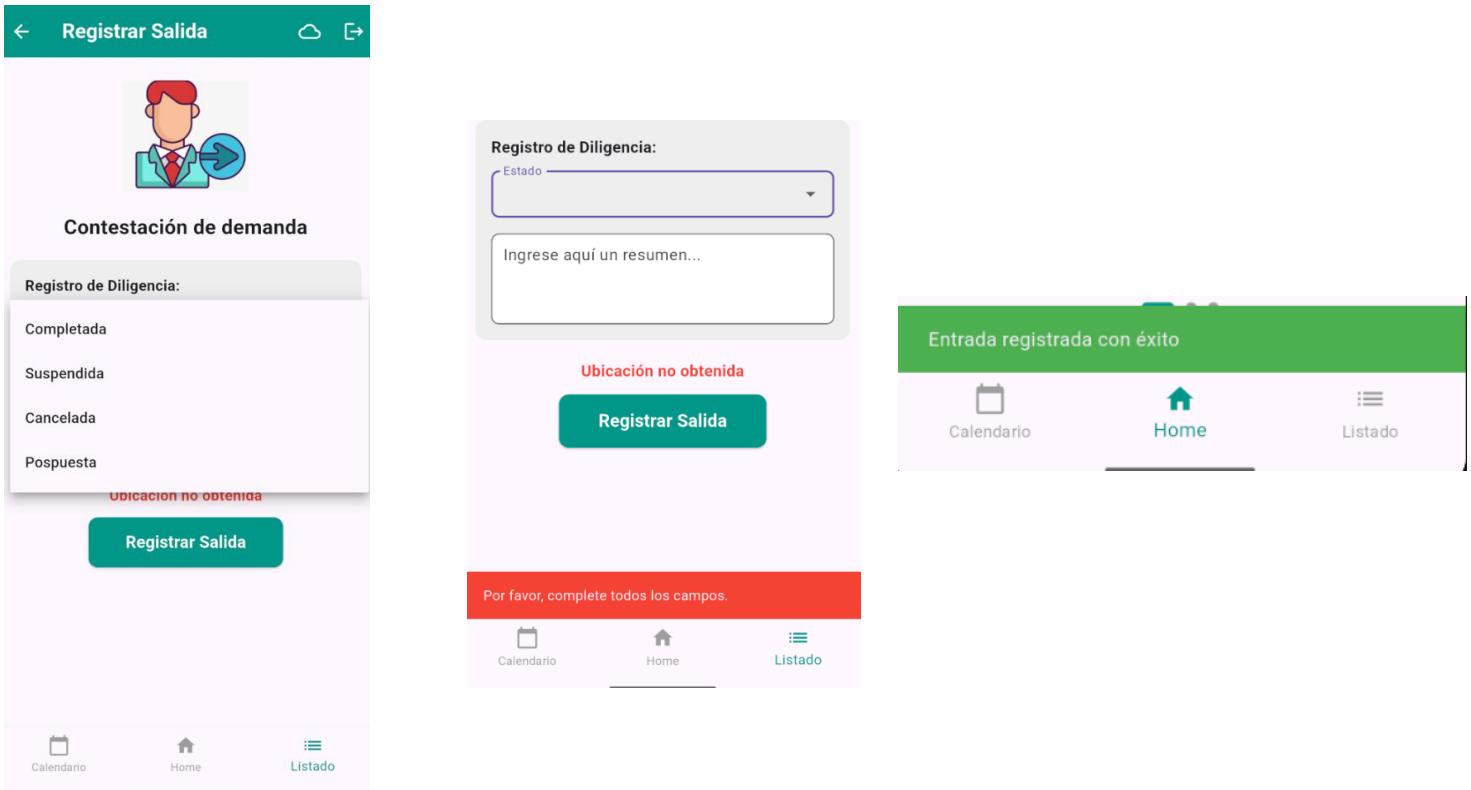
Figura 40

Registro de entrada de asistencia con verificación de ubicación

Nota: La figura muestra la pantalla de registro de entrada de una diligencia en la aplicación móvil. En esta vista se presenta la información detallada de la actividad asignada, junto con la opción *"Registrar Entrada"*. En caso de que el servicio de geolocalización no se encuentre habilitado, el sistema notifica al usuario mediante un mensaje de error y solicita la activación de los permisos de ubicación desde la configuración del dispositivo. Si el usuario rechaza inicialmente el acceso, la aplicación impide continuar con el registro hasta que los permisos sean concedidos. Una vez habilitado el acceso a la ubicación y verificado correctamente, el usuario puede registrar su asistencia, tras lo cual el sistema muestra un mensaje de confirmación indicando que la entrada ha sido registrada con éxito. Fuente: Elaboración Propia.

Figura 41

Registro de salida



Nota: La figura muestra la pantalla de registro de salida de una actividad en la aplicación móvil. En esta etapa final del proceso, el usuario debe seleccionar el estado de la diligencia y registrar una observación o resumen correspondiente. El sistema valida que todos los campos obligatorios hayan sido completados antes de permitir el envío de la información; en caso contrario, se presenta un mensaje informativo indicando que es necesario completar los datos requeridos. Una vez ingresada correctamente la información y verificada la ubicación del usuario, la aplicación procesa el registro y muestra un mensaje de confirmación indicando que la salida ha sido registrada con éxito. Fuente: Elaboración Propia.

Figura 42

Creación de actividad y visualización de registros de asistencia

Nota La figura muestra el proceso de creación de una actividad desde el sistema web y la visualización de los registros de asistencia asociados. En la primera interfaz se registran los datos principales de la actividad. Una vez creada y asignada, la actividad queda disponible para su seguimiento. La segunda interfaz presenta los registros de entrada y salida generados por el usuario. En esta vista se muestra la hora de registro, la ubicación capturada, el cumplimiento del tiempo establecido y la observación ingresada. Esto permite verificar de forma clara el desarrollo y cierre de la actividad. Fuente: Elaboración Propia.

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

- El desarrollo de la aplicación móvil para el Consultorio Jurídico Gratuito de la PUCE permitió mejorar ASIGANACIÓN DE ACTIVIDADES DE LOS ESTUDIANTES Y ABOGADOS significativamente la gestión y control de las actividades realizadas por los estudiante y abogados, proporcionando una herramienta tecnológica que optimiza el registro de asistencia y el seguimiento de las actividades asignadas.
- La funcionalidad de gestión de sesión garantiza un acceso seguro al sistema, asegurando que únicamente usuarios previamente registrados puedan utilizar la aplicación, mediante el uso de autenticación basada en tokens JWT y almacenamiento seguro de la información de sesión en el dispositivo móvil.
- El registro de asistencia mediante geolocalización permitió documentar de manera precisa la entrada y salida de las actividades, fortaleciendo el control y la confiabilidad de la información registrada, además de reducir errores derivado de registros manuales.
- La incorporación del modo offline y la sincronización automática constituyen uno de los principales aportes de la aplicación, ya que garantiza la continuidad operativa del sistema en escenarios en los que la conectividad es limitada, asegurándonos que la información registrad se conserve y sincronice adecuadamente con el backend.
- La arquitectura modular facilitó la separación de funcionalidades y mejoró la mantenibilidad del código y permitió una implementación coherente con los requerimientos funcionales y casos de uso definidos durante la fase de análisis y diseño.
- El diseño y construcción de las interfaces de la aplicación móvil mejora exponencialmente la experiencia del usuario (UX) llevandola a ser minimalista y fácil de usar.

- Usar la metodología de prototipado evolutivo no solo fue eficiente, si no también adaptable, ya que al ser un proceso iterativo con mejoras contina las retroalimentaciones del usuario ayudaron a refinar cada aspecto visual y de funcionalidad ajustando la lógica del negocio a los requerimientos presentados, asegurándonos que el producto satisface al usuario final.

6.2 Recomendaciones

- Se recomienda continuar adicionar a la aplicación móvil nuevas funcionalidades que fortalezcan la experiencia del usuario, tales como notificaciones automáticas para recordar registros y alertas sobre actividades próximas a realizarse
- Se recomienda implementar métricas de uso, para determinar la frecuencia de utilización de la aplicación móvil.
- Para futuras versiones se propone integrar funcionalidades analíticas que permitan generar reportes de productividad directamente desde la aplicación móvil, facilitando la toma de decisiones por parte de los coordinadores del consultorio jurídico

REFERENCIAS

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

Apple. (2024). *Human interface guidelines: Security & privacy*. Apple Developer. https://developer.apple.com/design/human-interface-guidelines

Arias, D. (2021). *Hashing in action: Understanding bcrypt*. Auth0. https://auth0.com/blog/hashing-in-action-understanding-bcrypt/

Bass, L., Clements, P., & Kazman, R. (2013). *Software architecture in practice* (3rd ed.). Addison-Wesley.

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

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress. https://git-scm.com/book/en/v2

Coronel, C., & Morris, S. (2019). *Database systems: Design, implementation, & management* (13th ed.). Cengage Learning.

Crockford, D. (2013). *The JSON data interchange format: Syntax and semantics*. ECMA International.

Date, C. J. (2021). *An introduction to database systems* (9th ed.). Pearson.

De Luca, S. (2022). *Buenas prácticas para el desarrollo web*. Universidad Nacional de La Plata. https://sedici.unlp.edu.ar/bitstream/handle/10915/145469/Documento_completo.pdf

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?script=sci_arttext&pid=S2227-18992021000100037

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/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

Google Developers. (2023). *Android Studio overview*. https://developer.android.com/studio/intro

Google Developers. (2023). *About device emulation*. https://developer.android.com/studio/run/emulator

Google Developers. (2023). *Run apps on the Android Emulator*. https://developer.android.com/studio/run/emulator

Google Developers. (2023). *Build apps for any screen with Flutter*. https://developer.android.com

ISO. (2019). *ISO 9241-210: Ergonomics of human–system interaction — Human-centred design for interactive systems*. International Organization for Standardization.

JetBrains. (2023). *What is an IDE (Integrated Development Environment)?* https://www.jetbrains.com/idea/resources/what-is-ide/

Kumar, A. (2024). *A guide to web development frameworks*. Built In.

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*. 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

Microsoft. (2023). *GitHub Copilot documentation*. https://docs.github.com/en/copilot

Microsoft. (2023). *Visual Studio Code documentation*. https://code.visualstudio.com/docs

Mozilla Developer Network. (2023). *HTTP access control (CORS)*. https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

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

Nielsen, J. (1994). *Usability engineering*. Academic Press.

NIST. (2020). *SP 800-63B: Digital identity guidelines – Authentication and lifecycle management*. National Institute of Standards and Technology.

Oracle. (2023). *What is a database?* https://www.oracle.com/database/what-is-database/

Oracle. (2023). *What is a database management system (DBMS)?* https://www.oracle.com/database/what-is-a-dbms/

Oracle. (2023). *What is JSON?* https://www.oracle.com/database/what-is-json/

Oracle. (2023). *What is SQL?* https://www.oracle.com/database/what-is-sql/

OWASP. (2023). *Mobile Application Security Verification Standard (MASVS) & Mobile Security Testing Guide (MSTG)*. https://owasp.org

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/

Pressman, R. S., & Maxim, B. R. (2015). *Software engineering: A practitioner's approach* (8th ed.). McGraw-Hill.

PrimeVue. (2025). *The next-gen UI suite for Vue.js*. https://primevue.org/

Red Hat. (2023). *What is an integrated development environment (IDE)?* https://www.redhat.com/en/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://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

Shneiderman, B., & Plaisant, C. (2010). *Designing the user interface* (5th ed.). Pearson.

Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.

Stack Overflow. (2023). *2023 developer survey results*. https://survey.stackoverflow.co/2023/

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

Universidad Tecnológica Nacional. (2025). *Introducción a la programación*.

https://sanfrancisco.utn.edu.ar/documentos/archivos/ingreso/Intro_a_la_Programacion.pdf

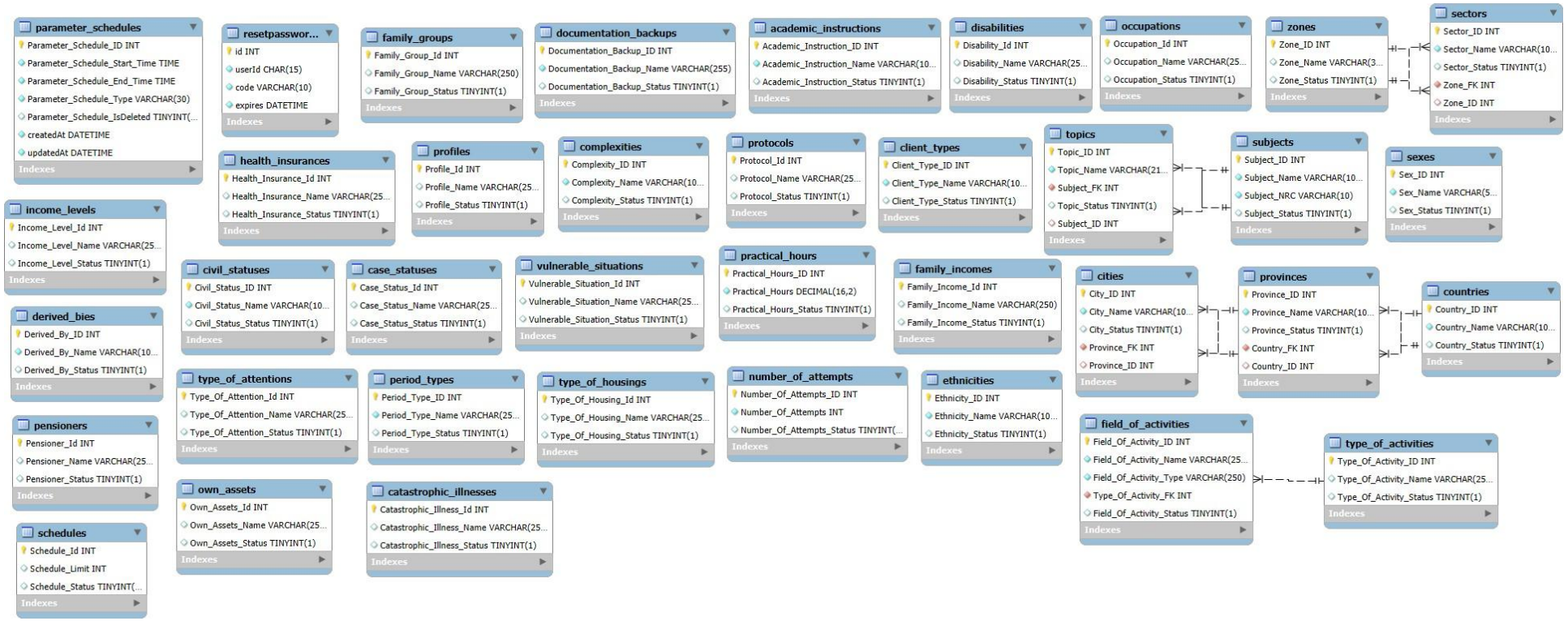
Vite. (2025). *Introducción*. https://es.vite.dev/guide/

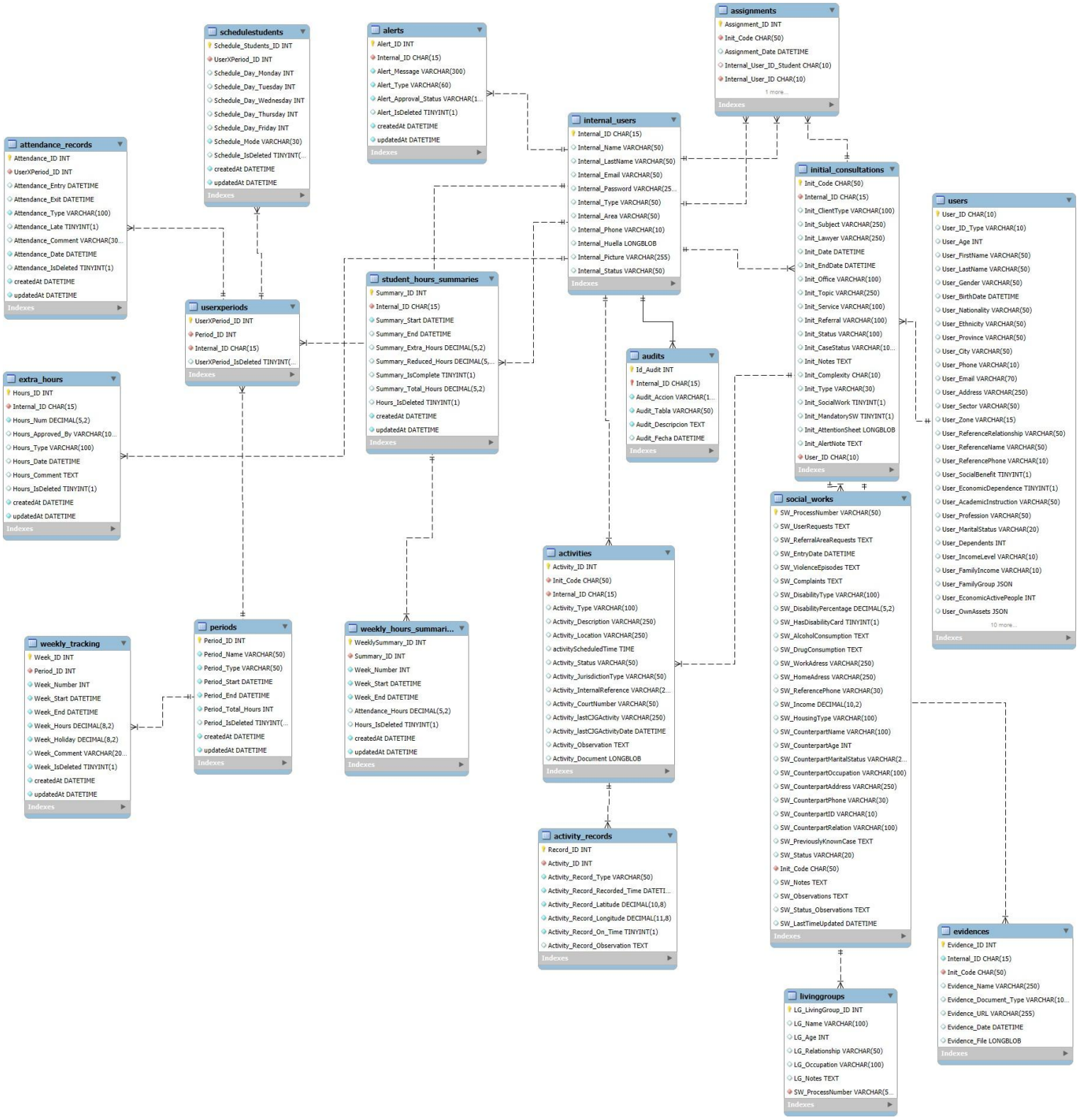
W3C. (2018). *Web content accessibility guidelines (WCAG) 2.1*. World Wide Web Consortium. https://www.w3.org/TR/WCAG21/

W3C. (2023). *Cross-origin resource sharing (CORS) specification*. https://www.w3.org/TR/cors/

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

ANEXOS


parameter_schedules
Parameter_Schedule_ID INT
Parameter_Schedule_Start_Time TIME
Parameter_Schedule_End_Time TIME
Parameter_Schedule_Type VARCHAR(30)
Parameter_Schedule_IsDeleted TINYINT(...
createdAt DATETIME
updatedAt DATETIME
Indexes
resetpasswor...
id INT
userId CHAR(15)
code VARCHAR(10)
expires DATETIME
Indexes
family_groups
Family_Group_Id INT
Family_Group_Name VARCHAR(250)
Family_Group_Status TINYINT(1)
Indexes
documentation_backups
Documentation_Backup_ID INT
Documentation_Backup_Name VARCHAR(255)
Documentation_Backup_Status TINYINT(1)
Indexes
academic_instructions
Academic_Instruction_ID INT
Academic_Instruction_Name VARCHAR(10...
Academic_Instruction_Status TINYINT(1)
Indexes
disabilities
Disability_Id INT
Disability_Name VARCHAR(25...
Disability_Status TINYINT(1)
Indexes
occupations
Occupation_Id INT
Occupation_Name VARCHAR(25...
Occupation_Status TINYINT(1)
Indexes
zones
Zone_ID INT
Zone_Name VARCHAR(3...
Zone_Status TINYINT(1)
Indexes
sectors
Sector_ID INT
Sector_Name VARCHAR(10...
Sector_Status TINYINT(1)
Zone_FK INT
Zone_ID INT
Indexes
health_insurances
Health_Insurance_Id INT
Health_Insurance_Name VARCHAR(25...
Health_Insurance_Status TINYINT(1)
Indexes
profiles
Profile_Id INT
Profile_Name VARCHAR(25...
Profile_Status TINYINT(1)
Indexes
complexities
Complexity_ID INT
Complexity_Name VARCHAR(10...
Complexity_Status TINYINT(1)
Indexes
protocols
Protocol_Id INT
Protocol_Name VARCHAR(25...
Protocol_Status TINYINT(1)
Indexes
client_types
Client_Type_ID INT
Client_Type_Name VARCHAR(10...
Client_Type_Status TINYINT(1)
Indexes
topics
Topic_ID INT
Topic_Name VARCHAR(21...
Subject_FK INT
Topic_Status TINYINT(1)
Subject_ID INT
Indexes
subjects
Subject_ID INT
Subject_Name VARCHAR(10...
Subject_NRC VARCHAR(10)
Subject_Status TINYINT(1)
Indexes
sexes
Sex_ID INT
Sex_Name VARCHAR(5...
Sex_Status TINYINT(1)
Indexes
income_levels
Income_Level_Id INT
Income_Level_Name VARCHAR(25...
Income_Level_Status TINYINT(1)
Indexes
civil_statuses
Civil_Status_ID INT
Civil_Status_Name VARCHAR(10...
Civil_Status_Status TINYINT(1)
Indexes
case_statuses
Case_Status_Id INT
Case_Status_Name VARCHAR(25...
Case_Status_Status TINYINT(1)
Indexes
vulnerable_situations
Vulnerable_Situation_Id INT
Vulnerable_Situation_Name VARCHAR(25...
Vulnerable_Situation_Status TINYINT(1)
Indexes
practical_hours
Practical_Hours_ID INT
Practical_Hours DECIMAL(16,2)
Practical_Hours_Status TINYINT(1)
Indexes
family_incomes
Family_Income_Id INT
Family_Income_Name VARCHAR(250)
Family_Income_Status TINYINT(1)
Indexes
cities
City_ID INT
City_Name VARCHAR(10...
City_Status TINYINT(1)
Province_FK INT
Province_ID INT
Indexes
provinces
Province_ID INT
Province_Name VARCHAR(10...
Province_Status TINYINT(1)
Country_FK INT
Country_ID INT
Indexes
countries
Country_ID INT
Country_Name VARCHAR(10...
Country_Status TINYINT(1)
Indexes
derived_bies
Derived_By_ID INT
Derived_By_Name VARCHAR(10...
Derived_By_Status TINYINT(1)
Indexes
type_of_attentions
Type_Of_Attention_Id INT
Type_Of_Attention_Name VARCHAR(25...
Type_Of_Attention_Status TINYINT(1)
Indexes
period_types
Period_Type_ID INT
Period_Type_Name VARCHAR(25...
Period_Type_Status TINYINT(1)
Indexes
type_of_housings
Type_Of_Housing_Id INT
Type_Of_Housing_Name VARCHAR(25...
Type_Of_Housing_Status TINYINT(1)
Indexes
number_of_attempts
Number_Of_Attempts_ID INT
Number_Of_Attempts INT
Number_Of_Attempts_Status TINYINT(...
Indexes
ethnicities
Ethnicity_ID INT
Ethnicity_Name VARCHAR(10...
Ethnicity_Status TINYINT(1)
Indexes
field_of_activities
Field_Of_Activity_ID INT
Field_Of_Activity_Name VARCHAR(25...
Field_Of_Activity_Type VARCHAR(250)
Type_Of_Activity_FK INT
Field_Of_Activity_Status TINYINT(1)
Indexes
type_of_activities
Type_Of_Activity_ID INT
Type_Of_Activity_Name VARCHAR(25...
Type_Of_Activity_Status TINYINT(1)
Indexes
pensioners
Pensioner_Id INT
Pensioner_Name VARCHAR(25...
Pensioner_Status TINYINT(1)
Indexes
own_assets
Own_Assets_Id INT
Own_Assets_Name VARCHAR(25...
Own_Assets_Status TINYINT(1)
Indexes
catastrophic_illnesses
Catastrophic_Illness_Id INT
Catastrophic_Illness_Name VARCHAR(25...
Catastrophic_Illness_Status TINYINT(1)
Indexes
schedules
Schedule_Id INT
Schedule_Limit INT
Schedule_Status TINYINT(...
Indexes


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
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
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
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
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
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