Tecnológico Nacional de México
Instituto Tecnológico Superior de Purísima del Rincón

ESTUDIOS CON RECONOCIMIENTO DE VALIDÉZ OFICIAL
NÚMERO: 11EIT0006A

TITULACIÓN INTEGRAL

TIPO DE PROYECTO:
RESIDENCIA PROFESIONAL

TÍTULO:
MAIA App - Feature-Driven Development - CORVUZ

PARA OBTENER EL GRADO DE
INGENIERO EN INFORMÁTICA

PRESENTA:
Victor Manuel Castillo Olivetto
RS21110113

ASESOR:
Mtro. Dolorez Ayala Muñoz

PURÍSIMA DEL RINCÓN, GTO. Junio 2025

AGRADECIMIENTOS

Agradezco a mi familia por apoyarme a lo largo de mi carrera y a esos maestros que siempre me apoyaron.

RESUMEN

El presente proyecto tuvo como objetivo el desarrollo de una aplicación móvil multiplataforma para la gestión de citas médicas denominada Maia, la cual complementa la plataforma web ya existente. La aplicación fue creada para optimizar el trabajo de los especialistas, permitiéndoles agendar, reagendar, eliminar y visualizar citas, así como consultar el corte del día y administrar su información personal de manera rápida y segura. El desarrollo se llevó a cabo durante el periodo de residencia profesional, aplicando una metodología ágil que permitió avanzar de forma organizada y controlada.

Durante el proceso de implementación se realizaron diversas actividades que incluyeron el análisis de la plataforma web Agenda Maia, la evaluación de las APIs existentes en Laravel y la adaptación de aquellas que requerían ajustes para comunicarse con la nueva aplicación. La metodología utilizada fue Feature-Driven Development (FDD), que facilitó el trabajo incremental por funcionalidades, permitiendo diseñar, codificar y probar cada módulo de manera independiente. Las principales herramientas empleadas fueron Flutter y Dart para el desarrollo móvil, PHP con Laravel para la gestión de las APIs, y GitHub para el control de versiones y respaldo del código.

Se realizaron pruebas de integración y usabilidad para validar la correcta comunicación entre la aplicación móvil y los servicios web. Estas pruebas permitieron identificar y corregir errores, optimizar la estructura del código y garantizar una experiencia de usuario fluida. Los resultados demostraron que la aplicación cumplió con los requerimientos planteados, ofreciendo una interfaz intuitiva, tiempos de respuesta adecuados y una sincronización eficiente de datos en tiempo real.

Como conclusión, el proyecto permitió integrar de manera exitosa la nueva aplicación móvil con la plataforma web existente, mejorando el acceso a las funciones de gestión de citas y ofreciendo una solución tecnológica escalable para

futuros desarrollos. La implementación de Maia evidenció la importancia de las metodologías ágiles y el uso de herramientas modernas para el desarrollo de software, logrando un producto final funcional, estable y adaptable a las necesidades de los especialistas y usuarios.

ÍNDICE DE CONTENIDO

INDICE DE TABLAS Y FiguraS

CAPÍTULO I GENERALIDADES DEL PROYECTO

1.1 INTRODUCCIÓN

El presente proyecto tiene como objetivo la creación de la aplicación móvil Maia, una herramienta que optimiza la gestión de citas médicas y profesionales, el control de ingresos diarios mediante el corte del día y la administración de la información de los especialistas. Este desarrollo se plantea como una extensión de la plataforma web Agenda Maia, pero con un enfoque simplificado y adaptado a dispositivos móviles, respondiendo a la necesidad de los especialistas de contar con una solución práctica, accesible y disponible en cualquier momento.

La hipótesis que guía este trabajo es que el desarrollo de la aplicación móvil Maia permitirá a los especialistas organizar sus citas e información de manera más eficiente, reduciendo tiempos administrativos y ofreciendo un mayor control sobre su actividad diaria. Al enfocarse en tres funciones clave ,gestión de citas, perfil de especialista y corte del día,, se busca no solo mejorar la productividad de los usuarios, sino también incrementar la satisfacción de los clientes finales al contar con procesos más rápidos y confiables.

Para alcanzar este propósito, se utilizará la metodología Feature-Driven Development (FDD), que facilita dividir el trabajo en características específicas, medibles y verificables. Esta metodología permitirá asegurar la calidad en cada módulo desarrollado, avanzando de forma ordenada desde el diseño de las APIs hasta la implementación de las vistas móviles. Con ello, la aplicación Maia se proyecta como una herramienta clave en la operación diaria de los consultorios y empresas que utilizan la plataforma web, aportando valor agregado y modernizando su experiencia tecnológica.

1.2. MARCO CONTEXTUAL

1.2.1 Generalidades de la Empresa

Nombre: Corvuz

Dirección: C. Hermenegildo Bustos 215, La Barda, 36404 Purísima de Bustos, Guanajuato.

Teléfono: 476 113 3570

Página web: www.corvuz.com

Año de fundación: 2013

Giro de la empresa: Desarrollo de soluciones tecnológicas personalizadas (software web, ecommerce, aplicaciones móviles y contenido digital).

1.2.2 Antecedentes de la Empresa

Corvuz fue fundada en 2013 por Andrea y Héctor, quienes detectaron la necesidad de ofrecer soluciones tecnológicas a medida, más allá de las plataformas y plantillas predefinidas como Shopify. Su primer acercamiento se dio gracias a la recomendación de un amigo con un cliente que requería un desarrollo web personalizado.

La empresa surge con el objetivo de brindar a las marcas un diferenciador en el ámbito digital, creando software web y aplicaciones que se adaptan a los procesos

internos de cada cliente. A lo largo de los años, Corvuz ha evolucionado en sus metodologías, pasando de estructuras monolíticas a arquitecturas modernas basadas en **microservicios, APIs, jobs, eventos y uso de CDNs** para lograr soluciones más escalables y robustas.

Actualmente, Corvuz trabaja con diferentes clientes en el ámbito nacional, apoyándolos en la digitalización de procesos, la gestión de comercio electrónico, aplicaciones móviles y el desarrollo de sistemas administrativos personalizados.

1.2.3 Filosofía de la Empresa

Misión

Transformar las ideas de los clientes en soluciones tecnológicas prácticas y eficientes, que no solo cumplan con sus expectativas, sino que también las superen.

Visión

Ser el aliado tecnológico preferido de las empresas mexicanas, reconocido por la innovación y calidad de sus soluciones, contribuyendo al crecimiento y eficiencia de sus clientes.

Objetivos

- Ofrecer soluciones tecnológicas personalizadas que se adapten a las necesidades de cada cliente.

- Mantenerse a la vanguardia tecnológica mediante la actualización constante.
- Brindar productos y servicios de calidad que incrementen la eficiencia de las empresas.
- Posicionarse como líder en el desarrollo de software y aplicaciones en México.

Valores

- **Innovación:** búsqueda constante de nuevas tendencias tecnológicas.
- **Compromiso:** con el éxito de los proyectos y clientes.
- **Calidad:** desarrollo de productos confiables y eficientes.
- **Responsabilidad:** ética en cada acción y decisión.
- **Servicio al cliente:** atención personalizada y cercana.
- **Empatía:** entendimiento de las necesidades y perspectivas de cada cliente.

Política de Calidad

Corvuz se compromete a entregar soluciones tecnológicas de alta calidad, alineadas a los objetivos del cliente, garantizando eficiencia, confiabilidad e innovación en cada proyecto.

1.2.4 Descripción del Área de Negocios.

El área de negocios a la que corresponde el proyecto es el **Departamento de Desarrollo de Software**, encargado de la planeación, diseño, programación e implementación de aplicaciones web y móviles.

Este departamento colabora estrechamente con:

- **Área de diseño:** para definir la maquetación y experiencia de usuario.

Su contribución principal al logro de los objetivos institucionales radica en la creación de herramientas tecnológicas personalizadas que aumentan la competitividad de los clientes y consolidan la reputación de Corvuz como empresa innovadora y confiable.

1.2.5 Puesto Asignado y Funciones.

- Puesto asignado: Residente en Desarrollo de Software.
- Conocimientos requeridos: Programación en aplicaciones móviles (Flutter/Dart), manejo de APIs REST, bases de datos, control de versiones con Git, buenas prácticas de programación y análisis de requerimientos.

Funciones desempeñadas:

- Análisis de la página web Maia para comprender su funcionamiento.
- Creación y consumo de APIs necesarias para la aplicación móvil.
- Implementación de módulos en la app Maia:

    - Gestión de citas (agendar, reagendar, cancelar).
    - Corte del día (consulta de ingresos diarios).
    - Manejo de información del especialista (perfil y edición de datos).
- Estructuración del proyecto móvil (organización de carpetas, estados, rutas y dependencias).
- Integración de la interfaz de usuario con la lógica del negocio.
- Pruebas funcionales y corrección de errores.
- Optimización de código y sincronización con repositorio Git.

1.3 MARCO DE REFERENCIA

1.3.1 Antecedentes del Problema

En la actualidad, la gestión de citas y servicios en consultorios médicos y de especialidades representa un reto debido a la necesidad de mantener un control eficiente de horarios, pagos y disponibilidad de especialistas. Muchas plataformas web, como la propia Agenda Maia, ya ofrecen soluciones robustas para este fin; sin embargo, la movilidad de los profesionales de la salud y de otras áreas exige herramientas más flexibles que puedan ser utilizadas desde cualquier lugar.

El problema radica en que, si bien la página web Maia cubre las necesidades administrativas desde un entorno de escritorio, los especialistas requieren de una aplicación móvil simplificada que les permita organizar sus citas, consultar

información de clientes y visualizar ingresos diarios sin depender de un equipo de cómputo. Además, la ausencia de una interfaz móvil limita la agilidad en la atención a clientes potenciales y reduce la capacidad de respuesta inmediata en escenarios fuera del consultorio.

Disciplinas como la ingeniería de software, el desarrollo móvil multiplataforma y la gestión de datos en tiempo real son fundamentales para abordar este problema. La implementación de metodologías ágiles, en este caso Feature-Driven Development (FDD), permite estructurar el desarrollo en fases centradas en características específicas, garantizando un producto modular, usable y confiable.

1.3.2 Planteamiento del Problema.

Los principales problemas identificados que motivan este proyecto son:

1. Falta de una aplicación móvil para especialistas que permita acceder de forma ágil a la información de citas y servicios.
2. Dependencia total de la versión web, lo que limita el uso en situaciones donde no se tiene acceso a un equipo de cómputo.
3. Carencia de una herramienta para agendar, cancelar o crear citas desde el dispositivo móvil en tiempo real.
4. Dificultad de los especialistas para visualizar de manera inmediata el corte del día, afectando el control de ingresos y la toma de decisiones.
5. Necesidad de optimizar la administración del perfil del especialista, reduciendo la duplicidad de datos y errores administrativos.

1.3.2 Objetivos

- **Objetivo General**

Desarrollar la aplicación móvil **Maia** como una extensión simplificada de la Agenda Maia web, que permita a los especialistas gestionar citas, visualizar el corte del día y administrar la información de usuario de manera ágil, confiable y accesible desde cualquier dispositivo móvil.

- **Objetivos Específicos**

    - Implementar un sistema móvil para crear, reagendar y cancelar citas en tiempo real.
    - Desarrollar la funcionalidad de corte del día, permitiendo visualizar ingresos diarios a nivel individual y general.
    - Diseñar e implementar un módulo de perfil de especialista para la consulta y edición de datos básicos.
    - Asegurar la sincronización correcta entre la aplicación móvil y el sistema central de la Agenda Maia.
    - Aplicar la metodología Feature-Driven Development (FDD) para garantizar un desarrollo estructurado, modular y con entregables verificables.
    - Realizar pruebas de funcionalidad y usabilidad para validar el correcto desempeño de la aplicación y su facilidad de uso para los especialistas.

1.3.3. Justificación

El proyecto es importante porque responde a la necesidad de contar con una herramienta tecnológica que facilite la gestión de citas y el control de información para especialistas. Actualmente, la dependencia de la versión web limita la flexibilidad y accesibilidad del sistema, lo que ocasiona dificultades en la administración de citas y en la consulta de información en tiempo real. La aplicación móvil Maia atiende esta problemática al ofrecer una plataforma intuitiva y accesible que mejora la experiencia tanto de los especialistas como de los usuarios finales.

Además, la aplicación contribuye al fortalecimiento de la propuesta de valor de la empresa Corvuz, ya que amplía su portafolio de soluciones digitales al integrar un producto móvil moderno, escalable y adaptado a las tendencias actuales del mercado. Con ello, se refuerza la innovación y competitividad de la organización, consolidándose como un aliado tecnológico confiable.

A futuro, el impacto de este proyecto se refleja en la optimización del tiempo de los especialistas, la reducción de errores administrativos y una mayor satisfacción del cliente, gracias a un sistema que permite gestionar citas y visualizar información crítica desde cualquier dispositivo móvil. Asimismo, la aplicación servirá como base para futuras actualizaciones y nuevas funcionalidades, generando un ciclo de mejora continua que beneficiará tanto a la empresa como a sus clientes, posicionando a Maia como una solución integral en la gestión de citas y servicios.

CAPÍTULO II MARCO TEÓRICO

2.1 Teorías y Enfoques Relacionados

El desarrollo de aplicaciones móviles orientadas a la gestión de servicios y citas se enmarca en la teoría de la transformación digital, la cual sostiene que la implementación de tecnologías modernas mejora la eficiencia operativa y la experiencia del usuario. Bajo esta perspectiva, los sistemas móviles permiten la optimización de procesos administrativos al ofrecer accesibilidad, inmediatez y escalabilidad.

Asimismo, se retoma el enfoque de la experiencia de usuario (UX), que establece que una aplicación debe centrarse en la usabilidad, accesibilidad y simplicidad, garantizando que los especialistas y administradores puedan gestionar citas y consultar información de manera rápida y sin fricciones. Dicho enfoque asegura que la aplicación cumpla con su propósito práctico en un entorno competitivo. **(Kuusinen & Mikkonen, 2013, 535-540)**

2.2 Investigaciones y Antecedentes

El estudio “Diseño y Validación de Arquitecturas de Aplicaciones Empresariales ” demuestra que las aplicaciones móviles de gestión de servicios incrementan la productividad y reducen los tiempos administrativos al centralizar información en un sistema único. Investigaciones sobre aplicaciones empresariales concluyen que la integración de módulos de citas, control de ingresos y manejo de información fortalece el control organizacional y la toma de decisiones. Estos antecedentes

respaldan la relevancia del proyecto Maia al buscar una solución práctica y adaptada a las necesidades de los especialistas. **(A & J, 2015)**

2.3 Metodología de Desarrollo: Feature-Driven Development (FDD)

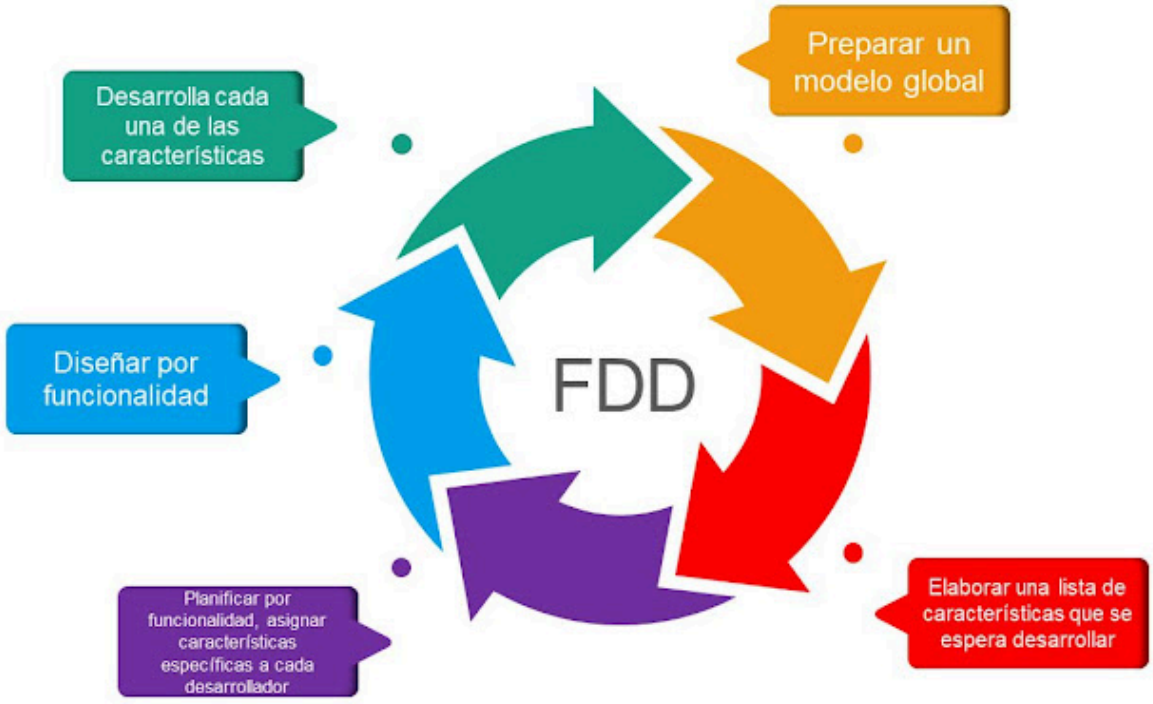

Figura 1.- (Rivera, 2020)

Para el desarrollo de la aplicación móvil Maia, se adoptó la metodología Feature-Driven Development (FDD), como se puede ver en la Figura 1 es un enfoque ágil enfocado en la construcción de funcionalidades específicas y medibles. Esta metodología resultó ideal para el proyecto, ya que permitió dividir el trabajo en partes pequeñas (features) que podían desarrollarse, probarse y

entregarse en períodos cortos de tiempo. Gracias a su estructura, se garantiza un avance constante, la reducción de errores y una mejor gestión del tiempo durante la residencia. **(F, 2017, 1-10)**

Etapa 1: Desarrollo del Modelo General

Desarrollar un modelo global: Al inicio del desarrollo se construye un modelo teniendo en cuenta la visión, el contexto y los requisitos que debe tener el sistema a construir. Este modelo se divide en áreas que se analizan detalladamente. Se construye un diagrama de clases por cada área.

Durante esta fase se realizó el análisis del sistema web Maia, identificando los módulos principales que conformaban la plataforma y determinando cuáles serían trasladados a la aplicación móvil.
Se estudiaron los procesos de:

- Creación y gestión de citas.
- Administración de usuarios y especialistas.
- Generación de cortes del día.

El resultado fue un modelo funcional general donde se definió la interacción entre el usuario (especialista), la aplicación móvil, las APIs de Laravel y la base de

datos. Este modelo sirvió de base para establecer la arquitectura del proyecto y planificar los módulos de desarrollo.

Etapa 2: Construcción de la Lista de Funcionalidades

Construir una lista de funcionalidades: Se elabora una lista que resuma las funcionalidades que debe tener el sistema, cuya lista es evaluada por el cliente. Cada funcionalidad de la lista se divide en funcionalidades más pequeñas para un mejor entendimiento del sistema.

Una vez definido el modelo general, se elaboró un listado detallado de características, priorizando las más importantes para el funcionamiento inicial de la aplicación.

Las principales funcionalidades identificadas fueron:

1. Autenticación y login del especialista.
2. Visualización del calendario de citas.
3. Creación, reagendamiento y cancelación de citas.
4. Consulta del corte del día.
5. Visualización y edición de la información del especialista.

Cada característica fue descrita con sus requerimientos, dependencias, flujo de datos y endpoints necesarios para comunicarse con el backend.

Etapa 3: Planificación por Funcionalidades

Planear por funcionalidades: Se procede a ordenar los conjuntos de funcionalidades conforme a su prioridad y dependencia, y se asigna a los programadores jefes.

En esta etapa se diseñó un plan de desarrollo por módulos, estableciendo la secuencia de trabajo y los recursos necesarios.
El orden de desarrollo se definió considerando la prioridad funcional:

1. Login.
2. Gestión de citas.
3. Corte del día.
4. Perfil del especialista.
5. Optimización y pruebas.

Se establecieron entregables por semana, asegurando que cada módulo estuviera funcional antes de avanzar al siguiente.

Etapa 4: Diseño por Funcionalidades (Feature Design)

Diseñar por funcionalidades: Se selecciona un conjunto de funcionalidades de la lista. Se procede a diseñar y construir la funcionalidad mediante un proceso iterativo, decidiendo que funcionalidad se van a realizar en cada iteración. Este

proceso iterativo incluye inspección de diseño, codificación, pruebas unitarias, integración e inspección de código.

En esta fase se realizó el diseño técnico y visual de cada módulo.

- Se definieron los flujos de usuario, las pantallas y los eventos que desencadenaron acciones dentro de la app.
- Se estructuraron los controladores y servicios en Laravel que serían consumidos desde Flutter.
- En la parte visual, se aplicó un diseño limpio y funcional,este diseño fue dado por el departamento de diseño .

Cada feature fue diseñada considerando su relación con las demás, garantizando la coherencia en navegación, datos y experiencia de usuario.

Etapa 5: Construcción por Funcionalidades

Construir por funcionalidades: Se implementa y prueba cada funcionalidad. Estos pasos garantizan que el desarrollo siga siendo congruente y que el proyecto pueda crecer, permitiendo que los nuevos miembros del equipo se adapten al ritmo de trabajo más rápido.

En la última fase se llevó a cabo el desarrollo y codificación de las funcionalidades planificadas.

- En Flutter, se implementaron las vistas, estados, eventos y controladores de cada módulo.
- En Laravel, se ajustaron y crearon las APIs necesarias para la sincronización de datos.
- Se realizó la integración completa entre frontend y backend mediante llamadas REST, garantizando la actualización inmediata de la información.

Durante esta etapa también se realizaron:

- Pruebas unitarias y de integración, validando la comunicación entre app y servidor.
- Depuración de errores, optimización de rutas y refactorización del código.
- Pruebas de usabilidad, verificando que la aplicación respondiera correctamente en dispositivos móviles reales.

El resultado de esta fase fue una aplicación móvil funcional, estable y sincronizada con la plataforma web, cumpliendo todos los objetivos propuestos.

2.4 Tecnologías y Arquitectura Utilizadas

El desarrollo de la aplicación móvil Maia se sustenta en una arquitectura moderna basada en microservicios y APIs, lo que permite una comunicación eficiente entre los diferentes módulos de la aplicación y el sistema central de la empresa. Este enfoque facilita la escalabilidad, la integración de nuevas funcionalidades y el

mantenimiento independiente de cada componente, asegurando así una solución robusta y adaptable a las necesidades futuras de la organización.

PHP con Laravel

El backend del proyecto se implementa en PHP, un lenguaje de programación ampliamente utilizado en el desarrollo de aplicaciones web gracias a su flexibilidad y capacidad para integrarse con múltiples sistemas. Sobre PHP se emplea el framework Laravel, que destaca por su estructura elegante y su enfoque en la productividad. Laravel provee una arquitectura MVC (Modelo-Vista-Controlador) que separa la lógica de negocio, la interfaz de usuario y el manejo de datos, favoreciendo la organización del código y la reutilización de componentes. **(Stauffer, 2019)**

Entre sus ventajas principales se encuentran:

- Eloquent ORM para una gestión simplificada de bases de datos.
- Sistema de rutas y controladores que permite manejar peticiones de manera clara y eficiente.
- Mecanismos de seguridad integrados, como protección CSRF y encriptación de datos.
- Herramientas para pruebas, migraciones de base de datos y colas de trabajo, que aceleran el desarrollo y facilitan el mantenimiento.

Gracias a estas características, Laravel garantiza un backend seguro, escalable y de fácil mantenimiento, ideal para la administración de citas, usuarios, especialistas y el corte del día en la aplicación Maia.

JavaScript (JS)

En el frontend, JavaScript es la tecnología principal para dotar a la aplicación de interactividad y dinamismo. Este lenguaje permite manipular la interfaz en tiempo real, gestionar eventos de usuario y realizar peticiones asincrónicas a las APIs del sistema, ofreciendo una experiencia fluida. Su versatilidad y compatibilidad con múltiples entornos facilitan el desarrollo de una aplicación móvil que puede funcionar tanto en navegadores como en dispositivos móviles, asegurando una interacción amigable y responsiva para el usuario final. **(*Javascript - Javascript En Español*, n.d.)**

Microservicios

La arquitectura de microservicios permite que las distintas funciones del sistema (como la gestión de citas, el manejo de usuarios y especialistas, y el corte del día) se ejecuten como servicios independientes que se comunican a través de APIs.
Las principales ventajas de este enfoque son:

- **Escalabilidad independiente**: cada módulo puede actualizarse o ampliarse sin afectar el resto del sistema.

- **Resiliencia**: si un microservicio falla, los demás pueden seguir operando.
- **Despliegue ágil**: facilita la integración continua y la entrega rápida de nuevas funciones.

En el caso de Maia, esta arquitectura es esencial para mantener un sistema que pueda crecer en volumen de usuarios y funcionalidades sin comprometer su estabilidad. **(*¿Qué Son Los Microservicios?*, n.d.)**

APIs REST

Para la comunicación entre la aplicación móvil y el servidor, se utilizan APIs REST (Representational State Transfer), un estándar que permite el intercambio de datos a través de peticiones HTTP. Las APIs facilitan que la aplicación consulte, cree o actualice información en el servidor, como los datos de citas, especialistas, usuarios y cortes del día. **(*¿Qué Es REST?*, 2025)**

Entre sus beneficios destacan:

- **Interoperabilidad**: permite que diferentes aplicaciones o servicios se integren de manera sencilla.
- **Eficiencia**: el formato de respuesta en JSON es ligero y rápido de procesar.
- **Escalabilidad**: favorece la adición de nuevos endpoints sin afectar los existentes.

Integración de la Arquitectura

La combinación de Laravel, JavaScript, microservicios y APIs REST proporciona una base tecnológica sólida para Maia. Esta arquitectura garantiza:

- **Sincronización en tiempo real** entre la aplicación móvil y el sistema central.
- **Modularidad** para agregar nuevos servicios sin reestructurar todo el sistema.
- **Seguridad y confiabilidad** en el manejo de datos sensibles de usuarios y especialistas.

En conjunto, estas tecnologías permiten que Maia ofrezca una plataforma móvil ágil, escalable y alineada con las mejores prácticas de desarrollo actuales, cumpliendo así con los objetivos de eficiencia y calidad establecidos para el proyecto. **(Kaur, 2023)**

Uso de Flutter para el Desarrollo de la Aplicación Móvil

Para el desarrollo de la aplicación móvil Maia se utilizó Flutter, un framework de código abierto creado por Google que permite construir aplicaciones nativas para Android e iOS a partir de una sola base de código. Esta tecnología ofrece una

solución altamente eficiente para proyectos que requieren rapidez de desarrollo, mantenimiento simplificado y una experiencia de usuario de alto rendimiento.

Flutter utiliza el lenguaje de programación Dart, también desarrollado por Google, el cual está optimizado para el desarrollo de interfaces gráficas y cuenta con un motor de renderizado propio. Esto permite que las aplicaciones creadas con Flutter no dependan de componentes nativos del sistema operativo, logrando una apariencia y comportamiento uniforme en diferentes plataformas.

Entre las principales ventajas que Flutter aporta a Maia se encuentran:

- **Desarrollo multiplataforma**: permite compilar una única base de código para Android e iOS, reduciendo costos y tiempos de desarrollo.
- **Hot Reload**: facilita la actualización en tiempo real de los cambios realizados en el código, acelerando las pruebas y el proceso de diseño.
- **Rendimiento nativo**: gracias a su motor de renderizado, las aplicaciones alcanzan una velocidad y fluidez comparables a las desarrolladas de forma totalmente nativa.
- **Amplia biblioteca de widgets personalizables**: Flutter ofrece una gran variedad de elementos gráficos que permiten crear interfaces atractivas, intuitivas y adaptadas a las necesidades del proyecto.

En la aplicación Maia, Flutter fue fundamental para la implementación de las diferentes pantallas y funcionalidades, como la gestión de citas, la consulta del

corte del día y la administración de perfiles de usuario y especialista. Además, su capacidad de integrarse con APIs REST facilitó la comunicación con el backend desarrollado en Laravel, asegurando la sincronización de datos en tiempo real y una experiencia de usuario ágil y confiable. **(*Flutter*, n.d.)**

Base de datos del sistema

El sistema utiliza una base de datos relacional MySQL, gestionada desde el entorno de Laravel mediante Eloquent ORM. Esta base de datos almacena toda la información esencial del sistema, entre la que se incluye:

- Datos de usuarios y especialistas (nombre, teléfono, correo, rol, credenciales).
- Información de servicios (nombre, descripción, precio, duración).
- Registro de citas (fecha, hora, especialista asignado, estado, cliente).
- Datos del corte del día (ingresos, pagos confirmados, citas atendidas).

La estructura relacional de MySQL permite realizar consultas complejas de forma eficiente y mantener la integridad referencial entre las tablas. Además, las migraciones de Laravel facilitaron la creación y actualización del esquema de la base de datos durante el desarrollo. **(*Eloquent: Getting Started*, 2020)**

Servicios utilizados

Para el funcionamiento de Maia se integraron diversos servicios que complementan la infraestructura principal:

- **Stripe**: utilizado en la versión web de Maia para procesar pagos en línea. **(*Stripe*, n.d.)**
- **Factura.com**: servicio externo que maneja la generación de facturas electrónicas, implementado en la versión web. **(*Factura.com*, n.d.)**
- **APIs internas de Maia**: servicios propios desarrollados en Laravel para gestionar usuarios, citas y reportes desde el backend.
- **Servicios de mensajería (WhatsApp API)**: permiten enviar notificaciones automáticas a los clientes cuando se crea o confirma una cita. **(*Get Started - WhatsApp Cloud API - Documentation*, n.d.)**

La aplicación móvil Maia consume principalmente los servicios internos de la plataforma mediante endpoints REST, asegurando comunicación constante y actualizaciones inmediatas.

CAPÍTULO III DESARROLLO

En este capítulo se describe el proceso completo de desarrollo de la aplicación móvil Maia, desde la planeación hasta la implementación y pruebas finales. Se detallan los métodos, técnicas, procedimientos y herramientas utilizadas a lo largo del proyecto, así como las etapas aplicadas de la metodología Feature-Driven Development (FDD).

El desarrollo de Maia se realizó con el propósito de ofrecer una herramienta tecnológica que optimiza la gestión de citas, la visualización del corte del día y la administración de información de especialistas, funcionando como una extensión móvil de la plataforma web existente.

Las preguntas que responde este capítulo son:

A través de una metodología ágil basada en la división del proyecto en funcionalidades específicas, con desarrollo incremental y pruebas continuas. Fueron utilizadas las tecnologías como Flutter y Dart para la aplicación móvil, PHP con Laravel para el backend, MySQL como base de datos, y una arquitectura de microservicios y APIs REST para la comunicación entre los módulos del sistema.

3.1 Metodología aplicada

El proyecto se desarrolló bajo la metodología Feature-Driven Development (FDD), un modelo ágil que se centra en la creación de software a partir de features.Cada característica del sistema se diseñó, implementó y probó de manera independiente, garantizando avances continuos y un control de la calidad del producto. Las etapas aplicadas fueron las siguientes:

1. Desarrollo del modelo general
2. Construcción de la lista de funcionalidades
3. Planificación por características

4. Diseño por características
5. Construcción por funcionalidades

3.2 Fase 1: Desarrollo del modelo general

Esta etapa, núcleo de la Fase 1 de FDD, se centró en la creación de un modelo de objeto global y la delimitación precisa del alcance del módulo móvil Maia App. El objetivo fue asegurar una integración exitosa con la plataforma web Agenda Maia existente, basando las decisiones en un análisis exhaustivo:

- **Análisis del Sistema Web**: Se realizó un análisis detallado del sistema web Agenda Maia para comprender la lógica de negocio, las estructuras de datos (MySQL) y la disponibilidad de las APIs existentes en Laravel.

Para este proyecto se va a usar la misma base de datos que se usa en la Página Web Maia. Solo las tablas Cliente, Citas, Corte_del_dia, Especialistas y horarios de la base de datos van a ser usadas.

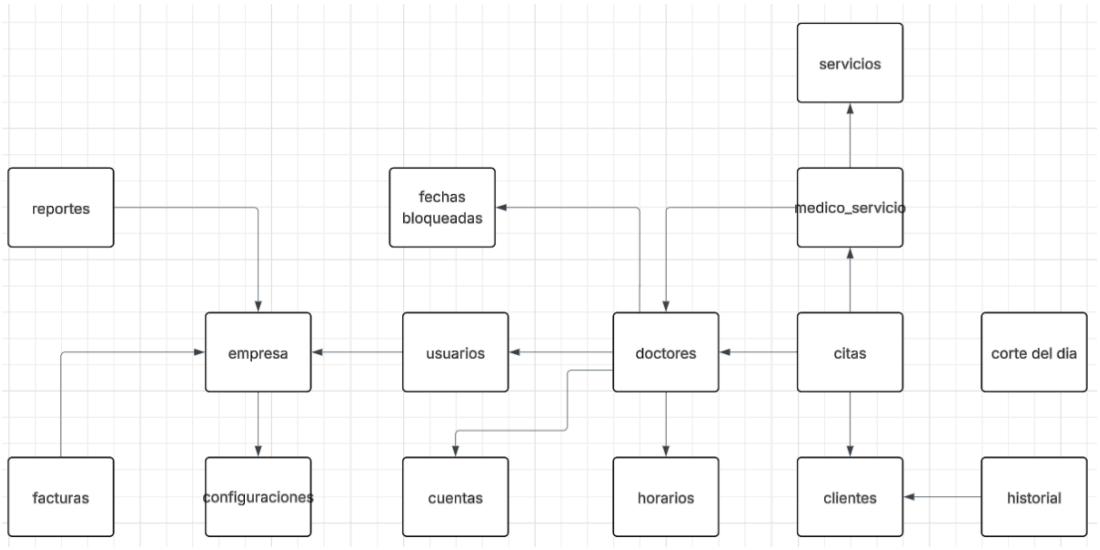

Figura 2. Diagrama de la base de datos.

Se seleccionaron solo las APIs getClientes, getCitas, postCitas, deleteCitas, updateCitas, getCorte_del_dia, getEspecialista, updateEspecialista, getHoradios, postLogin, postLogout.

postLogin (POST /auth/login)

Autentica al usuario/especialista. Devuelve token y bootstrap (usuario, especialista, permisos).

Se utilizó en la pantalla de inicio de sesión, donde la app autentica al especialista. Al recibir el token y el bootstrap (usuario, especialista, permisos), la app inicializa el estado global (BLoC de Auth) y habilita las rutas protegidas como Agenda y Corte del Día.

postLogout (POST /auth/logout)

Cierra sesión e invalida el token actual.

Se implementó en el menú de perfil, permitiendo cerrar sesión e invalidar el token. Tras ejecutarse, la app limpia la sesión local y redirige al login.

getClientes (GET /clientes?query=)

Lista/busca clientes (p. ej., por nombre o WhatsApp). Sirve para asignar un cliente al crear una cita.

Se consumió en el formulario para crear citas, donde el usuario busca clientes por nombre o WhatsApp.

Permitió llenar selectores y validar si el cliente ya existe antes de registrar la cita.

getEspecialista (GET /perfil o /especialistas/{id})

Obtiene datos del especialista (nombre, correo, teléfono, foto, permisos).

Se usó en la pantalla de Perfil, para mostrar los datos actuales del especialista (nombre, correo, teléfono, foto).

También se aplicó al iniciar sesión para configurar permisos (por ejemplo, acceso a corte global).

updateEspecialista (PATCH /perfil)

Edita datos del especialista (nombre, correo, teléfono, contraseña).

Se aplicó en la edición del perfil, dentro del formulario donde se actualizan nombre, correo, teléfono o contraseña.

Después de enviarlo, se refresca el perfil en la app con la nueva información.

getCitas (GET /citas?fecha=&especialista_id=&page= o /citas/semana?desde=)

Lista citas por día o semana (con filtros). Se usa para la pantalla de agenda.

Se utilizó en la pantalla principal de Agenda, para:

- Listar las citas del día
- Cargar citas de la semana al cambiar de fechas
- Actualizar la agenda después de crear, reagendar o cancelar

Se integró al calendario y a los estados de carga, vacío y error.

postCitas (POST /citas)

Crea una cita: especialista, servicio, cliente (nuevo o existente), fecha/hora.

Se aplicó en el formulario de crear citas, enviando:

- especialista
- servicio

- cliente
- fecha/hora

Tras crearse, la agenda se refresca automáticamente con getCitas.

updateCitas (PATCH /citas/{id})

Reagenda una cita (solo fecha/hora, no cambia especialista/servicio).

Se utilizó en la opción Reagendar, permitiendo modificar únicamente fecha/hora sin cambiar cliente o servicio.

Después de la actualización, la app sincroniza la agenda mostrando el nuevo horario.

deleteCitas (DELETE /citas/{id})

Cancela una cita. Actualiza la agenda al instante.

Se usó en el botón de cancelar cita, con confirmación previa.

Una vez eliminada, la cita desaparece de la lista sin recargar toda la pantalla, usando refresco parcial.

getHoradios (prob. “getHorarios”) (GET /horarios?especialista_id)

Devuelve horarios disponibles para un especialista/servicio en una fecha o rango. Se usa al crear/reagendar.

Se llamó en dos lugares:

- Crear cita
- Reagendar cita

Sirvió para cargar las horas disponibles según especialista, servicio y fecha seleccionada, asegurando evitar choques o solapes.

getCorte_del_dia (GET /corte?fecha=&scope=propio|global)

Regresa totales y listado de citas pagadas del día (propias o globales según permisos) y permite histórico por fecha.

Se utilizó en la pantalla de Corte del Día, para mostrar:

- total generado
- citas pagadas
- desglose propio o global (según permisos)
- navegación entre fechas

También permitió generar el histórico por fecha.

- **Definición de Alcance**: El cliente estableció el modelo de la aplicación móvil diseñada para soportar las operaciones diarias esenciales del especialista: permite iniciar y cerrar sesión de forma segura, gestionar su agenda (ver citas por día o semana, crear nuevas citas, cambiar la fecha u hora de una cita y cancelar citas) y consultar los horarios disponibles para cada servicio. Además ofrece la visualización y edición de los datos personales del especialista y un módulo de corte del día que muestra los ingresos y las citas pagadas, con acceso al histórico según los permisos del usuario. La aplicación se comunica con el sistema central para mantener la información sincronizada y consistente con la plataforma web; se entregó la documentación técnica y se realizaron pruebas de integración para validar los flujos principales.

- **Diseño de la Arquitectura Inicial**: El cliente definió la arquitectura de la solución: Flutter (Dart) para el desarrollo móvil (utilizando el patrón BLoC),

PHP-Laravel para el backend y la gestión de las APIs REST, y MySQL como base de datos. como se puede apreciar en la figura 3.

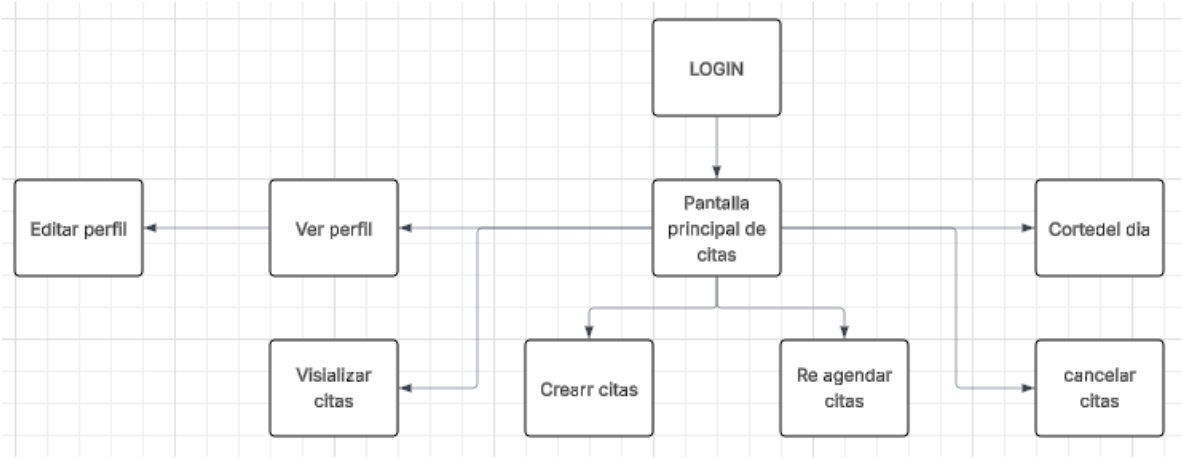

Figura 3. Diagrama de la arquitectura inicial.

Casos de uso

Para esta fase se elaboraron los diagramas de casos de uso correspondientes a la aplicación móvil Maia, considerando únicamente un tipo de usuario con todos los permisos operativos. Estos diagramas permitieron visualizar con claridad cómo interactúa el especialista con cada módulo de la aplicación y qué acciones puede ejecutar dentro de cada proceso. Además, se desarrolló un modelo conceptual de clases para el modo offline, con el objetivo de identificar las entidades clave y la relación de cardinalidad entre cada una de ellas. Todo el modelado se realizó utilizando la herramienta Draw.io, lo que facilitó la organización, la lectura y la trazabilidad del diseño.

Para los diagramas de casos de uso se siguieron los siguientes pasos:

1. Considerar al actor(es) del sistema.

2. Poner al actor del sistema con el icono de persona y darle un nombre o rol (el icono se encuentra en la sección general o de diagramas de UML).
3. Para las acciones se utilizan los óvalos (el ícono se llama “Ellipse” y está en la sección de general o diagramas de UML).
4. Dentro de los óvalos se debe poner la acción comenzando por un verbo.
5. Las acciones principales están conectadas al actor, es decir, son las que el puede realizar.
6. Algunas acciones pueden desencadenar obligatoriamente otras acciones, o algunas acciones pueden o no pueden ser obligatorias. Esto se representa con el “extend” y “include”.
    - “extend”: En ciertos casos, se añade comportamiento adicional al caso de uso principal.
    - “include”: Este caso de uso siempre ejecuta otro caso de uso como parte de su flujo normal.

A continuación, se plasman algunos ejemplos de los diagramas de caso de uso

En la figura 4 se muestra el diagrama de casos de uso del sistema para el usuario con todos los permisos, agrupando las acciones por módulo: acceso (El crud de agendar citas).

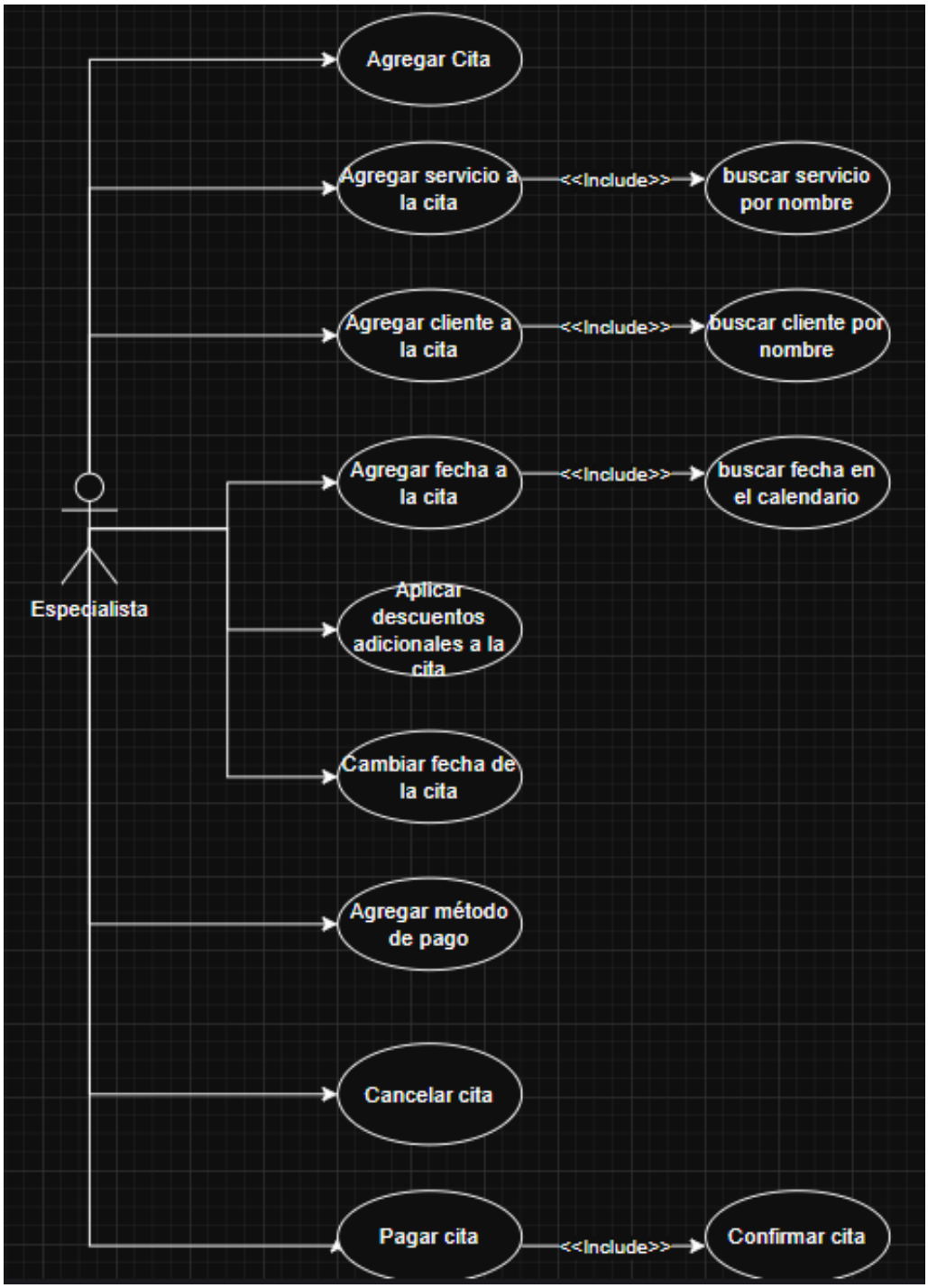

Figura 4. Diagrama de casos de uso de la app maia.

Resultado: Se tomaron las APIs seleccionas de la web y se copiaron a la carpeta “móvil” dentro del mismo proyecto, sentando las bases de la comunicación entre la app y la web. **Como se puede apreciar en la figura 4** estando las apis del login, agenda, corte del dia y perfil en la carpeta “móvil” separada de las apis de la web.

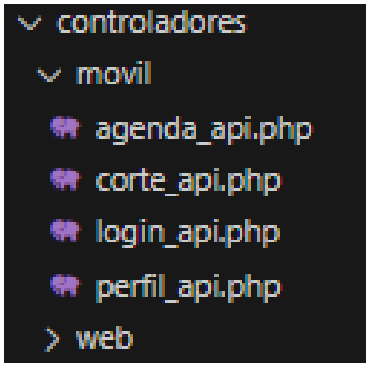

Figura 4. Representación del modelo de las Apis

.

3.3 Fase 2: Construcción de la lista de funcionalidades

La construcción de la lista de funcionalidades partió de una premisa clara: la app Maia no incluye facturación y se concentra en cuatro capacidades núcleo ,login y conexión con backend, agenda de citas (CRUD), perfil del especialista y corte del día que debían estar disponibles de forma confiable, rápida y coherente con la web. Todo esto dado por la arquitectura que dio el cliente.

Se estandarizaron las respuestas de las apis: todas las respuestas de las apis responderán con un sobre JSON común ({"ok": true|false, ...}), los códigos HTTP se normalizaron (200/201 para éxito; 400/401/403/404/409/422/500 para casos

típicos). También se fijaron convenciones de nombres (métodos HTTP en APIs REST + nombre(login,agenda,corte,perfil)), formato de fechas en ISO 8601 (YYYY-MM-DDThh:mm:ss). Esta estandarización fue crítica para que Flutter pudiera usar las respuestas que vienen de las apis del backend.

Se detalló un mapeo de dependencias funcionales: Auth es la principal, Agenda depende de catálogos (especialistas/servicios), y Corte del día depende de permisos y del estado "pagada" de las citas. Este mapeo sirvió para organizar el trabajo, establecer límites claros y definir cada funcionalidad con su propósito, entradas, salidas, reglas, validaciones y criterios de aceptación, complementado con un contrato de API validado en Postman.

3.3.1 Login y conexión con backend.

Para Login y conexión con backend, se decidió que la primera respuesta del servidor incluyera, además del token, un bootstrap con usuario, especialista y permisos. La razón fue operativa: reducir llamadas posteriores y permitir que la app pinte la pantalla de inicio sin pedir más datos.

En Postman se modelaron el POST /auth/login, acordamos mensajes de error (“credenciales inválidas”), y se documentaron la expiración/renovación del token. En Flutter definimos el DTO correspondiente y el BLoC de autenticación esperando exactamente ese contrato; cualquier cambio en backend se reflejaba inmediatamente en la colección de Postman y en la interfaz del repositorio AuthRepository.

3.3.2 Agenda (listar, crear, reagendar, cancelar).

En Agenda, la lista de funcionalidades se redujo hasta quedar cuatro acciones atómicas: listar, crear, reagendar y cancelar. se observaron que el GET /citas original de la web respondía por rango; para móvil propusimos un GET /citas?fecha=YYYY-MM-DD (día) y se añadió un GET /citas/semana?desde=YYYY-MM-DD para soportar el calendario semanal sin sobrecargar el payload.

Reagendar solo usa PATCH /citas/{id} donde solo se cambia únicamente fecha/hora. Para conflictos de horario se adoptó 409 CONFLICT con un code específico y detalles de la colisión; y se normaliza la zona horaria para que lista y calendario no tuviera error por offsets. Se alimentó estos endpoints con datos de prueba (citas adyacentes, solapes intencionales, días vacíos) y, con EXPLAIN en MySQL, se validaron índices compuestos por fecha/especialista/estado para mantener la latencia en p95 < 1.2s.

3.3.3 Perfil del especialista (ver/editar).

En Perfil del especialista, se acotaron los campos editables a nombre, correo, teléfono y contraseña. El GET /perfil reutiliza el bootstrap del login para el primer pintado y, si es necesario, refresca datos. El PATCH /perfil implementa validaciones por campo (formato de correo y teléfono, políticas de contraseña) y devuelve 422 con details por atributo en caso de error; Flutter consume ese detalle para marcar inputs específicos y mostrar mensajes precisos. Se optó por actualización optimista con sincronización posterior del caché local: al éxito, el BLoC actualiza el estado y persiste el nuevo perfil.

3.3.4 Corte del día (propio/global + histórico).

Para Corte del día se definió que el cliente pidiera fecha y alcance (propio/global), y que el servidor devolviera totales y conteos junto con un listado resumido, manteniendo un payload pequeño.

Además del día actual, habilitamos histórico para días anteriores, reutilizando la misma estructura de respuesta; si el usuario no contaba con permiso de vista global, el backend respondía 403 y la app degradar el alcance a “propio” con un aviso suave. Aquí también se fijó la meta de rendimiento (p95 < 1.2 s) y formatos numéricos consistentes.Cada ajuste se probaba en Postman; así el catálogo de features no era solo una lista, sino un conjunto de contratos verificables. Con ello, cerrar una funcionalidad significaba tener su API probada.

3.4 Fase 3: Planificación por características

Para esta fase ya se tenían las pantallas de la maqueta y las APIs probadas en Postman. Con eso listo se empezó a trabajar. La secuencia de trabajo fue Login, Agenda, Perfil, Corte del día, porque todas dependen de autenticación y la Agenda concentra la mayor complejidad. Cada entrega pasó primero por staging con datos de prueba y, una vez validada end-to-end, se integró al flujo principal.

La planificación del desarrollo mediante Feature-Driven Development (FDD) se estructuró de manera secuencial, priorizando la implementación de las funcionalidades según sus dependencias técnicas y operativas. La característica de Login y conexión con backend se definió como la de mayor prioridad (Alta), ya que establece la base de autenticación y conectividad a las APIs de Laravel, siendo un prerrequisito indispensable para que cualquier otro módulo pueda operar. A esta le siguió de cerca el módulo de Agenda (listar, crear, reagendar, cancelar), que es el corazón del proyecto y su objetivo principal, priorizándolos la capacidad de listar y crear citas como la entrega más inmediata. Finalmente, la implementación del Perfil del especialista (ver/editar) y el Corte del día (propio/global + histórico) se planificaron con una prioridad Media y Media-Baja, respectivamente, dado que son funciones de soporte y reportes que dependen de que las funcionalidades de la Agenda ya están operativas y generando datos.

En cuanto a las dependencias, la característica de Login y conexión con backend actúa como el pilar técnico, requiriendo las APIs de autenticación y la infraestructura del backend en Laravel. Todos los demás módulos dependen, lógica y técnicamente, de un inicio de sesión exitoso. El módulo de Agenda

requiere además de un conjunto específico de APIs para sus operaciones CRUD y del modelo de datos de citas. Por su parte, el módulo de Perfil del especialista necesita sus propias APIs para consultar y actualizar los datos del usuario. La dependencia más marcada se encuentra en la funcionalidad de Corte del día, la cual no solo requiere la conexión con el backend, sino que también depende directamente de que el módulo de Agenda haya generado los datos de transacciones de ingresos de las citas.

El proceso de desarrollo se dividió en tres iteraciones lógicas, asegurando entregas incrementales y funcionales. La Iteración 1, "Base de Operación", se centró en establecer los pilares del sistema, completando el Login y conexión con backend, y las funcionalidades esenciales de la Agenda: listar y crear citas. La Iteración 2, "Refinamiento de la Agenda y Perfil Básico", se enfocó en completar la gestión de citas con las funciones de reagendar y cancelar, además de incluir la visualización básica del Perfil del especialista. Finalmente, la Iteración 3, "Reportes y Cierre", integró las funcionalidades restantes: el módulo completo de Corte del día (incluyendo el alcance propio/global y el histórico), la capacidad de Editar el Perfil, y concluyó con las pruebas de integración finales y la documentación del sistema.

3.4.1 Login y conexión con backend.

La planificación de Login comenzó fijando su propósito operativo: autenticar y “precargar” el contexto del especialista para reducir llamadas posteriores. Con la maqueta aprobada, se redactó el flujo: ingreso de credenciales, llamada a /auth/login, almacenamiento seguro del token y arranque con datos de usuario,

especialista y permisos. En paralelo se normalizó el contrato del endpoint (payload de entrada, respuesta con token y objetos compactos, y mensajes de error coherentes para 401/422). En Flutter se definió el AuthBloc con eventos de LoginRequested, TokenExpired y LogoutRequested, más un repositorio GIT que está alojado en GITLab que centraliza el código y controla versiones del mismo.

Se sembraron usuarios de prueba (uno con permiso de corte global y otro sin él) y se escribieron casos de aceptación: sesión persistente tras reinicio, manejo de token expirado volviendo a la pantalla de login sin perder la intención de navegación y mensajes claros en credenciales inválidas. Esta preparación permitió implementar y validar el módulo en uno y medio a dos días, dejando una base de sesión estable que el resto de la app reutiliza.

3.4.2 Agenda (listar, crear, reagendar, cancelar).

La Agenda se planificó como la pieza central y se dividió en dos olas: primero reagendar, después agendar. Se trazó en papel un único flujo reutilizable de formulario (misma plantilla para crear y reagendar) y se documentaron reglas críticas: reagendar solo cambia fecha/hora, prohibido modificar especialista/servicio en esa acción, y rechazo inmediato de solapes con un 409 estandarizado.

Con Postman se cerraron los contratos de GET /citas (día), GET /citas/semana, POST /citas, PATCH /citas/{id} y DELETE /citas/{id}, verificando tiempos objetivo (p95 < 1.2 s) y mensajes de error consistentes. En Flutter se modeló AppointmentsBloc, se diseñó la pantalla de calendario/listas con estados de cargando/vacío/error y se sembraron 25 citas con casos distontos (adyacencias y

solapes intencionales). La primera ola (reagendar) tomó ~3 días porque se resolvieron normalización de zona horaria y conflictos de modelo/JSON; gracias a esa plantilla compartida, la segunda ola (agendar) quedó en menos de un día. El plan contempló pantallas de carga previas al pintado, eliminación de dos clases redundantes heredadas y una pasada de optimización de rutas para simplificar navegación.

El criterio de aceptación exigía CRUD extremo a extremo, refresco instantáneo de la lista tras cualquier operación y validaciones de negocio activas.

3.4.3 Perfil del especialista (ver/editar).

Para Perfil se planificó un flujo muy directo: primer pintado con los datos que ya trae el bootstrap de login (para evitar otra llamada inicial) y edición con validaciones por campo. Se cerró el contrato de /perfil (GET/PATCH), dejando a los errores por campo en 422 para que la UI mostrará mensajes precisos. A nivel de estado se definió ProfileBloc con un ciclo de “cargar, editar, confirmar, sincronizar caché local”, y se acordó que, al guardar, la app actualiza de inmediato el modelo en memoria para mantener consistencia sin depender de una recarga.

Los casos de aceptación incluyeron colisiones de correo (mostrando detalle de validación), cambios parciales (sin tocar contraseña) y preservación de sesión. Con la maqueta como base, solo se adaptaron los widgets a los estilos del proyecto y se documentó el pequeño mapa de navegación (Perfil, Editar, Perfil), de forma que pruebas manuales e integración se pudieran ejecutar en una tarde.

3.4.4 Corte del día (propio/global + histórico).

La planificación de Corte partió de una decisión de producto: la app muestra ingresos del día y permite moverse a días anteriores, con alcance propio o global según permisos. Se preparó el endpoint GET /corte?fecha=YYYY-MM-DD y, si aplica, un histórico acotado; se acordaron agregaciones en servidor (totales y conteos) para cumplir latencias objetivo.

A nivel de UI se diseñó una pantalla con estados claros: cargando, vacío cuando no hay pagos y error con acción de reintento. En el estado ReportBloc se añadió memoización por fecha para no repetir la misma consulta en la misma sesión. Los casos de aceptación exigían precisión de totales respecto a una verificación manual y transición fluida entre fechas sin perder el estado de filtros. Finalmente, se probó con usuarios con y sin permiso de “corte global”, asegurando que un 403 mantuviera el alcance en “propio” sin romper la navegación.

En conjunto, esta planificación garantizó que cada entrega contratos modelados (BLoC, DTOs, repositorios) y criterios de verificación medibles (DoD). Gracias a ello, se pudo implementar, probar e integrar cada feature con muy poca fricción, manteniendo un ritmo constante y una trazabilidad clara desde el objetivo de negocio hasta el código y las pruebas que lo validan.

3.5 Fase 4: Diseño por características

El punto de partida fue una aplicación ya maquetada conforme al diseño aprobado. A partir de ese entregable, se estructuró la capa de lógica y datos para conectar las pantallas con el backend en Laravel mediante APIs REST, sin alterar estilos ni remaquetar vistas. Para ello se definieron los contratos de API (endpoints, métodos, parámetros, esquemas JSON y códigos de respuesta) y se trasladaron a DTOs en Flutter, añadiendo normalización de fechas/horas y mapeo defensivo de campos opcionales.

Sobre estos DTOs se construyeron los repositorios alojados en GitLab (consumo HTTP, manejo de tokens, cabeceras y errores) y la capa de estado con BLoC (eventos/estados por pantalla), de forma que la UI maquetada reaccionara correctamente a los estados cargando, éxito, vacío y error sin requerir cambios visuales. También se definió la navegación protegida (rutas que exigen sesión válida) y la gestión de expiración del token: ante un 401, la app redirige a login, conserva la intención del usuario y regresa al punto exacto tras reautenticar.

En Login, se busca que se conecté el formulario ya maquetado con el endpoint /auth/login, se validen credenciales en cliente, se guardé el token en almacenamiento seguro y ejecuté un bootstrap inicial (usuario, especialista y permisos) para reducir llamadas posteriores.

El BLoC centraliza mensajes de error (“credenciales inválidas”, “sin conexión”) y bloquea el botón durante la petición para evitar envíos duplicados; todo esto se integró sobre la UI existente, sin modificar su apariencia.

Podemos apreciar que en la figura 5 está el login, la pantalla que recibe al usuario cuando este abre la aplicación y no está logueado.

Figura 5.- Login de la app Maia

En Agenda, se trabajó sobre las pantallas maquetadas de lista (día/semana) y calendario. Se unificó la lógica de agendar y reagendar usando la misma pantalla ya maquetada, pero bloqueando en lógica los campos que no deben cambiar al

reagendar (especialista/servicio) y permitiendo solo fecha/hora, según las reglas del negocio. Se implementó la detección de solapes: si el backend devuelve 409, el BLoC mantiene la pantalla en contexto y muestra el mensaje correspondiente, mientras la lista se actualiza puntualmente tras crear/reagendar/cancelar, sin recargar toda la vista. Toda la normalización de zona horaria y formatos ISO se resolvió en la capa de datos para que la UI no requiriera cambios.

Podemos apreciar que en la figura 6 están las pantallas de agenda, estas pantallas sirven para ver las citas del día, programar citas, cancelar citas, detalle de citas y reprogramar citas.

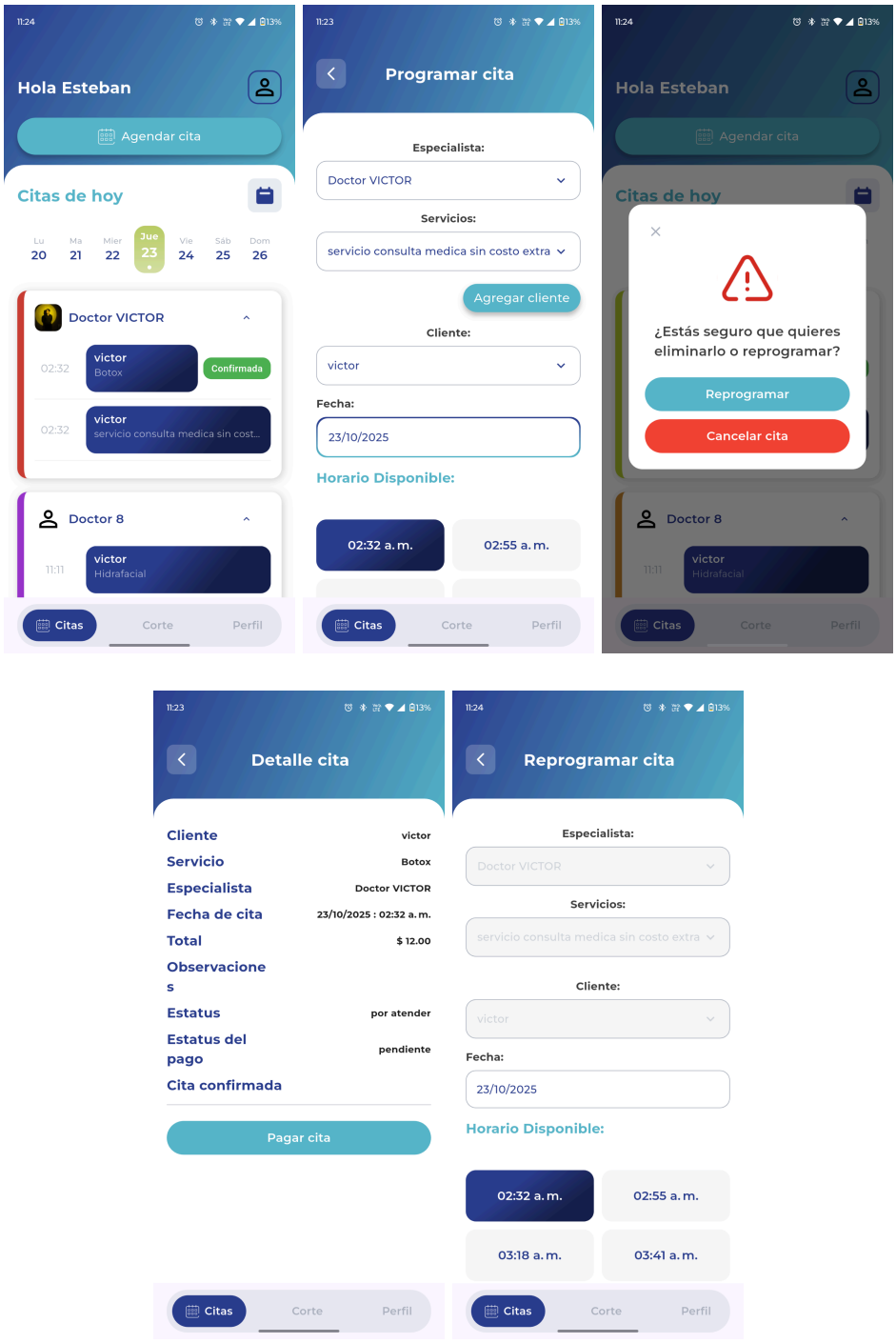

Figura 6.- Pantallas del crud de la agenda maia

En Perfil, se usaron los datos de bootstrap para que la pantalla “ver perfil” mostrará información inmediata sin llamadas extra. Al entrar en “editar”, la lógica realiza un fetch fresco, aplica validaciones por campo (correo/teléfono/contraseña si se cambia) y envía un PATCH al guardar. Se aplique actualización optimista: la UI refleja el cambio y, si el servidor responde 422, se revierte sólo el campo afectado y se muestra el error contextual, respetando por completo la maquetación.

Podemos apreciar que en la figura 7 está el corte del día, en esta pantalla el usuario podrá ver el corte del día dependiendo del permiso (global y local).

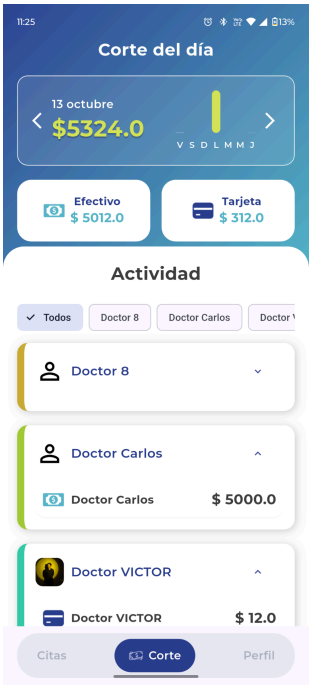

Figura 7.- Pantalla con el corte del dia de la app Maia

En Corte del día, se conectó la pantalla de métricas y listado con los endpoints de reporte. El alcance global se habilita o no desde permisos cargados en el login; si

el usuario intenta una vista para la que no tiene permiso, la lógica lo impide y muestra la razón, sin necesidad de modificar la interfaz.Se Implementó navegación conservando filtros, y estados “vacío”/“cargando” gestionados desde BLoC con los componentes ya presentes en la maquetación.

Podemos apreciar que en la figura 8 están las pantallas del perfil, en esta pantalla el usuario podrá modificar y visualizar los datos del doctor.

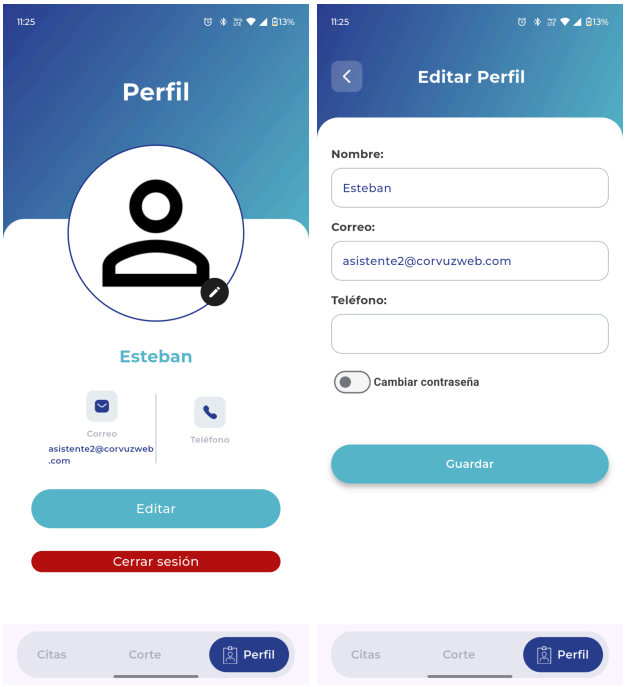

Figura 8.- Pantalla para editar el perfil del especialista y asistente de la app Maia

Finalmente, se documentaron todos los contratos de API y se validaron en Postman (200/4xx/5xx), se dejaron versionados los DTOs, repositorios alojados en GitLab y BLoCs, y se aseguro que cada pantalla maquetada tuviera su ciclo de

estados bien definido desde la lógica (cargando , éxito / vacío / error), manteniendo intacta la presentación visual.

Lo que se realizó fue conectar y robustecer las pantallas entregadas, garantizando reglas de negocio, seguridad de sesión, manejo de errores y rendimiento percibido, sin tocar el trabajo de Diseño/Maquetación.

3.6 Fase 5: Construcción por características

La primera construcción real fue Login. se implementó el formulario con validaciones en cliente y se conectó AuthRepository.login() al endpoint; al recibir 200, guardaba el token en almacenamiento seguro y ejecutaba un “bootstrap” que traía usuario, especialista y permisos.

Se decidió resolver la persistencia de sesión en el propio AuthBloc: si el token vencía, interceptaban la respuesta, mostraba un aviso corto y redirige al login conservando la intención de navegación. Los errores 401/422 se mapean a mensajes específicos (“credenciales inválidas”, “verifique sus datos”), y se deshabilita el botón mientras el estado era AuthLoading para evitar solicitudes duplicadas.Con la sesión estable, se construyó la Agenda empezando por el listado diario/semanal. Lo primero que apareció fueron discrepancias entre los modelos móviles y las respuestas del backend (nombres y tipos); Se soluciono normalizando los DTOs y acordando el contrato con backend para fechas en ISO.

En paralelo, se ajustó la conversión de zona horaria para que lo que se seleccionaba en el DateTimePicker coincidiera exactamente con lo que veía el especialista en la web. Sobre esa base se desarrolló una plantilla única para agendar y reagendar: el mismo formulario, pero con reglas distintas en el BLoC; reagendar permitía exclusivamente cambiar fecha/hora y conservaba el resto del payload.

Esa parte tomó más tiempo porque se incorporaron validaciones de colisiones: al recibir un 409 desde el servidor, el BLoC traducía la causa a un mensaje claro y mantenía el estado del formulario, evitando que el usuario perdiera lo capturado. Gracias a esa plantilla, la pantalla de agendar quedó muy rápida: sólo se cambiaron las precondiciones y el dispatch del evento. Para cancelar, se utilizó un diálogo de confirmación; al confirmar, hacía DELETE y actualizaba la lista aplicando un refresco fino: elimina o modifica el elemento afectado sin reconstruir toda la pantalla, lo que hizo más fluida la experiencia en listas largas.

Durante esta fase se eliminaron dos clases que habían quedado sin uso (“citas asistentes” y “agendar cita” separadas) y se consolidó la navegación entre pantallas para que el flujo de Agenda fuese predecible: regresar desde reagendar debía volver a la misma fecha y posición de scroll.

Se añadieron loaders no bloqueantes y skeletons antes del pintado real para que el usuario no percibiera “saltos”. En rendimiento, se activó itemBuilder y se utilizó memorización ligera de catálogos (especialistas/servicios) para no pedirlos en cada navegación.El módulo de Perfil se construyó aprovechando que en el login ya traía los datos: la pantalla de visualización renderiza desde ese bootstrap y sólo hace GET /perfil si el caché está ausente.

Para editar, conecté PATCH /perfil con validaciones por campo. Los errores 422 venían con details por atributo; se mapean a mensajes debajo de cada TextField y, al éxito, se actualicen de inmediato el estado local para que el usuario viera los cambios sin recargar. También que un cambio de correo no invalidará la sesión en curso.En Corte del día se priorizo la precisión antes que el aspecto visual. Se probó con Postman los filtros scope=propio|global y la consulta por fecha. La UI muestra tarjetas de totales y un resumen de citas pagadas; si el usuario no tiene permiso para “global”, el servidor responde 403 y el BLoC fuerza el estado “propio” con un aviso breve.

Se agregó la navegación a histórico sin perder filtros: al cambiar de fecha, se mantiene el scope y la posición; si el día no tiene pagos, aparece un estado vacío informativo en vez de un error genérico. Para sostener el p95 por debajo de 1.2 s, coordiné con backend índices en citas(fecha, especialista_id, estado_pagado) y agregados más eficientes.

En el backend Laravel cada request valida y autoriza con políticas; todas las respuestas siguen un contrato JSON estandarizado (ok, data, y en errores code, message, details).

Las pruebas no se dejaron para el final. Por cada feature se hicieron tests unitarios de BLoC (transiciones de estado), validadores y mapeos DTO; se ejecutó la colección de Postman para contrato y escenarios negativos (401/403/409/422/500); y se cerró con pruebas de integración en emulador y dispositivo físico recorriendo el flujo completo: login , agenda (crear/reagendar/cancelar) , corte , perfil. y muestra mensajes claros cuando no hay conectividad.

Cada incremento terminaba con un Pull Request a develop. Si en regresión aparecía un fallo, lo corregía en la misma rama y volvía a pasar la colección de Postman antes de fusionarla. Sólo cuando el recorrido completo estuvo estable y sin regresiones se etiquetó la versión y se liberó. Este ritmo de pequeñas entregas funcionales, verificables y documentadas fue el que permitió detectar errores temprano, mantener la app sincronizada con la web y sostener la calidad mientras el sistema crecía.

CAPÍTULO IV RESULTADOS

En la figura 9 se presenta la pantalla inicial del sistema, en la cual se establece el punto de ingreso a la aplicación. En esta etapa, el usuario tiene la posibilidad de iniciar sesión y proporcionar sus credenciales de acceso (correo electrónico y contraseña) con el fin de permitir la validación correspondiente y la posterior apertura de sesión o recuperar su contraseña en dado caso que el usuario no la recuerde.

Figura 9.- Login de la app Maia

En la figura 10 se puede apreciar la primer pantalla de tres dedicada a ayudar al usuario a recuperar la contraseña de su cuenta, en esta pantalla únicamente pide un correo electrónico el cual se usará para mandar un código de verificación.

Figura 10.- Pantalla 1/3 para recuperar la contraseña

En la figura 11 se puede busca ayudar al usuario a recuperar la contraseña de su cuenta, en esta pantalla únicamente pide el código de verificación que fue enviado al correo electrónico que la pantalla de la figura 12, esta pantalla da la opción de validar el código o de re enviar otro al mismo correo, esto cada 30 seg.

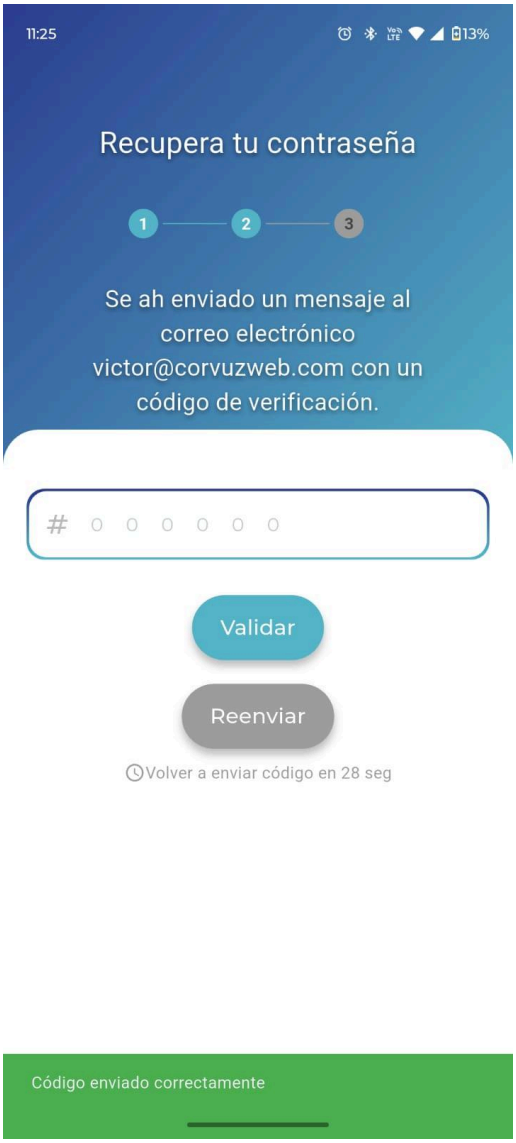

Figura 11.- Pantalla 2/3 para recuperar la contraseña

En la figura 12 se puede apreciar la tercer pantalla de tres dedicada a recuperar la contraseña de su cuenta, a esta pantalla solo se puede llegar si el código de verificación fue validado en la pantalla de la figura 13, en esta pantalla nos pide una nueva contraseña y nos pide confirmar esta misma contraseña.

Figura 12.- Pantalla 3/3 para recuperar la contraseña

En la figura 13 se puede apreciar la pantalla principal de la app Maia después de ser logueado, se pueden apreciar un saludo al doctor logueado, la opcion rapida para ver el perfil del especialista de la pantalla de la figura XX y el boton para agendar una cita, después de aprecian las citas registradas para el dia de hoy de los diferentes doctores.

Figura 13.- Pantalla mostrando las citas del día actual

En la figura 14 se puede apreciar la pantalla principal de la app Maia después de ser logueado, se puede apreciar el calendario completo para seleccionar cualquier fecha y ver las citas de estos días.

Figura 14.- Pantalla mostrando mostrando el calendario de las citas

En la figura 16 se puede apreciar la pantalla para programar una cita, aquí te pide que se escoja el especialista, el servicio, el cliente, el día y la hora, todos estos datos los trae desde el backend asegurando congruencia con la web.

Figura 16.- Pantalla con datos para programar una cita de la app Maia

En la figura 17 se puede apreciar la pantalla que te avisa cuando una cita ha sido creada,aparece una notificación en la aplicación que informa cuando se crea una cita con la información del doctor, la fecha y la hora.

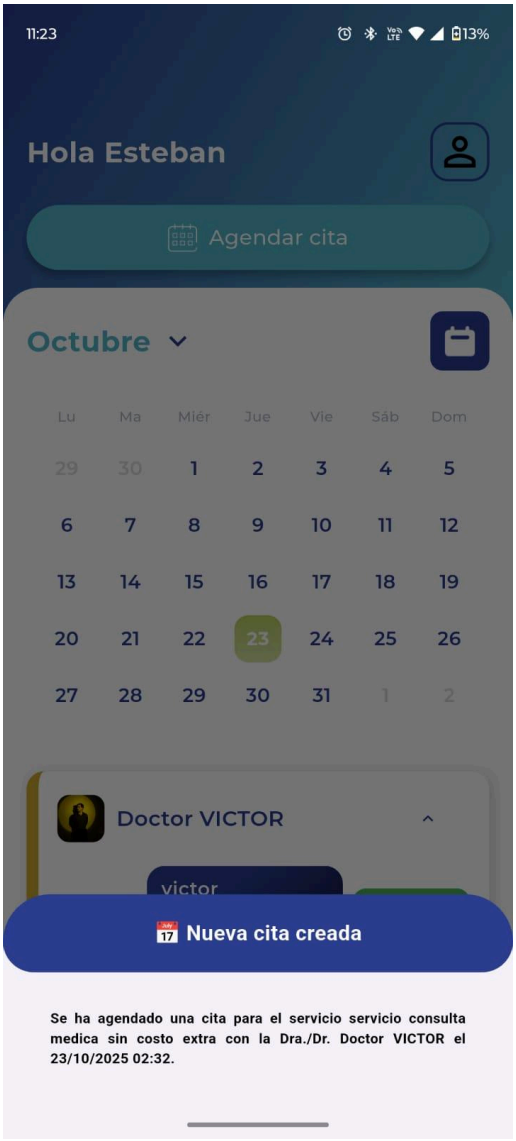

Figura 17.- Pantalla de notificación interna cuando se crea una cita

En la figura 18 se puede apreciar la pantalla del detalle de la cita con toda la información relevante que se guarda de esta como el cliente, servicio, especialista, fecha, total,observaciones, estatus, estatus de pago y si está confirmada, después está el botón para pagar la cita.

Figura 18.- Pantalla para mostrar el detalle de la citas de la app Maia

En la figura 19 se puede apreciar las opciones para reprogramar una cita y para cancelar una cita en una ventana emergente que está sobre la pantalla de la figura 15.

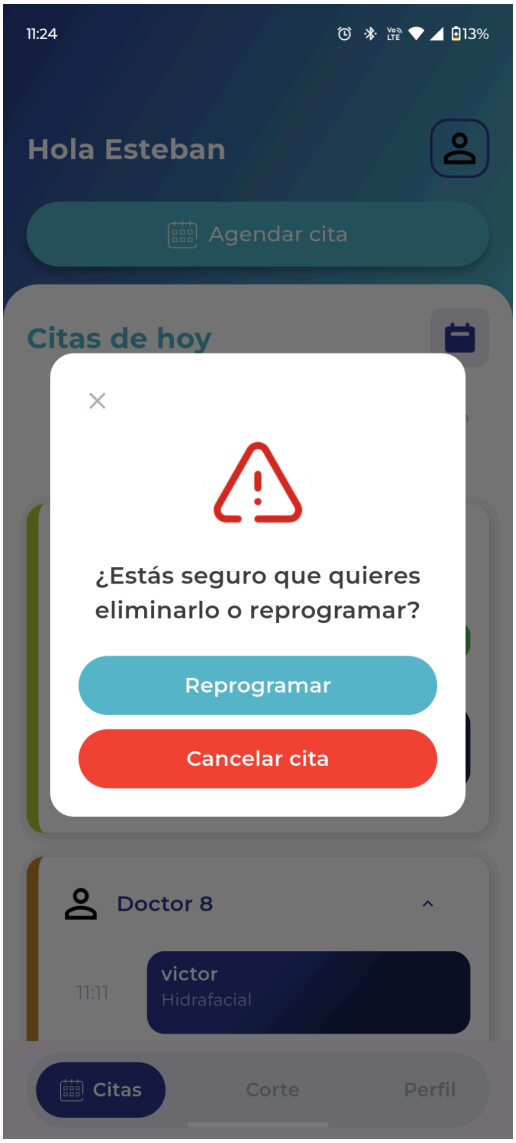

Figura 19.- Pantalla para mostrar las opciones de re programar y cancelar una cita

En la figura 20 se puede apreciar la pantalla para re programar una cita, aquí te pide que se escoja el nuevo día y la hora, ya que no se pueden cambiar los demás datos, todos estos datos los trae desde el backend asegurando congruencia con la web.

Figura 20.- Pantalla para el asistentes de listar citas con calendario de la app Maia

En la figura 21 se puede apreciar la opción para confirmar una cita en una ventana emergente que está sobre la pantalla de la figura 15.

Figura 21.- Pantalla para confirmar una cita

En la figura 22 se pueden apreciar los detalles de la cita y también las opciones de pago que son cortesía, efectivo, tarjeta de crédito y tarjeta de débito, al final el botón para pagar.

Figura 22.- Pantalla para pagar una cita

En la figura 23 se puede apreciar la pantalla del corte del día con datos para diferentes doctores, lo primero que hay es el día y el total de dinero de las citas que se pagaron ese día con una pequeña gráfica comparando con los demas dias de la semana, la aplicación separa el dinero por doctor y por tipo de pago.

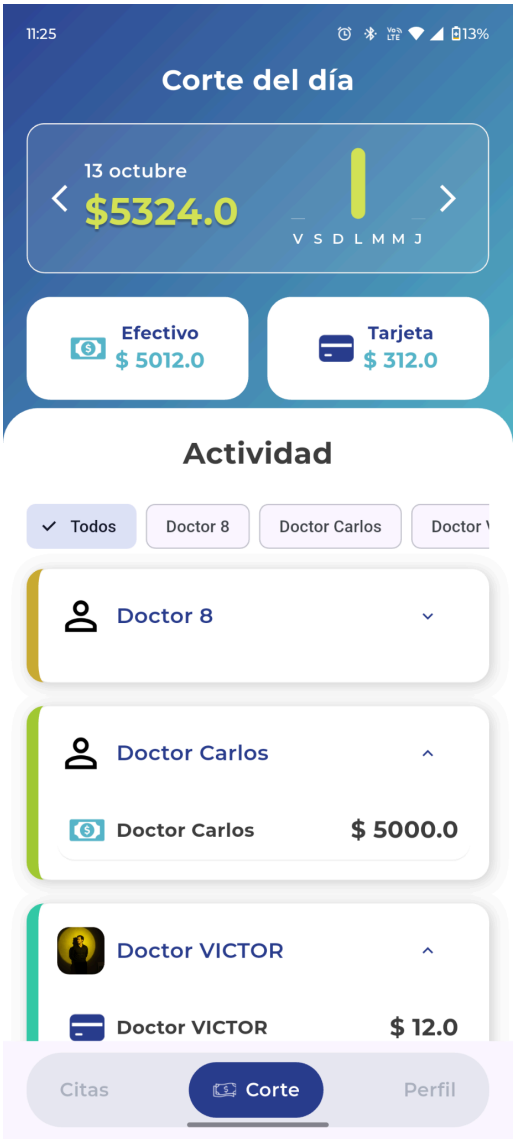

Figura 23.- Pantalla con el corte del dia de la app Maia

En la figura 24 se pueden apreciar los datos principales del doctor que esté logueado como su foto de perfil, correo electronico y telefono ademas de el boton para editar estos datos y el botón para cerrar la sesión que te redirige al login que se aprecia en la figura 11.

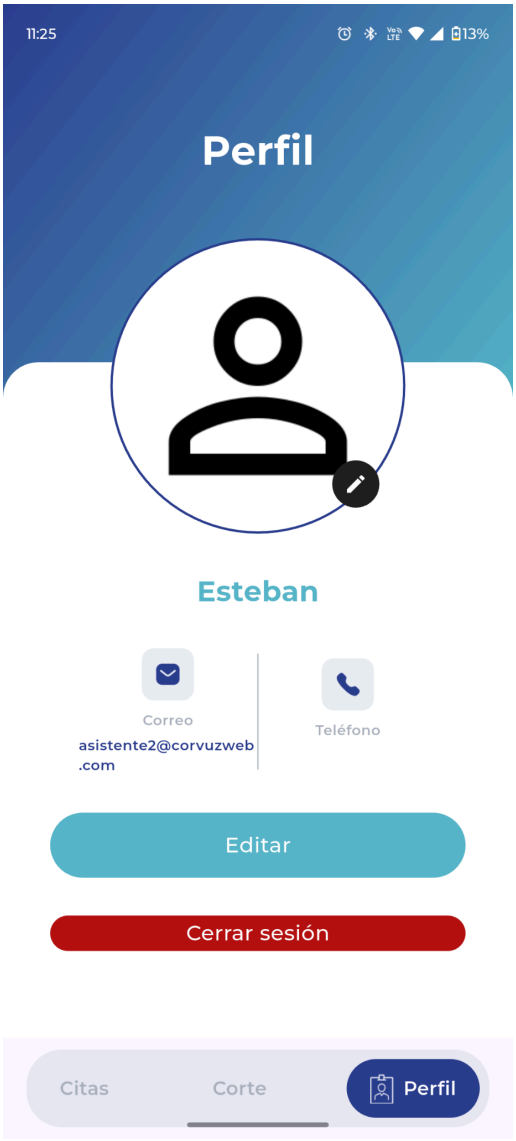

Figura 24.- Pantalla con el perfil de especialista o el asistente de la app Maia

En la figura 25 se puede apreciar la pantalla para editar los datos del perfil del doctor con los datos de correo electrónico, nombre, contraseña y teléfono además de el botón para guardar estos cambios.

Figura 25.- Pantalla para editar el perfil del especialista y asistente de la app Maia

La figura 26 muestra una ventana emergente de confirmación que aparece al intentar salir de la aplicación, preguntando al usuario si está seguro de que desea cerrarla.

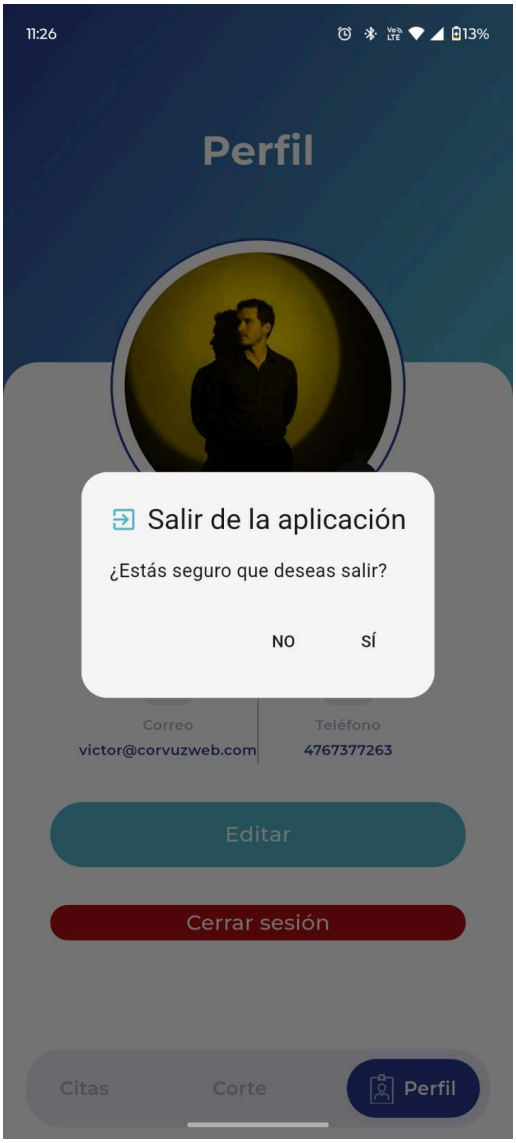

Figura 26.- Pantalla emergente cuando quieres salir de la app

CAPÍTULO V. CONCLUSIONES, RECOMENDACIONES Y EXPERIENCIA PROFESIONAL ADQUIRIDA

5.1 CONCLUSIONES

Conclusión General

El proyecto Maia cumplió su objetivo: proveer a los especialistas una aplicación móvil ágil, confiable y accesible para gestionar citas, administrar su perfil y consultar el corte del día, complementando la plataforma web. La adopción de Feature-Driven Development (FDD) permitió entregas incrementales verificables, asegurando usabilidad y rendimiento en cada funcionalidad priorizada. La arquitectura Flutter (BLoC) + Laravel (APIs REST) + MySQL demostró ser coherente y escalable, y la app móvil—sin incluir facturación—quedó integrada de forma estable con la web.

Conclusiones Parciales

Gestión de Citas (CRUD): creación, reagendado (solo fecha/hora) y cancelación operan de extremo a extremo con actualización inmediata del listado; se validan solapes de horario (409).

Corte del Día: consulta de ingresos diarios con filtros propio/global (según permisos) e histórico; cálculos en servidor con tiempos de respuesta adecuados.

Perfil del Especialista: visualización y edición con validaciones por campo y actualización optimista; coherencia entre estado local y remoto.

Calidad e Integración: comunicación app–backend validada con pruebas de contrato; tiempos de respuesta dentro de meta (p95 < 1.2 s) y primera carga fluida (< 2.5 s); sin regresiones en los flujos principales.

Aportaciones a la Disciplina

Endpoints a medida para móvil y manejo de errores estandarizado (ok/code/message/details), mejorando testabilidad y mantenimiento.

5.2 RECOMENDACIONES

1. Sugerencias para mejorar los métodos de estudio/desarrollo (FDD)

- **Formalizar las métricas de FDD:** Aunque la metodología Feature-Driven Development (FDD) se aplicó con éxito, se recomienda formalizar métricas claras y cuantificables para cada etapa del proceso (como el tiempo promedio para "Diseñar por Funcionalidad" y "Construir por Funcionalidad"). Esto permitiría una gestión más precisa de proyectos futuros y una mejora continua en la estimación de tiempos.

2. Acciones específicas en base a las consecuencias

- **Creación de un Plan de Mantenimiento y Actualización Post-lanzamiento:** Ante el éxito de la integración de la App Maia con la plataforma web, se recomienda establecer un ciclo de vida de mantenimiento claro. Esto incluye un plan para monitorear el rendimiento de las APIs ajustadas, corregir errores reportados en producción y asegurar la compatibilidad con las nuevas versiones de los sistemas operativos móviles (iOS/Android).

- **Capacitación Formal a Usuarios Clave:** Dado que la aplicación está diseñada para optimizar el trabajo de los especialistas, se debe elaborar un

manual de usuario detallado y realizar sesiones de capacitación para los especialistas y el personal de soporte de CORVUZ/Agenda Maia, maximizando la adopción y el uso eficiente de las nuevas funcionalidades .

3. Sugerencias para futuras investigaciones o desarrollos

- **Integración de Notificaciones Push Avanzadas:** Investigar e implementar un sistema de notificaciones push más inteligente, que no solo alerte sobre la programación o cancelación de citas, sino que también envíe recordatorios automatizados a los clientes.

5.3 COMPETENCIAS DESARROLLADAS Y/O APLICADAS

Competencias instrumentales

- **Análisis y síntesis:** Se realizó un análisis detallado del funcionamiento de la plataforma web Maia para identificar los procesos esenciales que debían trasladarse a la aplicación móvil. Esto permitió sintetizar flujos completos ,inicio de sesión, gestión de citas, consulta del perfil y corte del día, en interfaces y acciones móviles más simples y directas. Este proceso reforzó la capacidad de transformar requerimientos amplios en funcionalidades concretas y bien delimitadas.

- **Planificación (FDD):** Se aplicó Feature-Driven Development para organizar el trabajo en torno a funcionalidades completas. Esto permitió priorizar de manera estratégica (Login, Agenda, Perfil y Corte del día) y planificar entregas progresivas. El uso de Git como soporte de control ayudó a

mantener una estructura ordenada de versiones y a garantizar que cada funcionalidad pudiera revisarse, probarse y mejorar de forma independiente. Esta experiencia fortaleció la capacidad de planificar de forma ordenada y orientada a resultados.

- **Técnicas de la carrera:** Se aplicaron herramientas aprendidas durante la formación profesional, como Flutter con BLoC para la capa móvil, Laravel 10 para la lógica del backend y MySQL para el almacenamiento de datos. La combinación de estas tecnologías permitió construir un sistema robusto y modular. El proceso facilitó reforzar conocimientos técnicos adquiridos en materias como desarrollo móvil y bases de datos, aplicándolos en un contexto real y con requerimientos de calidad profesional.

- **Solución de problemas:** Se diseñó una plantilla unificada para crear y modificar citas, reduciendo duplicidad y haciendo más sencillo el mantenimiento. Esta solución surgió de la necesidad de simplificar flujos que originalmente eran distintos entre sí. El proceso fortaleció la capacidad de detectar patrones, optimizar procesos y encontrar alternativas más eficientes sin afectar la funcionalidad.

Competencias interpersonales

- **Trabajo en equipo:** Se colaboró estrechamente con los miembros del equipo responsables de la maquetación y del backend. La integración entre diseño, lógica de negocio y funcionamiento real requirió coordinación constante. Esta dinámica permitió mejorar la colaboración profesional y la capacidad de contribuir activamente dentro de un proyecto multidisciplinario.

- **Comunicación efectiva:** Durante el desarrollo se mantuvo comunicación clara y puntual con el equipo para resolver dudas, validar decisiones y

evitar malentendidos técnicos. Se explicaron flujos, restricciones y necesidades de manera accesible para todos los involucrados,para ello se utilizó la herramienta Click que es la que usa la empresa para mensajes internos con el equipo fortaleciendo la habilidad de comunicar aspectos técnicos de forma comprensible.

- **Crítica/autocrítica:** Se identificaron retrasos en los tiempos de entrega inicialmente planeados debido a la complejidad de la implementación y ajustes requeridos. Esta situación motivó un enfoque de la priorización de tareas, implementando un mejor control del avance para asegurar el cumplimiento de los siguientes hitos sin comprometer la calidad.

- **Ética profesional:** Se siguieron estrictamente las solicitudes del cliente en cuanto al manejo de código, datos y decisiones funcionales. No se almacenó información no autorizada ni se modificaron flujos fuera del alcance acordado. Esto fortaleció el compromiso ético y la responsabilidad profesional dentro del proyecto.

Competencias sistémicas

- **Aplicación práctica y autonomía:** Se implementó la arquitectura por capas completa: capas de presentación (pantallas maquetadas), lógica de estado con BLoC, repositorios que encapsulan llamadas HTTP, y consumo de APIs en Laravel con persistencia en MySQL. Desplegó y probó features de punta a punta (login → bootstrap → agenda CRUD → corte → perfil) de manera autónoma. Para garantizar separación de responsabilidades, facilitar pruebas y permitir que cada cambio afecte lo mínimo posible al resto del sistema.Se aprendió a como orquestar un flujo end-to-end en un proyecto real, decidir límites entre responsabilidades (qué queda en UI vs. BLoC vs. repositorio), y automatizar pasos repetitivos del desarrollo (scripts, seeds, Postman collections).

- **Adaptabilidad y aprendizaje:** Se gestionaron las migraciones y compatibilidades (actualizar dependencias a Laravel 10), adaptó el consumo de APIs cuando se modificaron payloads (permisos para corte global). Esto fue porque el entorno productivo y las dependencias cambian; adaptarse rápido evita bloqueos y re trabajo y asegura que la app mantenga coherencia con la web. Se aprendieron las técnicas prácticas para evaluar compatibilidad, priorizar cambios (qué migrar primero). También mejoró la capacidad de diagnosticar errores de dependencia y de rollback controlado.

- **Calidad:** Se definieron metas de calidad y rendimiento de los tiempos de respuesta de las apis (objetivos p95), implementó pruebas de contrato con Postman y se estandarizaron respuestas de API (ok/code/message/details). Lideró las decisiones técnicas sobre índices en MySQL y normalización de TZ para mejorar precisión y rendimiento. esto se hizo para asegurar que las funcionalidades entregadas no sólo funcionaran, sino que fueran mantenibles, medibles y confiables en producción. La estandarización facilita debugging y soporte operativo. Se aprendió a como traducir requisitos de negocio a métricas técnicas, cómo priorizar acciones de optimización (índices, paginación, reducción de payloads) y cómo guiar al equipo en acuerdos técnicos que mejoren mantenimiento y soporte.

References

A, C., & J, C. (2015). Diseño y validación de arquitecturas de aplicaciones empresariales. *RISTI-Revista Ibérica de Sistemas y Tecnologías de Información*, *1*(4), 79-91. https://pdfs.semanticscholar.org/029e/e612f9ccd1155c7d5ca5cd791a9a718f0ba2.pdf

Eloquent: Getting Started. (2020, March 4). Laravel. Retrieved October 24, 2025, from https://laravel.com/docs/12.x/eloquent

F, A. (2017). *Agile software development models tdd, fdd, dsdm, and crystal methods: A survey* (Vol. 1-10). International journal of multidisciplinary sciences and engineering. https://es.scribd.com/document/557781420/Agile-Software-Development-Models-TDD-FDD-DSDM-And-Crystal-Methods-a-Survey

Factura.com. (n.d.). Servicio de Facturación Electrónica - Factura.com. Retrieved October 24, 2025, from https://factura.com/

Flutter. (n.d.). Flutter - Build apps for any screen. Retrieved October 24, 2025, from https://flutter.dev/

Get Started - WhatsApp Cloud API - Documentation. (n.d.). Meta for Developers. Retrieved October 24, 2025, from https://developers.facebook.com/docs/whatsapp/cloud-api/get-started?locale=es_LA

Javascript - Javascript en español. (n.d.). Lenguaje JS. Retrieved October 24, 2025, from https://lenguajejs.com/javascript/

Kaur, P. (2023, January 3). *Top 5 PHP REST API Frameworks*. Moesif. Retrieved October 24, 2025, from https://www.moesif.com/blog/api-product-management/api-analytics/Top-5-PHP-REST-API-Frameworks/

Kuusinen, k., & Mikkonen, T. (2013). Designing user experience for mobile apps: Long-term product owner perspective. *Asia-Pacific Software Engineering Conference*, *1*(20), 535 - 540. https://www.researchgate.net/publication/271550077_Designing_User_Experience_for_Mobile_Apps_Long-Term_Product_Owner_Perspective

¿Qué es REST? (2025, April 1). REST API Tutorial: What is REST? Retrieved October 24, 2025, from https://restfulapi.net/

¿Qué son los microservicios? (n.d.). IBM. Retrieved October 24, 2025, from https://www.ibm.com/mx-es/think/topics/microservices

Rivera, I. (2020, 1 17). *¿En qué consiste el Desarrollo Dirigido por Características (FDD)?* IVAN RIVERA PMP. https://ivanrivera-pmp.blogspot.com/2020/01/en-que-consiste-el-desarrollo-dirigido.html

Stauffer, M. (2019). *Laravel: Up & Running, a Framework for Building Modern PHP Apps*. O'Reilly Media, Incorporated. http://oreilly.com/catalog/errata.csp?isbn=9781492041214

Stripe. (n.d.). Stripe | Una infraestructura para pagos online. Retrieved October 24, 2025, from https://stripe.com/es

ANEXOS

- Carta de autorización por parte de la empresa u organización para la titulación.

CORVUZ

Hermenegildo Bustos 215
La Barda, C. P. 36400
Purísima del Rincón, Gto.
4761133570
hola@corvuz.com

Purísima del Rincón 21/11/2025

ASUNTO: Carta de autorización por parte de la empresa u organización para la titulación

Ing. Héctor Javier Ríos Flores
Director de Operaciones de Corvuz
Presente

Por medio de la presente hago constar que el estudiante **VICTOR MANUEL CASTILLO OLIVETTO** con número de control **RS21110113** del programa educativo de **INGENIERÍA INFORMÁTICA** del **INSTITUTO TECNOLÓGICO SUPERIOR DE PURÍSIMA DEL RINCÓN**, quien realizó su **RESIDENCIA PROFESIONAL** durante el periodo **AGOSTO – DICIEMBRE 2025** en **Corvuz**, por lo que manifiesto que dicho trabajo ha sido liberado.

Hago de su conocimiento lo anterior, para los fines que al interesado/a convengan.

Héctor Javier R.F.

(SELLO DPTO. CORRESPONDIENTE)
FIRMA Y NOMBRE DEL ASESOR EXTERNO
PUESTO

- Cronograma de actividades

mes		julio				agosto				septiembre				octubre			
semana		semana 5	semana1	semana2	semana3	semana4	semana1	semana 2	semana 3	semana4	semana1	semana 2	semana 3	semana 4
actividad	fecha de entrega																
1.1 Análisis y diseño	24 de julio - 31 de julio																
Análisis del flujo de facturación con Factura.com.																	
Diseño del módulo de citas externas (flujo: cliente → especialista → servicio → horario → cuenta → pago → facturación).																	
1.2 Desarrollo de facturación	1 de agosto - 8 de agosto																
Creación de APIs propias para facturación.																	
Integración con API de Factura.com para alta de usuarios y generación de facturas.																	
Creacion del modulo para gestionar las facturas																	
Pruebas del modulo																	
1.3 Desarrollo del módulo de citas externas	11 de agosto - 25 de agosto																
Creación del módulo accesible vía URL.																	
Implementación de visualización de especialistas, servicios, precios y horarios.																	
Integración con Stripe para pagos en línea.																	
Creación de cuenta de usuario antes de confirmar la cita.																	
Adaptación de APIs para usarse sin login (acceso a información pública de especialistas y servicios).																	
Inclusión de opción de facturación en el flujo del módulo.																	
Pruebas del modulo																	
1.4 Pruebas y validación	26 de agosto - 29 de agosto																
Pruebas del flujo completo de citas (selección → pago → confirmación).																	
Pruebas de facturación con Factura.com y validación de almacenamiento de UUID en perfil de usuario.																	
1.5 Entrega	1 de septiembre - 8 de septiembre																
Documentación del módulo de citas.																	
Documentación técnica de la integración de facturación.																	
2.1 Análisis inicial	9 de septiembre - 13 de septiembre																
Identificación de módulos que se migrarán a la app (citas, corte del día, perfil).																	
Definición de roles de usuario y permisos (doctor, administrador, especialista).																	
Documentación de requerimientos funcionales y no funcionales.																	
2.2 Diseño de la aplicación	13 de septiembre - 15 de septiembre																
Definición de la arquitectura cliente-servidor (App ↔ API ↔ Base de datos).																	
Diagramas de flujo para procesos principales: agendar cita, reagendar, cancelar, corte del día, edición de perfil.																	
2.3 Análisis y creación de las APIs	16 de septiembre - 24 de septiembre																
Revisión de APIs existentes de la página web *Agenda Maya*.																	
Definición de APIs necesarias para la app móvil.																	
Creación y/o ajuste de APIs:																	
Autenticación (login).																	
Gestión de citas (crear, modificar, eliminar).																	
Corte del día (consultar ingresos del día, filtrados por especialista o global).																	
Gestión de usuarios/especialistas (ver y editar información).																	
Pruebas de funcionamiento y seguridad en las APIs.																	
2.4 Configuración del entorno de desarrollo	24 de septiembre - 26 de septiembre																
Configuración del repositorio (Git).																	
Preparación del entorno para desarrollo móvil.																	
Integración de la app con el servidor donde corren las APIs.																	
2.5 Desarrollo de los módulos de la app	27 de septiembre - 17 de octubre																
Login y autenticación: pantalla de inicio de sesión y validación de credenciales.																	
Gestión de citas:																	
Ver citas del día.																	
Seleccionar citas por semana o calendario.																	
Agendar cita (nuevo cliente o cliente existente).																	
Reagendar cita (solo cambiar fecha/hora).																	
Cancelar cita.																	
Pruebas del modulo de citas.																	
Perfil del especialista:																	
Visualización de datos (nombre, correo, teléfono, foto).																	
Edición de datos básicos y contraseña.																	
Menú de navegación: acceso rápido a Citas, Corte del día y Perfil.																	
Pruebas del modulo de perfil y menu de navegacion.																	
Corte del día:																	
Consulta de citas pagadas en el día.																	
Filtro por doctor o todos los doctores (según permisos).																	
Pruebas del modulo de corte del dia.																	
2.6 Pruebas y validaciones extra	20 de octubre - 24 de octubre																
Pruebas unitarias de cada módulo.																	
Pruebas de integración con las APIs.																	
Pruebas de usabilidad de la app en distintos dispositivos.																	
Corrección de errores y optimización.																	
2.7 Documentación y entrega	27 de octubre - 31 de octubre																
Documentación técnica de la app (estructura, módulos, APIs usadas).																	
Preparación de la versión final para despliegue																	

- Evaluación de seguimiento de asesor interno y externo. (2 Reportes)

I.T.S.P.R.	INSTITUTO TECNOLÓGICO SUPERIOR DE PURÍSIMA DEL RINCÓN FORMATO DE EVALUACIÓN Y SEGUIMIENTO DE RESIDENCIA PROFESIONAL NORMA ISO 9001:2015	Código: P017-07 No Rev: 0 Fecha: 14/03/17 Hoja: 1 de 2

ASESOR INSTITUCIÓN (INTERNO)

Nombre de Residente: VICTOR MANUEL CASTILLO OLIVETTO **Número de Control:** RS21110113

Nombre del Proyecto: MAIA App - Feature-Driven Development

Carrera: INFORMATICA **Periodo a Evaluar:** Agosto-diciembre 2025

Criterios a Evaluar		Valor	Evaluación
Evaluación del Asesor Interno	Asistió puntualmente a las reuniones de asesoría.	10	10
	Demuestra conocimiento en el área de su especialidad.	20	20
	Trabaja en equipo y se comunica de forma efectiva (oral y escrita).	15	15
	Es dedicado y proactivo en las actividades encomendadas.	20	20
	Es ordenado y cumple satisfactoriamente con las actividades encomendadas en los tiempos establecidos en el cronograma.	20	20
	Propone mejoras al proyecto.	15	15
	Calificación Total	**100**	**100**

Observaciones:

INSTITUTO TECNOLÓGICO SUPERIOR DE PURÍSIMA DEL RINCÓN SUBDIRECCIÓN ACADÉMICA **DOLORES AYALA MUÑOZ** **Asesor Interno**	**DIEGO EDUARDO MORALES LOPEZ** **Jefe de división**	**02/10/25** **Fecha de Evaluación**

	INSTITUTO TECNOLÓGICO SUPERIOR DE PURÍSIMA DEL RINCÓN	Código: P017-07
	FORMATO DE EVALUACIÓN Y SEGUIMIENTO DE RESIDENCIA PROFESIONAL	No Rev: 0 Fecha: 14/03/17
	NORMA ISO 9001:2015	Hoja: 2 de 2

ASESOR EMPRESA (EXTERNO)

Nombre de Residente: Victor Manuel Castillo Olivetto **Número de Control:** BS21110113

Carrera: Informatica **Empresa:** Corvuz

Nombre del Proyecto: Maia App - Feature-Driven Development

Periodo a Evaluar: Agosto - Diciembre

Criterios a Evaluar		Valor	Evaluación
Evaluación del Asesor Externo	Asiste puntualmente en el horario establecido	5	5
	Trabaja en equipo y se comunica de forma efectiva (oral y escrita)	10	5
	Tiene iniciativa para colaborar	5	5
	Propone mejoras al proyecto	10	5
	Cumple con los objetivos correspondientes al proyecto	15	15
	Es ordenado y cumple satisfactoriamente con las actividades encomendadas en los tiempos establecidos del cronograma	15	15
	Demuestra liderazgo en su actuar	10	8
	Demuestra conocimiento en el área de su especialidad	20	20
	Demuestra un comportamiento ético (es disciplinado, acata órdenes, respeta a sus compañeros de trabajo, entre otros)	10	10
	Calificación Total	**100**	88

Observaciones:

Héctor Javier R.F.	No contamos con sello	2/10/2025
Nombre y Firma del Asesor Externo	**Sello de la Empresa, Organismo o Dependencia**	**Fecha de Evaluación**

I.T.S.P.R.	INSTITUTO TECNOLÓGICO SUPERIOR DE PURÍSIMA DEL RINCÓN FORMATO DE EVALUACIÓN Y SEGUIMIENTO DE RESIDENCIA PROFESIONAL NORMA ISO 9001:2015	Código: P017-07 No Rev: 0 Fecha: 14/03/17 Hoja: 1 de 2

ASESOR INSTITUCIÓN (INTERNO)

Nombre de Residente: VICTOR MANUEL CASTILLO OLIVETTO **Número de Control:** RS21110113

Nombre del Proyecto: MAIA App - Feature-Driven Development

Carrera: INFORMATICA **Periodo a Evaluar:** agosto-diciembre 2025

Criterios a Evaluar		Valor	Evaluación
Evaluación del Asesor Interno	Asistió puntualmente a las reuniones de asesoría.	10	10
	Demuestra conocimiento en el área de su especialidad.	20	20
	Trabaja en equipo y se comunica de forma efectiva (oral y escrita).	15	15
	Es dedicado y proactivo en las actividades encomendadas.	20	20
	Es ordenado y cumple satisfactoriamente con las actividades encomendadas en los tiempos establecidos en el cronograma.	20	17
	Propone mejoras al proyecto.	15	15
	Calificación Total	**100**	**97**

Observaciones:

INSTITUTO TECNOLÓGICO SUPERIOR DE PURÍSIMA DEL RINCÓN
SUBDIRECCIÓN ACADÉMICA

DOLORES AYALA MUÑOZ **Asesor Interno**	**DIEGO EDUARDO MORALES LOPEZ** **Jefe de división**	**06 de noviembre de 2025** **Fecha de Evaluación**

	INSTITUTO TECNOLÓGICO SUPERIOR DE PURÍSIMA DEL RINCÓN	Código: P017-07
	FORMATO DE EVALUACIÓN Y SEGUIMIENTO DE RESIDENCIA PROFESIONAL	No Rev: 0 Fecha: 14/03/17
	NORMA ISO 9001:2015	Hoja: 1 de 2

ASESOR EMPRESA (EXTERNO)

Nombre de Residente: VICTOR MANUEL CASTILLO OLIVETTO **Número de Control:** RS21110113

Carrera: INFORMATICA **Empresa:** CORVUZ

Nombre del Proyecto: MAIA App - Feature-Driven Development

Periodo a Evaluar: Agosto- Diciembre 2025

	Criterios a Evaluar	Valor	Evaluación
Evaluación del Asesor Externo	Asiste puntualmente en el horario establecido	5	5
	Trabaja en equipo y se comunica de forma efectiva (oral y escrita)	10	8
	Tiene iniciativa para colaborar	5	5
	Propone mejoras al proyecto	10	8
	Cumple con los objetivos correspondientes al proyecto	15	12
	Es ordenado y cumple satisfactoriamente con las actividades encomendadas en los tiempos establecidos del cronograma	15	12
	Demuestra liderazgo en su actuar	10	8
	Demuestra conocimiento en el área de su especialidad	20	20
	Demuestra un comportamiento ético (es disciplinado, acata órdenes, respeta a sus compañeros de trabajo, entre otros)	10	10
	Calificación Total	**100**	**88**

Observaciones:

Héctor Javier R.F. Héctor Javier Ruiz Flores **Nombre y Firma del Asesor Externo**	No contamos con sello **Sello de la Empresa, Organismo o Dependencia**	**6 de Noviembre del 2025** **Fecha de Evaluación**

- Evaluación de reporte de residencia profesional.

I.T.S.P.R.	INSTITUTO TECNOLÓGICO SUPERIOR DE PURÍSIMA DEL RINCÓN FORMATO DE EVALUACIÓN Y SEGUIMIENTO DE RESIDENCIA PROFESIONAL NORMA ISO 9001:2015	Código: P017-08 No Rev: 0 Fecha: 14/03/17 Hoja: 1 de 2

ASESOR EMPRESA

Nombre de Residente: VICTOR MANUEL CASTILLO OLIVETTO **Número de Control:** RS21110113
Carrera: INFORMATICA **Empresa:** CORVUZ
Nombre del Proyecto: MAIA App - Feature-Driven Development - CORVUZ
Periodo a Evaluar: AGOSTO – DICIEMBRE

Criterios a Evaluar		Valor	Evaluación
EVALUACIÓN ASESOR EXTERNO (EMPRESA)	Portada	2	2
	Agradecimientos	2	2
	Resumen	2	2
	Índice	2	2
	Introducción	2	2
	Problemas a Resolver, priorizándolos.	5	5
	Objetivos	5	5
	Justificación	5	5
	Marco Teórico	10	10
	Procedimiento y Descripción de las Actividades Realizadas	5	5
	Resultados, planos, prototipos, manuales, análisis estadísticos...	40	40
	Conclusiones, Recomendaciones y Experiencia Profesional Adquirida	15	15
	Competencias Desarrolladas y/o Aplicadas	3	3
	Fuentes de Información	2	2
	Calificación Total	**100**	100

Observaciones:

Héctor Javier Rf.	No contamos con sello	28 de noviembre del 2025
Nombre y Firma del Asesor Externo	**Sello de la Empresa, Organismo o Dependencia**	**Fecha de Evaluación**

I.T.S.P.R.	INSTITUTO TECNOLÓGICO SUPERIOR DE PURÍSIMA DEL RINCÓN FORMATO DE EVALUACIÓN Y SEGUIMIENTO DE RESIDENCIA PROFESIONAL NORMA ISO 9001:2015	Código: P017-08 No Rev: 0 Fecha: 14/03/17 Hoja: 1 de 2

ASESOR INSTITUCIÓN (INTERNO)

Nombre de Residente: VICTOR MANUEL CASTILLO OLIVETTO **Número de Control:** RS21110113
Carrera: INFORMATICA **Nombre del Proyecto:** MAIA App - Feature-Driven Development - CORVUZ
Periodo a Evaluar: AGOSTO – DICIEMBRE

Criterios a Evaluar		Valor	Evaluación
EVALUACIÓN ASESOR EXTERNO (EMPRESA)	Portada	2	2
	Agradecimientos	2	1
	Resumen	2	2
	Índice	2	2
	Introducción	2	2
	Problemas a Resolver, priorizándolos.	5	5
	Objetivos	5	5
	Justificación	5	5
	Marco Teórico	10	9
	Procedimiento y Descripción de las Actividades Realizadas	5	4
	Resultados, planos, prototipos, manuales, análisis estadísticos...	40	35
	Conclusiones, Recomendaciones y Experiencia Profesional Adquirida	15	15
	Competencias Desarrolladas y/o Aplicadas	3	3
	Fuentes de Información	2	2
	Calificación Total	**100**	**92**

Observaciones:

Dolores Ayala Muñoz Nombre y Firma del Asesor Interno	Diego Eduardo Morales López Firma del Jefe de División	3 de diciembre del 2025 Fecha de Evaluación