UNIVERSIDAD LAICA "ELOY ALFARO" DE MANABÍ
EXTENSIÓN EN EL CARMEN
CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN
Creada Ley No. 10 – Registro Oficial 313 de Noviembre 13 de 1985

PROYECTO INTEGRADOR

PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERIA EN TECNOLOGÍAS DE LA INFORMACIÓN

TEMA

Aplicación Móvil con Sincronización inteligente para la gestión de reservas de los espacios físicos compartidos en la Uleam-El Carmen.

AUTOR
Rey Josias Miranda Llerena
TUTOR
ING. Wladimir Minaya, MG.

EL CARMEN, 2026

CERTIFICACIÓN DEL TUTOR

Uleam ELOY ALFARO DE MANABÍ	NOMBRE DEL DOCUMENTO: CERTIFICADO DE TUTOR(A).	CÓDIGO: PAT-04-F-004
	PROCEDIMIENTO: TITULACIÓN DE ESTUDIANTES DE GRADO BAJO LA UNIDAD DE INTEGRACIÓN CURRICULAR	REVISIÓN: 1
		Página 1 de 1

CERTIFICACIÓN

En calidad de docente tutor de la Extensión El Carmen de la Universidad Laica "Eloy Alfaro" de Manabí, CERTIFICO:

Haber dirigido, revisado y aprobado preliminarmente el Trabajo de Integración Curricular bajo la autoría del estudiante **MIRANDA LLERENA REY JOSIAS**, legalmente matriculado/a en la carrera de Ingeniería en Tecnología de la Información, período académico 2025(2)-2026(1), cumpliendo el total de 384 horas, cuyo tema del proyecto o núcleo problémico es "**APLICACIÓN MÓVIL CON SINCRONIZACIÓN OFFLINE -FIRST PARA LA GESTIÓN DE RESERVAS DE LOS ESPACIOS FÍSICOS COMPARTIDOS EN ULEAM EL CARMEN**".

La presente investigación ha sido desarrollada en apego al cumplimiento de los requisitos académicos exigidos por el Reglamento de Régimen Académico y en concordancia con los lineamientos internos de la opción de titulación en mención, reuniendo y cumpliendo con los méritos académicos, científicos y formales, y la originalidad del mismo, requisitos suficientes para ser sometida a la evaluación del tribunal de titulación que designe la autoridad competente.

Particular que certifico para los fines consiguientes, salvo disposición de Ley en contrario.

El Carmen, 27 de Julio de 2026.

Lo certifico,

Wladimir Minaya Macias, Mg.Sc.
Docente Tutor
Área: Tecnología de la Información

TRIBUNAL DE SUSTENTACIÓN

TRIBUNAL DE SUSTENTACIÓN

Universidad Laica "Eloy Alfaro de Manabí"
Extensión El Carmen
Carrera de Ingeniería en Tecnologías de la información

Título del trabajo de titulación:

Aplicación Móvil con Sincronización inteligente para la gestión de reservas de los espacios físicos compartidos en la Uleam-El Carmen.

Modalidad:
Proyecto Integrador

Autor:
Miranda Llerena Rey Josias

Tutor:
A.S Minaya Macias Renelmo Wladimir, Mg

Tribunal de Sustentación

- Presidente — Ing. Mora Marcillo Alex Bladimir
- Miembro — Ing. Arévalo Hermida Rómulo Danilo
- Miembro — Ing. Arce Zavala Jefferson Omar

Fecha de sustentación:

24 agosto del 2026

DECLARACIÓN DE AUTORÍA

UNIVERSIDAD LAICA "ELOY ALFARO" DE MANABÍ
EXTENSIÓN EN EL CARMEN

Uleam

DECLARACIÓN DE AUTORÍA

La responsabilidad del contenido de este Trabajo de titulación, cuyo tema es: Aplicación Móvil con Sincronización inteligente para la gestión de reservas de los espacios físicos compartidos en la Uleam-El Carmen. Corresponde exclusivamente a: Miranda Llerena Rey Josias y los derechos patrimoniales de la misma corresponden a la Universidad Laica "Eloy Alfaro" de Manabí.

Miranda Llerena Rey Josias
C.I. 1550109910

DEDICATORIA

Primeramente a Dios por haber sido mi guía y mi sustento en todo tiempo, y por permitirme cumplir este sueño que parecía imposible, y si hoy lo estoy logrando es por su ayuda y por nunca haberme dejado solo.

A mis queridos padres, Rocío y Rey, por guiarme desde pequeño con sus consejos y valores, que me han permitido llegar hasta aquí. A mi madre, por ser mi fortaleza y enseñarme a ser valiente, perseverante y a creer en mí incluso en los momentos de duda. A mi papá, por ser un hombre luchador y responsable, cuyo amor y ejemplo han sido una motivación constante durante esta etapa.

A mi querida abuela y a mi hermana, América y Débora, por ocupar un lugar especial en mi vida. A mi abuelita, por su cariño, compañía y por ser siempre una fuente de fortaleza y motivación. A Débora, por ser una de mis motivaciones para seguir adelante y por acompañarme con sus palabras de aliento en este camino.

A mi querida Mayerli, por compartir conmigo este camino y permanecer a mi lado en los momentos de frustración y tristeza. Por su paciencia, amor incondicional y por creer siempre en mí, siendo una parte importante de este proceso y de esta meta que hoy alcanzo.

Y por último, a mí mismo, por elegir seguir adelante ante las adversidades, por mi valentía y fortaleza, y por no rendirme hasta alcanzar esta meta. A quien fui, por luchar para llegar hasta aquí, y a quien soy hoy, por demostrarme que sí pude, que sí se puede y que, mientras crea en mí mismo, siempre podré.

Miranda Llerena Rey Josias

AGRADECIMIENTO

Agradezco a Dios por permitirme haber llegado hasta esta parte de mi carrera, por ser mi apoyo y ayudarme a permanecer frente a cualquier dificultad.

A mi Familia agradecimiento eterno, por siempre estar a mi lado y no dejar que baje los brazos, este logro es para ustedes, porque nunca me dejaron solo, por esa comprensión y amor que siempre me brindaron, por recordarme siempre que no debo rendirme.

A mis queridos amigos y compañeros de clases, por hacer de este camino una experiencia más llevadera con cada sonrisa, charla y momento compartido. A pesar de nuestras ocupaciones, siempre encontramos la manera de apoyarnos y estar presentes. A Melanie, Melvin y Angie, a quienes considero mis hermanos y parte de mi familia, gracias por estar siempre ahí, por sus consejos, motivación y por ser esas primeras manos amigas que encontré al iniciar este camino.

A mi querida Uleam, gracias por permitirme pasar los mejores 5 años de mi vida, gracias por cada aprendizaje en sus aulas, gracias por cada vivencia en sus instalaciones.

A mi tutor, Ing. Wladimir Minaya Macías, MG., y a cada ingeniero que formó parte de este camino, gracias por su paciencia, enseñanza y motivación durante todo este tiempo. Por compartir conmigo sus conocimientos, los cuales hoy se ven reflejados en este trabajo.

A mí mismo, porque nunca me rendí. A pesar de las madrugadas, el cansancio y las dificultades, tuve el valor de seguir adelante y no rendirme. Por cada esfuerzo, cada desvelo y cada momento en que decidí continuar luchando por alcanzar esta meta. Hoy me agradezco por perseverar y demostrarme que, con esfuerzo y determinación, era posible llegar hasta aquí.

Miranda Llerena Rey Josias

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE ANEXOS

RESUMEN

La gestión de los espacios físicos compartidos en la Universidad Laica "Eloy Alfaro" de Manabí, Extensión El Carmen (aulas, laboratorios, canchas y auditorio) se ha desarrollado tradicionalmente mediante procesos manuales, sin un mecanismo centralizado, lo que ha generado conflictos de horario, doble asignación de espacios y falta de disponibilidad en tiempo real. Ante esta problemática, la investigación tuvo como objetivo desarrollar una aplicación móvil con sincronización inteligente que automatice la consulta de disponibilidad, la solicitud y aprobación de reservas, y la notificación de su estado, con una arquitectura offline-first que permita su funcionamiento incluso sin conexión a internet.El estudio se enmarcó en una investigación aplicada de enfoque mixto, combinando una encuesta estructurada aplicada a 256 miembros de la comunidad universitaria (230 estudiantes y 24 docentes, además de 2 respuestas incidentales de personal administrativo) y entrevistas semiestructuradas al coordinador académico y a un docente con alta demanda de espacios. Los resultados evidenciaron que el 42.2% identificó la falta de información sobre disponibilidad como su principal inconveniente, y que el 95.7% se mostró dispuesto a utilizarla.El sistema se desarrolló bajo la metodología Scrum, en cuatro sprints, con Flutter y Dart para el cliente móvil, Node.js, Express y MySQL para el backend, desplegado en Railway, Socket.io y Firebase Cloud Messaging para las notificaciones, y SQLite para el almacenamiento local en modo sin conexión. Las pruebas de caja negra, caja blanca y usabilidad confirmaron el correcto funcionamiento de todos los módulos, validando la aplicación como una solución viable para la gestión de espacios físicos compartidos en la institución.

Palabras clave: aplicación móvil, sincronización offline-first, gestión de reservas, Flutter, notificaciones push.

ABSTRACT

The management of shared physical spaces at Universidad Laica "Eloy Alfaro" de Manabí, El Carmen Extension (classrooms, laboratories, sports courts, and the auditorium) has traditionally been carried out through manual processes, without a centralized mechanism, which has led to schedule conflicts, double booking of spaces, and a lack of real-time availability information. In response to this problem, the research aimed to develop a mobile application with intelligent synchronization that automates availability queries, booking requests and approvals, and status notifications, incorporating an offline-first architecture that allows it to function even without an internet connection. The study followed a mixed-methods applied research approach, combining a structured survey administered to 256 members of the university community (230 students and 24 professors, plus 2 incidental responses from administrative staff) and semi-structured interviews with the academic coordinator and a professor with high demand for spaces. The results showed that 42.2% identified the lack of availability information as their main concern with the current process, and that 95.7% expressed willingness to use a mobile application. The system was developed under the Scrum methodology, across four sprints, using Flutter and Dart for the mobile client, Node.js, Express, and MySQL for the backend deployed on Railway, Socket.io and Firebase Cloud Messaging for notifications, and SQLite for local offline storage. Black-box, white-box, and usability testing confirmed the correct functioning of all modules, validating the application as a viable solution for the management of shared physical spaces at the institution.

Keywords: mobile application, offline-first synchronization, booking management, Flutter, push notifications.

CAPÍTULO I

1 INTRODUCCIÓN

1.1 Introducción

El presente trabajo busca desarrollar una aplicación móvil que facilite y mejore la gestión de reservas de los espacios físicos compartidos de la extensión El Carmen de la Universidad Laica "Eloy Alfaro" de Manabí (ULEAM-El Carmen). La idea nace de una necesidad real que se ha observado: actualmente el proceso de consultar disponibilidad, reservar y controlar el uso de aulas, laboratorios y otros espacios es bastante tedioso y poco eficiente. Con esta app se busca simplificar todo eso aprovechando lo práctico que resultan los celulares hoy en día.

En Ecuador la mayor parte de las universidades han implementado instrumentos digitales para manejar tareas administrativas y académicas. No obstante, todavía es raro hallar soluciones móviles que incorporen una arquitectura offline-first y sincronización bidireccional, sobre todo si se enfocan en la administración de espacios físicos. En otras naciones, este tipo de aplicaciones han resultado ser una herramienta efectiva, pues optimizan la eficiencia operacional y brindan al usuario una experiencia más ágil y fiable.

En la ULEAM-El Carmen, en particular, hoy en día hay restricciones para acceder a los datos acerca de la disponibilidad de aulas, laboratorios y otros espacios desde dispositivos móviles. Por este motivo, la propuesta de este proyecto es una aplicación móvil incorpora una arquitectura offline-first que permite consultar espacios disponibles y reservas previamente almacenadas en el dispositivo cuando no existe conexión a Internet, actualizando la información local al recuperar la conectividad. El propósito no es solo optimizar los procesos internos, sino también consolidar la

administración universitaria y fomentar un uso más eficiente de la tecnología dentro de la institución.

1.2 Presentación del tema

Aplicación móvil con sincronización inteligente para la gestión de reservas de los espacios físicos compartidos en la Uleam-El Carmen.

1.3 Ubicación y contextualización de la problemática

La Universidad Laica "Eloy Alfaro" de Manabí, Extensión en El Carmen, está ubicada en la provincia de Manabí, específicamente en la Avenida 3 de Julio, en el cantón El Carmen. Esta institución brinda capacitación presencial en varias carreras y necesita administrar bien sus espacios físicos compartidos para las tareas académicas, deportivas, culturales y administrativas.

La institución cuenta con varios espacios compartidos: aulas, laboratorios, auditorio, canchas deportivas, biblioteca y salas de reuniones, usados por estudiantes, docentes, personal administrativo y visitantes.

El campus cuenta con dos niveles: en el nivel inferior se encuentran las aulas que emplean las carreras de Administración de Empresas y Contabilidad durante la mañana y la tarde, mientras que Fisioterapia y Enfermería las ocupan por la noche. También hay espacios temporales para Electromecánica. En la planta alta, hay tres laboratorios de computación que están diseñados, para las carreras de Desarrollo de Software y Tecnologías de la Información, además de un aula que se comparte con la carrera de Enfermería.

Actualmente está gestión se hace mediante procesos manuales y sistemas de escritorio, lo que limita el acceso a la información desde dispositivos móviles. Esto genera inconvenientes para quienes necesitan consultar disponibilidad, reservar o recibir notificaciones estando lejos del

campus universitario. La ausencia de solución móvil eficaz está provocando conflictos de horario frecuentes, uso inadecuado de los espacios compartidos y una gestión de recursos deficiente. Implementar una aplicación con sincronización inteligente y diseño offline-first permitiría: mejorar notablemente el acceso a la información en tiempo real, gestión reservas desde cualquier lugar y garantizar que el sistema siga operativo incluso en entornos con conectividad limitada o intermitente.

1.4 Planteamiento del problema

1.4.1 Problematización

Gestión deficiente de espacios físicos compartidos debido a limitaciones de acceso móvil y dependencia de conectividad constante.

El principal inconveniente es la falta de una aplicación móvil, que facilite a los usuarios de la ULEAM-El Carmen la gestión eficaz de reservas de espacios compartidos desde sus dispositivos. El acceso a información esencial y la posibilidad de reservar estando fuera del campus se ven seriamente restringidos por la dependencia presente de sistemas de escritorio y procedimientos manuales.

1.4.2 Génesis del problema

En la ULEAM-El Carmen, el problema vinculado a cómo administrar espacios compartidos de manera móvil proviene del historial en la creación de sistemas de información institucionales que se enfocaron en plataformas de escritorio sin tener en cuenta las nuevas tendencias en móvil y conectividad ubicua.

El aumento en la cantidad de teléfonos inteligentes, el avance en la cobertura de datos móviles y la transformación en los patrones de uso tecnológico de alumnos y profesores han

generado una diferencia importante entre lo que los usuarios esperan y lo que los sistemas institucionales actuales pueden ofrecer. Los usuarios actuales desean tener la posibilidad de manejar casi todos los aspectos de su vida académica desde sus dispositivos móviles, tal como hacen con las compras, las operaciones bancarias y las comunicaciones.

El crecimiento gradual de la matrícula estudiantil y el incremento en la variedad de programas académicos, sin que se haya desarrollado a la par una infraestructura tecnológica móvil adecuada, han empeorado las cosas. La falta de los métodos convencionales basados en sistemas de escritorio y presencialidad ha quedado al descubierto debido a la ampliación del número de usuarios potenciales del sistema y a la creciente dificultad para coordinar espacios.

Además, eventos recientes, como las interrupciones en la conectividad o climáticas de conectividad, han demostrado la fragilidad de los sistemas que dependen por completo de una conexión permanente a Internet y carecen de capacidades offline para asegurar un funcionamiento básico ininterrumpido.

1.4.3 Estado actual del problema

Actualmente, la administración de espacios compartidos en la extensión El Carmen de la ULEAM se maneja mediante una combinación poco eficiente de sistemas de escritorio, intercambio constante de correos, coordinación por teléfono y hojas de cálculo colaborativas. Este conjunto de herramientas está disperso y no tiene ninguna opción móvil que ofrezca acceso unificado desde cualquier dispositivo.

Como resultado, cualquier persona que necesite verificar disponibilidad o reservar un espacio debe desplazarse hasta una computadora fija, generalmente ubicada en los laboratorios de informática o en las áreas administrativas, lo que genera demoras y limitaciones importantes.

1.5 Diagrama causa – efecto del problema

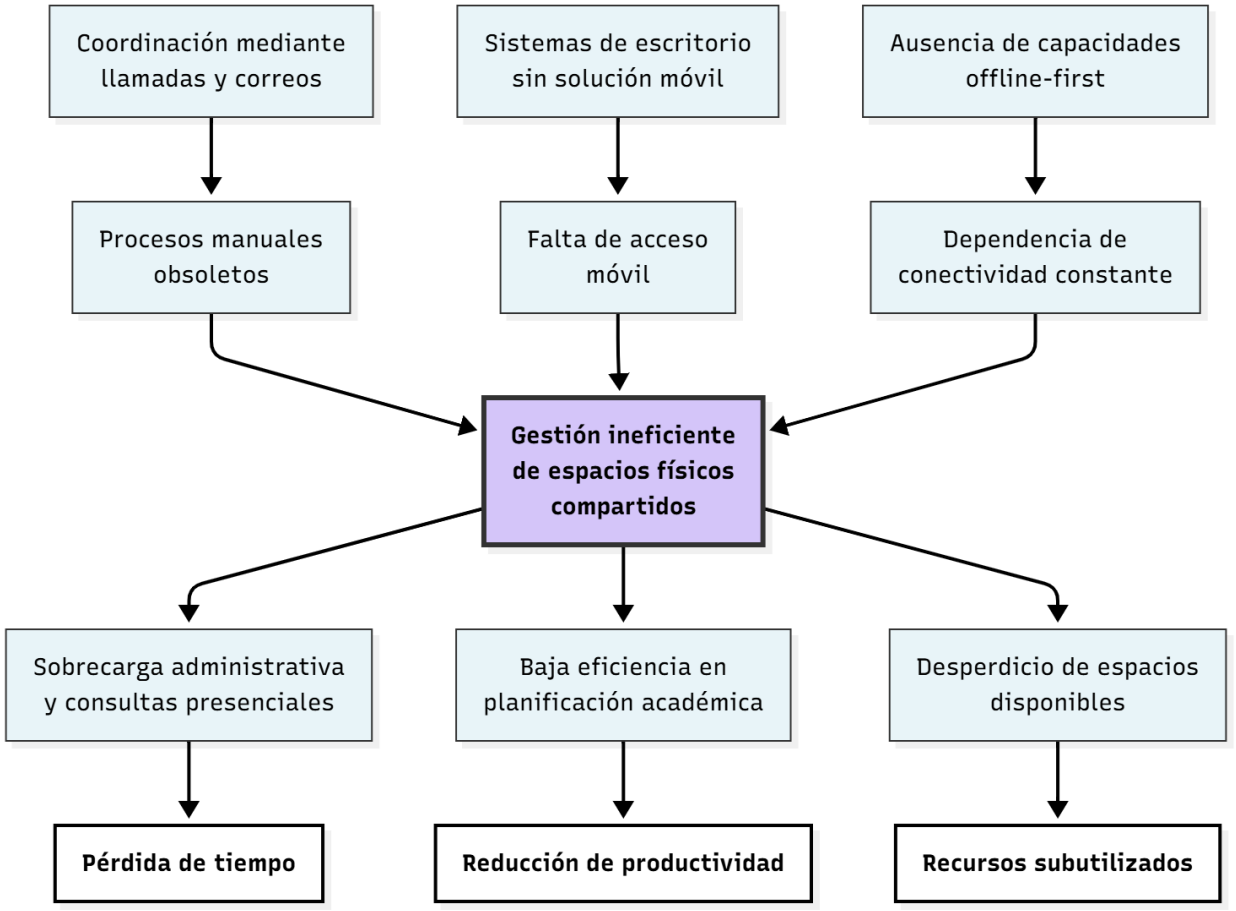


Figura 1 Árbol de problema

1.6 Objetivos

1.6.1 Objetivo general

Desarrollar una Aplicación Móvil con Sincronización inteligente para la gestión de reservas de los espacios físicos compartidos en la Uleam-El Carmen

1.6.2 Objetivos específicos

- Analizar los fundamentos teóricos de las aplicaciones móviles nativas y los sistemas de sincronización offline-first para establecer la base conceptual del proyecto.

- Evaluar la situación presente de la administración de espacios físicos compartidos en la ULEAM-El Carmen para determinar requerimientos puntuales de conectividad y movilidad.
- Diseñar la arquitectura del sistema móvil con capacidades de sincronización inteligente y notificaciones push multiplataforma.
- Implementar y comprobar el funcionamiento de la aplicación móvil utilizando frameworks de desarrollo nativo con base de datos local y sincronización bidireccional, evaluando su desempeño en la administración móvil de espacios compartidos, rendimiento offline y experiencia de usuario.

1.7 Justificación

Este proyecto es necesario para modernizar la administracion de los espacios comunes en la ULEAM-El Carmen, con una solución movil que se ajuste a las expectativas tecnológicas actuales del alumnado universitario. Desplegar una aplicación con sincronización inteligente responde a las tendencias mundiales en acceso universal a la información, movilidad y experiencia de usuario enfocada en dispositivos móviles.

1.7.1.1 Beneficiarios directos:

Personal administrativo: Su carga laboral asociada a las consultas en persona y a la coordinación telefónica disminuirá notablemente. Si se automatizan los procesos de reserva y consulta a través de la app móvil, se tendrá más tiempo disponible para tareas con mayor valor añadido y para brindar atención individualizada a situaciones complejas.

Docentes: Podrán verificar la disponibilidad, reservar y recibir notificaciones desde sus celulares, sin ir a las oficinas administrativas ni interrumpir sus actividades académicas. Su planificación y eficiencia mejorarán al poder gestionar reservas sobre la marcha.

Estudiantes: Podrán consultar la información de espacios desde cualquier lugar, en tiempo real, lo que facilita la planificación de proyectos y actividades académicas. Las notificaciones push les avisaran de inmediato sobre confirmaciones o cambios.

Autoridades institucionales: podrán mostrar su compromiso con la innovación tecnológica y con mejorar la experiencia para los usuarios, lo que ayudará a posicionar a la universidad como una institución moderna y atenta a las demandas de su comunidada.

1.8 Impactos esperados

1.8.1.1 Impacto tecnológico

La puesta en marcha de la aplicación móvil con sincronización inteligente colocará a la ULEAM-El Carmen como una institución líder en la adopción de tecnologías móviles para la gestión administrativa. El proyecto incorpora al ecosistema tecnológico de la universidad elementos como arquitecturas offline-first, notificaciones push multiplataforma, sincronización bidireccional y principios actuales de desarrollo móvil.

Se prevé que la aplicación disminuya en más de un 80% el tiempo dedicado a los trámites de reserva y consulta de espacios, convirtiendo tareas que hoy tardan entre 15 y 30 minutos en procesos que se completan en menos de dos minutos desde el dispositivo móvil del usuario.

1.8.1.2 Impacto social

La mejora notable de la experiencia y la satisfacción de la totalidad de la comunidad universitaria será el reflejo del impacto social del proyecto. Al gestionar reservas desde el celular,

en cualquier momento y lugar, se reduce la frustración y el estrés que están asociados los procedimientos actuales.

Los docentes tendrán más autonomía y flexibilidad en su planeación académica, ya que podrán revisar o cambiar reservas mientras se encuentran dentro o fuera del campus. Esto se traduce en un uso más eficiente del tiempo y en una mayor concentración en tareas académicas de calidad.

Al quitar las limitaciones de tiempo y ubicación para manejar reservas, se benefician especialmente los estudiantes con responsabilidades familiares o laborales. El acceso se democratiza con la posibilidad de hacer solicitudes fuera del horario laboral, por las noches, tardes o fines de semana.

1.8.1.3 Impacto ambiental

Desde el punto de vista medioambiental, la aplicación móvil ayudará a disminuir notablemente el uso de papel al suprimir los documentos impresos, las anotaciones en papel y las comunicaciones en papel que se emplean hoy en día para la administración de espacios. Se prevé que se reduzcan alrededor de 10,000 hojas de papel al año, lo que equivale a la economía de medio árbol y a una disminución en las emisiones de CO_2 relacionadas con su producción y transporte.

La optimización de la distribución de espacios tendrá un impacto positivo indirecto en la eficiencia energética de las instituciones. El sistema, al hacer posible la visualización en tiempo real de la disponibilidad, fomentará que las actividades se consoliden en espacios adecuados a su tamaño y evitará el empleo innecesario de auditorios grandes para grupos pequeños que podrían operar en aulas ordinarias. La optimización mencionada disminuirá el consumo de climatización e iluminación.

Disminuirá la huella de carbono vinculada al transporte si se reduce el traslado innecesario de los usuarios y del personal administrativo a las oficinas para realizar verificaciones de información o consultas presenciales. Los usuarios podrán aclarar consultas desde su ubicación actual sin tener que desplazarse específicamente para ello.

CAPÍTULO II

2 Marco Teórico

2.1 Antecedentes históricos

2.1.1 Aplicaciones móviles

El desarrollo de las aplicaciones móviles tiene sus orígenes a principios de los años 2000, cuando los dispositivos portátiles comenzaron a incorporar sistemas operativos capaces de ejecutar programas de terceros. Según Humberto y Casariego (2026), las aplicaciones móviles se han consolidado como herramientas esenciales para la gestión del aprendizaje y la administración institucional en la educación superior latinoamericana, ya que popularizaron el concepto de acceso digital inmediato en entornos híbridos, permitiendo a desarrolladores independientes y organizaciones crear soluciones personalizadas para millones de usuarios en todo el mundo.

Posteriormente, con el lanzamiento del sistema operativo Android por parte de Google en 2008, el ecosistema móvil se expandió considerablemente. Como señalan Pascuas-Rengifo. et al. (2020), la integración de tecnologías de la información y la comunicación mediante dispositivos móviles ha favorecido la innovación de procesos educativos, permitiendo que las aplicaciones evolucionen desde funciones simples hacia servicios con mayor interacción, conectividad y capacidad de procesamiento.

En el contexto sudamericano, Guale (2024), destaca que la adopción de aplicaciones móviles en instituciones educativas ha crecido significativamente en los últimos años, impulsada por el aumento en el uso de smartphones entre los estudiantes y el acceso extendido a datos móviles. Los autores documentan que las instituciones que implementaron soluciones móviles para gestión académica reportan mejoras sustanciales en eficiencia operativa y satisfacción de usuarios.

2.1.2 Sincronización de datos y arquitecturas offline-first

La necesidad de mantener datos actualizados en entornos con conectividad intermitente dio origen al concepto de sincronización bidireccional en aplicaciones móviles. Según Quisaguano Collaguazo et al., (2022) al analizar el desarrollo híbrido con Flutter, describen el enfoque offline-first como un cambio de paradigma en el diseño de aplicaciones móviles: en vez de depender solamente de una conexión activa a Internet, estas aplicaciones guardan los datos localmente y se sincronizan automáticamente cuando se restablece la conectividad. Esto resulta particularmente relevante en entornos educativos con cobertura intermitente, donde la continuidad operativa es esencial para la experiencia del usuario.

Azketa et al., (2021) analiza cómo las tecnologías de sincronización en sistemas distribuidos incorporan algoritmos de resolución de conflictos, replicación incremental y gestión de versiones, lo que permite mantener la consistencia de datos en entornos multiusuario sin necesidad de requerir una conexión permanente.

2.1.3 Gestión de espacios compartidos en instituciones educativas

Históricamente, la gestión de espacios físicos compartidos en universidades se llevó a cabo mediante registros manuales, pizarras de clasificación y comunicación directa entre responsables administrativos. Como indican Ruiz Muñoz & Santos Tomalá, (2024), el crecimiento exponencial de la matrícula universitaria y la diversificación de actividades académicas generaron la necesidad imperante de sistemas digitales que optimicen el uso de la infraestructura física. Los hallazgos muestran que la adopción de sistemas automatizados para la gestión de espacios contribuyo de manera significativa a mejorar la eficiencia operativa y la satisfacción de la comunidad universitaria.

Engel y Coll (2022) analizan, en ese mismo contexto, la transición de los entornos compartidos en universidades desde modelos fijos hacia configuraciones híbridas. Destacan que usar enfoques y tecnologías digitales para difuminar las fronteras entre lo virtual y lo presencial permite optimizar y personalizar el aprovechamiento de los recursos disponibles. Su estudio subraya la importancia de implementar sistemas de gestión que integren calendarización, notificaciones y roles de usuario, para evitar solapamientos y garantizar un uso equitativo de los recursos.

2.2 Antecedentes de investigaciones relacionadas al tema presentado

La revisión de la literatura científica y técnica permite ver una tendencia creciente en el desarrollo de soluciones digitales orientadas a la gestión de espacios físicos en los entornos educativos. A continuación se presenta una sistematización de las investigaciones más relevantes, organizadas según su alcance, metodología y contribución al estado del conocimiento en el área.

2.2.1.1 Investigaciones sobre sistemas web de reservas de espacios

El trabajo de Omar et al. (2021), titulado "Implementación de una Aplicación Web para la Gestión de Reservas y de Espacios para la Dirección Técnica de Administración e Inventarios de la Universidad Politécnica Salesiana, Sede Guayaquil", implementó un sistema web para automatizar la reserva de espacios compartidos en un entorno universitario ecuatoriano. El autor uso la metodología en cascada, con fases de análisis de requisitos, diseño UML y pruebas unitarias, y lograron como resultado principal una disminución del 88% en errores y cruces de horarios según las pruebas realizadas. Esta investigación demuestra que las soluciones digitales para la gestión de espacios son viables, aunque se limita a entornos de escritorio sin considerar el acceso móvil.

Por su parte, Culqui Escobar (2015) desarrolló un sistema web para el registro de reservaciones y control del hospedaje en un hotel de Ambato. Aunque el contexto es distinto al

educativo, pero aporta un modelo pionero para sistemas de reservas distribuidas: se utilizaron tecnologías como PHP para el backend, MySQL para la base de datos y HTML/CSS/JavaScript para interfaces responsivas, integrando un módulo de calendario para ver disponibilidades. Los resultados mostraron una reducción del 65% en tiempos de procesamiento manual, mejorando la eficiencia operativa y satisfacción de usuarios.

2.2.1.2 Investigaciones sobre gestión de laboratorios universitarios

El trabajo de titulación de Figueroa Quimis (2022) titulado "Sistema web para la gestión y reserva de equipos para los laboratorios de computación de la carrera de tecnologías de la Información de la Universidad Estatal del Sur de Manabí", desarrolló una plataforma web para automatizar las reservas de equipos en laboratorios universitarios. Empleó metodología en cascada y tecnologías como PHP con Laravel para el backend, MySQL para la base de datos y JavaScript con Bootstrap para interfaces responsivas. Los resultados mostraron una reducción del 75% en tiempos de procesamiento y una mejora significativa en la satisfacción de estudiantes y docentes, aunque la solución implementada no contempla acceso móvil ni sincronización offline.

Posteriormente, Emiro Francisco (2023) desarrolló una aplicación web para la automatización del proceso de administración y control en los laboratorios de informática de la Facultad de Ingeniería en Ciencias Aplicadas de la Universidad Técnica del Norte. Las características clave de este estudio son el uso de la metodología Scrum, con un ciclo de iteración de cuatro semanas, y la aplicación de tecnologías más modernas: Angular para el front-end, Java y Spring Boot para el backend, y PostgreSQL para la base de datos. Una evaluación de usabilidad realizada con SUS arrojó una puntuación de 85/100, lo que indica una buena aceptabilidad. Esta investigación representa un avance significativo en la integración de metodologías ágiles, pero no aborda la necesidad de acceso sin conexión en dispositivos móviles.

2.2.1.3 Investigaciones sobre sincronización y movilidad en aplicaciones educativas

En el ámbito de las aplicaciones móviles con capacidades de sincronización, Cabezuelo (2023) explora la gestión de espacios compartidos en entornos en una biblioteca universitaria mediante una aplicación móvil basada en geolocalización, donde el sistema y el control de acceso reducen conflictos de horarios en un 50% en instituciones con alta movilidad estudiantil. El autor propone un marco metodológico para entornos universitarios que optimiza el uso de los espacios y permite al usuario planificar su visita, alineándose con la necesidad de priorizar reservas según el tipo de usuario y el nivel de demanda.

Finalmente, Azketa et al., (2021) proporcionan los fundamentos técnicos de los sistemas distribuidos que sustentan las arquitecturas de sincronización modernas. Sus planteamientos sobre replicación de datos, consistencia y tolerancia a fallos son directamente aplicables al diseño de aplicaciones móviles que deben operar en entornos de conectividad variable, constituyendo una base esencial para el desarrollo del presente proyecto.

2.2.1.4 Síntesis y oportunidad de innovación

En este sentido, la propuesta se diferencia porque busca atender simultáneamente la movilidad de los usuarios, la disponibilidad intermitente de la conexión a internet y la necesidad de mantener informados a los participantes sobre el estado de sus solicitudes. La arquitectura offline-first permite que la aplicación consulte información almacenada localmente y sincronice los cambios cuando se restablezca la conectividad, mientras que las notificaciones push complementan el proceso de comunicación entre el sistema y los usuarios (Quisaguano Collaguazo et al., 2022; Azketa et al., 2021).

2.3 Definiciones conceptuales

2.3.1 Variable independiente: Aplicación móvil con sincronización inteligente

2.3.1.1 Aplicación móvil

Una aplicación móvil es un programa de software diseñado específicamente para ejecutarse en dispositivos portátiles como teléfonos inteligentes y tablets. Según Humberto y Casariego (2026), las aplicaciones móviles modernas se caracterizan por su interfaz de usuario adaptada a pantallas pequeñas, su capacidad de aprovechamiento de sensores integrados del dispositivo y su distribución mediante plataformas comerciales. En el contexto institucional, estas aplicaciones sirven como punto de acceso principal para interactuar con sistemas backend mediante APIs RESTful, permitiendo que los usuarios realicen operaciones complejas desde cualquier ubicación.

2.3.1.2 Arquitectura offline-first

La arquitectura offline-first es un paradigma de diseño de aplicaciones que prioriza el funcionamiento sin conexión a Internet como estado por defecto. Como describen Quisaguano Collaguazo et al. (2022) en su análisis del desarrollo híbrido con Flutter, las aplicaciones modernas almacenan todos los datos necesarios en el dispositivo local y operan de manera completa sin requerir conectividad. Cuando se restablece la conexión, el sistema sincroniza automáticamente los cambios realizados en modo desconectado con el servidor central, resolviendo posibles conflictos mediante algoritmos determinísticos.

2.3.1.2.1 Principios fundamentales

La arquitectura offline-first se fundamenta en varios principios que permiten conservar la continuidad operativa de una aplicación cuando la conexión a internet es limitada o inexistente. El primer principio es el almacenamiento local, mediante el cual los datos necesarios para las

funciones principales se conservan en el dispositivo del usuario. El segundo es la sincronización automática, que permite transferir al servidor los cambios realizados localmente cuando se recupera la conexión. El tercero es la resolución de conflictos, necesaria cuando dos usuarios modifican un mismo registro durante períodos de desconexión. Finalmente, se encuentra el funcionamiento continuo, cuyo propósito es evitar que la experiencia del usuario dependa completamente de la disponibilidad de la red (Kleppmann, 2017; Azketa et al., 2021).

2.3.1.3 Sincronización bidireccional

La sincronización bidireccional permite el intercambio de datos entre el dispositivo móvil y el servidor central en ambas direcciones. Según Azketa et al. (2021), este mecanismo utiliza algoritmos de replicación incremental que identifican los cambios realizados desde la última sincronización y transfieren únicamente los datos modificados, optimizando el consumo de ancho de banda y la velocidad de actualización. En el contexto de la gestión de espacios, esto implica que las reservas realizadas offline se transmiten al servidor cuando se recupera la conexión, y los cambios efectuados por otros usuarios en el servidor se descargan automáticamente al dispositivo.

2.3.1.4 Notificaciones push

Las notificaciones push son mensajes enviados desde un servidor hacia un dispositivo conectado, sin que el usuario tenga que iniciar una solicitud en ese momento. Estos mensajes pueden incluir información visible para el usuario o datos que la aplicación procesa internamente. Firebase Cloud Messaging permite enviar mensajes a dispositivos Android, iOS y plataformas web mediante una conexión administrada por el servicio de Firebase (Google, 2026).

En una aplicación móvil, el comportamiento de una notificación depende del estado de la aplicación. Cuando esta se encuentra en segundo plano, el SDK puede mostrar automáticamente

una notificación visible; cuando está en primer plano, la aplicación debe definir cómo procesar y presentar el mensaje recibido (Google, 2026). Por ello, las notificaciones push constituyen un mecanismo adecuado para comunicar confirmaciones, rechazos, cancelaciones y cambios relacionados con las reservas.

2.3.1.5 Tecnologías de desarrollo móvil

2.3.1.5.1 Flutter

Flutter es un framework de desarrollo de aplicaciones móviles multiplataforma creado por Google, lanzado oficialmente en 2018. A diferencia de otros frameworks híbridos, Flutter no utiliza componentes nativos del sistema operativo, sino que renderiza su propia interfaz gráfica mediante el motor de renderizado Skia (y más recientemente Impeller), lo que le permite lograr un rendimiento visual consistente y fluido de 60 o 120 fps en cualquier dispositivo. Según Quisaguano Collaguazo et al. (2022), Flutter garantiza una experiencia de usuario homogénea e independiente de la versión del sistema operativo, ya que esta permite diseñar y codificar aplicativos móviles con un único código base, eliminando inconsistencias visuales que suelen presentarse en otros enfoques multiplataforma.

Para este proyecto, Flutter representa la elección tecnológica más adecuada por varias razones como lo son: su lenguaje de programación nativo Dart ofrece tipado estático y compilación ahead-of-time (AOT), lo que resulta en un rendimiento superior al de frameworks basados en JavaScript como React Native (Google, 2026). Además, Flutter tiene soporte oficial de Google para integrarse con Firebase Cloud Messaging, cuenta con el paquete sqflite -muy usado para implementar bases de datos SQLite locales con arquitectura offline-first-, y tiene una comunidad activa y en crecimiento.

2.3.1.5.2 Dart

Dart es el lenguaje de programación desarrollado por Google sobre el cual está construido Flutter. Según Pascuas-Rengifo et al. (2020), los lenguajes de programación modernos orientados a objetos con tipado estático son la base para el desarrollo de aplicaciones educativas e institucionales confiables; Dart incorpora estas características junto con programación asíncrona nativa mediante async/await, null safety y un sistema de tipos robusto que detecta errores en tiempo de compilación. Estas características lo hacen una opción sólida para aplicaciones empresariales que requieren de alta confiablilidad y mantenibilidad.

En el sistema propuesto, Dart facilita implementar la lógica de sincronización bidireccional y resolución de conflictos, mediante su soporte nativo para programación reactiva y manejo de estados (Google, 2026). Su compilación AOT produce código de maquia optimizado que corre directamente en el dispositivo, sin la sobrecarga de un puente JavaScript como el que usan otros frameworks, resultando en tiempos de respuesta menores y menor consumo de bateria durante las operaciones de sincronización offline-first.

2.3.1.6 Base de datos local

2.3.1.6.1 SQLite

SQLite es un motor de base de datos relacional ligero que opera como una biblioteca integrada en la aplicación, sin requerir un servidor externo. Según Quisaguano Collaguazo et al. (2022), SQLite es ampliamente utilizado en el desarrollo de aplicaciones móviles híbridas debido a su bajo consumo de recursos, compatibilidad con las plataformas principales y soporte completo para transacciones ACID. En el contexto del sistema propuesto, SQLite almacena localmente los datos de espacios, reservas y usuarios a través del paquete sqflite de Flutter, permitiendo el funcionamiento completo de la aplicación en modo offline.

2.3.1.7 Tecnologías backend y comunicación

2.3.1.7.1 Node.js y Express

Node.js es un entorno de ejecución de JavaScript del lado del servidor que utiliza un modelo de entrada/salida no bloqueante, permitiendo manejar múltiples solicitudes concurrentes de manera eficiente. Express es un framework minimalista construido sobre Node.js que facilita la creación de APIs RESTful mediante un sistema de rutas y middleware. Según Pressman et al., (2021), esta combinación es particularmente adecuada para aplicaciones que requieren alta escalabilidad y comunicación en tiempo real con dispositivos móviles.

2.3.1.7.2 Firebase Cloud Messaging (FCM)

Firebase Cloud Messaging es un servicio desarrollado por Google que permite enviar notificaciones push a dispositivos Android e iOS de manera confiable. Como indica Humberto y Casariego (2026) al analizar el uso de herramientas digitales en la educación superior, la entrega de mensajes instantáneos a través de plataformas nativas garantiza que los usuarios reciban alertas incluso cuando la aplicación no está activa en segundo plano. Este servicio es compatible de forma nativa con Flutter mediante el paquete oficial firebase_messaging, lo que simplifica su integración en el sistema de notificaciones en tiempo real del proyecto.

2.3.1.7.3 MySQL

MySQL es un sistema de gestión de bases de datos relacional (RDBMS, por sus siglas en inglés) de código abierto que utiliza el lenguaje de consulta estructurado (SQL) para almacenar, gestionar y manipular datos. El sistema organiza la información en tablas con filas y columnas, siguiendo el modelo relacional, y ofrece cumplimiento ACID (atomicidad, consistencia, aislamiento y durabilidad) para garantizar la integridad de las transacciones (Oracle, 2026).

Entre sus características principales se incluyen alto rendimiento en la recuperación y procesamiento de datos, escalabilidad para manejar desde pequeñas hasta grandes bases de datos, soporte para múltiples motores de almacenamiento (como InnoDB y MyISAM), replicación maestro-esclavo para redundancia y disponibilidad, y compatibilidad con múltiples plataformas (Linux, Windows, macOS) (Oracle, 2026).

2.3.1.8 Seguridad y autenticación

2.3.1.8.1 JSON Web Tokens (JWT)

JWT es un estándar para la creación de tokens de autenticación que permite transmitir información de forma segura entre dos partes. Según Pressman et al. (2021), los tokens JWT contienen las credenciales del usuario en formato cifrado y pueden ser verificados por el servidor sin necesidad de mantener sesiones activas, lo que facilita la implementación de autenticación sin estado en aplicaciones móviles.

2.3.1.8.2 Bcrypt

Bcrypt es un algoritmo adaptativo de derivación de claves diseñado para almacenar contraseñas de forma más resistente frente a ataques de fuerza bruta. Su funcionamiento incorpora un factor de trabajo que puede ajustarse para aumentar el costo computacional requerido durante la generación y verificación del hash (OWASP Foundation, 2026). A diferencia del cifrado reversible, el hash de una contraseña no está diseñado para recuperarse como texto original. Durante el inicio de sesión, la contraseña introducida por el usuario se compara con el hash almacenado mediante una función de verificación.

2.3.2 Variable dependiente: Gestión de reservas de espacios físicos compartidos

2.3.2.1 Gestión de espacios compartidos

La gestión de espacios compartidos engloba el conjunto de procesos, políticas y herramientas utilizadas para administrar el uso de recursos físicos comunes dentro de una institución. Según Ruiz Muñoz y Santos Tomalá (2024), una gestión efectiva de espacios se mide mediante indicadores de desempeño tales como la tasa de ocupación, el tiempo de respuesta ante solicitudes y el porcentaje de conflictos horarios. En el contexto de la ULEAM-El Carmen, estos espacios incluyen aulas, laboratorios de computación, el auditorio y las canchas deportivas.

2.3.2.2 Proceso de reserva digital

El proceso de reserva digital comprende una secuencia de etapas que incluyen la consulta de disponibilidad, la solicitud de reserva, la validación automática y la confirmación. Como indica Omar et al. (2021), el sistema debe garantizar que una vez aprobada una reserva, el espacio quede bloqueado para otros usuarios durante el intervalo horario correspondiente. Las reglas operativas típicamente incluyen restricciones sobre la duración máxima de reserva, priorización según el rol del usuario y validación de capacidad según el número de personas registradas.

2.3.2.3 Actores y responsabilidades en la gestión de reservas

En los sistemas de reservas intervienen actores con responsabilidades distintas. Los usuarios solicitantes consultan disponibilidad y generan solicitudes, mientras que los responsables administrativos aplican criterios de validación, autorización y control. La diferenciación de responsabilidades ayuda a organizar el proceso y a reducir ambigüedades sobre quién puede solicitar, aprobar o modificar el uso de un recurso compartido (Omar., 2021; Engel y Coll, 2022).

2.3.2.4 Notificaciones automáticas

Las notificaciones automáticas forman parte de los mecanismos de comunicación de un sistema de reservas, ya que permiten informar sobre solicitudes, aprobaciones, rechazos, cancelaciones o cambios de disponibilidad. Su utilidad radica en reducir la dependencia de comunicaciones manuales y ofrecer al usuario información oportuna sobre el estado de una gestión. En aplicaciones móviles, los servicios de mensajería push permiten entregar avisos aun cuando la aplicación no se encuentre activa en primer plano (Google, 2026).

2.3.2.5 Administración de recursos físicos institucionales

La administración de recursos físicos institucionales comprende la planificación, asignación, control y seguimiento de los espacios que sostienen las actividades de una organización. En el ámbito universitario, esta gestión debe relacionar la disponibilidad de aulas, laboratorios, auditorios y otros ambientes con las necesidades académicas y administrativas, procurando que la infraestructura se utilice de manera ordenada y con el menor número posible de interferencias entre usuarios. Una gestión eficiente no depende únicamente de disponer de espacios suficientes, sino de contar con información que permita organizarlos y tomar decisiones oportunas sobre su uso (Ruiz Muñoz y Santos Tomalá, 2024).

La digitalización de estos procesos favorece que la información sobre los espacios deje de estar dispersa en registros aislados y pueda consultarse desde un punto común. Los estudios relacionados con reservas universitarias muestran que la automatización permite estructurar solicitudes, horarios y responsables, reduciendo la dependencia de coordinaciones informales. En consecuencia, la gestión de recursos físicos puede entenderse como un proceso administrativo que requiere reglas, información actualizada y mecanismos de control para sostener el uso compartido de la infraestructura (Omar, 2021; Figueroa Quimis, 2022).

2.3.2.6 Disponibilidad y ocupación de espacios

La disponibilidad representa la condición de un espacio para ser utilizado durante un periodo determinado, mientras que la ocupación expresa el uso efectivo o planificado de ese recurso. Ambas dimensiones son esenciales en un sistema de reservas porque permiten distinguir entre horarios libres, horarios comprometidos y periodos no habilitados. La consulta anticipada de disponibilidad facilita la planificación de actividades y disminuye la incertidumbre asociada a la búsqueda de un espacio adecuado (Cabezuelo, 2023).

El control de ocupación también aporta información para identificar patrones de utilización, como franjas de alta demanda, periodos con baja utilización o espacios que concentran un número elevado de solicitudes. Este tipo de información puede apoyar decisiones administrativas sobre distribución de actividades, reorganización de horarios y aprovechamiento de la infraestructura. Por ello, la disponibilidad no debe considerarse únicamente como un dato de consulta, sino como un componente de gestión que permite observar y mejorar el uso de los recursos físicos (Ruiz Muñoz & Santos Tomalá, 2024).

2.3.2.7 Planificación y calendarización institucional

La calendarización organiza temporalmente el uso de los espacios y relaciona cada reserva con una fecha, una hora de inicio, una hora de finalización y un recurso determinado. En entornos donde varias personas solicitan los mismos espacios, el calendario constituye un mecanismo básico para visualizar compromisos existentes y evitar que las asignaciones se realicen de manera aislada. Los sistemas de reservas analizados en contextos universitarios incorporan calendarios precisamente para facilitar la consulta y el control de la disponibilidad (Omar, 2021); Figueroa Quimis, 2022).

Una planificación institucional adecuada requiere que las reservas ocasionales convivan con actividades previamente programadas, como clases, prácticas o eventos. Esto implica reconocer que la disponibilidad de un espacio depende del conjunto de compromisos registrados y no solamente de las solicitudes creadas en un momento específico. La integración de la calendarización con mecanismos digitales permite representar estas restricciones de forma más clara y contribuye a una coordinación más consistente entre los distintos actores que utilizan la infraestructura (Engel y Coll, 2022).

2.3.2.8 Conflictos de horario y doble asignación

Un conflicto de horario ocurre cuando dos o más solicitudes pretenden utilizar el mismo espacio durante intervalos de tiempo que se superponen. La doble asignación es una consecuencia directa de este problema y puede generar reprogramaciones, pérdida de tiempo y dificultades para desarrollar actividades académicas. En los sistemas digitales de reservas, la validación de cruces de horario constituye una función central porque permite comparar una nueva solicitud con los registros previamente existentes antes de confirmar el uso del recurso (Omar., 2021).

La prevención de conflictos requiere que las reglas de disponibilidad sean consistentes y que la información utilizada para validar las solicitudes esté actualizada. Cuando el proceso se administra mediante mecanismos dispersos, la comprobación depende de revisiones manuales y aumenta la posibilidad de omisiones. Por el contrario, un registro centralizado facilita reconocer superposiciones y mantener una referencia común para quienes solicitan y administran espacios, fortaleciendo el control del proceso de reserva (Figueroa Quimis, 2022).

2.3.2.9 Asignación y priorización de espacios

La asignación consiste en vincular una solicitud aceptada con un espacio y un intervalo de uso. En instituciones con recursos limitados, este proceso puede requerir criterios de prioridad cuando existen varias solicitudes para horarios similares. Los criterios pueden considerar el tipo de actividad, las características del espacio, la cantidad de usuarios o el rol institucional de quien solicita el recurso. La gestión de puestos y espacios compartidos muestra la utilidad de establecer reglas que orienten la asignación y ayuden al usuario a planificar su utilización (Cabezuelo, 2023).

La priorización no implica asignar recursos de manera arbitraria, sino aplicar criterios previamente definidos y conocidos por los usuarios. Esto favorece una gestión más transparente, porque permite justificar por qué una solicitud puede recibir atención antes que otra o por qué un espacio resulta más adecuado para una actividad determinada. En consecuencia, la asignación forma parte de la gobernanza del recurso físico y debe relacionarse con reglas institucionales que promuevan un uso ordenado y equitativo (Ruiz Muñoz & Santos Tomalá, 2024).

2.3.2.10 Centralización y actualización de la información

La centralización de la información busca reunir en un mismo registro los datos relevantes para la gestión de reservas: espacios, horarios, solicitudes, estados y responsables. Cuando estos elementos se administran mediante distintos canales, aumenta la dificultad para conocer cuál es la información vigente. Los antecedentes sobre sistemas universitarios de reservas resaltan la importancia de disponer de una fuente común de datos para reducir inconsistencias y mejorar la coordinación entre usuarios y administradores

Además de centralizar, es necesario mantener la información actualizada. Una reserva aprobada, cancelada o modificada cambia la disponibilidad del espacio y, por tanto, debe reflejarse

en las consultas posteriores. La actualización oportuna permite que las decisiones se basen en un estado reciente del recurso y evita que los usuarios planifiquen actividades a partir de datos desfasados. Esta necesidad se vuelve especialmente relevante cuando existen múltiples usuarios que consultan o gestionan recursos de forma concurrente (Figueroa Quimis, 2022).

2.3.2.11 Trazabilidad y estados de una reserva

La trazabilidad permite seguir el recorrido de una solicitud desde su creación hasta su resolución. Para ello, los sistemas de reservas suelen utilizar estados que identifican la situación de cada registro, por ejemplo, pendiente, aprobada, rechazada o cancelada. La existencia de estados diferenciados ayuda a separar una solicitud que todavía requiere decisión de otra que ya compromete efectivamente la disponibilidad del espacio, ofreciendo mayor claridad sobre el proceso administrativo (Omar et al., 2021).

El seguimiento de estados también mejora la comunicación entre quien solicita y quien administra el recurso. Cuando el usuario puede conocer si su solicitud continúa pendiente o ya fue atendida, disminuye la necesidad de realizar consultas adicionales por canales externos. Desde una perspectiva de gestión, conservar el historial de las decisiones permite revisar cambios y comprender cómo se utilizó un espacio durante un periodo determinado, fortaleciendo el control del proceso (Cabezuelo, 2023).

2.3.2.12 Eficiencia administrativa y tiempos de respuesta

La eficiencia administrativa se relaciona con la capacidad de ejecutar un proceso utilizando de manera adecuada el tiempo y los recursos disponibles. En la gestión de espacios, los tiempos de respuesta pueden verse afectados cuando la disponibilidad debe verificarse manualmente o cuando la solicitud circula por varios canales antes de recibir una confirmación. La incorporación

de herramientas digitales contribuye a estructurar el flujo de trabajo y a reducir actividades repetitivas asociadas con la consulta y coordinación de reservas (Ruiz Muñoz y Santos Tomalá, 2024).

Los sistemas de reservaciones desarrollados en distintos contextos muestran que automatizar el registro y la consulta puede disminuir la carga asociada al procesamiento manual. Aunque el impacto depende de las condiciones de cada organización, la automatización permite trasladar tareas rutinarias hacia el sistema y reservar la intervención humana para decisiones que requieren criterio administrativo. De esta forma, el tiempo de respuesta constituye un indicador útil para valorar el desempeño de un proceso de reservas (Culqui Escobar, 2015).

2.3.2.13 Acceso y equidad en el uso de espacios compartidos

El acceso a los espacios compartidos debe organizarse de modo que los diferentes grupos institucionales puedan conocer las condiciones de uso y realizar solicitudes bajo reglas comprensibles. Los entornos educativos contemporáneos combinan actividades presenciales y recursos digitales, por lo que la gestión de la infraestructura requiere mecanismos que faciliten la coordinación sin depender exclusivamente de la presencia física del usuario en una oficina administrativa (Engel & Coll, 2022).

La disponibilidad de información y la claridad de los criterios de reserva contribuyen a un acceso más equitativo, ya que reducen la ventaja de quienes poseen información informal o pueden gestionar una solicitud de forma presencial con mayor facilidad. Los sistemas digitales de gestión de espacios pueden apoyar esta transparencia al mostrar opciones disponibles y registrar las solicitudes mediante un procedimiento común para los usuarios habilitados (Cabezuelo, 2023).

2.3.2.14 Gestión de reservas en instituciones de educación superior

Las universidades presentan una dinámica particular de uso de espacios, debido a que deben coordinar clases, prácticas, reuniones, actividades administrativas, eventos y otras acciones que comparten infraestructura. Esta diversidad incrementa la necesidad de contar con mecanismos que integren información temporal y funcional de los recursos. Los antecedentes desarrollados en universidades ecuatorianas evidencian que los sistemas de reserva pueden apoyar la administración de aulas, laboratorios y otros recursos cuando estructuran de manera explícita la disponibilidad y las solicitudes (Omar, 2021; Figueroa Quimis, 2022).

En este contexto, la gestión no se limita a registrar una petición, sino que debe relacionar la planificación académica con reglas de uso, responsables y comunicación de cambios. El empleo de tecnologías digitales en entornos educativos permite ampliar las posibilidades de coordinación y adaptar el acceso a las necesidades de una comunidad que utiliza recursos desde diferentes escenarios. Por ello, la gestión de reservas constituye un componente de apoyo a la organización institucional y al aprovechamiento de su infraestructura (Engel & Coll, 2022).

2.3.2.15 Indicadores para evaluar la gestión de reservas

La evaluación de la gestión requiere indicadores que permitan observar si el proceso cumple sus objetivos. Entre los aspectos relevantes se encuentran el tiempo requerido para atender una solicitud, la frecuencia de conflictos de horario, el nivel de ocupación de los espacios y la percepción de los usuarios sobre el servicio. Estos indicadores ayudan a transformar situaciones operativas en información que puede ser comparada y utilizada para orientar decisiones de mejora (Ruiz Muñoz & Santos Tomalá, 2024).

El seguimiento periódico de estos indicadores permite identificar problemas que no siempre son visibles a partir de casos aislados. Por ejemplo, un número elevado de conflictos puede evidenciar fallas en la actualización de disponibilidad, mientras que una ocupación desigual puede mostrar oportunidades para redistribuir actividades. En consecuencia, medir el desempeño complementa el proceso de reserva, ya que aporta evidencia para revisar reglas, procedimientos y formas de asignación de los espacios compartidos (Ruiz Muñoz & Santos Tomalá, 2024).

2.3.3 Metodología propuesta

Se ha seleccionado la metodología Scrum como marco de desarrollo para el presente proyecto, adaptada al contexto académico de la Universidad Laica "Eloy Alfaro" de Manabí, Extensión El Carmen. Esta elección se sustenta en las experiencias exitosas implementadas en proyectos de titulación recientes como los de Emiro Francisco (2023) y Figueroa Quimis (2022), que demostraron que la metodología ágil permite la entrega incremental de funcionalidades, retroalimentación constante del tutor y usuarios finales, y flexibilidad ante cambios en los requisitos institucionales.

Desde la perspectiva de la ingeniería de software, este enfoque permite reducir la distancia entre la planificación, el desarrollo y la retroalimentación, facilitando la revisión de los requisitos a medida que avanza el proyecto. La entrega progresiva de funcionalidades permite identificar problemas de forma temprana y disminuir el riesgo de descubrir al final del desarrollo que la solución implementada no responde adecuadamente a las necesidades establecidas. De esta manera, la retroalimentación frecuente y la revisión de prioridades contribuyen a orientar los esfuerzos hacia aquellas funcionalidades que generan mayor valor para los usuarios (Pressman & Maxim, 2021).

De manera complementaria, Scrum establece cinco valores que orientan el comportamiento del equipo: compromiso, enfoque, apertura, respeto y coraje. El compromiso permite mantener la responsabilidad sobre los objetivos establecidos; el enfoque concentra los esfuerzos en las actividades correspondientes al Sprint; la apertura facilita comunicar avances, dificultades y cambios; el respeto reconoce las capacidades y responsabilidades de cada integrante; y el coraje permite afrontar problemas y tomar las decisiones necesarias durante el desarrollo. Estos valores contribuyen a la adecuada aplicación de los pilares empíricos y favorecen la colaboración dentro del equipo (Schwaber & Sutherland, 2020).

El trabajo se desarrolla mediante un Scrum Team compuesto por Product Owner, Scrum Master y Developers. El Product Owner tiene la responsabilidad de maximizar el valor del producto y gestionar el Product Backlog; el Scrum Master promueve la correcta comprensión y aplicación del marco de trabajo y contribuye a resolver impedimentos que afecten la efectividad del equipo; mientras que los Developers se encargan de crear un Incremento usable durante cada Sprint. Scrum promueve equipos pequeños, multifuncionales y autogestionados, capaces de organizar internamente la manera en que transformarán el trabajo seleccionado en resultados funcionales, reduciendo dependencias y favoreciendo una comunicación continua entre sus integrantes (Schwaber & Sutherland, 2020).

Para organizar el proceso, Scrum emplea eventos y artefactos que proporcionan oportunidades formales de inspección y adaptación. Los artefactos principales son el Product Backlog, el Sprint Backlog y el Incremento, cada uno asociado con un compromiso específico que permite mejorar la transparencia y evaluar el progreso. El Product Backlog se relaciona con el Objetivo del Producto, que establece una meta de largo plazo; el Sprint Backlog se vincula con el Objetivo del Sprint, que define la finalidad concreta del ciclo de trabajo; y el Incremento se

relaciona con la Definición de Terminado, mediante la cual se establecen los criterios de calidad necesarios para considerar un trabajo como completado (Schwaber & Sutherland, 2020).

2.4 Conclusiones relacionadas al marco teórico

El marco teórico de este capítulo establece las bases conceptuales, históricas y tecnológicas que sustentan el desarrollo de una aplicación móvil con sincronización inteligente para la gestión de reservas de espacios físicos compartidos en la Universidad Laica “Eloy Alfaro de Manabí, Extensión el Carmen.

Desde la perspectiva histórica, se ve una evolución desde las aplicaciones móviles simples de los años 2000 hasta los sistemas complejos de hoy en día, que integran sincronización bidireccional, notificaciones en tiempo real y arquitecturas offline-first. Esta mejora, junto con el crecimiento del uso de smartphones y la mejora en cobertura de datos móviles, creo las condiciones tecnológicas necesaria para implementar soluciones móviles robustas en entornos educativos.

Conceptualmente, el sistema propuesto es una aplicación móvil desarrollada con Flutter y Dart, que implementa una arquitectura offline-first con sincronización bidireccional mediante SQLite (a través del paquete sqflite) como base de datos local, y MySQL en el servidor. La integracion de Firebase Cloud Messaging mediante el paquete oficial firebase_messaging garantiza la entrega de notificaciones push en tiempo real, mientras que JWT y Bcrypt se asegura de la autenticación y protección de datos. La decisión de seleccionar Flutter como framework de desarrollo se fundamentaba, por su rendimiento superior gracias a la incorporación motor Skia, por su capacidad de generar aplicaciones nativas para Android e iOS desde un único código fuente en Dart, y por su integración nativa con el ecosistema de servicios de Google. Por estos motivos, las soluciones tecnológicas garantizan escalabilidad, eficiencia, seguridad y accesibilidad móvil,

en línea con las buenas prácticas de la literatura revisada. Adoptar Scrum en el contexto académico permite un desarrollo iterativo y flexible, incorporando requisitos y entregando valor progresivamente a los usuarios finales.

En síntesis, el marco teórico justifica plenamente la viabilidad y pertinencia del proyecto: una aplicación móvil con sincronización inteligente no solo resuelve las ineficiencias actuales en la gestión de espacios compartidos en ULEAM-El Carmen, sino que representa una mejora innovadora y sostenible, que optimiza la utilización de la infraestructura física, garantiza la continuidad operativa ante interrupciones de conectividad, y fortalece a la transformación digital de la institución educativa.

CAPÍTULO III

3 Marco Investigativo

3.1 Introducción

El capítulo que se presenta a continuación describe el diseño del método que se utilizó para realizar la investigación orientada al desarrollo de una aplicación móvil con sincronización inteligente para la gestión de las reservas de espacios físicos compartidos en la Universidad Laica "Eloy Alfaro" de Manabí, Extensión El Carmen. El marco investigativo establece los procedimientos, técnicas e instrumentos utilizados para recopilar, analizar e interpretar la información necesaria que sustenta el desarrollo del proyecto.

La metodología responde al objetivo de evaluar la situación actual de la administración de espacios físicos compartidos y determinar necesidades relacionadas con movilidad, disponibilidad de información y conectividad. Para ello se integraron datos cuantitativos procedentes de la encuesta y datos cualitativos obtenidos mediante entrevistas semiestructuradas.

La evidencia recopilada en este marco investigativo, se proporcionó el sustento empírico para la toma de decisiones de diseño y desarrollo de la aplicación móvil, para que la solución tecnológica propuesta responda de verdad las necesidades identificadas en la comunidad universitaria de la ULEAM-El Carmen.

3.2 Tipos de investigación

Este estudio se enmarca en un enfoque de investigación aplicada, con características descriptivas y exploratorias. Según Hernández-Sampieri y Mendoza Torres (2018), la investigación aplicada tiene como objetivo utilizar el conocimiento científico existente para

resolver problemas prácticos específicos, lo que se alinea con el objetivo de desarrollar soluciones tecnológicas específicas para la gestión de espacios compartidos institucionales.

3.2.1 Investigación aplicada

La investigación aplicada utiliza conocimientos y procedimientos científicos con el propósito de atender problemas concretos dentro de un contexto determinado. Su interés principal se centra en emplear el conocimiento disponible para producir una respuesta útil frente a una necesidad práctica, manteniendo un proceso sistemático de análisis y comprobación (Hernández-Sampieri y Mendoza Torres, 2018). En este proyecto, la investigación aplicada sirvió para transformar la información obtenida sobre la gestión de espacios de la ULEAM-El Carmen en requerimientos para una aplicación móvil. Los resultados del diagnóstico se utilizaron para orientar las funciones de consulta de disponibilidad, solicitud de reservas, control de conflictos, notificaciones y funcionamiento con conectividad limitada.

3.2.2 Investigación descriptiva

La investigación descriptiva busca especificar las propiedades, características y condiciones de un fenómeno a partir de la observación y medición de sus componentes. Este tipo de estudio permite representar cómo se manifiesta una situación en un contexto específico sin modificar deliberadamente las variables analizadas (Hernández-Sampieri y Mendoza Torres (2018). En la investigación, el componente descriptivo permitió caracterizar la forma en que estudiantes, docentes y personal relacionado con la gestión utilizan los espacios compartidos, así como identificar los canales empleados para reservar, los tiempos de gestión, la frecuencia de conflictos, el nivel de satisfacción y las funcionalidades esperadas de una solución móvil.

3.2.3 Investigación exploratoria

La investigación exploratoria se empleó para examinar aspectos poco estudiados en el contexto institucional, como el uso de una aplicación móvil con funcionamiento offline-first, las preferencias de los usuarios y las expectativas sobre la sincronización de datos en escenarios de conectividad variable (Hernández-Sampieri y Mendoza Torres, 2018).

En este proyecto, la investigación exploratoria permitió indagar aspectos que no estaban formalmente documentados en la extensión, especialmente las experiencias de los responsables frente a conflictos de reserva y problemas de conectividad. Esta aproximación ayudó a reconocer necesidades que complementaron los resultados cuantitativos de la encuesta.

3.3 Métodos de investigación

3.3.1 Método cuantitativo

El método cuantitativo se fundamenta en la recolección de datos susceptibles de medición y análisis numérico. Utiliza procedimientos estandarizados para organizar las respuestas, identificar frecuencias, comparar resultados y describir tendencias relacionadas con las variables o dimensiones estudiadas (Hernández-Sampieri y Mendoza Torres, 2018). En la investigación, el método cuantitativo se utilizó mediante una encuesta estructurada aplicada. Los datos obtenidos permitieron cuantificar la frecuencia de uso de los espacios, los medios empleados para gestionar reservas, los conflictos de horario, el nivel de satisfacción y la disposición de los participantes para utilizar una aplicación móvil.

3.3.2 Método cualitativo

El método cualitativo busca comprender fenómenos a partir de las experiencias, significados y perspectivas de los participantes dentro de su contexto. Para ello emplea técnicas

flexibles que permiten profundizar en situaciones, prácticas y valoraciones que no pueden reducirse únicamente a datos numéricos (Denzin y Lincoln, 2022). En este estudio, el método cualitativo se aplicó mediante entrevistas semiestructuradas al coordinador académico y a un docente con alta demanda de espacios. Las entrevistas permitieron profundizar en el proceso de gestión, los casos de doble asignación, la utilización de los espacios, las necesidades administrativas y los problemas de conectividad percibidos en la institución.

3.4 Fuentes de información de datos

Las fuentes de información se organizaron según el origen de los datos utilizados en el diagnóstico. En concordancia con el procedimiento seguido en el estudio, se trabajó con fuentes primarias y se emplearon dos técnicas de recolección de datos: la encuesta estructurada y la entrevista semiestructurada (Arias, 2016).

3.4.1 Fuentes primarias

3.4.1.1 Encuesta

. La encuesta es una técnica de recolección de datos que permite obtener información de un grupo de participantes mediante un cuestionario organizado y aplicado bajo criterios comunes. Su estructura facilita recopilar respuestas comparables y posteriormente resumirlas mediante frecuencias, porcentajes u otras formas de análisis cuantitativo (Arias, 2016). En el proyecto, se aplicó una encuesta mediante Google Forms a estudiantes, docentes y dos participantes del personal administrativo, obteniendo 256 respuestas. Esto permitió identificar tendencias sobre el uso de espacios compartidos, las reservas, los conflictos de horario, la satisfacción y las expectativas sobre una aplicación móvil.

3.4.1.2 Entrevista

La entrevista semiestructurada combina una guía previa de preguntas con la posibilidad de profundizar en temas que surgen durante la conversación. Esta técnica permite obtener información detallada sobre experiencias, opiniones y prácticas de los participantes sin limitar el diálogo a respuestas completamente cerradas (Hernández-Sampieri & Mendoza Torres, 2018).

En la investigación, la entrevista se aplicó al coordinador académico y a un docente con alta demanda de espacios. Su utilización permitió conocer con mayor profundidad cómo se coordinan las reservas, qué situaciones generan conflictos, qué criterios se consideran al asignar espacios y cuáles son las funcionalidades que los responsables consideran necesarias para mejorar el proceso.

3.5 Estrategia operacional para la recolección de datos

3.5.1 Población

La población estuvo conformada por 903 miembros de la comunidad universitaria de la ULEAM-El Carmen vinculados con el uso o gestión de espacios físicos compartidos: 810 estudiantes, 85 docentes y 8 miembros del personal administrativo y autoridades.:

Segmento	Cantidad
Estudiantes (todas las carreras)	810
Docentes	85
Personal administrativo y autoridades	8
Total	903

Tabla 1 Distribución de la población objeto de estudio

3.5.2 Segmentación

Para el estudio, la población se divide en tres grupos según su relación con el sistema de gestión de espacios: estudiantes, como principales usuarios finales que consultan disponibilidad y solicitan espacios para actividades académicas y extracurriculares; docentes, quienes realizan reservas para clases, prácticas y otras actividades académicas con necesidades específicas de planificación; y personal administrativo y autoridades, responsables de aprobar reservas, gestionar el sistema, generar reportes y definir los requisitos administrativos necesarios para su funcionamiento.

3.5.2.1 Técnica de muestreo

Para los segmentos A y B, conformados por estudiantes y docentes, se utilizó un muestreo no probabilístico de tipo intencional o por conveniencia, seleccionando a los participantes según su disponibilidad y acceso mediante los canales institucionales, sin aplicar un proceso aleatorio. Según Hernández-Sampieri y Mendoza Torres (2018), este tipo de muestreo resulta adecuado cuando se busca obtener información de participantes accesibles en el momento de la recolección, priorizando la riqueza de los datos sobre la representatividad estadística exacta.

Bajo este criterio, se aplicó la encuesta a 256 miembros de la comunidad universitaria: 230 estudiantes, 24 docentes y 2 participantes del personal administrativo. La mayor participación estudiantil se relaciona con el carácter voluntario del instrumento y los medios de difusión utilizados, lo cual se reconoce como una limitación del muestreo empleado.

Para el segmento C se mantuvo el criterio intencional, seleccionando al coordinador académico por su responsabilidad directa en la gestión de espacios y a un docente con alta demanda de estos recursos, cuya experiencia permitió complementar la información cualitativa del estudio.

3.5.2.2 Tamaño de la muestra

Inicialmente se calculó un tamaño de muestra referencial mediante la fórmula para poblaciones finitas, con fines de planificación del trabajo de campo, aplicando un nivel de confianza del 95% y un margen de error del 5%:

$$n = (N \times Z^2 \times p \times q) / (e^2 \times (N - 1) + Z^2 \times p \times q)$$

Donde:

- N = Tamaño de la población
- Z = Nivel de confianza (1.96 para 95%)
- p = Probabilidad de ocurrencia (0.5)
- q = Probabilidad de no ocurrencia (0.5)
- e = Margen de error (0.05)

Aplicando la formula a la población combinada de estudiantes y docentes(N = 895):

$$n = (895 \times 1.96^2 \times 0.5 \times 0.5) / (0.05^2 \times 894 + 1.96^2 \times 0.5 \times 0.5)$$

$$n = (895 \times 3.8416 \times 0.25) / (0.0025 \times 894 + 0.9604)$$

$$n = 859.858 / 3.1954 = 269$$

El valor de 269 se utilizó únicamente como referencia, ya que la selección efectiva fue intencional y no aleatoria. La muestra final estuvo constituida por 256 participantes, de acuerdo con la disponibilidad real durante la aplicación de los instrumentos.

Segmento	Población	Muestra
Estudiantes	810	230
Docentes	85	24
Personal administrativo (incidental)	-	2
Total	895	256

Tabla 2 Distribución de la muestra por segmentos

3.5.3 Análisis de las herramientas de recolección de datos a utilizar

3.5.3.1 Encuesta estructurada

La encuesta es un instrumento cuantitativo que permite recopilar información estandarizada de un número considerable de participantes en poco tiempo. Según Arias (2016), la encuesta "es una técnica que pretende obtener información que suministra un grupo o muestra de sujetos acerca de sí mismos, o en relación con un tema en particular".

En este estudio, la encuesta se aplicó de forma digital mediante Google Forms y obtuvo 256 respuestas válidas. El instrumento permitió conocer la frecuencia de uso de los espacios, las formas de gestionar reservas, los tiempos de atención, la presencia de conflictos de horario, el nivel de satisfacción y las expectativas respecto a una aplicación móvil.

3.5.3.2 Entrevista semiestructurada

La entrevista semiestructurada es un instrumento cualitativo que combina preguntas previamente definidas con la posibilidad de profundizar en temas relevantes durante el diálogo. Hernández-Sampieri y Mendoza Torres (2018), este tipo de entrevista se apoya en una guía de preguntas, pero permite incorporar interrogantes adicionales para aclarar conceptos y obtener mayor información.

En este estudio, la entrevista semiestructurada se aplicó al coordinador académico y a un docente con alta demanda de espacios (n = 2). Las sesiones permitieron profundizar en el proceso de gestión, los conflictos de asignación, los criterios de priorización, las necesidades funcionales y las dificultades de conectividad que no podían explicarse únicamente mediante los datos cuantitativos.

3.5.3.3 Estructura de los instrumentos de recolección de datos aplicados

3.5.3.3.1 Estructura de la encuesta

El cuestionario de encuesta se organizó en tres secciones. La primera recopiló información general sobre el rol del participante y la frecuencia de uso de los espacios compartidos. La segunda evaluó el proceso actual de reservas, considerando aspectos como el tiempo empleado, los conflictos de horarios, el nivel de satisfacción y los principales inconvenientes. La tercera se enfocó en las expectativas sobre un sistema automatizado, incluyendo la disposición para utilizar una aplicación móvil, las funcionalidades más valoradas y el dispositivo de acceso.

En total, el cuestionario estuvo compuesto por 10 preguntas, de las cuales nueve fueron cerradas y una correspondió a una escala tipo Likert de cinco opciones, utilizada en la pregunta 6 para medir el nivel de satisfacción con el proceso actual.

3.5.3.3.2 Estructura de la entrevista

La guía de entrevista semiestructurada se organizó en varios ejes temáticos: descripción del proceso actual de gestión de espacios, actores involucrados y tiempos promedio; problemas y limitaciones existentes; criterios de priorización y resolución de conflictos; requisitos funcionales del sistema, como aprobaciones, reportes y gestión de usuarios; expectativas de integración con

otros sistemas institucionales y disposición al cambio tecnológico. La guía completa de preguntas orientadoras se presenta en el Anexo B.

3.5.4 Plan de recolección de datos

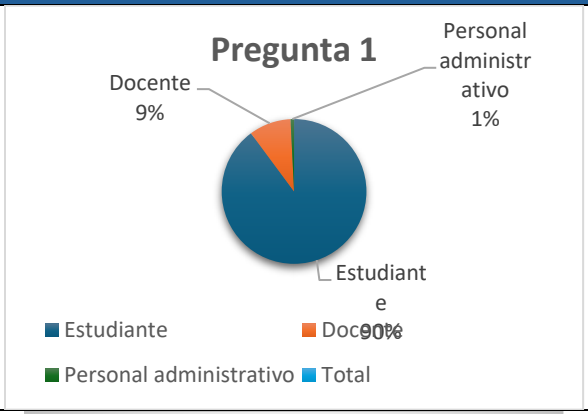
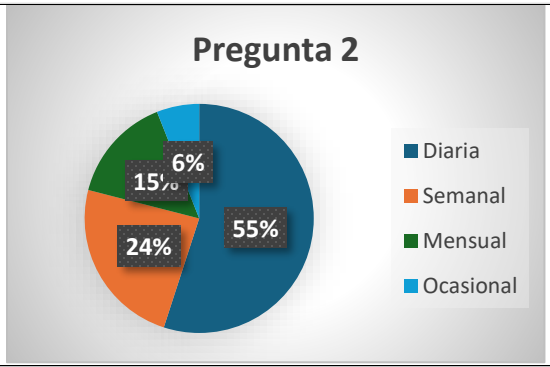
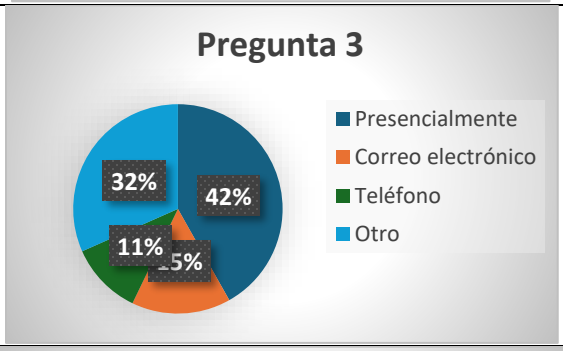
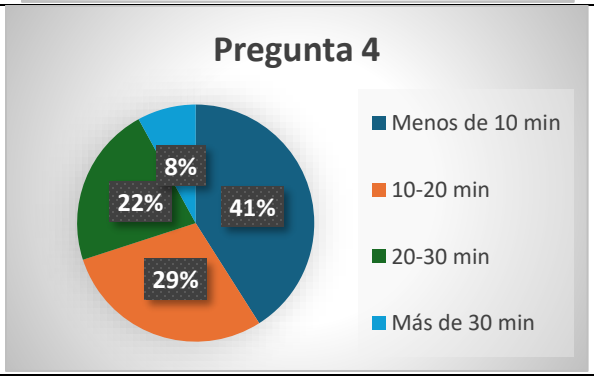
El plan de recolección de datos define de manera sistemática los aspectos operativos que guían la obtención de información primaria. A continuación se presenta la tabla que sintetiza las preguntas básicas del plan:

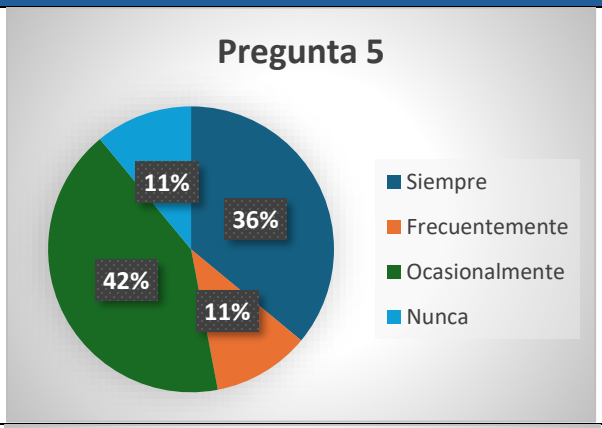
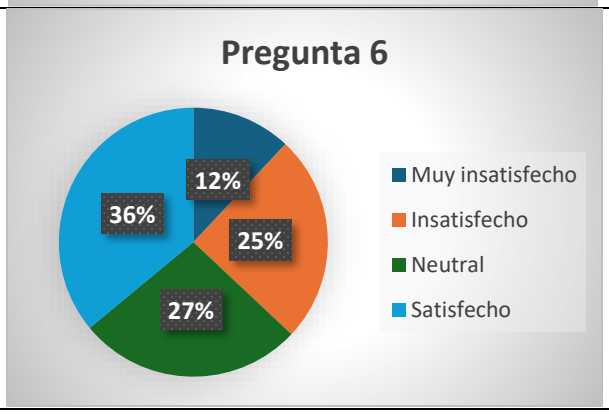
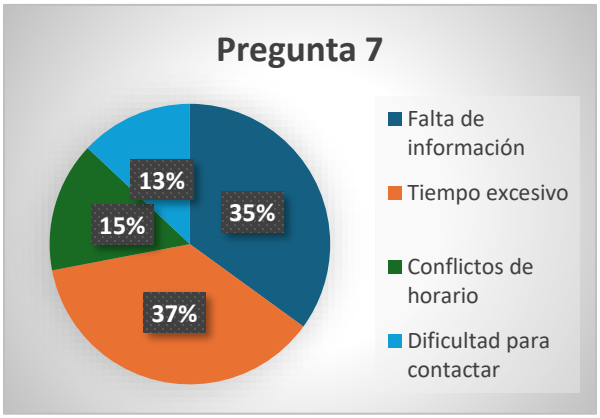
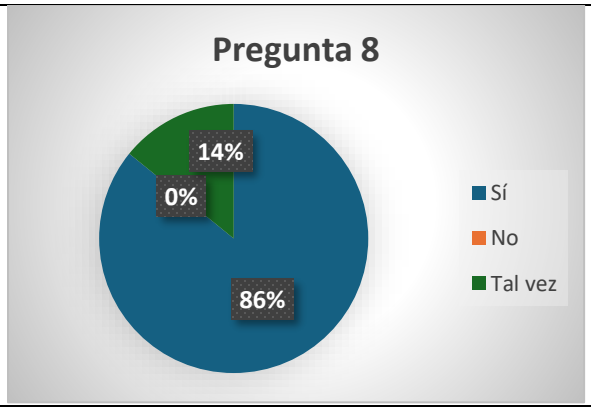
Preguntas básicas	Explicación
¿Para qué?	Para evaluar la situación presente de la administración de espacios físicos compartidos en la ULEAM-El Carmen y determinar requerimientos puntuales de conectividad y movilidad que orienten el desarrollo de la aplicación móvil.
¿Sobre qué aspectos?	Sobre la problemática de gestión de espacios compartidos, los procedimientos actuales, las deficiencias operativas, las necesidades tecnológicas de los usuarios y las expectativas respecto a una solución móvil con sincronización inteligente.
¿Quién?	Rey Josias Miranda Llerena, autor del proyecto de titulación, bajo la supervisión del Ing. Wladimir Minaya, MG., tutor asignado.
¿Cuándo?	Durante el período comprendido en enero de 2026, coincidiendo con el período de aplicación efectiva de los instrumentos a la comunidad universitaria.
¿Dónde?	En las instalaciones de la Universidad Laica "Eloy Alfaro" de Manabí, Extensión El Carmen; las entrevistas se realizaron de forma presencial con el coordinador académico y un docente de la institución.
¿Cuántas veces?	Se aplicaron los instrumentos de recolección de datos una sola vez a cada participante en una única ronda de recolección; La encuesta se distribuyó de forma digital, mientras que las entrevistas se realizan en sesiones individuales programadas.
¿Qué técnicas de recolección de datos?	Encuesta estructurada (para estudiantes y docentes), entrevista semiestructurada (para el coordinador académico y un docente con alta demanda de espacios).
¿Con qué?	Cuestionario estructurado digital (Google Forms) con 10 preguntas cerradas para la encuesta, y guía de entrevista con 10 preguntas orientadoras para las entrevistas semiestructuradas.

Tabla 3 Plan de recolección de datos

3.6 Análisis y presentación de resultados

3.6.1 Análisis de los resultados de la encuesta

Pregunta	Gráfico	Interpretación
¿Cuál es su rol dentro de la institución?		La mayoría de los participantes son estudiantes (89,8 %), seguidos por docentes (9,4 %) y personal administrativo (0,8 %), evidenciando que los estudiantes representan los principales usuarios potenciales del sistema.
¿Con qué frecuencia utiliza los espacios físicos compartidos de la institución?		El 55 % de los participantes utiliza los espacios compartidos diaria o semanalmente, evidenciando una demanda frecuente y la necesidad de un sistema que facilite la disponibilidad y gestión de reservas.
¿Cómo realiza actualmente la gestión de una reserva?		El 42 % gestiona sus reservas de forma presencial, el 15 % por correo electrónico y el 11 % por teléfono. Esto evidencia una gestión dispersa y la necesidad de centralizar las reservas en un solo sistema.
¿Cuánto tiempo dedica aproximadamente a gestionar una reserva?		El 41 % tarda menos de 10 minutos en gestionar una reserva, el 29 % entre 10 y 20 minutos y el 22 % más de 20 minutos, evidenciando demoras que podrían reducirse mediante una aplicación de reservas.

Pregunta	Gráfico	Interpretación
¿Con qué frecuencia experimenta conflictos de horario al reservar un espacio?		El 47% de los participantes enfrenta conflictos de horario siempre o frecuentemente, mientras que el 42,0% los experimenta ocasionalmente. Solo el 11,0% no presenta conflictos, evidenciando deficiencias en el control y visualización de la disponibilidad de los espacios.
¿Cuál es su nivel de satisfacción con el proceso actual de reserva?		El 37% de los participantes se muestra insatisfecho o muy insatisfecho con el proceso actual, frente al 36% que está satisfecho y el 27% que mantiene una posición neutral. Esto refleja una percepción dividida, con una ligera predominancia de la insatisfacción.
¿Cuál es el principal obstáculo que enfrenta durante el proceso de reserva?		El principal obstáculo es el tiempo excesivo para gestionar las reservas 37%, seguido de la falta de información sobre la disponibilidad 35,0%. En conjunto, representan el 72,0% de las respuestas, evidenciando que la demora y la falta de información actualizada son los principales problemas que una aplicación podría solucionar.
¿Estaría dispuesto a utilizar una aplicación móvil para consultar la disponibilidad y reservar espacios?		El 86 % de los participantes utilizaría una aplicación móvil para consultar y reservar espacios, mientras que el 14 % lo consideraría. En total, el 100 % muestra disposición o apertura hacia la propuesta, lo que respalda la pertinencia de desarrollar la aplicación móvil.

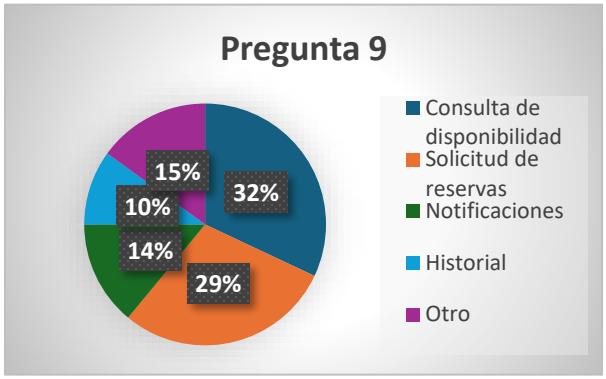
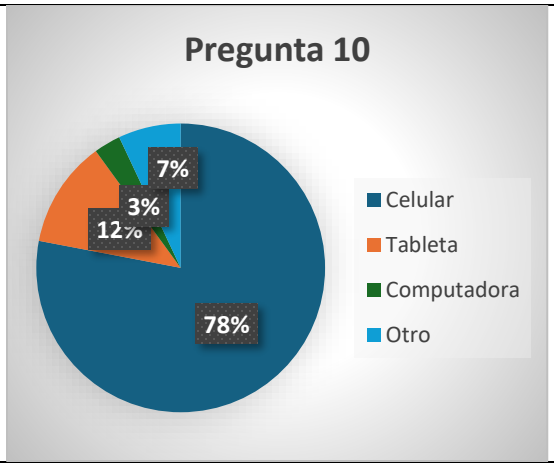
Pregunta	Gráfico	Interpretación
¿Qué funcionalidad considera más importante en una aplicación de gestión de reservas?		La consulta de disponibilidad 32 % y la solicitud de reservas 29 % son las funcionalidades más valoradas, concentrando el 61 % de las respuestas. Esto demuestra que los usuarios priorizan conocer la disponibilidad y gestionar reservas desde la aplicación. Las notificaciones 14 % también son relevantes para informar sobre cambios en las reservas.
¿Desde qué dispositivo accedería principalmente a la aplicación?		El celular concentra el 78 % de las respuestas, seguido por la tableta (12 %), otras opciones (7 %) y la computadora (3 %).

Tabla 4 Análisis de resultados de la encuesta

3.6.2 Resultados de las entrevistas

Las entrevistas realizadas al coordinador académico y a un docente con alta demanda de espacios (n=2) proporcionaron información cualitativa que enriquece la comprensión de la problemática.

Los principales hallazgos fueron los siguientes:

- El proceso vigente es manual: las reservas se coordinan principalmente mediante comunicación verbal, correo o llamadas, sin un registro centralizado de disponibilidad..
- Tiempo de gestión de conflictos: se reportaron casos de doble asignación que requieren coordinación manual y reubicación de actividades.

- Uso ineficiente de espacios: existen franjas horarias y aulas con menor utilización, asociadas a la falta de una planificación centralizada.
- Funciones esperadas del sistema: disponibilidad en tiempo real, reservas desde el móvil, confirmaciones, notificaciones automáticas y calendario institucional compartido.
- Riesgos percibidos: se identificó como principal desafío la adaptación inicial de los usuarios y la necesidad de capacitación.

3.6.3 Informe final del análisis de los datos

El análisis integral de los datos recopilados mediante encuestas, entrevistas y observación directa permite establecer las siguientes conclusiones que fundamentan el desarrollo del proyecto:

a. Validación de la problemática identificada

Los resultados confirman dificultades relacionadas con la disponibilidad de información, el tiempo empleado en la gestión y los conflictos de horario. Estos hallazgos respaldan la necesidad de centralizar el proceso de reservas.

b. Demanda confirmada de solución móvil

La encuesta evidencia una alta disposición de los participantes para utilizar una aplicación móvil de consulta y reserva de espacios, lo que respalda la pertinencia de la propuesta tecnológica.

La necesidad de una arquitectura offline-first no se midio con la encuesta: surgio como un hallazgo cualitativo de las entrevistas, que evidenciaron interrupciones intermitentes de conectividad en la institución lo que motivo que la aplicación no dependa de una conexión continua a internet.

c. Priorización de funcionalidades

Con base en los resultados obtenidos, se priorizan las siguientes funcionalidades para el desarrollo del sistema:

- Consulta de disponibilidad en tiempo real (32 %).
- Solicitud de reservas desde la aplicación (29 %).
- Notificaciones sobre confirmaciones y cambios (14 %).
- Consulta del historial de reservas (10 %).
- Otras funcionalidades señaladas por los participantes (15 %).

d. Necesidades administrativas específicas

El personal administrativo necesita: la aprobación manual de reservas según criterios institucionales, juntamente con reportes automáticos de ocupación por espacio y periodo, visualización de conflictos antes de aprobar solicitudes, y registro histórico de transacciones para auditoría.

e. Impacto esperado cuantificado

La automatización propuesta busca reducir tiempos de consulta y coordinación, mejorar la disponibilidad de información y disminuir los conflictos derivados del proceso manual.

En conjunto, los hallazgos orientaron los requisitos funcionales, la arquitectura y las tecnologías desarrolladas en el Capítulo IV.

CAPÍTULO IV

4 MARCO PROPOSITIVO

4.1 Introducción

La gestión manual de los espacios físicos compartidos en la Uleam-El Carmen (aulas, laboratorios, canchas y auditorio) ha evidenciado deficiencias operativas como conflictos de horario, doble asignación de espacios y ausencia de un mecanismo centralizado que permita a estudiantes, docentes y administradores conocer en tiempo real la disponibilidad de dichos recursos. Esa problemática fue confirmada mediante los resultados obtenidos en la encuesta y en las entrevistas aplicadas a la comunidad universitaria, presentados en el Capítulo III de esta investigación.

Ante esta situación, se propone el desarrollo de una aplicación móvil con sincronización inteligente que permita automatizar la consulta de disponibilidad, la solicitud y la aprobación de reservas, y la notificación de su estado, incorporando una arquitectura offline-first que garantice el acceso a la información aun sin conexión a internet, optimizando así el uso de la infraestructura institucional y la experiencia de los usuarios.

Se construyó el sistema mediante metodología Scrum, dividida en cuatro sprints iterativos, empleando tecnologías como Flutter y Dart para el frontend móvil, Node.js junto a Express en el servidor, MySQL como gestor de datos, Socket.io para la comunicación instantánea, Firebase Cloud Messaging para las alertas push y SQLite como base de datos local para el funcionamiento sin conexión.

4.2 Descripción de la propuesta

La aplicación móvil de reservas de espacios físicos para la Uleam-El Carmen tiene como finalidad optimizar los procesos de consulta, solicitud, y aprobación de reservas de los espacios institucionales. Para ello incorpora un módulo de autenticación con correo institucional, un módulo de consulta de disponibilidad mediante calendario interactivo, un módulo de solicitud y aprobación de reservas con validación automática de conflictos, un módulo de gestión de espacios y bloqueos de horario reservado al administrador, y un módulo de notificaciones que combina avisos push tiempo real y correo electrónico.

El sistema está dividido en tres roles de usuario: administradores, profesores y estudiantes. Los administradores tienen control total sobre la gestión de espacios físicos, pudiendo crear, modificar y cambiar el estado de un espacio, configurar bloqueos de horario, aprobar o rechazar solicitudes de reserva y consultar reportes de ocupación. Los profesores y estudiantes pueden consultar la disponibilidad, generar solicitudes de reserva y recibir notificaciones sobre el estado de las mismas; como medida de priorización, únicamente a los estudiantes se les exige una anticipación mínima de veinticuatro horas para solicitar una reserva.

Adicionalmente, la aplicación incorpora una arquitectura offline-first que permite a los usuarios consultar los espacios disponibles y su historial de reservas incluso sin conexión a internet, mediante una base de datos local SQLite que almacena una copia de los datos más recientes obtenidos del servidor.

4.3 Determinación de recursos

4.3.1 Humanos

Recurso	Cargo	Función
Rey Josias Miranda Llerena	Desarrollador	Encargado del diseño de la base de datos, desarrollo del frontend móvil en Flutter, integración con el backend compartido, configuración de Firebase Cloud Messaging y de la arquitectura offline-first.
Ing. Wladimir Minaya Macias, MG	Tutor/Supervisor del sistema	A cargo de supervisar el desarrollo del proyecto, validar las funcionalidades del sistema, y aprobar los entregables de cada sprint.
Melvin Emilio Zambrano Palacios	Colaborador-backend compartido	Responsable del desarrollo y mantenimiento del backend Node.js/Express + MySQL, consumido en conjunto por el sistema web y la aplicación móvil.
Coordinador académico, docentes y estudiantes de la Uleam-El Carmen	Usuarios de prueba	Encargados de probar las funcionalidades de la aplicación y dar retroalimentación sobre los procesos de gestión de espacios compartidos.

Tabla 5 Recursos Humanos

4.3.2 Tecnológicos

Recurso	Descripción
Visual Studio Code (v. 1.109.0)	Editor de código utilizado como entorno principal de desarrollo del backend y de apoyo para la revisión del proyecto Flutter.
Android Studio	IDE oficial de desarrollo Android, utilizado para la configuración del SDK, la gestión de emuladores y la generación de los paquetes de instalación (APK).
Flutter SDK (3.44.2) / Dart	Framework multiplataforma utilizado para el desarrollo de la aplicación móvil, con arquitectura basada en widgets y navegación declarativa mediante go_router.
Node.js (22.11.0) + Express	Entorno de ejecución y framework para el desarrollo del backend, la definición de la API REST y la gestión de sesiones.
MySQL	Sistema gestor de base de datos relacional para el almacenamiento estructurado de usuarios, recursos, reservas, bloqueos y tokens de notificación.
Socket.io	Librería para la comunicación en tiempo real entre el backend y los clientes: avisa de inmediato al administrador ante una nueva solicitud, y al usuario ante un cambio de estado.

Recurso	Descripción
Firebase Cloud Messaging + Firebase Admin SDK	Servicio usado para el envío de notificaciones push a los dispositivos Android, tanto en primer plano como en segundo plano.
SQLite (paquete sqflite)	Motor de base de datos local embebido en el dispositivo, usado para implementar la arquitectura offline-first de la aplicación.
Git y GitHub	Herramientas de control de versiones para el respaldo incremental del código fuente del proyecto.
Postman / Thunder Client	Herramientas para probar manualmente los endpoint del backend antes de integrarlos con la aplicación móvil.
Dispositivos físicos Android	Equipos usados para las pruebas funcionales, de notificaciones push y de modo offline en condiciones reales.

Tabla 6 Recursos Tecnológicos

4.3.3 Económicos

Cantidad	Recurso	Recurso	Subtotal
1	Laptop HP 15	$ 836,99	$ 836,99
1	Teléfono celular Samsung SM-A125U	$ 150,00	$ 150,00
1	Teléfono celular Infinix X678B	$ 230,00	$ 230,00
3 meses	Hosting (Railway)	$ 5,00	$ 15,00
4 meses	Internet	$ 23,90	$ 95,60
350 horas	Horas de programación	$ 8,00	$ 2.800,00
TOTAL			**$ 4.127,59**

Tabla 7 Recursos Económicos

4.4 Desarrollo del sistema según la metodología Scrum

El desarrollo de la aplicación se organizó mediante la metodología ágil Scrum, adaptada al contexto académico del proyecto. La implementación se distribuyó en cuatro sprints incrementales de dos semanas: el Sprint 1 estableció la base de autenticación y la arquitectura del proyecto; el Sprint 2 implementó la gestión de espacios físicos y los bloqueos de horario; el Sprint 3 desarrolló el flujo completo de reservas y su aprobación, además de integrar el backend compartido; y el Sprint 4 incorporó las notificaciones push mediante Firebase Cloud Messaging, la arquitectura offline-first y la identidad visual de la aplicación.

4.4.1 Descripción del Producto

4.4.1.1 Próposito del Producto

La aplicación móvil de reservas de espacios físicos para la Uleam-El Carmen tiene como finalidad optimizar los procesos de consulta, solicitud, y aprobación de reservas de los espacios institucionales. Para ello incorpora un módulo de autenticación con correo institucional, un módulo de consulta de disponibilidad mediant calendario interactivo, un módulo de solicitud y aprobación de reservas con validación automática de conflictos, un módulo de gestión de espacios y bloqueos de horario reservado al administrador, y un módulo de notificaciones que combina avisos push tiempo real y correo electrónico.

El producto final corresponde a una aplicación móvil orientada a estudiantes, docentes y personal administrativo de la ULEAM-El Carmen, capaz de centralizar la información de los espacios físicos compartidos, aplicar reglas de disponibilidad, gestionar solicitudes de reserva y mantener una copia local de información para consultas sin conexión.

4.4.1.2 Funcionalidades Clave

Registro e inicio de sesión mediante correo institucional, con control de acceso según rol.

Consulta de espacios físicos y disponibilidad mediante calendario interactivo.

Creación de solicitudes de reserva con validación automática de conflictos de horario.

Gestión administrativa de espacios, estados, imágenes y bloqueos por horario o rango de fechas.

Aprobación o rechazo de reservas pendientes por parte del administrador.

Notificaciones mediante Socket.io, Firebase Cloud Messaging y correo electrónico.

Consulta de espacios y reservas propias en modo sin conexión mediante SQLite y estrategia cache-first.

Reportes de ocupación y restricción de funciones administrativas mediante control de roles.

4.4.1.3 Usuarios Objetivo

Tipo de usuario	Funciones principales
Estudiante	Registrarse, iniciar sesión, consultar disponibilidad, solicitar reservas con anticipación mínima de 24 horas, revisar sus reservas y recibir notificaciones.
Profesor	Registrarse, iniciar sesión, consultar disponibilidad, solicitar reservas sin la restricción de anticipación aplicada a estudiantes, revisar sus reservas y recibir notificaciones.
Administrador	Gestionar espacios y bloqueos, revisar solicitudes pendientes, aprobar o rechazar reservas, consultar reportes y administrar la disponibilidad.

Tabla 8 Funciones de Usuarios

4.4.1.4 Condiciones de Éxito del Producto

El usuario puede autenticarse y mantener una sesión válida durante la navegación.

Los conflictos con clases programadas y reservas existentes son detectados antes de registrar una nueva solicitud.

Las funciones administrativas solo están disponibles para usuarios con rol ADMINISTRADOR.

Las notificaciones de cambios de estado llegan mediante los canales implementados.

La aplicación permite consultar información almacenada localmente cuando no existe conexión a internet.

La versión Android funciona en los dispositivos físicos empleados durante las pruebas y se comunica con el backend desplegado en Railway.

4.4.1.5 Requerimientos no funcionales

- RNF01. Disponibilidad: la aplicación debe permitir la consulta de información 24/7, incluso sin conexión, mediante la arquitectura offline-first.
- RNF02. Seguridad: las contraseñas son guardadas cifradas mediante bcrypt; la protección de los endpoints administrativos está protegido por middlewares de verificación de sesión y de rol.
- RNF03. Usabilidad: la navegación entre pantallas debe ser fluida, con una clara retroalimentación visual (indicadores de carga, banners de estado, colores por disponibilidad).
- RNF04. Rendimiento: las consultas de disponibilidad y de reservas se deben resolver en un tiempo que el usuario aprecie como inmediato, apoyándose en el cache local cuando corresponda.
- RNF05. Compatibilidad: la aplicación debe funcionar en dispositivos Android 5.0 (API 21) o superior.
- RNF06. Confiabilidad de la conectividad: detectar la conexión a internet debe hacerse mediante resolución DNS real, y no solo con el estado de la interfaz de red.
- RNF07. Mantenibilidad: el backend se organiza en rutas, middlewares y servicios independientes por módulo (auth, usuario, recursos, reservas, bloqueos, reportes), facilitando su mantenimiento.
- RNF08. Identidad visual: la aplicación debe presentar una identidad institucional coherente (logo, ícono adaptativo, pantalla de bienvenida).

4.4.1.6 Historias de Usuario

Los requerimientos funcionales definidos inicialmente se transformaron en historias de usuario para incorporarlos al Product Backlog. Cada historia expresa una necesidad desde la perspectiva del actor que recibe el valor de la funcionalidad y se asigna al sprint en el que fue implementada y validada.

4.4.2 Diseño de programa

4.4.2.1 Diagramas UML

A continuación se presentan los diagramas UML que modelan el comportamiento y la estructura del sistema: el diagrama de casos de uso, el diagrama de clases, el diagrama de estados de una reserva y diagrama de flujo del proceso de creación y aprobación de una reserva.

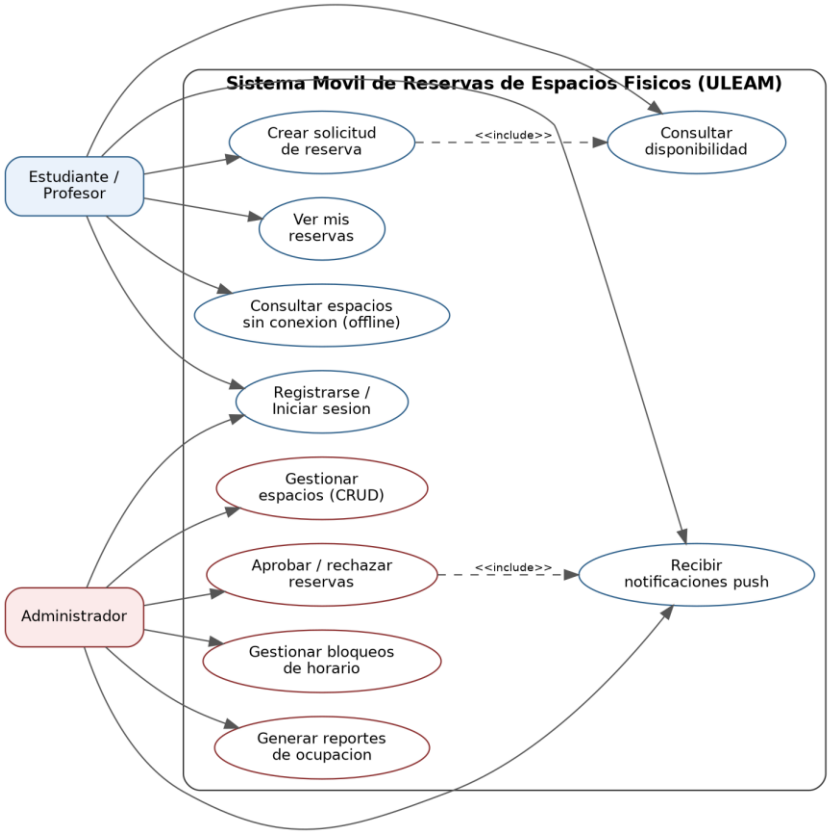


Figura 2 Diagrama de casos de uso del sistema

El diagrama de casos de uso agrupa las funcionalidades disponibles para los dos perfiles de usuario del sistema. El rol Estudiante/Profesor concentra las funciones orientadas al consumo de servicio (consulta, reserva y seguimiento), mientras que el rol Administrador concentra las funciones de gobierno del sistema (gestión de espacios, bloqueos, aprobación y reportes). Ambos roles comparten el caso de uso de autenticación y la recepción de notificaciones.

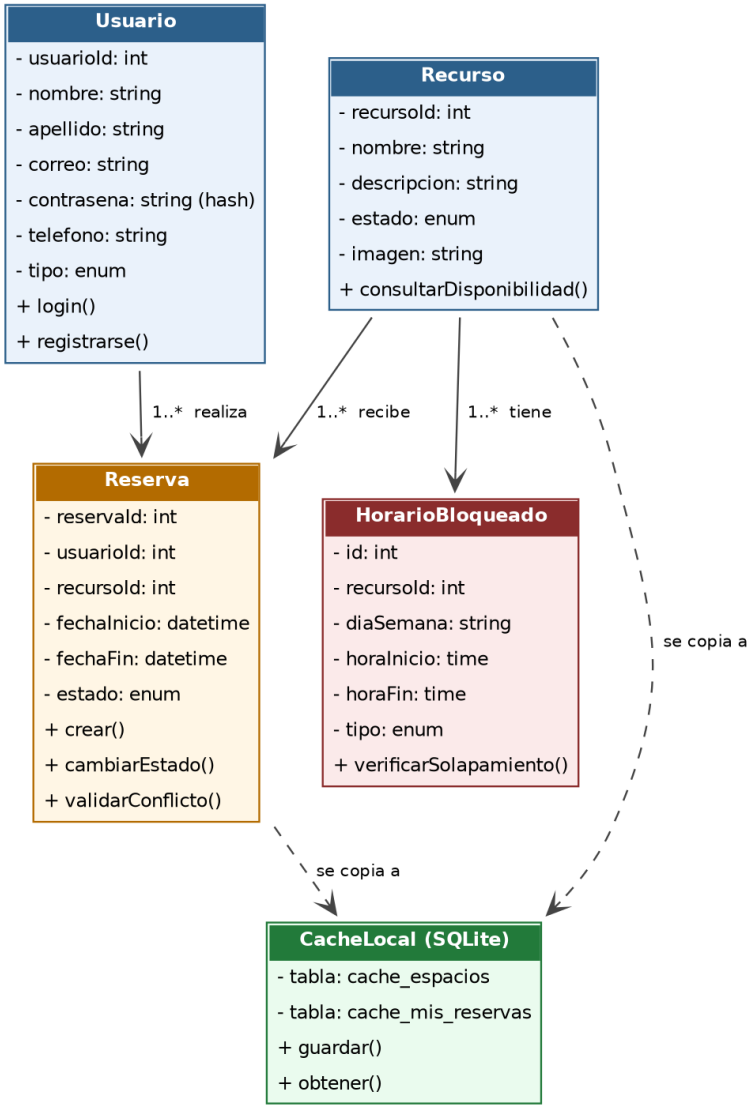


Figura 3 Diagrama de clases simplificado

El diagrama de clases representa las entidades principales del dominio y de su relación: un Usuario puede realizar múltiples solicitudes de Reserva sobre uno o varios Recursos; cada Recurso puede tener asociados varios HorarioBloqueador; y tanto los recursos como las reservas del usuario autenticando se replican en la clase CacheLocal para su consulta sin conexión.

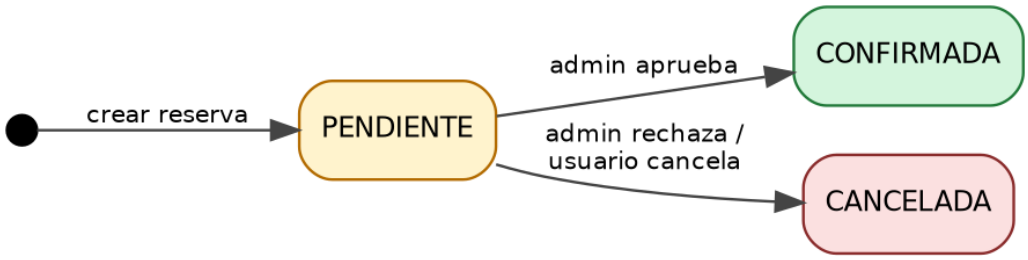


Figura 4 Diagrama de estados de una reserva

Toda reserva se crea en estado PENDIENTE. A partir de allí, únicamente el administrador puede transicionarla a CONFIRMADA (aprobada) o a CANCELADA (rechazo o cancelado por el propio usuario); no existen transiciones de retorno hacia el estado PENDIENTE.

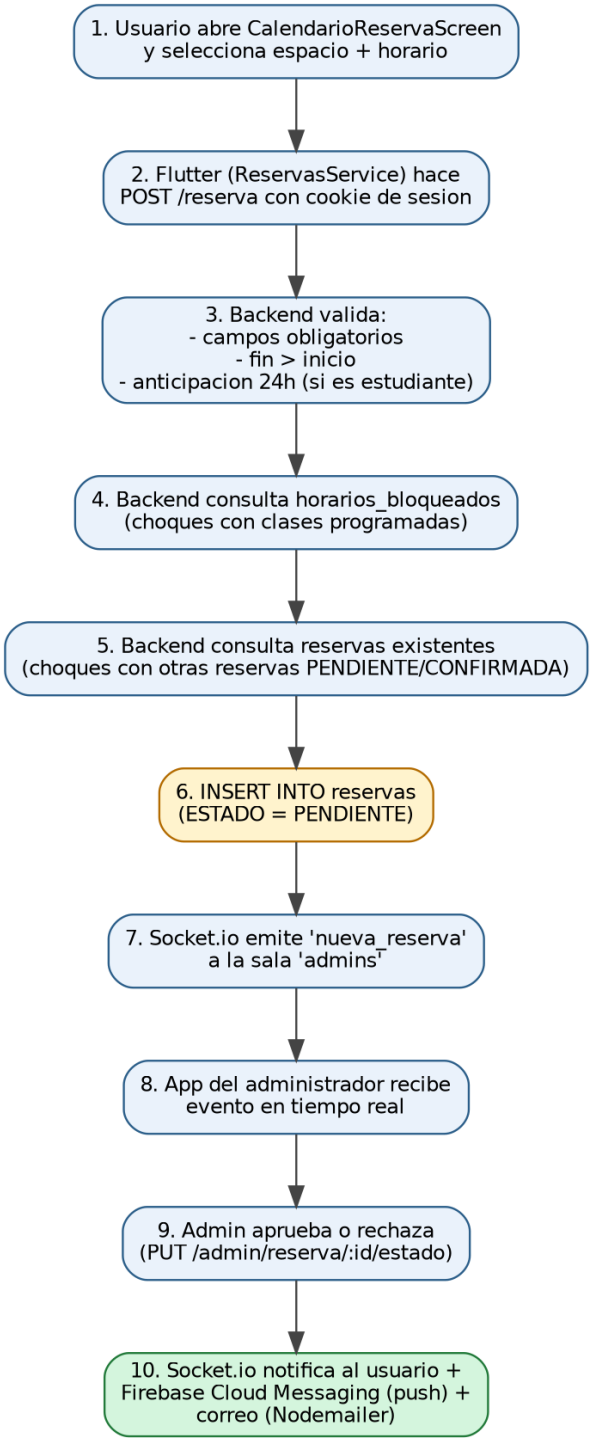


Figura 5 Diagrama de flujo/secuencia del proceso de creación y aprobación de una reserva

Este diagrama describe la secuencia completa desde que el usuario selecciona un horario en el calendario hasta que recibe la notificación del resultado de su solicitud, incluyendo las dos

validaciones de conflicto que realiza el backend (bloqueos por clase programada y solapamiento con otras reservas) y los tres canales de notificación que se activan una vez que el administrador resuelve la solicitud.

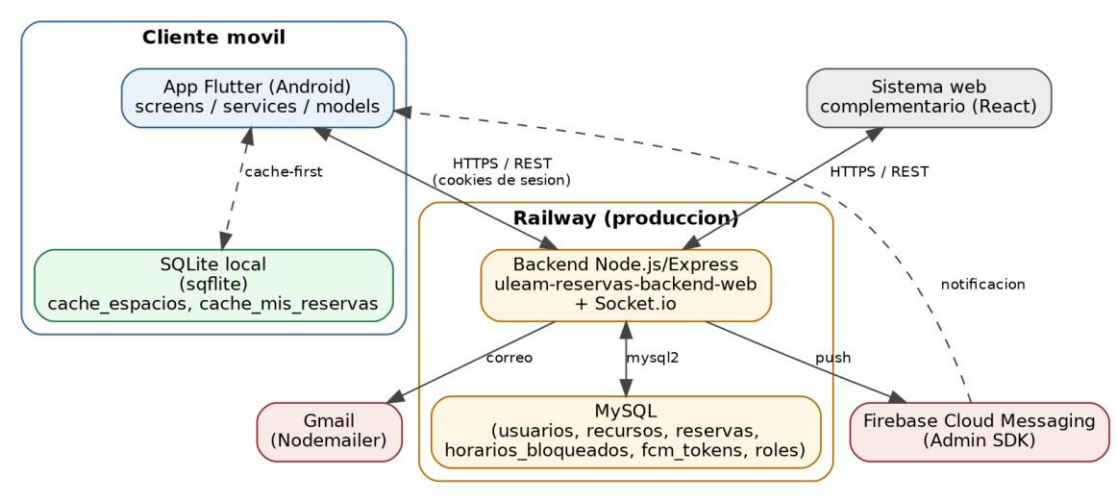


Figura 6 Diagrama de arquitectura general del sistema

El sistema sigue una arquitectura cliente-servidor en tres capas: el cliente móvil desarrollado en Flutter, el cual dispone de una copia local en SQLite para operar sin conexión a internet; el backend implementado con Node.js/Express + Socket.io), alojado en Railway junto con su base de datos en MySQL; y los servicios externos integrando servicios como Firebase Cloud Messaging para notificaciones push y Gmail vía Nodemailer para correo. Ademas, el backend es utilizado tanto por la aplicación móvil como por el sistema web complementario del proyecto institucional. Por esta razón, el diseño y las modificaciones relacionadas con el esquema de la base de datos deben ser coordinados entre los equipos responsables de ambas plataformas.

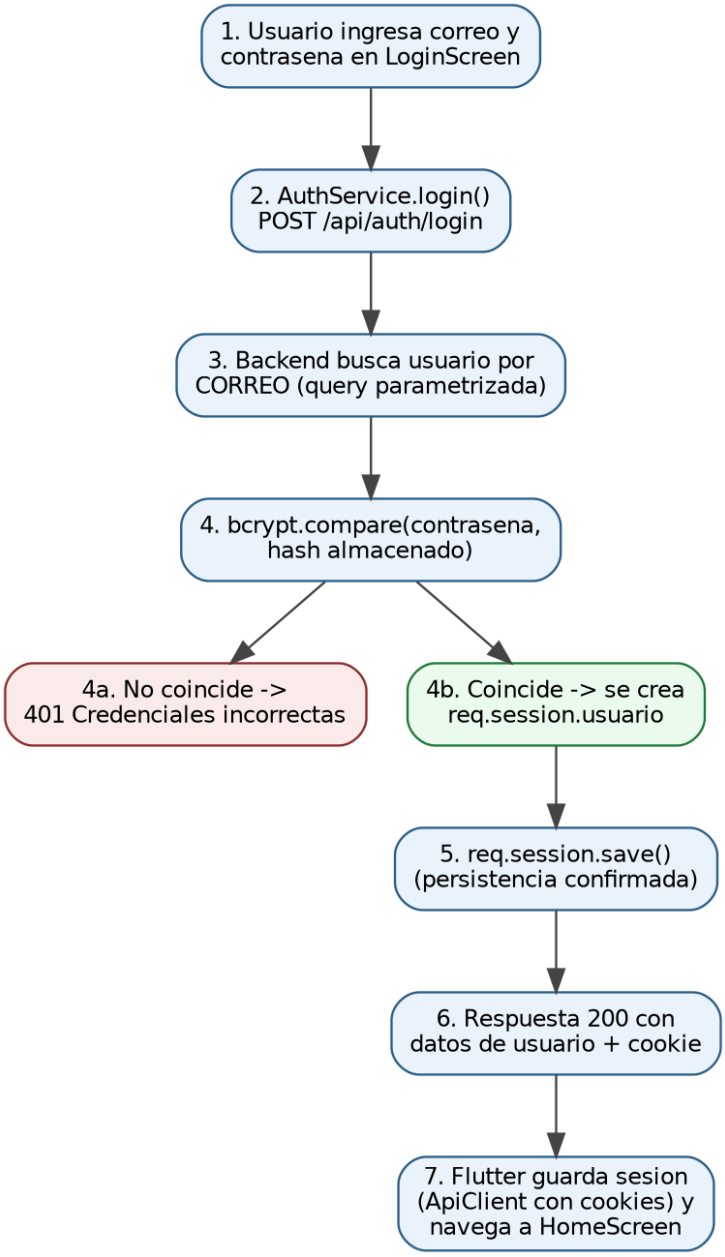


Figura 7 Diagrama de secuencia — inicio de sesión

El proceso de autenticación establece la comunicación entre la pantalla de login, el servicio de autenticación y el backend. La validación de las credenciales se realiza en dos etapas: primero se busca al usuario mediante su correo electrónico y, posteriormente, se compara la contraseña ingresada con el hash almacenado utilizando bcrypt. Una vez verificadas las credenciales, la sesión se guarda en el store de MySQL y solo se considera válida para el cliente cuando

`req.session.save()` confirma que la información fue persistida correctamente. De esta manera, se evita el problema de sesiones inexistentes o no disponibles durante el proceso de autenticación.

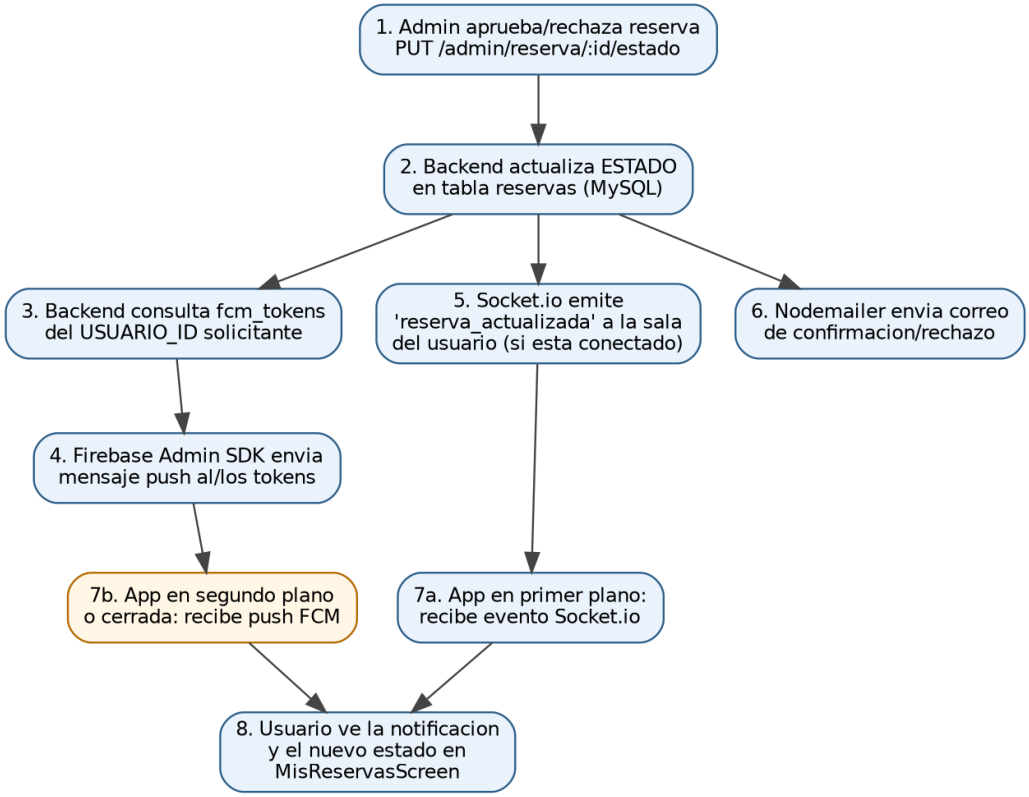


Figura 8 Diagrama de secuencia — notificación tras aprobar o rechazar una reserva

El Sistema utiliza tres canales de notificación de manera paralela para garantizar que los mensajes llegen al usuario en diferentes situaciones. Socket.io permite recibir eventos cuando la aplicación esta abierta o en segundo plano y mantiene una conexión activa. Firebase Cloud Messaging se encarga de enviar notificaciones cuando la aplicación está cerrada, mientras que el correo electrónico funciona como un medio de respaldo que permite recibir la información independientemente del estado de la aplicación.

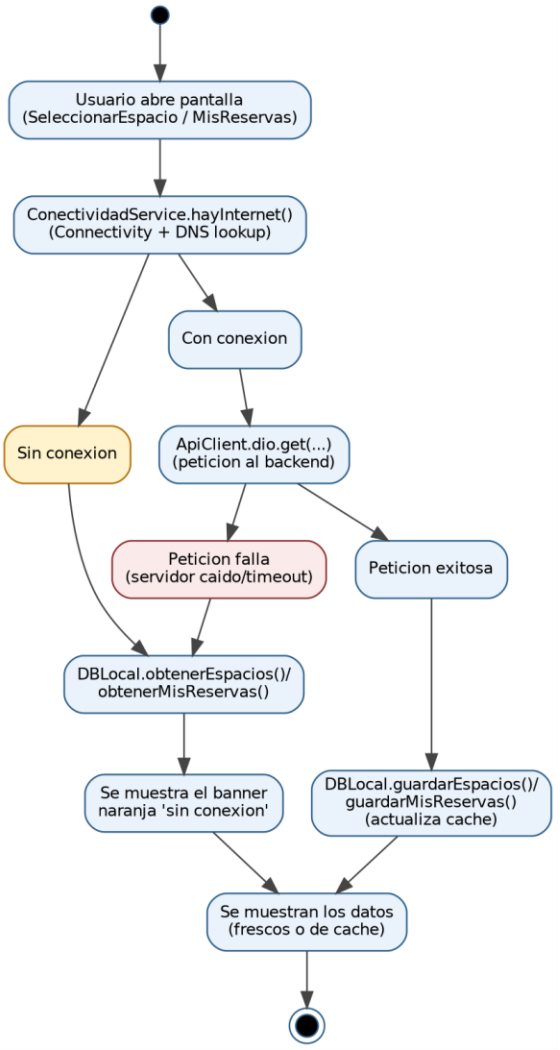


Figura 9 Diagrama de actividades — patrón cache-first en modo offline

Las funciones getEspaciosDisponibles() y getMisReservas() comprueban la conectividad mediante una resolución DNS real, no solo verificando el estado de la interfaz de red. Si la solicitud al servidor falla debido a una caída del servicio, aunque exista conexión a Internet, el sistema aplica el mismo flujo utilizado en el modo sin conexión y recupera los datos almacenados en la caché local de SQLite.

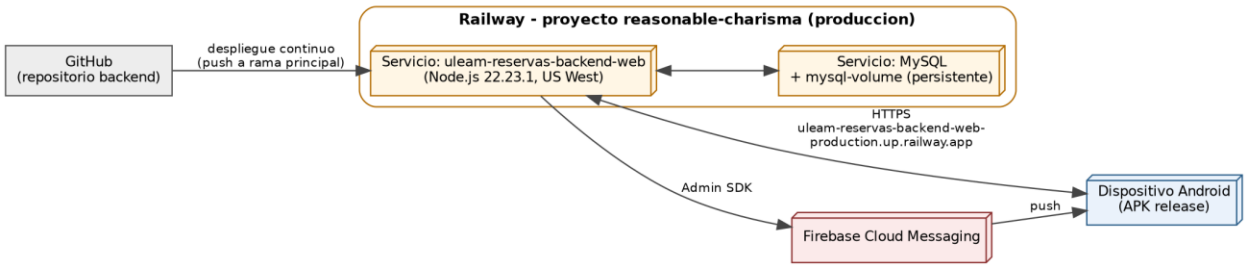


Figura 10 Diagrama de despliegue

El backend se despliega automáticamente en Railway, desde el repositorio de GitHub. Cada actualización de la rama principal se publica de forma continua en el servicio uleam-reservas-backend-web, que corre sobre Node.js 22.23.1 en a región US West. La base de datos MySQL funciona como un servicio independiente dentro del mismo proyecto de Railway, y cuenta con un volumen persistente que da garantía a la conservación de la información almacenada.

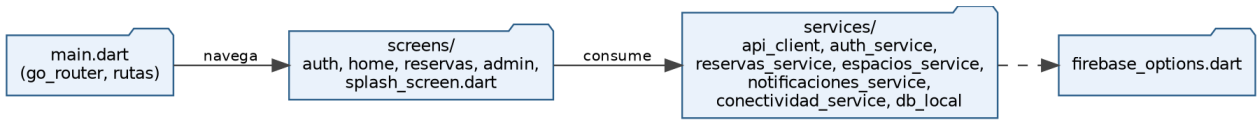


Figura 11 Diagrama de paquetes del cliente Flutter

El código del cliente se organiza en tres paquetes, cada uno con una responsabilidad claramente definida. El archivo main.dart utiliza go_router para configurar el enrutamiento. El paquete screens/ organiza las pantallas por dominio: autenticación, inicio, reservas y administración. El paquete services/ gestiona de forma centralizada la comunicación con el backend y el estado local. Por lo tanto, ninguna pantalla interactúa directamente con God o SQLite; todas estas operaciones se gestionan a través de los servicios correspondientes. Esta arquitectura

facilita la modificación y el mantenimiento del sistema: por ejemplo, puede cambiar la URL base del backend desde api_client.dart sin modificar ninguna de las pantallas.

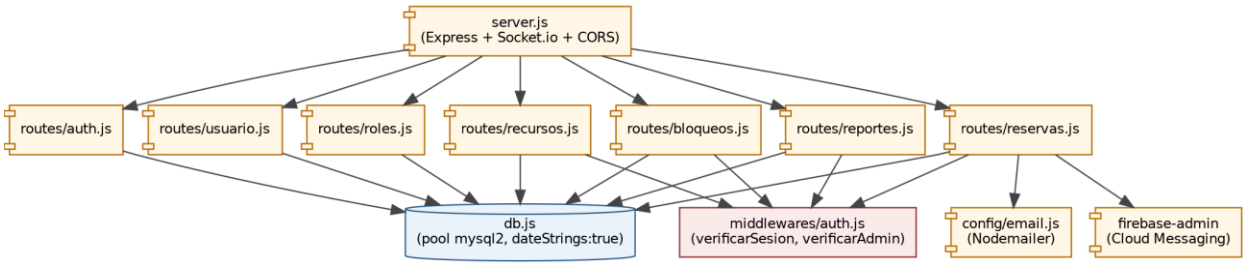


Figura 12 Diagrama de componentes del backend

El servidor Express centraliza la configuración y registro de siete módulos de rutas. De estos, cuatro —`recursos`, `reservas`, `bloqueos` y `reportes`— utilizan el middleware de sesión y control de roles definido en `middlewares/auth.js`, con el objetivo de limitar el acceso a las operaciones administrativas. Todos los módulos utilizan el mismo pool de conexiones establecido en `db.js`, incluyendo la configuración `dateStrings` mencionada en la función 7. Por otro lado, únicamente el módulo de `reservas` interactúa con servicios externos como el envío de correos mediante Nodemailer y las notificaciones push a través de Firebase Admin SDK, debido a que es el único componente del sistema en el que se produce un cambio de estado que requiere informar al usuario.

4.4.2.2 Diseño de la base de datos

La base de datos del sistema se implementó en MySQL y esta compuesta por cinco tablas principales. Usuarios, recursos, reservas, horarios_bloqueados y fcm_tokens. La tabla reservas mantiene una relación de muchos a uno con usuarios y con recursos; la tabla horarios_bloqueados mantiene una relación de muchos a uno con recursos; y la tabla fcm_tokens mantiene una relación

de muchos a uno con usuarios, permitiendo que un mismo usuario reciba notificaciones en más de un dispositivo.

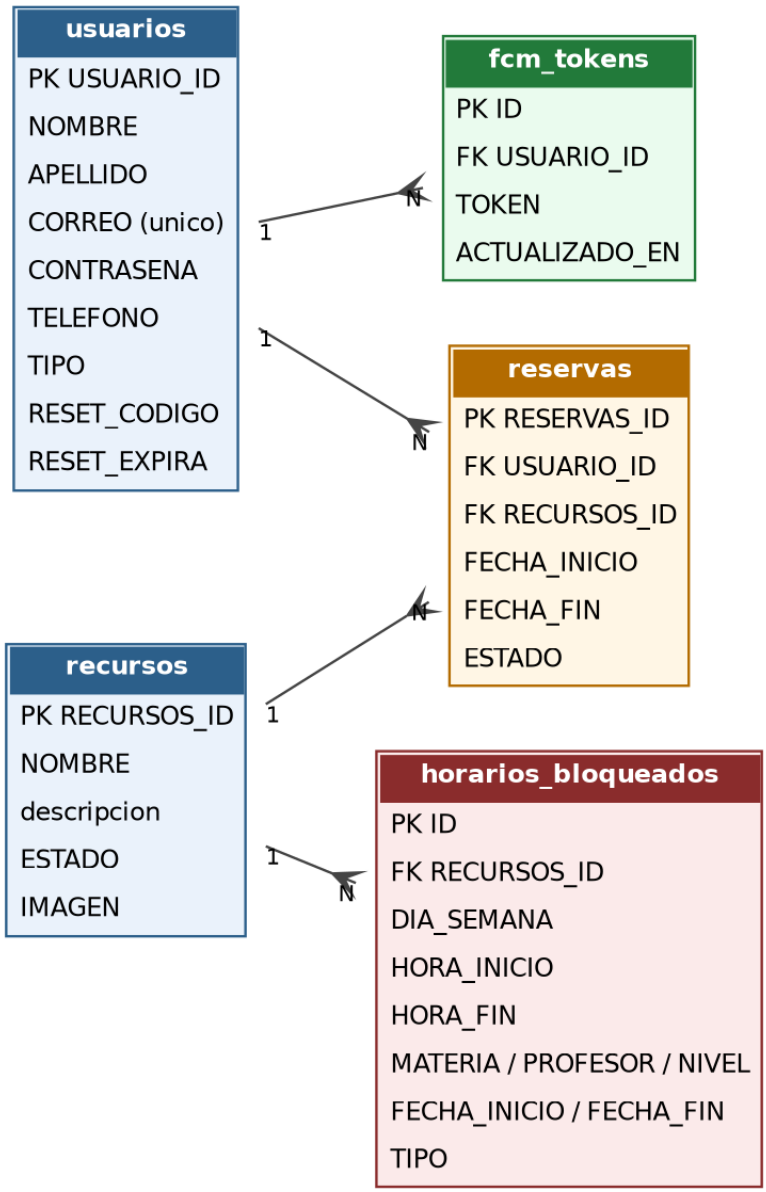


Figura 13 Diagrama entidad-relación de la base de datos

Tabla	Descripción
usuarios	Almacena los datos de autenticación y perfil de cada usuario: nombre, apellido, correo institucional (único), contraseña cifrada, teléfono, tipo de

Tabla	Descripción
	usuario (ESTUDIANTE, PROFESOR o ADMINISTRADOR) y los campos de recuperación de contraseña (código y expiración).
recursos	Registra los espacios físicos disponibles para reserva: nombre, descripción, estado (disponible, ocupado o mantenimiento) e imagen representativa.
reservas	Gestiona las solicitudes de reserva: usuario solicitante, recurso solicitado, fecha y hora de inicio y fin, y estado (pendiente, confirmada o cancelada).
horarios_bloqueados	Almacena los bloqueos de horario asociados a un recurso, ya sea por clases programadas (con materia, profesor y nivel) o por mantenimiento (con rango de fechas).
fcm_tokens	Guarda tokens de Firebase Cloud Messaging registrados por cada dispositivo de un usuario, utilizados para el envío de notificaciones push.

Tabla 9 Descripción de las tablas de la base de datos

4.4.3 Análisis de requerimiento

4.4.3.1 Requisitos del software

Software	Versión utilizada	Función
Flutter SDK	3.44.2	Desarrollo de la interfaz y lógica de la aplicación móvil (Android).
Node.js	22.11.0	Entorno de ejecución del servidor backend.
Express	5.x	Framework para la definición de la API REST del backend.
MySQL	8.x	Motor de base de datos relacional del sistema.
Firebase (Admin SDK / FCM)	—	Envío y recepción de notificaciones push.
sqflite	^2.3.0	Base de datos local SQLite para el modo offline-first.
Android SDK	API 21+	Compilación y ejecución de la aplicación en dispositivos Android 5.0 o superior.

Tabla 10 Requisitos de software

4.4.3.2 Requerimientos de hardware y software

Componente	Requisito mínimo
Dispositivo móvil	Android 5.0 (API 21) o superior, con acceso a Google Play Services para notificaciones push.
Conexión a internet	Requerida para sincronizar datos y crear/aprobar reservas; no requerida para consultar datos ya almacenados en cache local.

Componente	Requisito mínimo
Servidor backend	Node.js 18 o superior, motor MySQL 8.x, memoria y almacenamiento acordes al volumen de uso institucional.
Equipo de desarrollo	Procesador de 4 núcleos o superior, 8 GB de RAM como mínimo (16 GB recomendado para el emulador Android), 20 GB de espacio libre.

Tabla 11 Requerimientos de hardware y software

4.4.3.3 Implementación

4.4.3.3.1 Tipo de programación

El sistema se desarrollo bajo una arquitectura cliente-servidor: la aplicación móvil (cliente) desarrollada en Flutter consuma una API REST expuesta por el backend Node.js/Express (servidor), complementada con un canal de comunicación bidireccional mediante Socket.io para los eventos en tiempo real. Tanto el cliente como el servidor se estructuran siguiendo un paradigma de programación orientada a objetos combinada con una organización modular de tipo estructurada: el backend separa rutas, middlewares y configuración en archivos independientes por responsabilidad, y el cliente Flutter separa pantallas (screens), servicios (services) y modelos (models).

4.4.3.3.2 Lenguajes de programación

Lenguaje	Uso
Dart	Lenguaje del frontend móvil (Flutter): construcción de la interfaz, lógica de navegación y consumo de la API.
JavaScript (Node.js)	Lenguaje del backend: definición de rutas REST, middlewares de autenticación y lógica de negocio.
SQL	Lenguaje de consulta utilizado para la definición y manipulación de las tablas en MySQL.

Tabla 12 Lenguajes de programación utilizados

4.4.3.3.3 Herramientas de desarrollo

- Visual Studio Code, como editor principal del backend.
- Android Studio, para gestionar el SDK de Android, los emuladores y la depuración.
- MySQL (administrado mediante consola/cliente SQL), para diseñar y consultar la base de datos.
- Postman y Thunder Client, para probar manualmente los endpoints antes de integrarlos con Flutter.
- Firebase Console y FlutterFire CLI, para configurar el proyecto de notificaciones push.
- Git y GitHub, para el control de versiones y el respaldo del codigo fuente.
- Dispositivos físicos Android para pruebas en condiciones reales.

4.4.3.3.4 Métodos

En el cliente Flutter, la comunicación con el backend y el manejo del estado local se encapsulan en clases de servicio, cada una responsable de un módulo del dominio:

Servicio (Flutter)	Métodos principales
AuthService	login(), registro(), getToken()/getUsuario(), logout(), isLoggedIn()
EspaciosService	getEspacios(), getEspacio(id), crearEspacio(), editarEspacio(), eliminarEspacio(), cambiarEstado(), agregarBloqueoHorario(), agregarBloqueoFecha(), eliminarBloqueoHorario(), eliminarBloqueoFecha()
ReservasService	getEspaciosDisponibles(), getDisponibilidad(id, fecha), crearReserva(), getMisReservas(), cancelarReserva(), aprobarReserva()/rechazarReserva()
NotificacionesService	solicitarPermisos(), registrarToken(), configurarListeners(), actualización automática vía onTokenRefresh

Servicio (Flutter)	Métodos principales
ConectividadService	hayInternet() — doble verificación: estado de interfaz + resolución DNS con timeout de 3 s
DbLocal (cache SQLite)	guardarEspacios()/obtenerEspacios(), guardarMisReservas()/obtenerMisReservas() — operaciones batch con estrategia de conflicto REPLACE

Tabla 13 Métodos principales de los servicios del cliente Flutter

Los métodos getEspaciosDisponibles() y getMisReservas()implementan el patrón cache-first con actualización en línea: con conexión, consultan el backend y actualizan el cache local; sin conexión, devuelven directamente el cache; y si hay conexión pero el servidor no responde, el bloque try-catch captura el error y recurre igualmente al cache como respaldo. Las operaciones de escritura(crear, cancelar, aprobar o rechazar una reserva) no cuentan con soporte offline, ya que requieren confirmación del servidor para garantizar la integridad de los datos.

4.4.4 Roles y Responsabilidades

Scrum establece responsabilidades diferenciadas para priorizar el producto, facilitar el proceso y ejecutar el trabajo técnico. En este proyecto académico, una misma persona asumió más de una responsabilidad, manteniendo la separación funcional de los roles.

Rol Scrum	Responsable	Responsabilidades en el proyecto
Product Owner	Rey Josias Miranda Llerena (autor)	Priorizar las historias del Product Backlog, definir el valor esperado de las funcionalidades y validar los incrementos obtenidos.
Scrum Master	Tutor del proyecto	Orientar la aplicación de Scrum, facilitar el seguimiento académico, apoyar la resolución de impedimentos y revisar los avances del proyecto.
Developer	Rey Josias Miranda Llerena (autor)	Diseñar, programar, integrar, probar y documentar las funcionalidades comprometidas en cada sprint.

Tabla 14 Roles y responsabilidades

4.4.5 Diseño de la interfaz

4.4.5.1 Colores

La identidad de la aplicación se construyó sobre una paleta de colores institucional, reforzada con un código de colores funcional que permite al usuario interpretar el estado del sistema de un solo vistazo:

Color	Uso en la interfaz
Azul corporativo	Fondo de la pantalla de bienvenida (splash) y de la pantalla de inicio de sesión; color principal de la marca institucional.
Verde	Indicador de espacio DISPONIBLE y de reserva CONFIRMADA.
Naranja	Indicador de espacio OCUPADO, banner de modo sin conexión (offline) y color de acento en los correos de notificación.
Rojo institucional (#CC0000)	Indicador de espacio en MANTENIMIENTO, de reserva CANCELADA/RECHAZADA y color de marca en las plantillas de correo.

Tabla 15 Paleta de colores funcional de la interfaz

4.4.5.2 Diseño de pantallas

Las principales pantallas de la aplicación, desarrolladas siguiendo el patrón de arquitectura por capas de Flutter (screens, services y models), se describen a continuación:

Pantalla	Descripción
SplashScreen	Pantalla de bienvenida con el logo institucional sobre fondo azul corporativo (400 ms); verifica si existe una sesión guardada y redirige automáticamente a Home o a Login.
LoginScreen	Interfaz de autenticación que permite el de inicio de sesión, con validación de campos, indicador de carga durante el proceso y visualización del logo institucional.
RegistroScreen	Interfaz de registro que permite el registro de nuevos usuarios: nombre, apellido, correo institucional, contraseña, teléfono, y selección de rol (estudiante o profesor), incorporando validaciones para garantizar la integridad de la información.

Pantalla	Descripción
HomeScreen	Pantalla principal, con el nombre y rol del usuario autenticado, accesos a Aulas, Laboratorios, Canchas y Mis Reservas, opciones administrativas visibles solo para el rol Administrador.
SeleccionarEspacioScreen	Lista de espacios filtrable por tipo, con un banner naranja cuando la app opera sin conexión (los datos vienen del cache local).
CalendarioReservaScreen	Calendario de disponibilidad (table_calendar), con niveles de ocupación por color y un formulario modal para crear la solicitud de reserva.
MisReservasScreen	Historial de reservas del usuario, con filtros por estado y banner de modo offline cuando corresponde.
AprobarReservasScreen (admin)	Panel administrativo con el listado de reservas pendientes y acciones de aprobar o rechazar.
EspaciosScreen / FormularioEspacio (admin)	Listado de espacios en tarjetas con badge de estado por color y formulario tipo ModalBottomSheet para crear o editar un espacio.
BloqueosScreen (admin)	Pantalla con pestañas para administrar bloqueos por horario fijo y bloqueos por rango de fechas.

Tabla 16 Descripción de las principales pantallas de la aplicación

A continuación se presentan capturas de pantalla reales de la aplicación, correspondientes a las pantallas descritas en la tabla anterior.

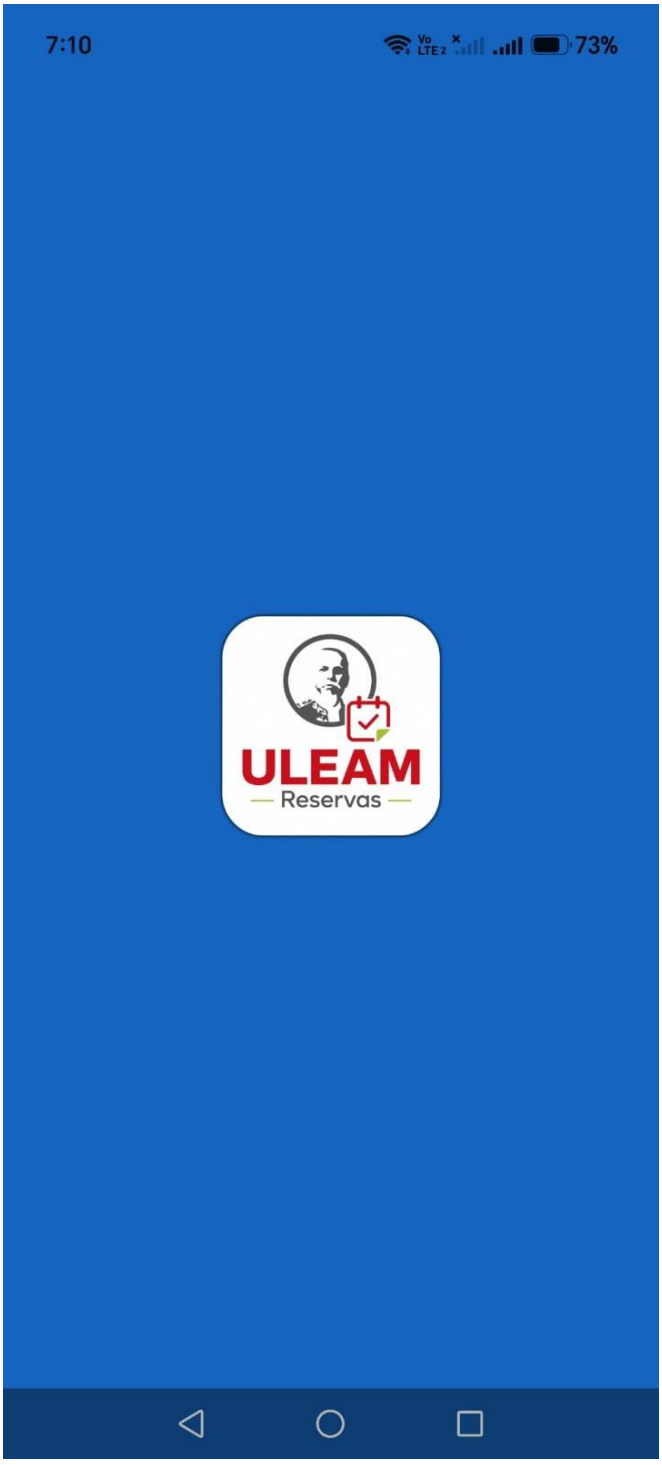


Figura 14 SplashScreen — pantalla de bienvenida

Scaffold con el logo centrado sobre el azul corporativo durante una ventana fija de 400 ms. Mientras se muestra, AuthService.isLoggedIn() revisa si hay una sesión guardada; según el

resultado, go_router redirige a HomeScreen o a LoginScreen sin dejar ver una pantalla en blanco de por medio.

Figura 15 LoginScreen — inicio de sesión

Los dos campos de texto (correo electrónico y contraseña, con un botón de icono para mostrar u ocultar la contraseña) se validan en el lado del cliente antes de enviarlos. Al pulsar la tecla "Intro" se activa la función AuthService.login(), que envía una solicitud POST a /api/auth/login. Si se ha creado una sesión en el servidor, el cliente HTTP almacena una cookie para usarla en solicitudes posteriores. Tras iniciar sesión correctamente, la pantalla llama inmediatamente a las funciones NotificationsService.requestPermissions() y registerToken() para permitir que el dispositivo reciba notificaciones push.

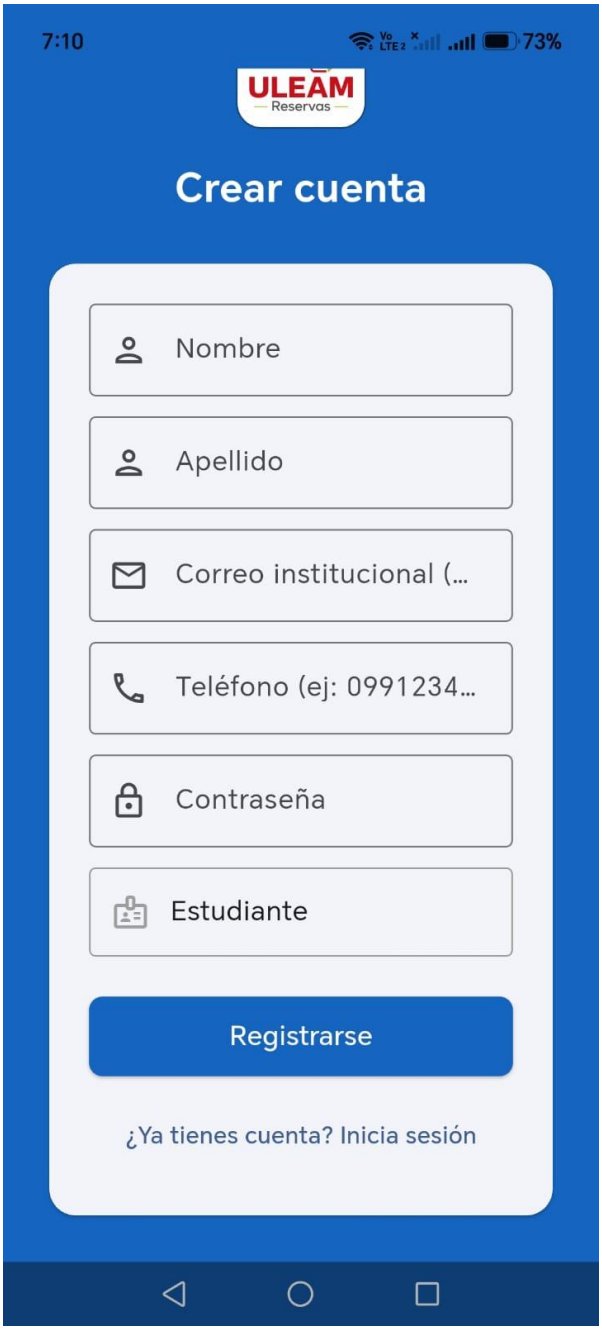


Figura 16 RegistroScreen — creación de cuenta

Seis campos, incluido el Dropdown de rol (Estudiante/Profesor). El teléfono ya restringe el formato de 10 dígitos iniciando en 0 desde el propio TextFormField, aunque la validación fuerte —incluyendo el dominio institucional @uleam.edu.ec— vuelve a aplicarse en el servidor, en /api/usuario/registro, por si el cliente fue manipulado.

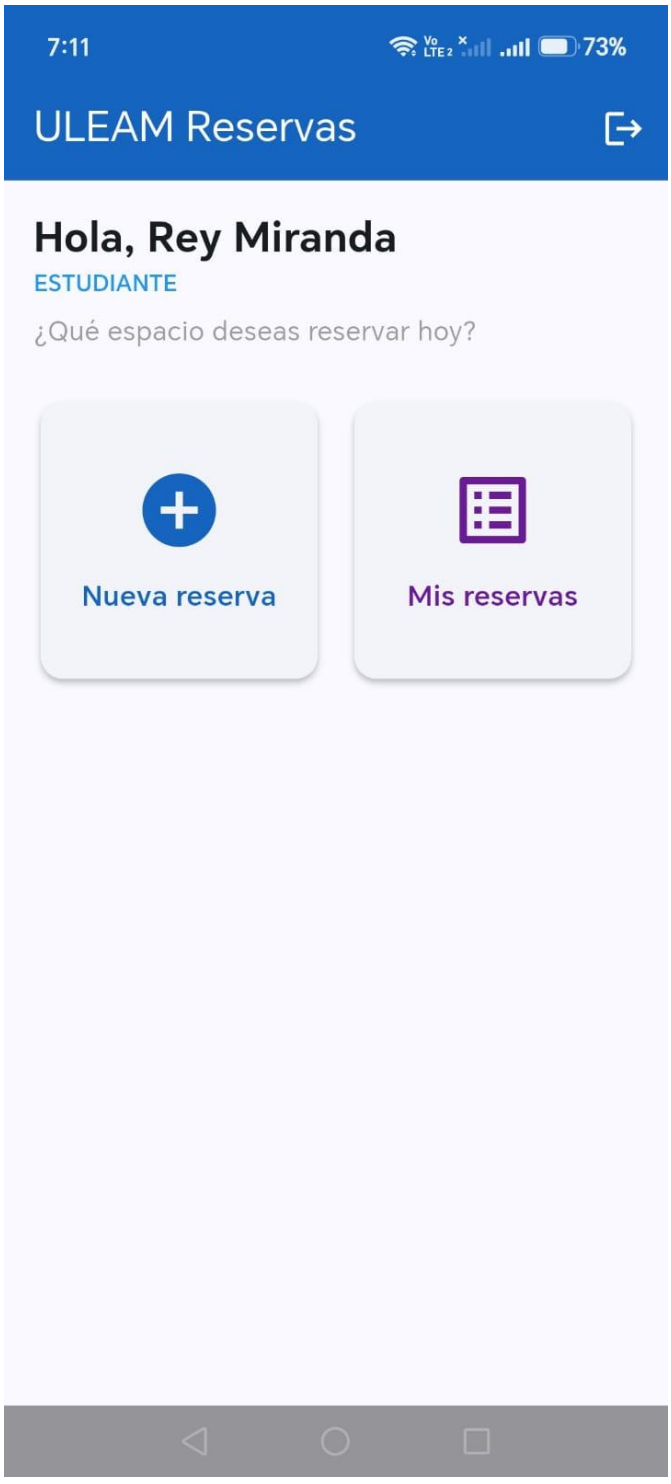


Figura 17 HomeScreen — pantalla principal del usuario

Las tarjetas de acceso rápido (Nueva reserva, Mis reservas) se generan según el TIPO de usuario guardado en sesión; con rol ESTUDIANTE, como en esta captura, no aparece ninguna

opción administrativa, a diferencia de la vista que obtiene un usuario con TIPO=ADMINISTRADOR.

Figura 18 CalendarioReservaScreen — disponibilidad del espacio, con bloqueos de clases programadas visibles.

table_calendar representa como un punto aquellos días que tienen reservas o bien clases programadas previamente. Al seleccionar el 20/7/2026, por ejemplo, la lista inferior consulta la

disponibilidad del Aula 208 y se mostrara tres bloques ocupados (iconos de prohibido), que vienen de horarios_bloqueados y no de la tabla reservas. Esto permite diferenciar un horario que ha sido bloqueado por una clase programada de una reserva confirmada.

Figura 19 MisReservasScreen — historial de reservas con filtros por estado

Los chips superiores (Todas, Pendientes, Confirmadas...) filtran la lista obtenida de GET /mis-reservas. El color del borde de cada tarjeta depende del campo ESTADO devuelto por el backend: verde para CONFIRMADA, rojo para CANCELADA, sin necesidad de lógica adicional en el cliente.

Figura 20 SeleccionarEspacioScreen en modo sin conexión, mostrando el banner naranja y los datos del cache local

Esta captura corresponde al patrón cache-first en acción: al no poder resolver DNS, getEspaciosDisponibles() no lanza una excepción visible al usuario, sino que sirve la última copia guardada en SQLite. El banner naranja (Container con Icons.wifi_off) se dibuja de forma condicional cuando ConectividadService.hayInternet() devuelve false, y advierte que no se podrán crear reservas hasta reconectarse.

Figura 21 EspaciosScreen (administrador) — gestión de espacios físicos

Cada tarjeta expone tres acciones mapeadas directamente a los endpoints administrativos: Editar (PUT /api/admin/espacios/:id), Estado (PATCH .../estado) y Eliminar (DELETE). El botón flotante "+ Nuevo espacio" abre el ModalBottomSheet de creación descrito en la sección de implementación.

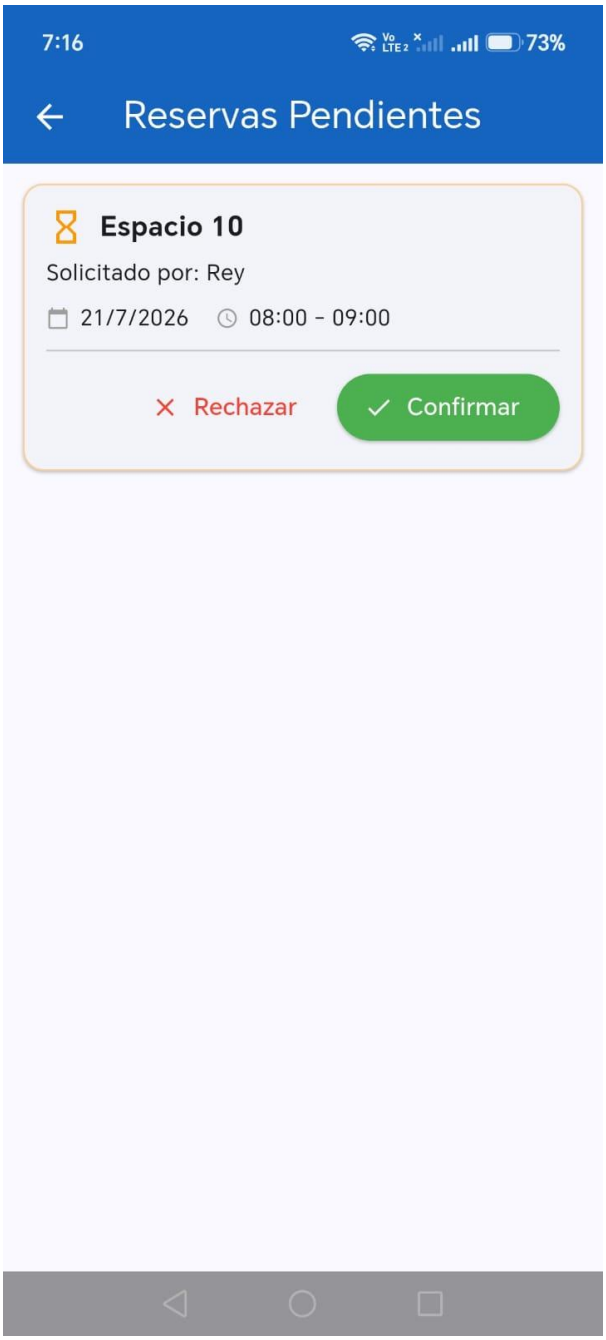


Figura 22 AprobarReservasScreen (administrador) — aprobación/rechazo de una solicitud

En este caso hay una única solicitud pendiente (Espacio 10, solicitada por Rey). Rechazar y Confirmar apuntan al mismo endpoint, PUT /admin/reserva/:id/estado, cambiando únicamente el valor de ESTADO enviado; ese cambio es el que dispara, del lado del servidor, el evento de Socket.io y el envío de la notificación push descritos en la función 4 de este capítulo.

Como evidencia adicional del funcionamiento del sistema de notificaciones descrito en la sección de requerimientos, se presentan las notificaciones push reales recibidas en los dos dispositivos utilizados durante las pruebas:

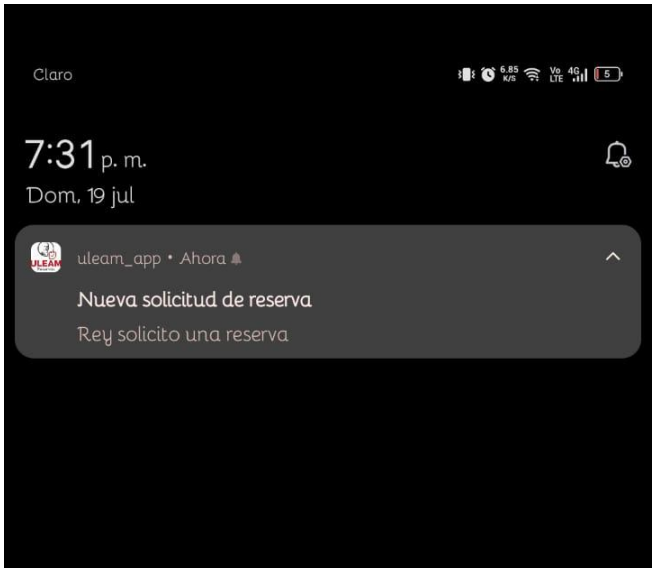


Figura 23 Notificación push recibida por el administrador ante una nueva solicitud de reserva

Notificación de sistema generada por el manejador de background de Firebase Cloud Messaging (visible con la app cerrada, no un simple SnackBar en primer plano). El payload "Rey solicito una reserva" corresponde al mismo evento que en paralelo emite Socket.io hacia la sala de administradores; FCM cubre el caso en que la app no está abierta en ese momento.

Figura 24 Notificación push recibida por el estudiante al ser aprobada su reserva

Notificación equivalente del lado del solicitante, disparada tras el PUT /admin/reserva/:id/estado con ESTADO=CONFIRMADA. Junto con el evento de Socket.io y el correo enviado por Nodemailer, completa los tres canales descritos en el requerimiento RF12.

4.4.5.3 Códigos fuente de las principales funciones

A continuación, se documentan siete funciones representativas del sistema, correspondientes a los módulos de autenticación, reservas, gestión de recursos, bloqueos y corrección de un defecto detectado durante la integración.

4.4.5.3.1 Inicio de sesión con verificación de credenciales (backend, routes/auth.js)

La contraseña nunca se compara dentro de la consulta SQL: primero se trae el registro por correo y recién después bcrypt.compare() revisa el hash. El req.session.save() al final no es decorativo — sin ese callback explícito, express-session puede responder al cliente con la cookie antes de que la sesión termine de escribirse en el store de MySQL, y la siguiente petición del

usuario llega sin sesión válida del otro lado. Costó encontrar ese bug la primera vez, porque el login "funcionaba" pero la petición inmediata después fallaba con 401.

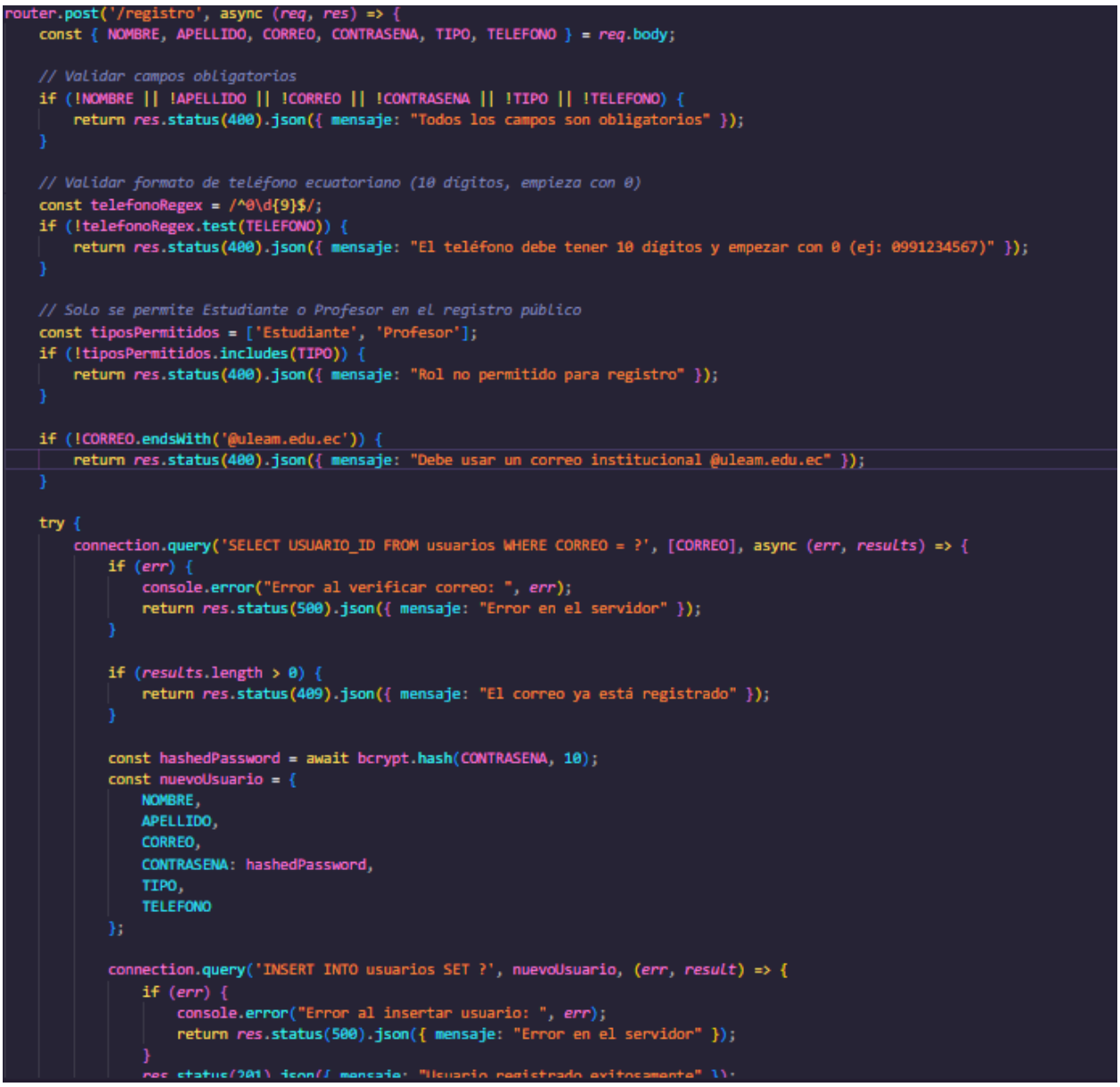
```
router.post('/login', async (req, res) => {
    const { CORREO, CONTRASENA } = req.body;

    try {
        connection.query(
            "SELECT USUARIO_ID, NOMBRE, APELLIDO, CORREO, TIPO, CONTRASENA FROM usuarios WHERE CORREO = ?",
            [CORREO],
            async (err, results) => {
                if (err || results.length === 0) {
                    return res.status(401).json({ mensaje: "Credenciales incorrectas" });
                }

                const usuario = results[0];
                const passwordValida = await bcrypt.compare(CONTRASENA, usuario.CONTRASENA);

                if (!passwordValida) {
                    return res.status(401).json({ mensaje: "Credenciales incorrectas" });
                }

                // Guardar en sesión (sin ROL_ID porque no existe en tu DB)
                req.session.usuario = {
                    id: usuario.USUARIO_ID,
                    nombre: usuario.NOMBRE,
                    apellido: usuario.APELLIDO,
                    correo: usuario.CORREO,
                    tipo: usuario.TIPO  // ← Esto es lo que usa el middleware
                };

                req.session.save((err) => {
                    if (err) {
                        console.error("Error al guardar sesión:", err);
                        return res.status(500).json({ mensaje: "Error al iniciar sesión" });
                    }

                    res.json({
                        mensaje: "Login exitoso",
                        usuario: req.session.usuario
                    });
                });
            }
        );
```

Figura 25 Inicio de sesión con verificación de credenciales

4.4.5.3.2 Registro de usuario con validación de correo institucional (backend, routes/usuario.js)

Las validaciones se hicieron en cascada, cortando en el primer error en vez de acumularlos todos y devolver una lista — es más simple de mantener, aunque el usuario tenga que corregir el formulario campo por campo si comete más de un error a la vez. El regex del teléfono (0 seguido

de 9 dígitos) fue un pedido puntual de la Secretaría de la extensión, para poder contactar al solicitante por WhatsApp si hace falta coordinar algo sobre la reserva.



```
router.post('/registro', async (req, res) => {
    const { NOMBRE, APELLIDO, CORREO, CONTRASENA, TIPO, TELEFONO } = req.body;

    // Validar campos obligatorios
    if (!NOMBRE || !APELLIDO || !CORREO || !CONTRASENA || !TIPO || !TELEFONO) {
        return res.status(400).json({ mensaje: "Todos los campos son obligatorios" });
    }

    // Validar formato de teléfono ecuatoriano (10 dígitos, empieza con 0)
    const telefonoRegex = /^0\d{9}$/;
    if (!telefonoRegex.test(TELEFONO)) {
        return res.status(400).json({ mensaje: "El teléfono debe tener 10 dígitos y empezar con 0 (ej: 0991234567)" });
    }

    // Solo se permite Estudiante o Profesor en el registro público
    const tiposPermitidos = ['Estudiante', 'Profesor'];
    if (!tiposPermitidos.includes(TIPO)) {
        return res.status(400).json({ mensaje: "Rol no permitido para registro" });
    }

    if (!CORREO.endsWith('@uleam.edu.ec')) {
        return res.status(400).json({ mensaje: "Debe usar un correo institucional @uleam.edu.ec" });
    }

    try {
        connection.query('SELECT USUARIO_ID FROM usuarios WHERE CORREO = ?', [CORREO], async (err, results) => {
            if (err) {
                console.error("Error al verificar correo: ", err);
                return res.status(500).json({ mensaje: "Error en el servidor" });
            }

            if (results.length > 0) {
                return res.status(409).json({ mensaje: "El correo ya está registrado" });
            }

            const hashedPassword = await bcrypt.hash(CONTRASENA, 10);
            const nuevoUsuario = {
                NOMBRE,
                APELLIDO,
                CORREO,
                CONTRASENA: hashedPassword,
                TIPO,
                TELEFONO
            };

            connection.query('INSERT INTO usuarios SET ?', nuevoUsuario, (err, result) => {
                if (err) {
                    console.error("Error al insertar usuario: ", err);
                    return res.status(500).json({ mensaje: "Error en el servidor" });
                }
                res.status(201).json({ mensaje: "Usuario registrado exitosamente" });
```

Figura 26 Registro de usuario con validación de correo institucional

4.4.5.3.3 Validación de conflictos al crear una reserva (backend, routes/reservas.js)

Van tres consultas anidadas en vez de una sola con JOIN, a propósito: apenas se detecta un choque se corta y se responde con el mensaje específico (con qué clase o con qué reserva), sin gastar la segunda consulta si la primera ya rechazó. Es menos elegante que resolverlo todo en una

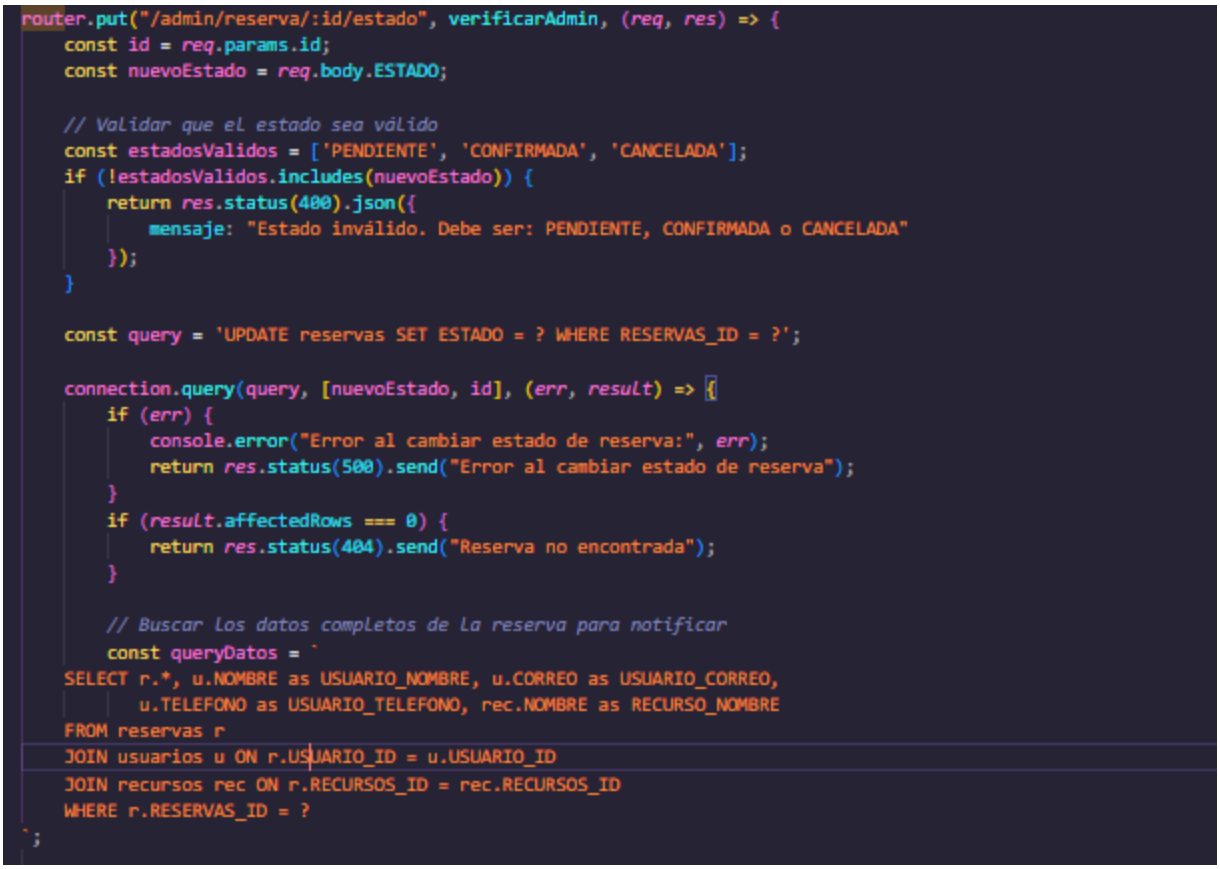
sola sentencia SQL, pero con el volumen de reservas que maneja un solo edificio no se notó ningún problema de rendimiento, y a cambio el mensaje de error que recibe el usuario es más útil.

```
// VALIDACIÓN DE ANTICIPACIÓN SEGÚN ROL
const ahora = new Date();
const horasDeAnticipacion = (fechaInicio - ahora) / (1000 * 60 * 60);
const esEstudiante = req.session.usuario.tipo === 'ESTUDIANTE';

if (esEstudiante && horasDeAnticipacion < 24) {
    return res.status(400).json({
        mensaje: '❌ Los estudiantes deben reservar con al menos 24 horas de anticipación'
    });
}

// 🔴 VALIDACIÓN DE HORARIOS BLOQUEADOS (CLASES)
const diasSemana = ['Domingo', 'Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sabado'];
const diaSemana = diasSemana[fechaInicio.getDay()];
const fechaSolo = fechaInicio.toISOString().split('T')[0];
// Obtener horarios bloqueados
const queryBloqueados = `
    SELECT HORA_INICIO, HORA_FIN, MATERIA
    FROM horarios_bloqueados
    WHERE RECURSOS_ID = ?
        AND DIA_SEMANA = ?
        AND ? BETWEEN FECHA_INICIO AND FECHA_FIN
`;
connection.query(queryBloqueados, [RECURSOS_ID, diaSemana, fechaSolo], (err, bloqueados) => {
    if (err) {
        console.error("Error al verificar horarios bloqueados:", err);
        return res.status(500).json({ mensaje: "Error al validar disponibilidad" });
    }
    // Convertir horas de la reserva a formato TIME
    const horaInicioReserva = fechaInicio.toTimeString().substring(0, 8);
    const horaFinReserva = fechaFin.toTimeString().substring(0, 8);
    // Verificar solapamiento con clases
    for (const bloqueado of bloqueados) {
        const horaBloqueadoInicio = bloqueado.HORA_INICIO;
        const horaBloqueadoFin = bloqueado.HORA_FIN;
        if (
            (horaInicioReserva >= horaBloqueadoInicio && horaInicioReserva < horaBloqueadoFin) ||
            (horaFinReserva > horaBloqueadoInicio && horaFinReserva <= horaBloqueadoFin) ||
            (horaInicioReserva <= horaBloqueadoInicio && horaFinReserva >= horaBloqueadoFin)
        ) {
            return res.status(400).json({
                mensaje: `❌ No disponible. Hay clase: ${bloqueado.MATERIA} de ${horaBloqueadoInicio.substring(0, 5)} a ${horaBloqueadoFin.substring(0, 5)}`
            });
        }
    }
```

Figura 27 Validación de conflictos al crear una reserva

4.4.5.3.4 Cambio de estado de una reserva y notificación multicanal (backend, routes/reservas.js)

Los tres canales no son redundantes, cada uno cubre un caso distinto: Socket.io es instantáneo, pero solo sirve si el usuario tiene la app abierta en ese momento; el correo es el que siempre llega, aunque tarde unos segundos más; y el push (Firebase) es el que cubre a la app cerrada. req.app.get('io') es simplemente la forma de llegar a la instancia de Socket.io creada en server.js sin tener que pasarla como parámetro por cada archivo de rutas.



```
router.put("/admin/reserva/:id/estado", verificarAdmin, (req, res) => {
    const id = req.params.id;
    const nuevoEstado = req.body.ESTADO;

    // Validar que el estado sea válido
    const estadosValidos = ['PENDIENTE', 'CONFIRMADA', 'CANCELADA'];
    if (!estadosValidos.includes(nuevoEstado)) {
        return res.status(400).json({
            mensaje: "Estado inválido. Debe ser: PENDIENTE, CONFIRMADA o CANCELADA"
        });
    }

    const query = 'UPDATE reservas SET ESTADO = ? WHERE RESERVAS_ID = ?';

    connection.query(query, [nuevoEstado, id], (err, result) => {
        if (err) {
            console.error("Error al cambiar estado de reserva:", err);
            return res.status(500).send("Error al cambiar estado de reserva");
        }
        if (result.affectedRows === 0) {
            return res.status(404).send("Reserva no encontrada");
        }

        // Buscar los datos completos de la reserva para notificar
        const queryDatos = `
    SELECT r.*, u.NOMBRE as USUARIO_NOMBRE, u.CORREO as USUARIO_CORREO,
           u.TELEFONO as USUARIO_TELEFONO, rec.NOMBRE as RECURSO_NOMBRE
    FROM reservas r
    JOIN usuarios u ON r.USUARIO_ID = u.USUARIO_ID
    JOIN recursos rec ON r.RECURSOS_ID = rec.RECURSOS_ID
    WHERE r.RESERVAS_ID = ?
`;
```

Figura 28 Cambio de estado de una reserva y notificación multicanal

4.4.5.3.5 Registro de un recurso con imagen (backend, routes/recursos.js)

El nombre del archivo se genera con Date.now() en vez de conservar el nombre original que sube el administrador, para que dos imágenes subidas el mismo día con el mismo nombre (por ejemplo, dos "foto.jpg" de celulares distintos) no se pisen entre sí en el disco. multer se configuró con diskStorage y no con memoria, así que las imágenes van directo al filesystem del servidor sin pasar por RAM.

```
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, path.join(__dirname, '../public/images/espacios'));
  },
  filename: (req, file, cb) => {
    const ext = path.extname(file.originalname);
    const nombre = Date.now() + ext;
    cb(null, nombre);
  }
});
const upload = multer({ storage });

// 1. CREAR RECURSO (Solo Admin)
router.post("/recurso", verificarAdmin, upload.single('IMAGEN'), (req, res) => {
  const nuevoRecurso = {
    NOMBRE: req.body.NOMBRE,
    descripcion: req.body.descripcion,
    ESTADO: req.body.ESTADO || 'DISPONIBLE',
    IMAGEN: req.file ? req.file.filename : null
  };

  const query = 'INSERT INTO recursos SET ?';
  connection.query(query, nuevoRecurso, (err, result) => {
    if (err) {
      return res.status(500).json({ mensaje: "Error al crear recurso", error: err.message });
    }
    res.json({ mensaje: "Recurso creado exitosamente", recursoId: result.insertId });
  });
});
```

Figura 29 Registro de un recurso con imagen

4.4.5.3.6 Registro de un bloqueo de horario (backend, routes/bloqueos.js)

MATERIA, PROFESOR y NIVEL solo se piden cuando TIPO es CLASE; si el bloqueo es por mantenimiento esos tres campos ni siquiera se validan. No fue así desde el principio del Sprint 2 -al inicio se pedían siempre, y los administradores de prueba se quejaron de tener que inventar un profesor solo para bloquear un aula en reparación, así que la validación se volvió condicional.

```
router.post("/bloqueos", verificarAdmin, (req, res) => {
    const {
        RECURSOS_ID,
        DIA_SEMANA,
        HORA_INICIO,
        HORA_FIN,
        MATERIA,
        PROFESOR,
        NIVEL,
        FECHA_INICIO,
        FECHA_FIN,
        TIPO
    } = req.body;

    // Validaciones
    if (!RECURSOS_ID || !DIA_SEMANA || !HORA_INICIO || !HORA_FIN || !FECHA_INICIO || !FECHA_FIN || !TIPO) {
        return res.status(400).json({ mensaje: "Faltan datos obligatorios" });
    }

    if (TIPO === 'CLASE' && (!MATERIA || !PROFESOR || !NIVEL)) {
        return res.status(400).json({
            mensaje: "Para bloqueos de clase se requiere: materia, profesor y nivel"
        });
    }

    const nuevoBloqueo = {
        RECURSOS_ID,
        DIA_SEMANA,
        HORA_INICIO,
        HORA_FIN,
        MATERIA: MATERIA || 'MANTENIMIENTO',
        PROFESOR: PROFESOR || '-',
        NIVEL: NIVEL || '-',
        FECHA_INICIO,
        FECHA_FIN,
        TIPO
    };
```

Figura 30 Registro de un bloqueo de horario

4.4.5.3.7 Corrección del desfase horario en timestamps (backend, db.js)

Durante la integración del módulo de reservas con el backend compartido se detectó que una reserva creada a las 08:00 se mostraba en la aplicación como las 13:00, un desfase de cinco horas equivalente a la diferencia entre la zona horaria de Ecuador (UTC-5) y UTC. El diagnóstico determinó que el driver mysql2 convertía automáticamente los campos DATETIME en objetos Date de JavaScript, los cuales se serializan en formato ISO 8601 con sufijo Z (UTC); Flutter interpretaba entonces ese valor como si ya estuviera en UTC y le restaba las cinco horas

nuevamente. Para resolverlo, se configuro el driver para que devolviera los campos de fecha como texto plano, sin conversión automática:

```
const pool = mysql.createPool({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME,
  port: process.env.DB_PORT,
  dateStrings: true
});
```

Figura 31 Corrección del desfase horario en timestamps

Etapa	Antes de la corrección	Después de la corrección
Hora seleccionada	8:00 AM	8:00 AM
Almacenado en MySQL	10:00:00(correcto)	10:00:00(correcto)
Devuelto por la API	2026-07-27T15:00:00.000Z (incorrecto)	2026-07-27 10:00:00 (correcto)
Mostrado en la app	13:00(incorrecto)	10:00(correcto)

Tabla 17 Efecto de la corrección dateStrings sobre los timestamps

La elección de este error tuvo más que ver con coordinación que con código: mientras cada backend se probaba por separado con su propio cliente, el desfase no se notaba, porque Flutter y el servidor “se entendían”en su propia convención interna. Recién se hizo visible al integrar ambos proyectos, y quedó como criterio para el resto del backend compartido: cualquier campo de fecha se devuelve como string plano, nunca como objeto Date serializado.

4.4.5.3.8 Cliente HTTP centralizado con sesión por cookies (Flutter, services/api_client.dart)

Todas las llamadas al backend pasan por esta única instancia de Dio, configurada una sola vez (patrón singleton perezoso) con la URL real de producción en Railway. El CookieManager es lo que permite que la sesión creada por express-session en el login sobreviva entre pantallas sin que el desarrollador tenga que leer o reenviar la cookie manualmente en cada petición; clearCookies() se invoca únicamente al cerrar sesión.

```
class ApiClient {
  static final Dio _dio = Dio();
  static final CookieJar _cookieJar = CookieJar();
  static bool _initialized = false;

  // static const String baseUrl = 'http://192.168.1.2:5000';
  static const String baseUrl = 'https://uleam-reservas-backend-web-production.up.railway.app';


  static Dio get dio {
    if (!_initialized) {
      _dio.options.baseUrl = baseUrl;
      _dio.options.connectTimeout = const Duration(seconds: 10);
      _dio.options.receiveTimeout = const Duration(seconds: 10);
      _dio.options.headers = {
        'Content-Type': 'application/json',
      };
      _dio.interceptors.add(CookieManager(_cookieJar));
      _initialized = true;
    }
    return _dio;
  }

  static Future<void> clearCookies() async {
    await _cookieJar.deleteAll();
  }
}
```

Figura 32 Cliente HTTP centralizado con sesión por cookies

4.4.5.3.9 Cambio de estado de un espacio desde un diálogo (Flutter, screens/admin/espacios_screen.dart)

El diálogo resalta con selected: true la opción que coincide con el estado actual del espacio, y cada ListTile dispara de inmediato la llamada al backend al tocarla —no hay un botón de "Guardar" separado—, recargando la lista completa (_cargarEspacios()) para reflejar el color actualizado en la tarjeta correspondiente.

```
void _cambiarEstado(Map<String, dynamic> espacio) {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: Text('Estado de ${espacio['NOMBRE']}'),
      content: Column(
        mainAxisSize: MainAxisSize.min,
        children: ['DISPONIBLE', 'OCUPADO', 'MANTENIMIENTO'].map((estado) {
          return ListTile(
            leading: CircleAvatar(
              backgroundColor: _colorEstado(estado),
              radius: 8,
            ), // CircleAvatar
            title: Text(estado),
            selected: espacio['ESTADO'] == estado,
            onTap: () async {
              Navigator.pop(context);
              await EspaciosService.cambiarEstado(
                  espacio['RECURSOS_ID'].toString(), estado);
              _cargarEspacios();
              _mostrarExito('Estado actualizado a $estado');
            },
          ); // ListTile
        }).toList(),
      ), // Column
    ), // AlertDialog
  );
}
```

Figura 33 Cambio de estado de un espacio desde un diálogo

4.4.5.3.10 Aprobación o rechazo de una reserva desde el panel admin (Flutter, screens/admin/aprobar_reservas_screen.dart)

La misma función _revisar() atiende tanto la aprobación como el rechazo, recibiendo el estado deseado ('CONFIRMADA' o 'CANCELADA') como parámetro y ajustando únicamente el título del diálogo según corresponda; esto evita duplicar la lógica de confirmación entre los dos botones de la tarjeta de reserva pendiente.

```
void _revisar(Map<String, dynamic> reserva, String estado) {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: Text(estado == 'CONFIRMADA'
          ? 'Aprobar reserva'
          : 'Cancelar reserva'), // Text
      content: Text(
        '${reserva['NOMBRE'] ?? 'Espacio ${reserva['RECURSOS_ID']}'} — ${reserva['USUARIO_NOMBRE'] ?? ''}',
        style: const TextStyle(fontWeight: FontWeight.bold),
      ), // Text
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: const Text('Cancelar'),
        ), // TextButton
        ElevatedButton(
          onPressed: () async {
            Navigator.pop(context);
            final result = await ReservasService.revisarReserva(
              id: reserva['RESERVAS_ID'].toString(),
              estado: estado,
            );
            if (result.containsKey('error')) {
              if (mounted) {
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(
                      content: Text(result['error']),
                      backgroundColor: Colors.red), // SnackBar
                );
              }
```

Figura 34 Aprobación o rechazo de una reserva desde el panel administrativo

4.4.5.3.11 Registro del token de notificaciones push (Flutter, services/notificaciones_services.dart)

El registro de token no se realiza únicamente una vez. Además de enviarlo inmediatamente después del inicio de sesión, la función también se suscribe a onTokenRefresh, ya que Firebase puede renovar o cambiar el token asociado a un dispositivo, por ejemplo, después de reinstalar la aplicación. Si el backend conserva el token anterior, las notificaciones push podrían dejar de llegar al usuario sin que se genere ningún error visible.

```dart
static Future<void> registrarToken() async {
  try {
    final token = await _messaging.getToken();
    if (token == null) return;
    final usuario = await AuthService.getUsuario();
    if (usuario == null) return;
    await ApiClient.dio.post(
      '/notificaciones/token',
      data: {'token': token},
    );
    _messaging.onTokenRefresh.listen((nuevoToken) async {
      await ApiClient.dio.post(
        '/notificaciones/token',
        data: {'token': nuevoToken},
      );
    });
  } catch (e) {
    // ignore: avoid_print
    print('No se pudo registrar el token FCM: $e');
  }
}
```

Figura 35 Registro del token de notificaciones push

4.4.5.3.12 Fusión de reservas y bloqueos para el calendario (Flutter, screens/reservas/calendario_reserva_screen.dart)

La pantalla del calendario no establece una diferencia visual entre una reserva confirmada y un bloqueo generado por una clase programada. Ambos se identifican mediante ESTADO=’BLOQUEADO’ y un MOTIVO obtenido a partir de la materia correspondiente. Esta unificación en el cliente permite mostrar en color rojo, utilizando un mismo widget de tarjeta, tanto las clases programadas como las reservas confirmadas correspondientes al día seleccionado.

```
Future<void> _cargarReservasDia(DateTime dia) async {
  try {
    final fechaStr =
        '${dia.year}-${dia.month.toString().padLeft(2, '0')}-${dia.day.toString().padLeft(2, '0')}';
    final disponibilidad = await ReservasService.getDisponibilidad(
      widget.espacio['RECURSOS_ID'].toString(),
      fechaStr,
    );
    final reservas = List<dynamic>.from(disponibilidad['reservas'] ?? []);
    final bloqueados = List<dynamic>.from(disponibilidad['bloqueados'] ?? []);
    setState(() {
      _reservasEspacio = [...reservas, ...bloqueados.map((b) => {
        'FECHA_INICIO': '$fechaStr ${b['HORA_INICIO']}',
        'FECHA_FIN': '$fechaStr ${b['HORA_FIN']}',
        'ESTADO': 'BLOQUEADO',
        'MOTIVO': b['MOTIVO'] ?? b['MATERIA'] ?? 'Clase',
      })];
    });
  } catch (e) {
    // silencioso
  }
}
```

Figura 36 Fusión de reservas y bloqueos para el calendario

4.4.6 Eventos Scrum

4.4.6.1 Sprint Review 1: Autenticación y estructura base

Aspecto	Descripción
Participantes	Product Owner, Scrum Master y Developer, de acuerdo con la adaptación académica definida para el proyecto.
Revisión de actividades	Se revisaron el registro, inicio de sesión, mantenimiento de sesión, visualización de nombre y rol, cierre de sesión y navegación mediante go_router.

Aspecto	Descripción
Retrospectiva / ajuste	Las pruebas registraron resultados exitosos en los flujos principales y permitieron establecer una base estable para incorporar funciones administrativas.

Tabla 18 Sprint 1

4.4.6.2 Sprint Review 2: Gestión de espacios y bloqueos

Aspecto	Descripción
Participantes	Product Owner, Scrum Master y Developer, de acuerdo con la adaptación académica definida para el proyecto.
Revisión de actividades	Se revisaron listar, crear, editar y eliminar espacios, cambiar su estado y administrar bloqueos de horario fijo y por rango de fechas.
Retrospectiva / ajuste	Durante este sprint se ajustó la validación de bloqueos para solicitar MATERIA, PROFESOR y NIVEL únicamente cuando el tipo de bloqueo es CLASE, evitando datos innecesarios para mantenimiento.

Tabla 19 Sprint 2

4.4.6.3 Sprint Review 3: Reservas y aprobación

Aspecto	Descripción
Participantes	Product Owner, Scrum Master y Developer, de acuerdo con la adaptación académica definida para el proyecto.
Revisión de actividades	Se revisaron calendario, creación de solicitudes, validación de conflictos, Mis Reservas, acceso administrativo, aprobación/rechazo y control de timestamps.
Retrospectiva / ajuste	La integración permitió detectar el desfase horario provocado por la conversión de DATETIME; se corrigió configurando la devolución de fechas como texto plano.

Tabla 20 Sprint 3

4.4.6.4 Sprint Review 4: Notificaciones y funcionamiento offline-first

Aspecto	Descripción
Participantes	Product Owner, Scrum Master y Developer, de acuerdo con la adaptación académica definida para el proyecto.
Revisión de actividades	Se revisaron notificaciones al administrador y al estudiante, consulta de espacios y reservas en modo avión, banner offline, splash screen e icono institucional.
Retrospectiva / ajuste	Las pruebas en dispositivos físicos confirmaron la recepción de notificaciones y el uso del cache local, completando el incremento final para despliegue.

Tabla 21 Sprint 4

4.4.7 Pruebas

4.4.7.1 Pruebas de caja negra

Las pruebas de caja negra se ejecutaron al terminar cada sprint, verificando el comportamiento observable de cada función desde la perspectiva del usuario, sin considerar la lógica interna de implementación. A continuación se consolidan los resultados obtenidos en los cuatro sprints de desarrollo.

4.4.7.1.1 Módulo de autenticación y base del sistema (Sprint 1)

Prueba	Resultado	Observación
POST /api/auth/registro	Exitoso (201)	Retorna los datos del usuario creado.
POST /api/auth/login	Exitoso (200)	Retorna la sesión y los datos del usuario.
Formulario de login (Flutter)	Exitoso	Validación de campos e indicador de carga.
Formulario de registro (Flutter)	Exitoso	Validación de campos y selección de rol.
Visualización de nombre y rol en Home	Exitoso	Datos reales provenientes de la base de datos.
Cierre de sesión	Exitoso	Limpieza de la sesión almacenada.
Navegación entre pantallas (go_router)	Exitoso	Transiciones fluidas entre rutas.

Tabla 22 Resultados de pruebas de caja negra — Módulo de autenticación y base del sistema (Sprint 1)

4.4.7.1.2 Módulo de gestión de espacios (Sprint 2)

Prueba	Resultado	Observación
Listar espacios	Exitoso	Carga datos reales desde MySQL.

Prueba	Resultado	Observación
Crear espacio	Exitoso	Formulario con validación completa.
Editar espacio	Exitoso	Precarga los datos del espacio seleccionado.
Eliminar espacio	Exitoso	Solicita confirmación antes de eliminar.
Cambiar estado del espacio	Exitoso	Diálogo con los tres estados disponibles.
Agregar bloqueo de horario fijo	Exitoso	Selector de día, hora de inicio y de fin.
Agregar bloqueo por fecha	Exitoso	DatePicker para inicio y fin del bloqueo.
Eliminar bloqueos	Exitoso	Eliminación inmediata con recarga de la lista.

Tabla 23 Resultados de pruebas de caja negra — Módulo de gestión de espacios (Sprint 2)

4.4.7.1.3 Módulo de reservas y aprobación (Sprint 3)

Prueba	Resultado	Observación
Visualizar calendario de un espacio	Exitoso	Marca los días con reservas existentes.
Crear solicitud de reserva	Exitoso	Queda registrada en estado PENDIENTE.
Validación de conflicto de horario	Exitoso	Rechaza las solicitudes superpuestas.
Ver Mis Reservas con filtros	Exitoso	Filtrada correctamente por estado.
Acceso al panel de aprobación (admin)	Exitoso	Lista las reservas pendientes.
Aprobar una reserva	Exitoso	Cambia de estado y desaparece de pendientes.
Rechazar una reserva	Exitoso	El estado se actualiza correctamente.
Restricción de acceso (estudiante)	Exitoso	No visualiza opciones administrativas.

Prueba	Resultado	Observación
Corrección de timestamps MySQL	Exitoso	La hora guardada y la mostrada coinciden.

Tabla 24 Resultados de pruebas de caja negra — Módulo de reservas y aprobación (Sprint 3)

4.4.7.1.4 Módulo de notificaciones e identidad offline (Sprint 4)

Prueba	Resultado	Dispositivo
Notificación push al admin por nueva reserva	Exitoso	Infinix X678B (admin)
Notificación push al estudiante por reserva aprobada	Exitoso	Samsung SM-A125U (estudiante)
Notificación push al estudiante por reserva rechazada	Exitoso	Samsung SM-A125U (estudiante)
Ver espacios en modo avión	Exitoso	Samsung SM-A125U
Ver Mis Reservas en modo avión	Exitoso	Samsung SM-A125U
Banner naranja de sin conexión visible	Exitoso	Samsung SM-A125U
Splash screen con sesión guardada → Home	Exitoso	Samsung SM-A125U
Splash screen sin sesión → Login	Exitoso	Samsung SM-A125U
Ícono del launcher con logo institucional	Exitoso	Ambos dispositivos

Tabla 25 Resultados de pruebas de caja negra — Módulo de reservas y aprobación (Sprint 3)

4.4.7.2 Pruebas de caja blanca

Las pruebas de caja blanca evaluaron las rutas internas de decisión del código del backend, y del cliente Flutter, verificando que cada rama condicional produjera el resultado esperado.

Función evaluada	Rutas de código cubiertas	Resultado
Anticipación mínima de reserva(routes/reservas.js)	1) TIPO=ESTUDIANTE y horasDeAnticipacion < 24 → rechaza. 2) TIPO=ESTUDIANTE y horasDeAnticipacion ≥ 24 → continúa. 3) TIPO≠ESTUDIANTE (profesor/admin) → continúa sin restricción.	El resultado de los tests análogamente esperados fue el que se devolvió por las tres rutas.
Choque con horario bloqueado(routes/reservas.js)	1) Existe solapamiento con una clase programada → rechaza indicando la materia. 2) No hay bloqueos para ese día/hora → continúa a la siguiente validación.	Ambas rutas presentaron un comportamiento correcto.
Choque con otra reserva(routes/reservas.js)	Se evaluaron las tres condiciones de solapamiento (inicio dentro de un rango existente, fin dentro de un rango existente, rango que contiene a uno existente) mediante la cláusula OR de la consulta SQL.	Las tres condiciones de solapamiento fueron detectadas correctamente, sin errores en la inserción de los horarios sin cruce.
Middleware verificarAdmin(middlewares/auth.js)	1) Sin sesión activa → 401. 2) Sesión activa, TIPO≠ADMINISTRADOR → 403. 3) Sesión activa, TIPO=ADMINISTRADOR → next().	Las tres rutas devolvieron el código HTTP esperado.
Cache-first en getEspaciosDisponibles() (Flutter, ReservasService)	1) Con internet: petición HTTP exitosa → actualiza cache y retorna datos frescos. 2) Sin internet: retorna directamente el cache local. 3) Con internet pero servidor caído: el try-catch captura el error y retorna el cache como respaldo.	Las tres rutas fueron revisadas manualmente activando y desactivando la conexión y parando el servidor de backend.
Verificación real de conectividad hayInternet() (Flutter)	1) ConnectivityResult.none → false de inmediato. 2) Interfaz activa pero sin resolución DNS (modo avión con WiFi asociado simulado) → false. 3) Interfaz activa y resolución DNS exitosa → true.	Las tres rutas se revisaron revocando permisos de notificación y forzando un reinicio de instalación de firebase en

Función evaluada	Rutas de código cubiertas	Resultado
		dispositivo de prueba.
Registro de token FCM con manejo de errores (NotificacionesService)	1) _messaging.getToken() retorna null (permiso denegado) → la función retorna sin llamar al backend. 2) Usuario no autenticado (getUsuario() null) → retorna sin registrar. 3) Token y usuario válidos → POST exitoso a /notificaciones/token. 4) Firebase rota el token (onTokenRefresh) → se reenvía automáticamente.	Las cuatro rutas fueron verificadas revocando permisos de notificación y forzando un reinicio de la instalación de Firebase en el dispositivo de prueba.
_revisar() para aprobar/rechazar (Flutter, AprobarReservasScreen)	1) estado='CONFIRMADA' → título "Aprobar reserva", backend recibe ESTADO=CONFIRMADA. 2) estado='CANCELADA' → título "Cancelar reserva", backend recibe ESTADO=CANCELADA. 3) Backend devuelve error (conflicto ya resuelto por otro admin) → se captura result['error'] sin cerrar la lista.	Las tres rutas se comportaron según lo esperado, incluyendo el caso de conflicto de doble revisión simultánea.
_cambiarEstado() con ListTile seleccionable (EspaciosScreen)	Se verificaron los tres estados posibles (DISPONIBLE, OCUPADO, MANTENIMIENTO) confirmando que selected: true resalta únicamente el estado actual del espacio antes de cualquier cambio, y que cada selección dispara la llamada a EspaciosService.cambiarEstado() de inmediato.	Los tres estados se reflejaron correctamente en el color de la tarjeta tras recargar la lista.

Tabla 26 Resultados de pruebas de caja blanca

4.4.7.3 Pruebas de usabilidad

Las pruebas de usabilidad se ejecutaron de forma funcional en condiciones reales: durante el Sprint1 sobre el navegador Chrome y desde el Sprint 4 sobre dos dispositivos Android físicos operando de manera simultánea, uno con rol de estudiante y otro con rol administrador, lo que permitió validar el flujo bidireccional completo entre ambos roles.

Tarea evaluada	Resultado	Observación
Registro e inicio de sesión con correo institucional	Exitoso	El usuario completo el flujo sin asistencia ni errores de validación.
Consulta de disponibilidad en el calendario	Exitoso	Los niveles de ocupación por color se interpretaron correctamente.
Creación de una solicitud de reserva	Exitoso	El formulario modal se completo sin necesidad de instrucciones adicionales.
Recepción de notificación push tras aprobar/rechazar	Exitoso	La notificación llego al dispositivo del solicitante en ambos escenarios (aprobada y rechazada).
Aprobación de una reserva desde el panel admin	Exitoso	El administrador ubicó y resolvió la solicitud sin dificultad.
Consulta de espacios y reservas en modo avión	Exitoso	El usuario reconoció el banner naranja como indicador de modo sin conexión.
Reapertura de la app con sesión previa	Exitoso	El splash screen redirigió automáticamente a Home sin solicitar login nuevamente.

Tabla 27 Resultados de pruebas de usabilidad

4.5 Incremento y Entregables

4.5.1 Incrementos por Sprint

Sprint	Incremento entregado
Sprint 1	Base de autenticación, sesión, navegación y control de acceso por roles.
Sprint 2	Gestión administrativa de espacios físicos, estados, imágenes y bloqueos.
Sprint 3	Flujo completo de disponibilidad, solicitud, validación, historial y aprobación/rechazo de reservas.
Sprint 4	Notificaciones multicanal, almacenamiento SQLite, estrategia cache-first, indicador offline, identidad visual y versión integrada.

Tabla 28 Incrementos por Sprint

4.5.2 Servicio/Lanzamiento

El backend del sistema se encuentra desplegado en producción sobre la plataforma Railway, junto con su base de datos MySQL, ambos como servicios independientes dentro del

mismo proyecto (reasonable-charisma). El servicio del backend (uleam-reservas-backend-web)se despliega automaticamente desde el repositorio de GitHub del proyecto, y expone la API en la URL pública: https://uleam-reservas-backend-web-production.up.railway.app, que consume tanto la aplicación móvil Flutter como el sistema web complementario. La base de datos MySQL corre como un servicio adicional dentro del mismo proyecto de Railway, con sus tablas(usuarios, recursos, reservas, horarios_bloqueados, fcm_tokens, roles) accesibles mediante el editor de datos integrado de la plataforma.

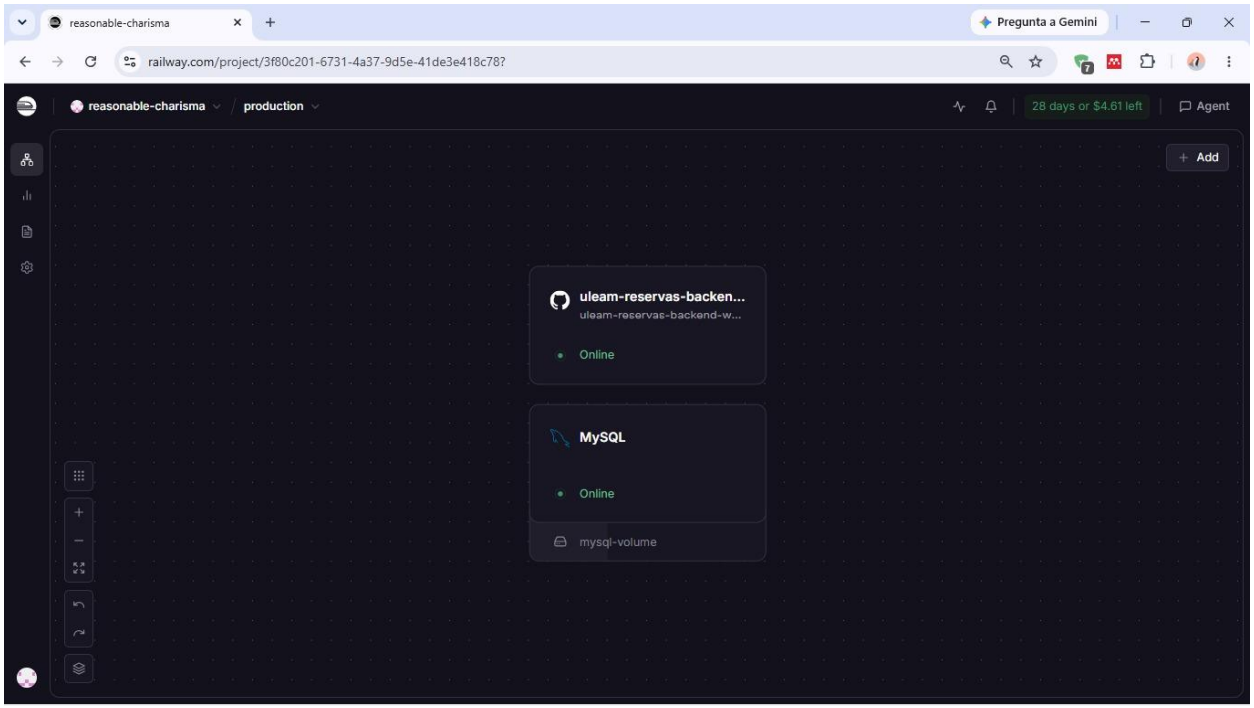


Figura 37 Panel del proyecto en Railway, con el backend y la base de datos MySQL en línea

Ambos servicios (uleam-reservas-backend-web y MySQL) corren dentro del mismo proyecto y ambiente de producción, cada uno con su propio estado (Online) monitoreado desde este panel.

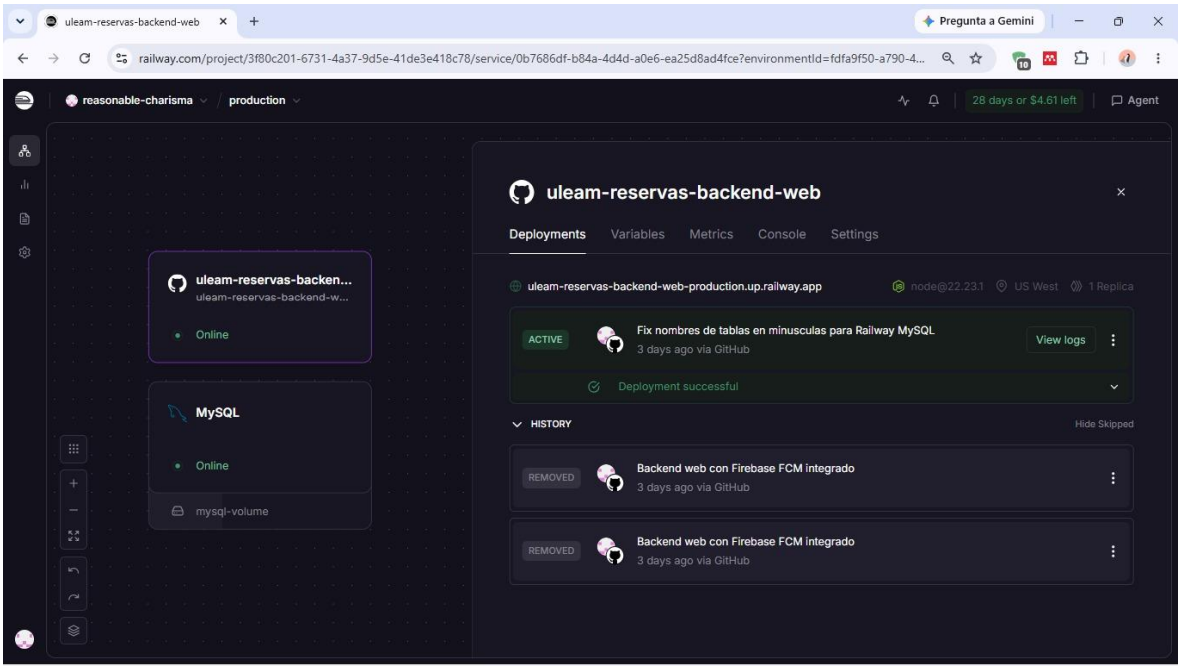


Figura 38 Historial de despliegues del servicio backend

Cada entrada del historial corresponde a un push al repositorio de GitHub; Railway reconstruye y publica automáticamente la nueva versión (despliegue continuo), y conserva el registro de versiones anteriores para poder revertir en caso de error.

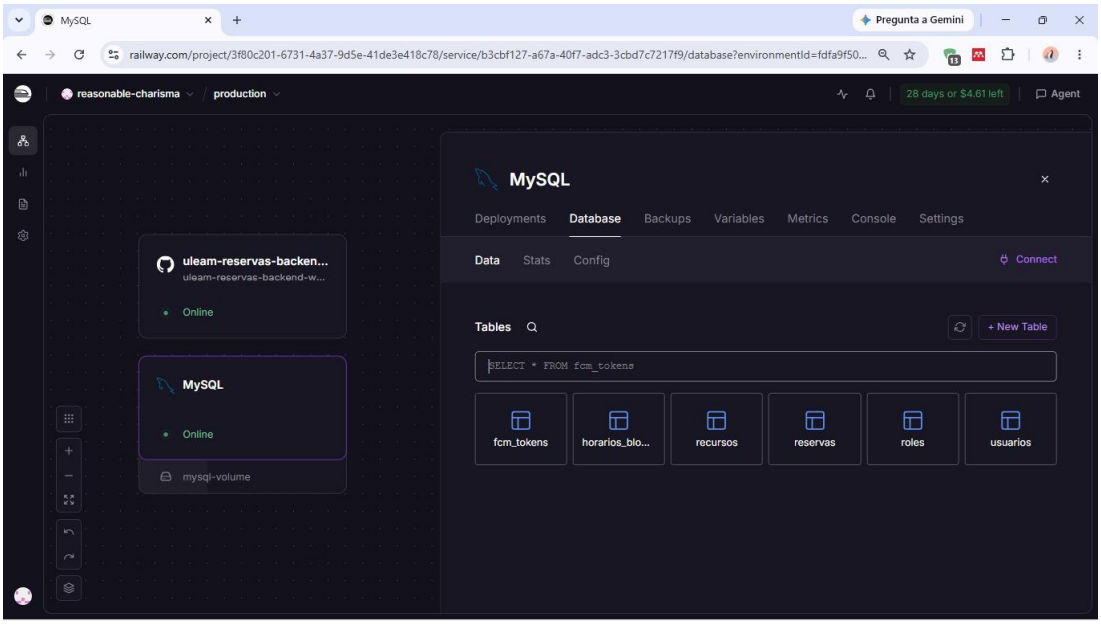


Figura 39 Editor de datos de la base MySQL en Railway

El panel de datos integrado de Railway permite consultar directamente las tablas de producción (fcm_tokens, horarios_bloqueados, recursos, reservas, roles, usuarios) sin necesidad de una herramienta externa como phpMyAdmin o MySQL Workbench, lo que agilizó la verificación de datos durante las pruebas descritas en este capítulo.

- Repositorio backend conectado a Railway para el desarrollo continuo (cualquier cambio realizado en la rama principal se publica automáticamente).
- El servicio MySQL se proporciona en el propio proyecto Railway, con un volumen persistente (mysql-volume) para el almacenamiento de datos.
- Las variables de entorno, incluidas las credenciales de inicio de sesión de la base de datos, el secreto de sesión y las credenciales de correo electrónico, se configuran directamente en el panel de control de Railway y no se exponen en el repositorio.
- Generación de la versión de producción de la aplicación móvil, mediante la compilación de la aplicación Flutter en modo release (flutter build apk), apuntando a la URL de producción del backend.
- Distribución del archivo APK generado a los dispositivos de prueba de la comunidad universitaria (estudiantes, docentes y administrador).
- Configuración del proyecto Firebase (google-services.json) en cada instalación para habilitar las notificaciones push.

CAPÍTULO V

5 EVALUACIÓN DE RESULTADOS

5.1 Introducción

La evaluación de los resultados del presente proyecto se realizó mediante un análisis comparativo entre la situación diagnosticada antes de la implementación del sistema, presentada en el capítulo III mediante datos recolectados con la encuesta(n=256) y las entrevistas (coordinador académico y un docente con alta demanda de espacios), y la capacidad técnica demostrada por la aplicación una vez desarrollada, documentada en el capítulo IV a través de los resultados de las pruebas funcionales de caja negra, caja blanca y usabilidad. El objetivo de esta evaluación es determinar el grado de mitigación de las problemáticas identificadas en el planteamiento del problema de esta investigación: la dependencia de procesos manuales y la falta de acceso móvil, los conflictos de horario por ausencia de control automático, y la dependencia de una conexión a internet constante para poder operar.

5.2 Presentación y monitoreo de resultados

La técnica empleada para la comparación de resultados consistió en contrastar, para cada una de las causas del problema, un indicador medido sin la variable independiente, es decir, bajo el proceso manual diagnosticado mediante la encuesta aplicada a estudiantes y docentes y las entrevistas al coordinador académico y a un docente, frente al mismo indicador evaluado con la variable independiente, una vez implementada la aplicación móvil, medido a través de las pruebas funcionales documentadas en el Capítulo IV.

Causa 1: Dependencia de procesos manuales y falta de un canal móvil

Indicador	Sin sistema (diagnóstico)	Con sistema (resultados de prueba)
Método de gestión de reservas	Según las entrevistas, las solicitudes se gestionaban mediante hojas de cálculo, correo electrónico o comunicación verbal, sin ningún registro centralizado(coordinador académico y docente entrevistado).	El 100% de las funciones de consulta, solicitud y aprobación pasan ahora por la aplicación; las 33 pruebas de caja negra ejecutadas en los cuatro sprints resultaron exitosas.
Acceso a información de disponibilidad	42.2% de los 256 encuestados señala la falta de información sobre disponibilidad como su principal inconveniente con el proceso actual.	Consulta de disponibilidad mediante calendario interactivo con niveles de ocupación por color, validada exitosamente en las pruebas de caja negra y de usabilidad.
Disposición a usar un canal móvil nativo	El 83.2% de los encuestados respondió que si usaría una aplicación móvil, y un 12.5% adicional respondió "tal vez" (95.7% en conjunto); el 64.8% señaló el celular como el dispositivo desde el cual accedería.	Desarrolló una aplicación nativa en Flutter para Android, con acceso directo desde el dispositivo del usuario, sin depender de un equipo de escritorio.

Tabla 29 Comparación de resultados — Causa 1

Causa 2: Conflictos de horario por falta de control automático

Indicador	Sin sistema (diagnóstico)	Con sistema (resultados de prueba)
Conflictos de horario reportados	El coordinador académico y el docente entrevistado relataron casos recientes de doble asignación de un laboratorio, resueltos manualmente trasladando o reprogramando la actividad afectada.	No se registro conflictos en las solicitudes de reservas creadas durante las pruebas; la validación automática de choques con clases programadas con otras reservas cubrió las tres rutas de solapamiento posibles (caja blanca).
Priorización de usuarios	Los entrevistados señalan la ausencia de un criterio único de priorización entre docentes y estudiantes.	La regla de anticipación mínima de 24 horas se aplica automáticamente solo al rol estudiante, y quedo verificada en las tres rutas de decisión (caja blanca).

Indicador	Sin sistema (diagnóstico)	Con sistema (resultados de prueba)
Mecanismo de aprobación	Las decisiones de aprobación se tomaban de manera informal, sin un flujo ni un registro formal.	Se implementó un flujo formal pendiente → confirmada/cancelada, con panel administrativo dedicado; el 100% de las pruebas de aprobación y rechazo resulto exitoso.

Tabla 30 Comparación de resultados — Causa 2

Causa 3: Dependencia de una conexión a internet constante

Indicador	Sin sistema (diagnóstico)	Con sistema (resultados de prueba)
Estabilidad de la conectividad institucional	Las entrevistas evidenciaron interrupciones intermitentes de conectividad en la institución, sobre todo en horas pico.	La arquitectura offline-first, con verificación real de conectividad mediante resolución DNS, permitió consultar espacios y reservas propias exitosamente en modo avión (pruebas de caja negra del Sprint 4).
Tiempo de gestión de una reserva	El 46.1% de los encuestados invierte más de 20 minutos en gestionar una reserva (29.3% entre 20 y 30 minutos, 16,8% más de 30 minutos); el coordinador académico dedica entre 4 y 6 horas semanales a esta labor.	La tarea de creación de una solicitud de reserva fue completada por el usuario de prueba sin asistencia ni instrucciones adicionales, consistente con la meta institucional de reducción de tiempos planteada en el Capítulo I.
Satisfacción con el proceso actual	Únicamente 25.1% de los encuestados se declara satisfecho o muy satisfecho con el proceso actual, frente a un 41.1% insatisfecho o muy insatisfecho.	Cache local en SQLite (cache_espacios, cache_mis_reservas) con patrón cache-first, probado exitosamente incluyendo el banner visual de modo sin conexión, como respuesta directa a la fragilidad de conectividad diagnosticada.

Tabla 31 Comparación de resultados — Causa 3

5.3 Interpretación objetiva

Las entrevistas pusieron de manifiesto que las reservas eran gestionadas de forma manual, acudiendo a el uso de hojas de calculo, correos y comunicación verbal, sin que exista ningún registro centralizado, lo cual coincide con la respuesta dada por el 42,2% de los encuestados que declaró que no tener información sobre la disponibilidad era su mayor inconveniente. Las 33 pruebas de caja negra realizadas tienen éxito, lo que indica que la aplicación sustituye funcionalmente estos procesos informales, ya que el 95,7% mostró disposición a usar una aplicación móvil, indicando un alto potencial de adopción, aunque su aceptación real habrá de comprobarse con el uso.

En cuanto a los conflictos de cursos, se detectaron algunos casos de asignaciones duplicadas durante las entrevistas, pero las pruebas del sistema no detectaron ningún conflicto. Esto se debe a los mecanismos de verificación implementados para evitar la superposición entre cursos y otras reservas. Las pruebas de caja blanca confirmaron

la cobertura de las rutas superpuestas, pero su efectividad dependerá de si los administradores pueden actualizar los horarios y la información de organización de los cursos de manera oportuna.

Finalmente, las entrevistas señalaron problemas recurrentes de conectividad; La arquitectura offline-first, con almacenamiento local y detección de conexión, fue validada exitosamente en dispositivos físicos, permitiendo consultar información sin internet. Asimismo, aunque el 46,1% tarda más de 20 minutos en gestionar una reserva y solo el 25,1% está satisfecho con el proceso actual, las pruebas de usabilidad mostraron un proceso más fluido. Sin embargo, la reducción real de tiempos deberá comprobarse mediante mediciones con usuarios en un entorno de uso continuo.

CAPÍTULO VI

6 CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

La investigación realizada permitió conocer a detalle los procesos que afectan la gestión de espacios físicos compartidos en la Uleam-El Carmen, confirmado mediante la encuesta y las entrevistas que dicha gestión dependía casi en su totalidad de canales manuales sin un mecanismo de consulta ni de control de conflictos.

El desarrollo de la aplicación móvil con sincronización inteligente respondió directamente a este problema, automatizando la consulta de disponibilidad, la solicitud y aprobación de reservas, y la notificación de su estado a los usuarios.

El desarrollo se sustentó en conceptos de aplicaciones móviles, arquitecturas offline-first, sincronización de datos y gestión de reservas, y se organizó siguiendo la metodología Scrum en cuatro sprints. La estructura incremental permitió incorporar sucesivamente autenticación, gestión de espacios, reservas, notificaciones y consulta local sin conexión.

La implementación permitió integrar el módulo de reservas con un backend compartido con el proyecto web institucional; unificando la información de espacios y reservas. Esta integración permitió ver la necesidad de mantener criterios técnicos coordinados entre los equipos, como ocurrió con la corrección del desfase horario que se detectó.

La evaluación del sistema mediante pruebas de caja negra, caja blanca y usabilidad en dispositivos Android, verificando el funcionamiento de los módulos, las principales validaciones internas y la facilidad de uso. Todas las pruebas documentadas en el Capítulo IV resultaron satisfactorias.

La comparación de los resultados del diagnóstico con los de la evaluación muestra una mitigación técnica de las principales causas del problema: los procesos manuales, los conflictos de horarios y la dependencia constante a internet. Aun así, el impacto real dependerá de que la comunidad universitaria de la ULEAM-El Carmen adopte, la aplicación y la usa de forma continua.

6.2 Recomendaciones

Se recomienda que el personal administrativo, docente y estudiantil se familiarice con las funciones de la aplicación, en especial, con el modo offline y las notificaciones Push, para aprovechar mejor sus capacidades.

Se sugiere implementar una bitácora de las acciones realizadas sobre la reservas, aprobaciones, rechazos y cambios de estado, para facilitar la auditoría y resolver posibles inconvenientes.

También se propone realizar pruebas de carga sobre los principales servicios del backend, en particular en la creación de reservas y en la consulta de disponibilidad, a fin de evaluar el rendimiento que dichos servicios poseen con una alta carga de usuarios concurrentes.

Además, se debe considerar el despliegue del backend compartido y plantear un mecanismo de distribución de la aplicación móvil ya sea de la distribución por Google Play Store o la distribución interna del APK, con el objetivo de garantizar su disponibilidad.

Finalmente, considerando el crecimiento institucional y la posible incorporación de nuevos tipos de espacios o de nuevas reglas de negocio, se recomienda mantener actualizaciones periódicas del sistema y de su base de datos, así como preservar la organización modular del backend (rutas, middlewares y servicios independientes por dominio) que facilitó su mantenimiento durante el desarrollo.

7 Bibliografía

Arias, F. G. (2016). *El proyecto de investigación: Introducción a la metodología científica* (7th ed.). Episteme.

Azketa, E., Mendialdua, X., Ibarguren, I., & Solís, A. (2021). Método de sincronización para sistemas distribuidos con seguridad funcional. *Revista Iberoamericana de Automática e Informática Industrial*, *18*(2), 109–114. https://doi.org/10.4995/RIAI.2020.14022

Cabezuelo, A. S. (2023a). Un sistema de reservas y gestión de puestos en una biblioteca universitaria utilizando geolocalización. *ATICA 2022: Aplicación de Tecnologías de La Información y Comunicaciones Avanzadas y Accesibilidad: Universidad Politécnica Salesiana Cuenca (Ecuador), Del 16 al 18 de Noviembre de 2022*, 171–178.

Cabezuelo, A. S. (2023b). Un sistema de reservas y gestión de puestos en una biblioteca universitaria utilizando geolocalización. *ATICA 2022: Aplicación de Tecnologías de La Información y Comunicaciones Avanzadas y Accesibilidad*, 171–178.

Culqui Escobar, A. E. (2015a). *Sistema Web para el registro de reservaciones y control de hospedaje en el Hotel Acapulco de la ciudad de Ambato.* Universidad Técnica de Ambato.

Culqui Escobar, A. E. (2015b). *Sistema web para el registro de reservaciones y control de hospedaje en el Hotel Acapulco de la ciudad de Ambato*.

Denzin, N. K., & Lincoln, Y. S. (2022). Manual de investigación cualitativa. Vol. IV: Métodos de recolección y análisis de datos. *Editorial Gedisa*, *31*(48), 166–177. https://doi.org/10.61303/07172257.V31I48.225

Emiro Francisco, L. R. (2023). *DESARROLLO DE UNA APLICACIÓN WEB PARA LA AUTOMATIZACIÓN DEL PROCESO DE ADMINISTRACIÓN Y CONTROL EN LOS LABORATORIOS DE INFORMÁTICA DE LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS*. Universiad Tecnica del Norte.

Engel, A., & Coll, C. (2022). Entornos híbridos de enseñanza y aprendizaje para promover la personalización del aprendizaje. *RIED-Revista Iberoamericana de Educación a Distancia*, *25*(1), 225–242. https://doi.org/10.5944/RIED.25.1.31489

Figueroa Quimis. (2022). *UNIVERSIDAD ESTATAL DEL SUR DE MANABÍ FACULTAD DE CIENCIAS TÉCNICAS*.

Google. (2026). *Firebase Cloud Messaging*. https://firebase.google.com/docs/cloud-messaging?hl=es-419

Guale, M. S. G., Guale, M. S. G., & Anormaliza, R. R. (2024). Aplicaciones móviles y el rendimiento académico en estudiantes de bachillerato en una Unidad Educativa en Ecuador. *Polo Del Conocimiento*, *9*(8), 1688–1708. https://doi.org/10.23857/pc.v9i8.7784

Hernández-Sampieri, R., & Mendoza Torres, C. P. (2018). *Metodología de la investigación: Las rutas cuantitativa, cualitativa y mixta*. McGraw-Hill Education.

Humberto, J., & Casariego, C. (2026). Aplicaciones móviles en la Educación Superior latinoamericana pospandemia: Una revisión del uso, logros y obstáculos. *Revista Cubana de Educación Superior*, *45*. https://revistas.uh.cu/rces/article/view/12473

Kleppmann, M. (2017). Designing Data-Intensive Applications: The Big Ideas behind Reliable, Scalable, and Maintainable Systems. *O'Reilly Media, Inc.*, 569. https://www.oreilly.com/library/view/designing-data-intensive-applications/9781491903063/

Omar, O., Ochoa, D., & Pizarro, M. G. (2021a). *Implementación de una aplicación web para la gestión de reservas y de espacios para la Dirección Técnica de Administración e Inventarios*.

Omar, O., Ochoa, D., & Pizarro, M. G. (2021b). *UNIVERSIDAD POLITÉCNICA SALESIANA SEDE GUAYAQUIL*.

Oracle. (2026). *MySQL 8.0 Reference Manual*. https://docs.oracle.com/cd/E17952_01/mysql-8.0-en/index.html

OWASP Foundation. (2026). *Password Storage Cheat Sheet*. https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

Pascuas-Rengifo, Y. S., García-Quintero, J. A., & Mercado-Varela, M. A. (2020). Dispositivos móviles en la educación: tendencias e impacto para la innovación. *Revista Politécnica*, *16*(31), 97–109. https://doi.org/10.33571/RPOLITEC.V16N31A8

Pressman, R. S. ., Maxim, B. R. ., Medellín Serna, L. Antonio., & Vidal Romero Elizondo, Alfonso. (2021). *Ingeniería de software : un enfoque práctico*. McGraw Hill Interamericana Editores.

Pressman, R. S., & Maxim, B. R. (2021). *Ingeniería de software: Un enfoque práctico*. McGraw Hill Interamericana.

Quisaguano Collaguazo, L. R., Pallasco Venegas, M. S., Andaluz Guerrero, A. A., Martínez Freire, M. N., & Corrales Beltrán, S. H. (2022). Desarrollo híbrido con Flutter. *Ciencia Latina Revista Científica Multidisciplinar*, *6*(4), 4594–4609. https://doi.org/10.37811/CL_RCM.V6I4.2959

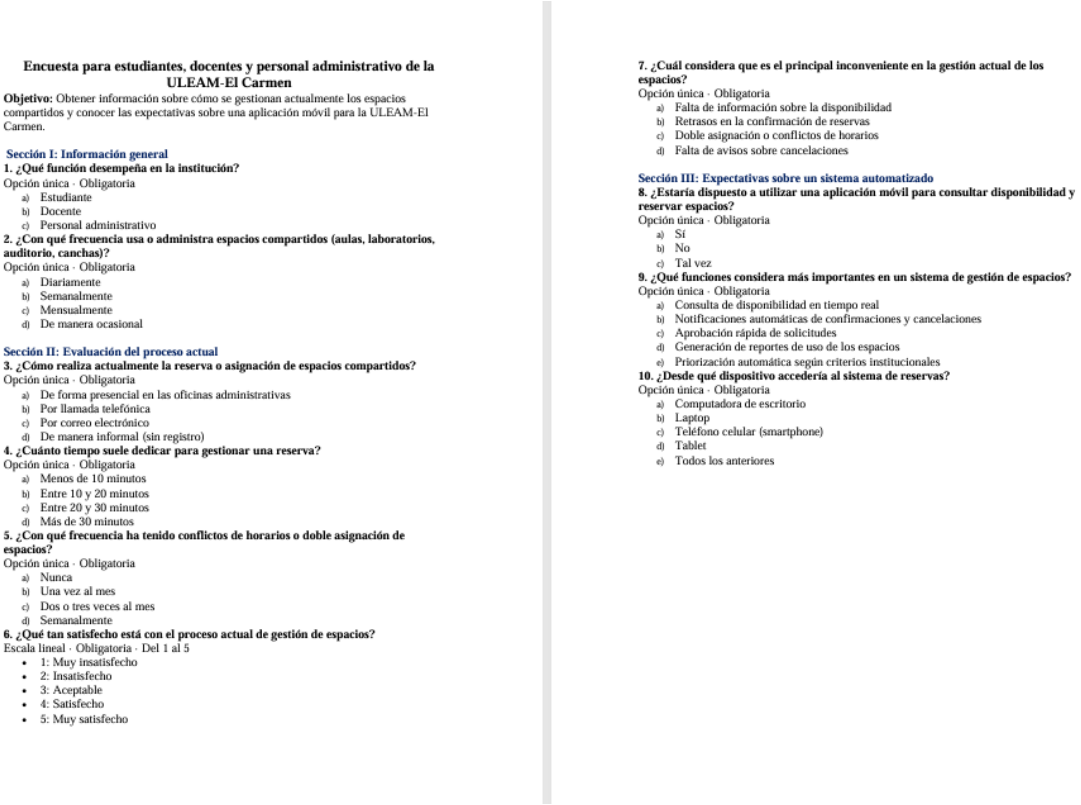
Quisaguano Collaguazo, Mg. L. R., Pallasco Venegas, Mg. M. S., Andaluz Guerrero, A. A., Martínez Freire, M. N., & Corrales Beltrán, Mg. S. H. (2022). DESARROLLO HÍBRIDO CON FLUTTER. *Ciencia Latina Revista Científica Multidisciplinar*, *6*(4), 4594–4609. https://doi.org/10.37811/CL_RCM.V6I4.2959

Ruiz Muñoz, G. F., & Santos Tomalá, J. I. (2024). Eficiencia administrativa y procesos de matriculación en instituciones de educación superior. *Revista Social Fronteriza*, *4*(2), e42240. https://doi.org/10.59814/RESOFRO.2024.4(2)240

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game*.

ANEXOS

Anexo A: Instrumento de Encuesta



Encuesta para estudiantes, docentes y personal administrativo de la ULEAM-El Carmen

Objetivo: Obtener información sobre cómo se gestionan actualmente los espacios compartidos y conocer las expectativas sobre una aplicación móvil para la ULEAM-El Carmen.

Sección I: Información general

1. ¿Qué función desempeña en la institución?
Opción única · Obligatoria
a) Estudiante
b) Docente
c) Personal administrativo

2. ¿Con qué frecuencia usa o administra espacios compartidos (aulas, laboratorios, auditorio, canchas)?
Opción única · Obligatoria
a) Diariamente
b) Semanalmente
c) Mensualmente
d) De manera ocasional

Sección II: Evaluación del proceso actual

3. ¿Cómo realiza actualmente la reserva o asignación de espacios compartidos?
Opción única · Obligatoria
a) De forma presencial en las oficinas administrativas
b) Por llamada telefónica
c) Por correo electrónico
d) De manera informal (sin registro)

4. ¿Cuánto tiempo suele dedicar para gestionar una reserva?
Opción única · Obligatoria
a) Menos de 10 minutos
b) Entre 10 y 20 minutos
c) Entre 20 y 30 minutos
d) Más de 30 minutos

5. ¿Con qué frecuencia ha tenido conflictos de horarios o doble asignación de espacios?
Opción única · Obligatoria
a) Nunca
b) Una vez al mes
c) Dos o tres veces al mes
d) Semanalmente

6. ¿Qué tan satisfecho está con el proceso actual de gestión de espacios?
Escala lineal · Obligatoria · Del 1 al 5
- 1: Muy insatisfecho
- 2: Insatisfecho
- 3: Aceptable
- 4: Satisfecho
- 5: Muy satisfecho

7. ¿Cuál considera que es el principal inconveniente en la gestión actual de los espacios?
Opción única · Obligatoria
a) Falta de información sobre la disponibilidad
b) Retrasos en la confirmación de reservas
c) Doble asignación o conflictos de horarios
d) Falta de avisos sobre cancelaciones

Sección III: Expectativas sobre un sistema automatizado

8. ¿Estaría dispuesto a utilizar una aplicación móvil para consultar disponibilidad y reservar espacios?
Opción única · Obligatoria
a) Sí
b) No
c) Tal vez

9. ¿Qué funciones considera más importantes en un sistema de gestión de espacios?
Opción única · Obligatoria
a) Consulta de disponibilidad en tiempo real
b) Notificaciones automáticas de confirmaciones y cancelaciones
c) Aprobación rápida de solicitudes
d) Generación de reportes de uso de los espacios
e) Priorización automática según criterios institucionales

10. ¿Desde qué dispositivo accedería al sistema de reservas?
Opción única · Obligatoria
a) Computadora de escritorio
b) Laptop
c) Teléfono celular (smartphone)
d) Tablet
e) Todos los anteriores

Anexo 1 Encuesta aplicada a estudiantes, docentes y personal administrativo

Anexo B: Instrumento de Entrevista

Entrevista dirigida al coordinador académico y docentes con alta demanda de espacios

Objetivo: Recopilar información detallada sobre la forma en que actualmente se administran los espacios compartidos, las dificultades que se presentan durante el proceso y las expectativas respecto a la implementación de una aplicación móvil para la gestión de reservas en la ULEAM-El Carmen.

1. ¿Podría explicar cómo se realiza actualmente el proceso para solicitar y asignar espacios compartidos, como aulas, laboratorios o auditorios? *(Relacionada con la pregunta 3 de la encuesta)*
2. ¿Qué herramientas o métodos utiliza para registrar y controlar las reservas o asignaciones de estos espacios?
3. En promedio, ¿cuánto tiempo dedica cada semana a gestionar las solicitudes de reserva y coordinar la asignación de espacios? *(Relacionada con la pregunta 4 de la encuesta)*
4. Desde su experiencia, ¿cuáles son las principales dificultades que enfrenta en el proceso de gestión de espacios? *(Relacionada con la pregunta 7 de la encuesta)*
5. ¿Podría compartir un caso reciente en el que se haya presentado un conflicto de horarios o una doble asignación de espacios? ¿Cómo se resolvió? *(Relacionada con la pregunta 5 de la encuesta)*
6. ¿Qué espacios considera que tienen un bajo nivel de utilización y cuáles cree que son las principales causas de esta situación?
7. ¿Qué funcionalidades considera esenciales o prioritarias en una aplicación móvil para la gestión de reservas de espacios? *(Relacionada con la pregunta 9 de la encuesta)*
8. ¿Qué posibles riesgos o desafíos considera que podrían presentarse al implementar un sistema automatizado para la gestión de espacios? *(Relacionada con la pregunta 8 de la encuesta)*
9. ¿Cómo considera que el sistema debería gestionar situaciones especiales, como eventos institucionales, mantenimientos de emergencia o actividades no programadas?
10. ¿Qué aspectos o indicadores tomaría en cuenta para evaluar si un sistema automatizado mejora la gestión de espacios en comparación con el proceso actual?

Anexo 2 Entrevista dirigida al coordinador académico y docentes con alta demanda de espacios

Anexo C: Certificado de coincidencia académica

Certificado de análisis
Compilatio Magister+ | ULEAM-ECU

COMPILATIO5 Miranda Rey
ID : 5533d4f622d8ca58d47fa0e94debb7343a085d25

9%
Textos sospechosos

Nombre del fichero : COMPILATIO5 Miranda Rey.txt
Tamaño del archivo original : 3,63 MB
Número de palabras : 18.065

Depositante : RENELMO WLADIMIR MINAYA MACIAS
Fecha de depósito : 7 de agosto de 2026
Tipo de carga : interface
fecha de fin de análisis : 7 de agosto de 2026

Resumen (sección 1/2)

Localización de los textos sospechosos en el documento :

Incluido en el porcentaje de textos sospechosos :

Similitudes 1%
Pasajes con similitudes a fuentes encontradas en diferentes colecciones.

Detección de IA 8%
Textos estilísticamente próximos a un texto generado por una IA.
Este índice es un indicador y no una prueba. Comprueba con el autor si domina los conocimientos mencionados en el documento.

Idiomas no reconocidos 0%
Pasajes en los que parte del vocabulario utilizado no forma parte del diccionario de la lengua.
Puede tratarse de un intento del autor de modificar el texto para evitar ser detectado.

No incluido en el porcentaje de textos sospechosos :

Textos entre comillas <1%
Pasajes entre comillas, a menudo indicativos de una cita.

UNIVERSIDAD LAICA "ELOY ALFARO" DE MANABÍ
EXTENSIÓN EN EL CARMEN
Uleam
COMISIÓN ACADÉMICA
EL CARMEN - MANABÍ

Anexo 3 Certificado de Coincidencia Académica

Glosario

A

Administrador: Rol de usuario con el máximo nivel de acceso en el sistema; puede gestionar espacios, aprobar o rechazar reservas, generar reportes y administrar usuarios.

API REST: Interfaz que permite el intercambio de datos entre la aplicación móvil y el backend mediante solicitudes HTTP (GET, POST, PUT, DELETE).

Aplicación Móvil: Programa de software diseñado para ejecutarse en dispositivos portátiles, que en este proyecto funciona como punto de acceso principal para consultar y gestionar reservas.

Arquitectura Cliente-Servidor: Modelo en el que el cliente (aplicación móvil) realiza solicitudes que son procesadas por el servidor (backend), el cual responde con la información solicitada.

Arquitectura Offline-first: Paradigma de diseño que prioriza el funcionamiento de la aplicación sin conexión a internet, almacenando los datos localmente y sincronizando los cambios cuando se restablece la conectividad.

B

Backend: Parte del sistema que gestiona la lógica de negocio, el acceso a la base de datos y las operaciones del servidor.

Base de Datos: Espacio estructurado donde se almacena y organiza la información del sistema, como usuarios, espacios físicos, reservas y bloqueos de horario.

Bcrypt: Función de hash utilizada para cifrar las contraseñas de los usuarios antes de almacenarlas en la base de datos, dificultando los ataques por fuerza bruta.

C

Caché Local: Copia de los datos del servidor almacenada en el dispositivo móvil, utilizada para consultar espacios y reservas cuando no hay conexión a internet.

CRUD: Conjunto de operaciones básicas (Crear, Leer, Actualizar y Eliminar) que permiten gestionar los registros de una base de datos.

D

Dart: Lenguaje de programación desarrollado por Google sobre el cual está construido Flutter, utilizado para implementar la lógica de la aplicación móvil.

Despliegue Continuo: Práctica mediante la cual cada cambio publicado en el repositorio del proyecto se compila y publica automáticamente en el entorno de producción, sin intervención manual.

Diagrama de Casos de Uso: Representación gráfica que agrupa las funcionalidades disponibles para cada tipo de usuario del sistema.

Diagrama de Clases: Representación gráfica de las entidades principales del sistema y de las relaciones existentes entre ellas.

Diagrama Entidad-Relación: Representación gráfica de las tablas de la base de datos y de las relaciones que existen entre ellas.

DNS: Sistema que resuelve nombres de dominio en direcciones IP; en este proyecto se utiliza para verificar de forma confiable si existe una conexión real a internet.

E

Endpoint: Dirección específica de una API a la que se envían las solicitudes para ejecutar una operación del sistema, como iniciar sesión o crear una reserva.

Express: Framework minimalista construido sobre Node.js que facilita la creación de la API REST del backend mediante un sistema de rutas y middlewares.

F

Firebase Cloud Messaging (FCM): Servicio desarrollado por Google que permite enviar notificaciones push a los dispositivos móviles de forma confiable, incluso cuando la aplicación no está activa.

Flutter: Framework de desarrollo de aplicaciones móviles multiplataforma creado por Google, utilizado para construir la interfaz y la lógica de la aplicación móvil del proyecto.

Frontend: Parte del sistema con la que interactúa directamente el usuario; en este proyecto corresponde a la aplicación móvil desarrollada en Flutter.

G

GitHub: Plataforma utilizada para el control de versiones y el resguardo del código fuente del proyecto, desde la cual se despliega automáticamente el backend.

I

Incremento: Artefacto de la metodología Scrum que corresponde al producto funcional entregable al finalizar cada sprint.

J

JSON Web Token (JWT): Estándar utilizado para generar tokens de autenticación que permiten verificar la identidad de un usuario sin necesidad de mantener sesiones activas en el servidor.

M

Middleware: Función que se ejecuta entre la solicitud y la respuesta de una API, utilizada para validar sesiones y permisos antes de permitir el acceso a una ruta.

MySQL: Sistema de gestión de bases de datos relacional utilizado para almacenar de forma centralizada la información de usuarios, espacios, reservas y bloqueos.

N

Node.js: Entorno de ejecución de JavaScript del lado del servidor, utilizado para construir el backend del sistema.

Notificaciones Push: Mensajes enviados desde el servidor a un dispositivo móvil sin que el usuario tenga que iniciar la comunicación, usados para avisar sobre confirmaciones, cancelaciones o cambios en una reserva.

P

PostgreSQL: Sistema de gestión de bases de datos relacional de código abierto, referenciado en el marco teórico como alternativa para el almacenamiento centralizado de datos.

Proceso de Reserva Digital: Secuencia de etapas —consulta de disponibilidad, solicitud, validación automática y confirmación— que sigue una reserva dentro del sistema.

Product Backlog: Lista priorizada de historias de usuario que orienta el trabajo de desarrollo del proyecto dentro de la metodología Scrum.

Product Owner: Rol de Scrum responsable de priorizar el Product Backlog, validar los entregables y representar los intereses de los usuarios finales.

Pruebas de Caja Blanca: Pruebas que evalúan las rutas internas de decisión del código, verificando que cada rama condicional produzca el resultado esperado.

Pruebas de Caja Negra: Pruebas que verifican el comportamiento observable de una función desde la perspectiva del usuario, sin considerar la lógica interna de su implementación.

Pruebas de Usabilidad: Pruebas que evalúan la facilidad con la que los usuarios reales completan tareas dentro del sistema.

R

Railway: Plataforma en la nube donde se despliegan en producción el backend y la base de datos MySQL del sistema.

RBAC (Control de Acceso Basado en Roles): Mecanismo de seguridad que restringe las funcionalidades disponibles según el rol del usuario autenticado: estudiante, docente o administrador.

Reserva: Solicitud realizada por un usuario para el uso de un espacio físico durante un horario determinado, la cual puede quedar pendiente, confirmada o cancelada.

Roles de Usuario: Niveles de acceso definidos dentro del sistema —estudiante, docente y administrador—, cada uno con funcionalidades y prioridades distintas.

S

Scrum: Metodología ágil de desarrollo de software organizada en sprints, utilizada para planificar y ejecutar el proyecto.

Scrum Master: Rol de Scrum encargado de facilitar las ceremonias del equipo, eliminar impedimentos y garantizar el cumplimiento de las prácticas ágiles.

Sincronización Bidireccional: Mecanismo que permite el intercambio de datos entre el dispositivo móvil y el servidor central en ambas direcciones, actualizando los cambios realizados por cualquiera de las dos partes.

Socket.io: Biblioteca que permite la comunicación en tiempo real entre el backend y el cliente mediante eventos, utilizada para notificar al instante cambios en las reservas.

Sprint: Periodo de tiempo fijo —dos semanas en este proyecto— durante el cual el equipo desarrolla un conjunto de funcionalidades planificadas.

Sprint Backlog: Conjunto de tareas seleccionadas del Product Backlog que se desarrollarán durante un sprint específico.

SQL: Lenguaje utilizado para consultar y manipular la información almacenada en una base de datos relacional.

SQLite: Motor de base de datos relacional ligero que opera como una biblioteca integrada en la aplicación móvil, sin requerir un servidor externo, usado para el almacenamiento local en modo offline.

SUS (System Usability Scale): Cuestionario estandarizado utilizado para medir la percepción de usabilidad de un sistema por parte de sus usuarios.

T

Token: Cadena de caracteres cifrada que permite verificar la identidad de un usuario autenticado sin necesidad de enviar sus credenciales en cada solicitud.

U

UML (Unified Modeling Language): Lenguaje de modelado utilizado para representar gráficamente la estructura y el comportamiento del sistema mediante diagramas.

Usabilidad: Grado en el que un sistema permite a los usuarios realizar sus tareas de forma sencilla, eficiente y satisfactoria.