AcessLab

Jhojan Arlex Brausin Rodríguez 825671
Wilder Gutiérrez Molina 825637
Gabriela Morales Torres 825255
Katerine Morales Vásquez 826149

Mag. Mayra Alexandra Amador Patiño

Universidad Cooperativa De Colombia
Facultad De Ingeniería
Ingeniería De Sistemas
Villavicencio – Meta
2025

Resumen

La Universidad Cooperativa de Colombia sede Villavicencio enfrenta limitaciones en su proceso de gestión de reservas de laboratorios, actualmente realizado mediante trámites físicos y presenciales. Este método genera pérdida de tiempo, errores en las asignaciones y falta de visibilidad en la disponibilidad de recursos, lo cual afecta la experiencia académica. Para superar estas deficiencias, se propone el desarrollo de un sistema digital de reservas que incorpore operaciones CRUD para la administración de usuarios, laboratorios y equipos, así como la integración de herramientas PL/SQL para validar horarios, generar reportes de ocupación y calcular el uso de equipos por semestre. La solución se implementará como una aplicación de escritorio con interfaz intuitiva, garantizando una gestión eficiente, confiable y moderna de los recursos académicos.

Palabras clave: gestión de reservas, laboratorios, automatización, PL/SQL, sistema digital, universidad.

Abstract

The Universidad Cooperativa de Colombia, Villavicencio campus, faces limitations in its laboratory reservation management process, which is currently carried out through physical and in-person procedures. This method results in time loss, allocation errors, and a lack of visibility regarding resource availability, which negatively impacts the academic experience. To overcome these deficiencies, the development of a digital reservation system is proposed, incorporating CRUD operations for the management of users, laboratories, and equipment, as well as the integration of PL/SQL tools to validate schedules, generate occupancy reports, and calculate equipment usage per semester. The solution will be implemented as a desktop application with an intuitive interface, ensuring efficient, reliable, and modern management of academic resources.

Keywords: reservation management, laboratories, automation, PL/SQL, digital system, university.

Tabla de contenido

Tabla de Figuras

1. **Introducción**

La gestión de reservas de laboratorios en la universidad Cooperativa sede Villavicencio enfrenta un desafío significativo, marcado por un proceso manual y obsoleto. El sistema actual, basado en formatos físicos y trámites presenciales, es propenso a errores, causa una considerable pérdida de tiempo y carece de la visibilidad necesaria sobre la disponibilidad de recursos. Esta ineficiencia no solo genera inconformidad entre los usuarios, sino que también limita la utilización efectiva de los recursos académicos.

Para superar estas limitaciones, se propone un sistema de reservas de laboratorios digital que optimice la gestión de recursos a través de operaciones CRUD para administrar usuarios, laboratorios y equipos. La integración de herramientas PL/SQL automatizará tareas críticas, como la validación de horarios para evitar conflictos y la generación de reportes de ocupación. Adicionalmente, el sistema calculará el uso de equipos por semestre, ofreciendo una visión clara del aprovechamiento de los recursos. La solución se materializará en una aplicación de escritorio, diseñada para que las facultades gestionen los laboratorios de manera más eficiente, mejorando así la experiencia del usuario en el entorno académico.

2. Planteamiento del problema

2.1 Descripción del problema

En la actualidad, la universidad Cooperativa de Colombia sede Villavicencio enfrenta dificultades en la gestión de reservas de laboratorios. El proceso tradicional requiere que los estudiantes o docentes diligencien un formato en papel o documento físico, el cual debe presentarse de manera presencial en la oficina correspondiente. Este procedimiento no solo resulta poco práctico, sino que también genera molestias por la pérdida de tiempo, la posibilidad de errores humanos en la asignación y la falta de visibilidad sobre la disponibilidad real de los laboratorios y equipos.

Estas limitaciones impiden que los recursos académicos se utilicen de manera eficiente y provocan inconformidad entre los usuarios, quienes esperan un sistema moderno, ágil y confiable.

2.2 Justificación

La implementación de un sistema digital para la gestión de reservas de laboratorios en la Universidad Cooperativa de Colombia sede Villavicencio se justifica como una necesidad estratégica urgente que transformará significativamente la experiencia académica y la eficiencia operativa institucional.

El proyecto se fundamenta en resolver las limitaciones críticas del proceso actual, que genera pérdida de tiempo valioso, errores en asignaciones y frustración en la comunidad universitaria. La digitalización eliminará estos obstáculos mediante una plataforma moderna que permitirá realizar reservas instantáneas desde cualquier dispositivo, visualizar disponibilidad en tiempo real y recibir confirmaciones automáticas.

Los beneficios principales incluyen la optimización del tiempo académico al eliminar desplazamientos y esperas innecesarias, la reducción significativa de errores operativos mediante automatización, y la maximización del uso de recursos educativos a través de una gestión inteligente de la disponibilidad. El sistema proporcionará trazabilidad completa de todas las transacciones, mejorando el control administrativo y la transparencia del proceso.

3. Objetivos

3.1 Objetivo general

Desarrollar un sistema para la gestión digital de reservas de laboratorios en la Universidad Cooperativa de Colombia sede Villavicencio, que permita optimizar la administración de recursos académicos mediante la automatización de procesos, la reducción de errores y la mejora de la experiencia de los usuarios.

3.2 Objetivos específicos

- Implementar operaciones CRUD para la gestión de usuarios, laboratorios y equipos, garantizando un control estructurado de los recursos disponibles.
- Integrar herramientas PL/SQL que automaticen la validación de horarios, evitando conflictos en la asignación de laboratorios y equipos.
- Desarrollar un módulo de generación de reportes de ocupación y cálculo del uso de equipos por semestre, que proporcione información clara y confiable sobre la utilización de los recursos académicos.
- Diseñar una interfaz intuitiva que facilite la interacción de estudiantes y docentes con el sistema.

4. Marco referencial

4.1 Estado del arte

El **Laboratory Information Management System (LIMS)** según el artículo "Research and Implementation of Big Data Technology Laboratory Equipment Reservation Management System", integra conceptos de gestión moderna y tecnología informática para administrar laboratorios de forma más eficiente. Este tipo de sistemas busca facilitar la reserva de equipos, la gestión de usuarios**,** el registro de actividades docentes y la generación de reportes estadísticos sobre el uso de los recursos.

En China, varias universidades han implementado redes de gestión de laboratorios**;** sin embargo, muchas soluciones se limitan a programar cursos experimentales y no resuelven plenamente la falta de mecanismos de intercambio de información ni de modelos de gestión innovadores**.** Se plantea la necesidad de sistemas abiertos, con modos de gestión innovadore**s** y capacidad para cubrir las actividades docentes diarias (Wang, Wei, Cao, & Liu, 2019).

El diseño de datos reportado en dicho sistema incluye tablas para usuarios, detalles de usuario, equipos de laboratorio, controles de mantenimiento y solicitudes de reserva. Esta estructura permite garantizar integridad de la información y optimizar la trazabilidad de las reservas**.**

El artículo afirma que el sistema desarrollado mejoró la eficiencia de trabajo en laboratorio universitario, aunque reconoce limitaciones y la necesidad de ajustes y actualizaciones continuas ante nuevas exigencias de gestión (Wang, Wei, Cao, & Liu, 2019). Esto respalda la adopción de un ciclo de mejora continua y métricas de uso

4.2 Marco teórico

Bases de datos relacionales

Una base de datos se puede definir como (Piattini et ál, 2006): Una colección o depósito de datos integrados con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de estas, y su definición y descripción únicas para cada tipo de dato, han de estar almacenadas junto con los mismos. Los procedimientos

de actualización y recupera-ción, comunes y bien determinados, habrán de ser capaces de conservar la seguridad (integridad, confidencialidad y disponibilidad) del conjunto de los datos.

Oracle Database

Oracle Database es un sistema de gestión de bases de datos relacionales (RDBMS) desarrollado por Oracle Corporation. Su función principal es almacenar, organizar y proteger grandes volúmenes de datos mediante estructuras como tablas, columnas y filas, permitiendo el acceso solo a usuarios autorizados. Fundada en 1977, Oracle sigue siendo líder mundial en bases de datos empresariales, gracias a su rendimiento, seguridad y flexibilidad para operar tanto en entornos locales como en la nube. Actualmente, sus versiones más destacadas son la 19c (soporte prolongado) y la 21c (innovación).

PL/SQL (Procedural Language / SQL)

Es una extensión de Oracle al lenguaje SQL que permite programar procedimientos, funciones y triggers dentro de bases de datos relacionales. Se caracteriza por ser estructurado, legible y accesible, además de estar integrado directamente en la base de datos Oracle, lo que facilita la automatización y el control de procesos. No funciona como lenguaje independiente, sino como un lenguaje embebido que se ejecuta dentro del entorno de Oracle, ofreciendo un alto rendimiento y herramientas específicas que mejoran la eficiencia en la gestión de datos. Según (Feuerstein, S., & Pribyl, B. 2005)

Restricciones de integridad (Integrity Constraints)

Las restricciones de integridad en bases de datos relacionales surgieron poco después del propio modelo relacional, hacia mediados de la década de 1970. Se definen como predicados booleanos que aseguran la validez y consistencia de los datos, impidiendo estados inválidos en la base de datos. Su importancia radica en que son teóricamente sólidas y, a la vez, de gran valor práctico, pues permiten preservar la integridad de la información. Con el tiempo, los principales sistemas de bases de datos incorporaron cada vez más soporte para distintos tipos de restricciones (clave primaria, foránea, unicidad, entre otras), convirtiéndolas en un pilar fundamental del diseño relacional. Según (Ceri, S., Cochrane, R., & Widom, J. 2000)

Triggers

Los triggers o disparadores nacieron como una extensión del concepto de restricciones de integridad, al plantearse la necesidad de acciones automáticas ante la violación de una regla o la ocurrencia de un evento en la base de datos. Se definen como reglas del tipo evento-condición-acción (ECA), también conocidas como reglas activas.

Aunque inicialmente no tuvieron la misma acogida que las restricciones, a finales de los años 80 y principios de los 90 se consolidaron dentro del campo de las bases de datos activas, convirtiéndose en un tema de investigación destacado y siendo adoptados por sistemas comerciales. En la práctica, los triggers permiten automatizar respuestas (como bloquear reservas duplicadas o registrar auditorías), lo que los convierte en una herramienta esencial para garantizar la coherencia de los datos y reducir la intervención manual. Según (Ceri, S., Cochrane, R., & Widom, J. 2000)

Índices

El propósito principal de un índice en una base de datos es optimizar la velocidad de recuperación de información, especialmente cuando las tablas contienen una gran cantidad de registros o cuando se ejecutan operaciones de ordenamiento y agrupamiento. Los índices permiten que el sistema localice los datos de manera más eficiente sin necesidad de recorrer toda la tabla.

Es fundamental determinar los campos sobre los cuales resulta conveniente crear un índice, priorizando aquellos que se utilizan con frecuencia en las búsquedas, como las claves primarias, las claves foráneas o los campos empleados para combinar tablas.

No se recomienda crear índices sobre columnas que se consultan con poca frecuencia o en tablas de tamaño reducido, ya que esto podría generar un consumo innecesario de recursos y no aportar beneficios significativos al rendimiento del sistema.

Funciones en PL/SQL

Son bloques de código almacenados que permiten encapsular una secuencia de instrucciones para realizar tareas específicas y devolver un valor como resultado. Estas funciones

son útiles para modularizar el código, mejorar la reutilización y simplificar el mantenimiento de los programas dentro de la base de datos.

Existen dos tipos principales de funciones PL/SQL: las funciones escalares y las funciones interconectadas.

- **Funciones escalares:** devuelven un único valor y pueden ser invocadas en cualquier contexto donde una expresión sea válida, como en sentencias SELECT, cláusulas WHERE o asignaciones de variables. Al evaluarse, la función devuelve un valor que sustituye la expresión en la que fue llamada.
- **Funciones interconectadas:** generan un conjunto de resultados que puede ser utilizado dentro de la cláusula FROM de una sentencia SELECT, actuando de manera similar a una tabla derivada o vista temporal. Estas funciones permiten procesar y devolver múltiples filas de datos de forma dinámica.

La creación de una función se realiza mediante la sentencia CREATE FUNCTION, donde se define su nombre, parámetros, tipo de retorno y cuerpo de ejecución. En caso de que se desee actualizar una función existente, puede utilizarse la opción OR REPLACE, que reemplaza la versión anterior sin necesidad de eliminarla previamente.

Si una función ya no es necesaria, puede eliminarse de la base de datos mediante la sentencia DROP FUNCTION.

Modelo relacional

El Modelo Relacional es un modelo de datos que se basa en la lógica de predicados y en la teoría de conjuntos, siendo uno de los más utilizados en los sistemas de gestión de bases de datos. Su estructura se organiza en tablas (también llamadas relaciones), formadas por filas o tuplas que representan registros, y columnas o campos que representan los atributos de esos registros. Cada tabla puede vincularse con otras mediante claves primarias y foráneas, lo que permite establecer relaciones entre conjuntos de datos distintos sin necesidad de duplicar información. Este modelo destaca por su simplicidad, consistencia y facilidad para realizar consultas mediante el lenguaje SQL (Structured Query Language). Gracias a esto, es ampliamente utilizado en

entornos empresariales donde se requiere integridad, seguridad y un manejo estructurado de grandes volúmenes de información.

Modelo orientado a objetos

El Modelo Orientado a Objetos propone una forma diferente de representar la información, combinando tanto los datos (atributos) como los comportamientos (métodos) dentro de una misma unidad denominada objeto. Este modelo se fundamenta en los principios de la programación orientada a objetos, destacando conceptos como clases, herencia, encapsulamiento y polimorfismo.

A diferencia del modelo relacional, donde los datos se dividen en tablas y registros, el modelo orientado a objetos permite representar entidades del mundo real de manera más natural y modular, ya que cada objeto puede interactuar con otros y mantener su propio estado y comportamiento.

Esto facilita el desarrollo de sistemas complejos, ya que los objetos pueden reutilizarse y extenderse, promoviendo la flexibilidad y escalabilidad del software. Sin embargo, la principal dificultad surge al conectar este modelo con bases de datos relacionales, generando lo que se conoce como desajuste de impedancia entre ambos enfoques.

ORM

El Mapeo Objeto–Relacional (Object Relational Mapping, ORM) es una técnica de programación que permite conectar el mundo de los objetos en los lenguajes de programación orientados a objetos con las bases de datos relacionales, las cuales almacenan información en tablas. En esencia, el ORM traduce los objetos del programa en registros que pueden ser almacenados en la base de datos, y realiza el proceso inverso cuando se necesita recuperar los datos.

Esto crea una especie de “base de datos virtual orientada a objetos” sobre un sistema relacional, haciendo posible que el programador trabaje con clases, herencia y polimorfismo sin preocuparse por las diferencias estructurales entre ambos modelos.

Los ORM automatizan el proceso de conversión entre objetos y registros, lo que evita tener que escribir código SQL manualmente para operaciones comunes. De esta manera, facilitan la comunicación entre la aplicación y la base de datos, incrementando la rapidez en el desarrollo, mejorando la reutilización del código, la abstracción de la capa de datos y reforzando la seguridad y el mantenimiento del sistema.

Sin embargo, esta capa de abstracción también presenta desventajas, como una ligera pérdida de rendimiento debido a la traducción constante de consultas y la curva de aprendizaje que requiere entender su funcionamiento. Aun así, su implementación se ha vuelto un estándar en la mayoría de frameworks modernos gracias a su eficiencia y a los patrones de diseño que integra, como Repository y Active Record, los cuales organizan la persistencia de los datos de manera estructurada y coherente con la arquitectura de software.

Django

Django es un software que permite desarrollar aplicaciones web de forma rápida y eficiente. La mayoría de las aplicaciones web comparten varias funciones comunes, como la autenticación, la recuperación de información de una base de datos y la gestión de cookies. Los desarrolladores deben codificar funcionalidades similares en cada aplicación web que crean. Django facilita esta tarea al agrupar las diferentes funciones en una amplia colección de módulos reutilizables, conocida como framework de aplicaciones web. Los desarrolladores utilizan el framework web Django para organizar y escribir su código de forma más eficiente y reducir significativamente el tiempo de desarrollo web.

API (Interfaz de Programación de Aplicaciones)

Es un conjunto de reglas y protocolos que permite la comunicación entre diferentes sistemas o aplicaciones para intercambiar datos y funcionalidades. Su objetivo principal es facilitar el desarrollo de software, permitiendo a los programadores integrar servicios o recursos existentes sin tener que crearlos desde cero.

Además, las API ofrecen una forma segura de compartir información, ya que exponen solo los datos necesarios para cada solicitud, manteniendo protegidos los detalles internos del sistema.

También pueden utilizarse para compartir o comercializar funciones y datos entre distintas áreas o empresas.

Finalmente, la documentación de una API actúa como un manual técnico que explica cómo usarla correctamente, mejorando la experiencia del desarrollador y garantizando su correcta implementación.

Flutter

Flutter es un framework de desarrollo móvil multiplataforma creado por Google, cuya primera versión estable se lanzó en 2018. Su objetivo principal es ofrecer alto rendimiento y una excelente experiencia de usuario. Utiliza el lenguaje Dart, también desarrollado por Google, el cual se compila directamente a código máquina (AOT) para optimizar la velocidad de ejecución.

En Flutter, todo es un widget: botones, textos, menús, colores o incluso la propia aplicación. Estos widgets pueden modificarse dinámicamente, y muchos incluyen animaciones integradas que mejoran la interacción con el usuario.

Actualmente, Flutter permite desarrollar aplicaciones para Android e iOS, pero su expansión incluye plataformas como web, escritorio (Windows, macOS, Linux)**,** dispositivos embebidos e incluso televisores o relojes inteligentes.

El framework se compone de dos partes principales:

- Framework, escrito en Dart, donde los desarrolladores crean la aplicación y su interfaz.
- Engine, escrito en C++, que ejecuta la aplicación y gestiona la comunicación con el hardware sin que el programador deba intervenir.

Arquitectura en capas

La Arquitectura de Capas es un estilo de diseño de software que organiza el sistema en niveles lógicos, donde cada capa cumple una función específica y se comunica únicamente con las capas cercanas. Este enfoque promueve la modularidad, mantenibilidad y escalabilidad, ya que separa las responsabilidades dentro de la aplicación.

Generalmente, se compone de cuatro capas principales:

- Presentación: gestiona la interfaz de usuario y la interacción con el sistema.
- Lógica de Aplicación: contiene las reglas de negocio y el procesamiento de datos.
- Acceso a Datos: maneja la comunicación con bases de datos u otras fuentes de información.
- Infraestructura: ofrece servicios de soporte como seguridad, configuración o registros.

Al mantener una comunicación estrictamente jerárquica entre capas, esta arquitectura permite modificar o reemplazar partes del sistema sin afectar su funcionamiento global, garantizando así una estructura más ordenada y flexible.

Arquitectura Rest

La arquitectura REST, conocida como Transferencia de Estado Representacional, es un estilo de diseño para sistemas distribuidos que facilita la comunicación entre aplicaciones a través del protocolo HTTP. En este modelo, los recursos del sistema, como datos o servicios, se identifican mediante URLs y se manipulan utilizando métodos estándar como GET, POST, PUT y DELETE. Cada solicitud en REST es independiente, lo que significa que el servidor no almacena información del estado del cliente entre peticiones. Esta característica la convierte en una arquitectura ligera, escalable y eficiente para el desarrollo de servicios web y APIs modernas.

4.3 Marco legal

Fundamento Normativo

El desarrollo del sistema de reservas de laboratorio se fundamenta en un marco normativo jerárquico que garantiza el adecuado tratamiento de datos personales en el contexto educativo colombiano. La Constitución Política de Colombia establece en sus artículos 15 y 20 los derechos fundamentales a la intimidad personal y familiar y el derecho a la información, constituyendo la base constitucional para la protección de datos personales (Constitución Política de Colombia, 1991).

La normativa específica en materia de protección de datos se encuentra desarrollada en la Ley 1581 de 2012, que establece el régimen general de protección de datos personales en Colombia. Esta ley tiene por objeto "desarrollar el derecho constitucional que tienen todas las personas a conocer, actualizar y rectificar las informaciones que se hayan recogido sobre ellas en

bases de datos o archivos" (Ley 1581 de 2012, art. 1). Los principios rectores establecidos incluyen legalidad, finalidad, libertad, veracidad, transparencia, acceso restringido, seguridad y confidencialidad (Ley 1581 de 2012, art. 4).

El Decreto 1377 de 2013 reglamenta parcialmente la Ley 1581 de 2012, estableciendo aspectos específicos sobre autorización del titular, políticas de tratamiento, ejercicio de derechos, transferencias internacionales y responsabilidad demostrada (Decreto 1377 de 2013). Este decreto define procedimientos concretos para la recolección de datos, especificando que "la recolección de datos deberá limitarse a aquellos datos personales que son pertinentes y adecuados para la finalidad para la cual son recolectados" (Decreto 1377 de 2013, art. 4).

A nivel institucional, la Universidad Cooperativa de Colombia adoptó mediante el Acuerdo 171 de 2014 sus políticas específicas de tratamiento de información y protección de datos personales. Este acuerdo establece que la universidad, como responsable del tratamiento de datos de estudiantes, empleados y miembros de la comunidad universitaria, debe aplicar los principios de legalidad y libertad en el uso, captura, recolección y tratamiento de datos personales (Acuerdo 171 de 2014, art. 1).

Responsable y Encargado del Tratamiento

Conforme a las definiciones establecidas en la Ley 1581 de 2012, se identifica como responsable del tratamiento a la Universidad Cooperativa de Colombia, persona jurídica de derecho privado con NIT 860002924-7, domiciliada en Medellín, Colombia, que en su calidad de institución de educación superior decide sobre la base de datos y el tratamiento de los datos personales (Acuerdo 171 de 2014). La universidad, como responsable, define las finalidades y medios del tratamiento, estableciendo las políticas y procedimientos aplicables al sistema de reservas.

El encargado del tratamiento corresponde al equipo desarrollador del sistema de reservas, que realiza el tratamiento de datos personales por cuenta del responsable. Esta figura, definida como "persona natural o jurídica que por sí misma o en asocio con otros, realice el tratamiento de datos personales por cuenta del responsable del tratamiento" (Ley 1581 de 2012, art. 3), opera bajo las directrices y políticas establecidas por la universidad, garantizando el cumplimiento de las obligaciones de seguridad y confidencialidad establecidas en la normativa.

Finalidad y Propósito del Sistema

El sistema de reservas de laboratorio se desarrolla con finalidades específicas y legítimas enmarcadas en la función educativa de la universidad. Las finalidades del tratamiento incluyen la gestión y programación eficiente de reservas de espacios y equipos de laboratorio, la verificación de identidad y autorización de usuarios según su rol institucional, y la garantía de seguridad y control en el uso de recursos académicos.

Estas finalidades se encuentran en concordancia con el principio de finalidad establecido en la Ley 1581 de 2012, según el cual "el tratamiento debe obedecer a una finalidad legítima de acuerdo con la Constitución y la Ley, la cual debe ser informada al titular" (art. 4, literal b). El tratamiento de datos se justifica en el ejercicio de la función educativa que desarrolla la universidad y en la necesidad de optimizar el uso de recursos académicos para beneficio de la comunidad universitaria.

Categorías de Datos Personales Tratados

El sistema procesa categorías específicas de datos personales necesarios para el cumplimiento de sus finalidades. Los datos de identificación académica incluyen el código institucional, nombre completo del usuario, programa académico o dependencia a la cual pertenece, y rol dentro de la institución. Esta información permite verificar la vinculación del usuario con la universidad y determinar sus niveles de acceso a los recursos del laboratorio.

Los datos de contacto comprenden el correo electrónico institucional como medio principal de comunicación y, de manera opcional, número telefónico para contacto directo. Adicionalmente, se registran datos específicos de cada reserva, incluyendo fecha y hora solicitada, duración estimada del uso, propósito académico específico de la actividad, equipos o recursos requeridos, y estado actual de la reserva.

Es importante señalar que el sistema no recolecta ni procesa datos sensibles según la definición del artículo 5 de la Ley 1581 de 2012, limitándose exclusivamente a información necesaria para los fines académicos y administrativos establecidos.

Principios Aplicables y Derechos de los Titulares

El tratamiento de datos personales en el sistema se rige por los principios establecidos en la Ley 1581 de 2012 y desarrollados en el Acuerdo 171 de 2014 de la universidad. El principio de

legalidad garantiza que el tratamiento se sujeta a las disposiciones legales vigentes, mientras que el principio de finalidad limita el uso de los datos a los propósitos académicos específicamente definidos.

Los usuarios del sistema, en su calidad de titulares de los datos, pueden ejercer los derechos fundamentales de acceso, rectificación, cancelación y oposición (derechos ARCO) establecidos en el artículo 8 de la Ley 1581 de 2012. Estos derechos incluyen conocer, actualizar y rectificar sus datos personales, solicitar prueba de la autorización otorgada, ser informados sobre el uso dado a su información, y revocar la autorización cuando no se respeten los principios y garantías legales.

El ejercicio de estos derechos se canaliza a través de los procedimientos institucionales de peticiones, quejas, reclamos y sugerencias (PQRS) establecidos por la universidad, garantizando respuesta oportuna dentro de los términos legales establecidos: máximo 10 días hábiles para consultas y 15 días hábiles para reclamos (Ley 1581 de 2012, arts. 14 y 15).

Medidas de Seguridad y Protección

La universidad implementa medidas técnicas, humanas y administrativas necesarias para garantizar la seguridad de los datos personales tratados en el sistema. Las medidas técnicas incluyen autenticación mediante credenciales institucionales integradas con el directorio activo de la universidad, cifrado de datos tanto en tránsito como en reposo utilizando estándares de seguridad reconocidos, y control granular de accesos basado en roles institucionales.

Las medidas administrativas comprenden la designación de personal autorizado con acceso restringido según su función específica, programas de capacitación en protección de datos para el personal técnico, y la implementación de políticas documentadas para el manejo seguro de la información. Adicionalmente, se establecen procedimientos de auditoría y monitoreo que permiten detectar y responder oportunamente a posibles incidentes de seguridad.

Estas medidas se alinean con las obligaciones establecidas en el artículo 17 de la Ley 1581 de 2012 para los responsables del tratamiento, especialmente el deber de "conservar la información bajo las condiciones de seguridad necesarias para impedir su adulteración, pérdida, consulta, uso o acceso no autorizado o fraudulento"

5 Resultados

5.1 Descripción de requisitos

Requisitos funcionales:

1. Gestión de Usuarios

RF-1.1 El sistema debe permitir el inicio de sesión y la autenticación de usuarios**.**

RF-1.2 El sistema debe permitir que solo usuarios autenticados accedan a las funciones internas.

RF-1.3 El sistema debe permitir registrar diferentes roles: Administrador y Solicitante

RF-1.4 El sistema debe permitir al Administrador crear, editar, eliminar y listar usuarios.

RF-1.5 El sistema debe permitir a cada usuario visualizar y editar su propio perfil.

RF-1.6 El sistema debe restringir el acceso a las secciones según el rol del usuario.

2. Gestión de Laboratorios

RF-2.1 El sistema debe permitir al Administrador registrar laboratorios.

RF-2.2 El sistema debe permitir listar y buscar laboratorios disponibles.

RF-2.3 El sistema debe permitir editar la información de un laboratorio.

RF-2.4 El sistema debe permitir marcar un laboratorio como disponible/no disponible.

3. Gestión de Objetos o Equipos

RF-3.1 El sistema debe permitir registrar los objetos o materiales asociados a los laboratorios.

RF-3.2 El sistema debe permitir listar, editar o eliminar objetos.

RF-3.3 El sistema debe permitir consultar el estado o disponibilidad de los objetos.

4. Gestión de Reservas (Solicitudes)

RF-4.1 El usuario Solicitante debe poder crear una solicitud de reserva indicando: Laboratorio, fecha y hora de inicio/fin, objetos necesarios.

RF-4.2 El sistema debe permitir al administrador listar todas las solicitudes registradas.

RF-4.3 El Administrador debe poder aprobar o rechazar las solicitudes.

RF-4.4 El sistema debe permitir agregar o eliminar participantes de una solicitud.

RF-4.5 El sistema debe validar que un laboratorio no esté doblemente reservado en el mismo horario.

Requisitos no funcionales

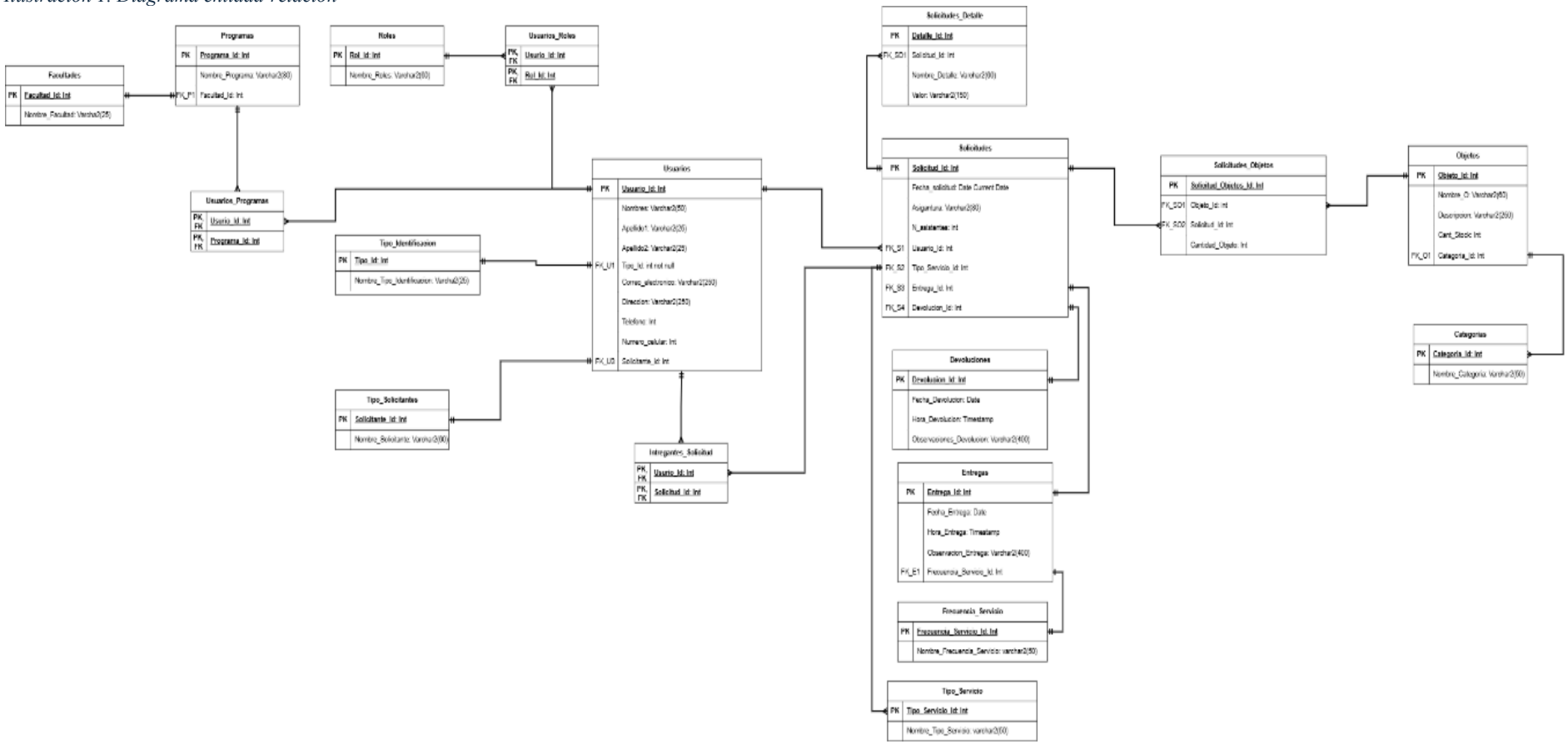
1. Rendimiento:
   El sistema debe responder a las solicitudes del usuario en un tiempo máximo de 3 segundos bajo carga normal.
2. Disponibilidad:
   El sistema debe estar disponible al menos el 99% del tiempo, garantizando el acceso constante a los usuarios.
3. Escalabilidad:
   La arquitectura debe permitir aumentar la capacidad del sistema (usuarios, datos o peticiones) sin afectar el rendimiento.
4. Seguridad:
   - La comunicación entre cliente y servidor debe realizarse mediante HTTPS.
   - Las contraseñas deben almacenarse cifradas.
   - Se deben implementar roles y permisos para controlar el acceso a la información
   5. Usabilidad:
      La interfaz debe ser intuitiva, accesible y compatible con dispositivos móviles y de escritorio.
   6. Mantenibilidad:
      El código debe estar modularizado y documentado, permitiendo actualizaciones o correcciones sin afectar otras partes del sistema.
   7. Portabilidad:
      La aplicación debe poder ejecutarse en diferentes sistemas operativos y navegadores sin necesidad de ajustes significativos.
   8. Confiabilidad:
      El sistema debe garantizar la integridad de los datos ante fallos inesperados o interrupciones en la conexión.
   9. Compatibilidad:
      Debe integrarse fácilmente con otros servicios o APIs externas mediante la arquitectura REST.

10. Escalabilidad técnica:

    La infraestructura debe soportar la migración a entornos en la nube sin necesidad de rediseñar la aplicación.

5.2 Modelado

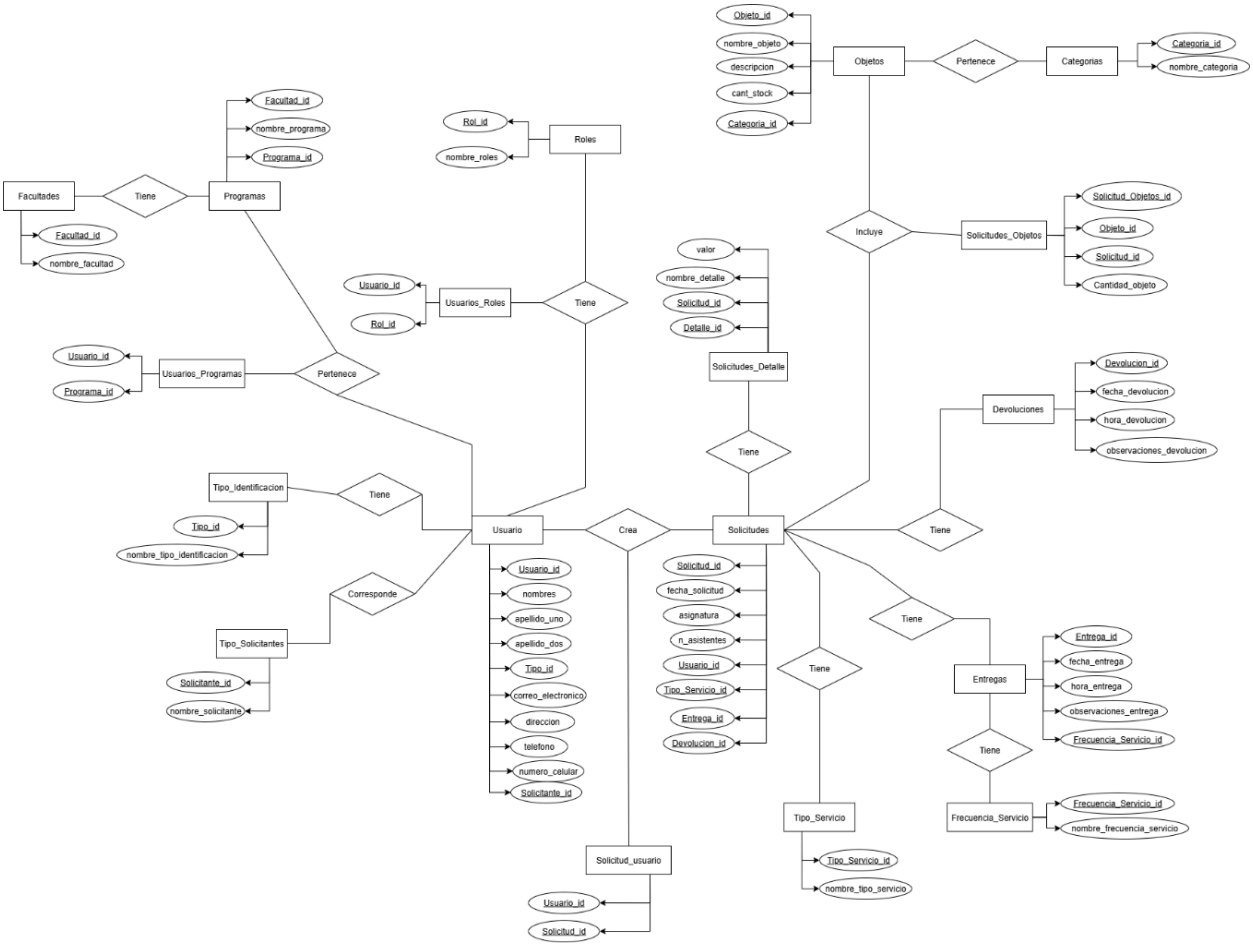
Ilustración 1. Diagrama entidad-relación



Fuente: Elaborado por los autore

Nota. Diagrama entidad-relación de la base de datos propuesta para el proyecto

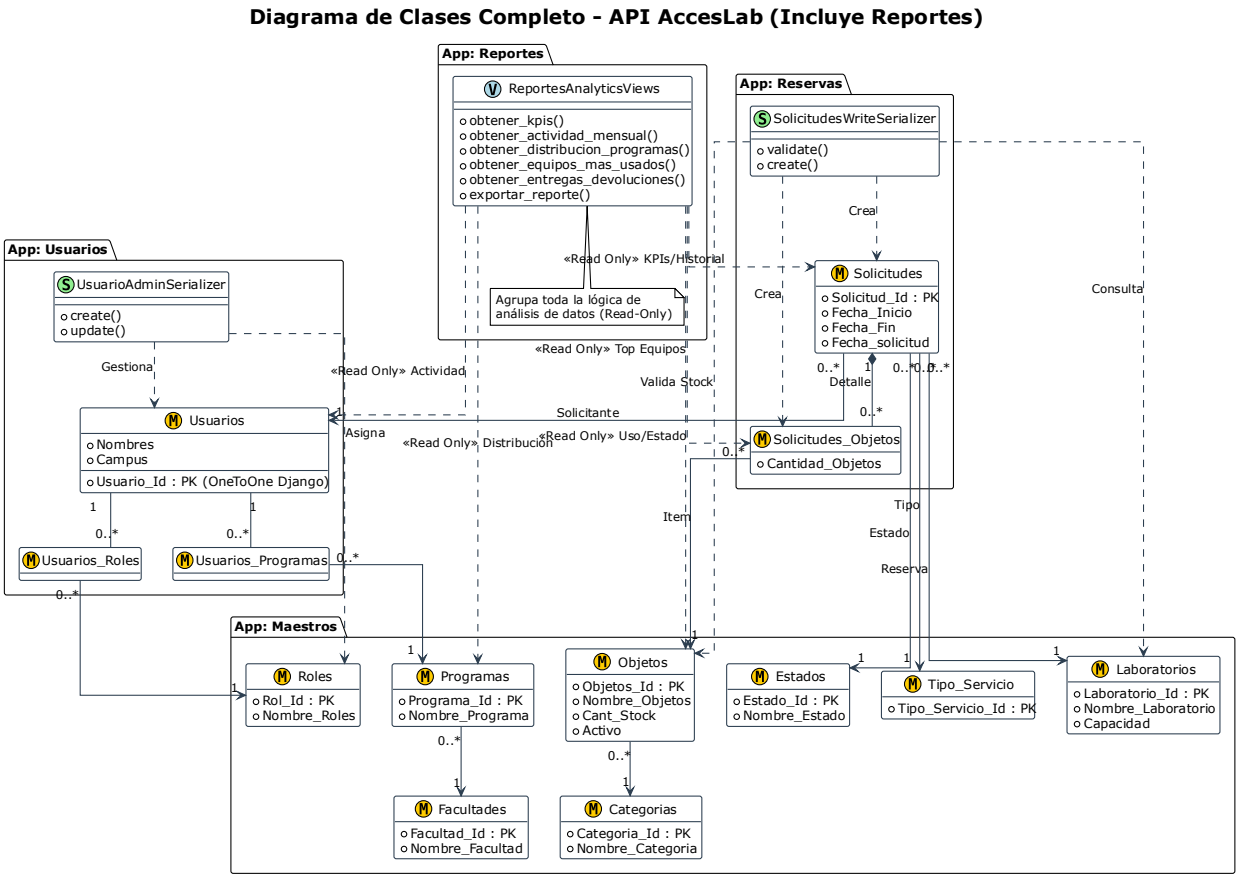
Ilustración 2. Diagrama relacional



Fuente: Elaborado por los autores

Nota. Diagrama relacional de la base de datos

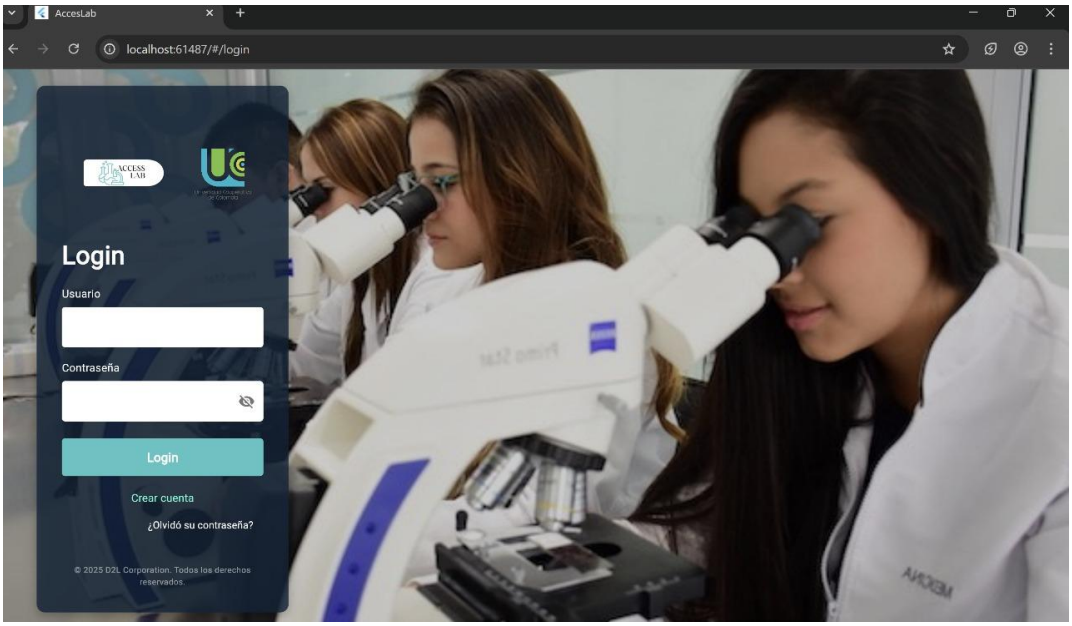
Ilustracion : diagrama de clases



Nota. *Diagrama de clases del proyecto*

5.3 Diseño de interfaz

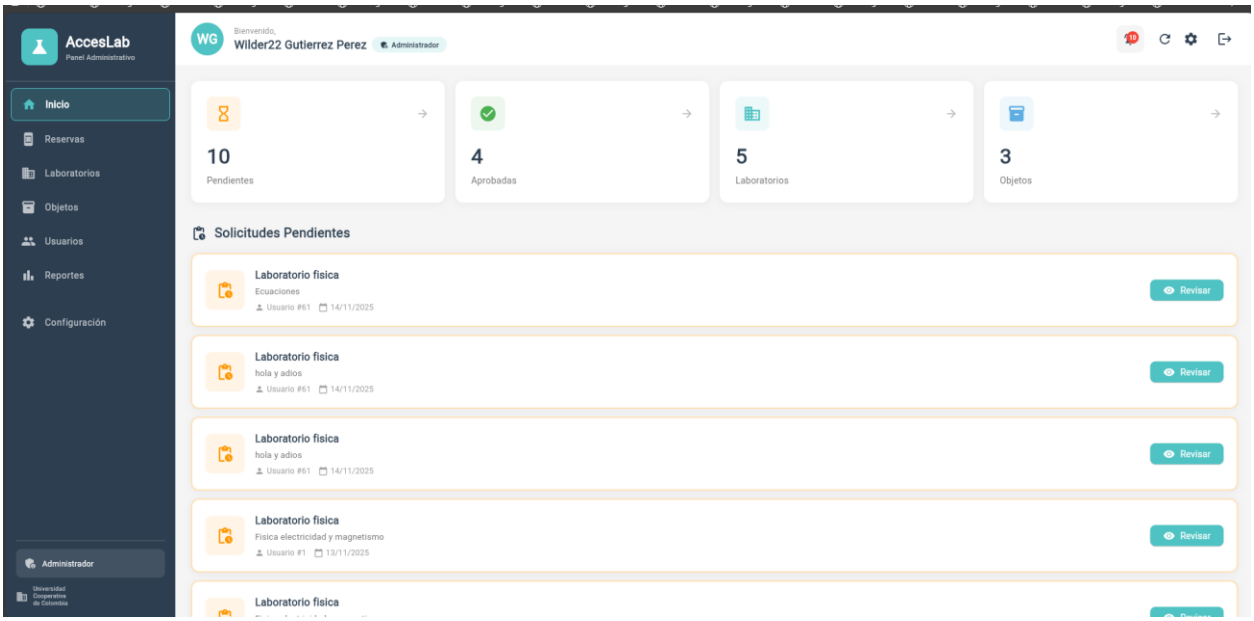
Ilustración 3. Panel del Login



Fuente: Elaborado por los autores

Nota. Esta es la vista inicial del proyecto, aquí los usuarios hacen el logueo de sus credenciales

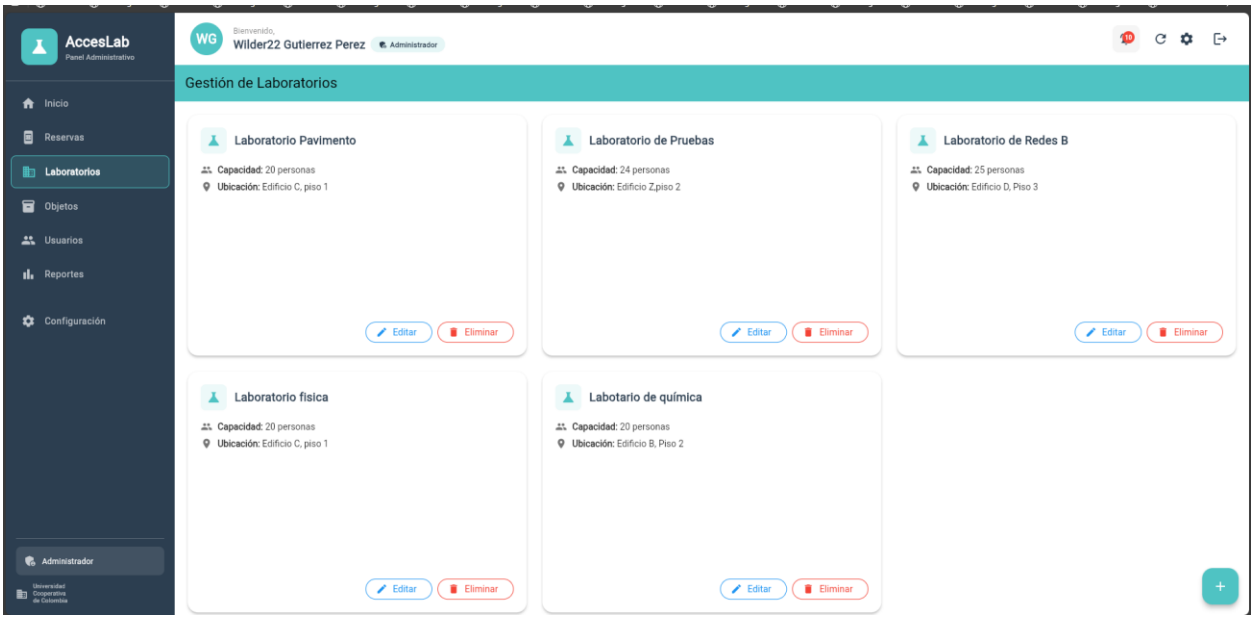
Ilustración 4. Panel inicial



Fuente: Elaborado por los autores

Nota. En esta vista usuario con rol administrador puede observar la página principal de la app.

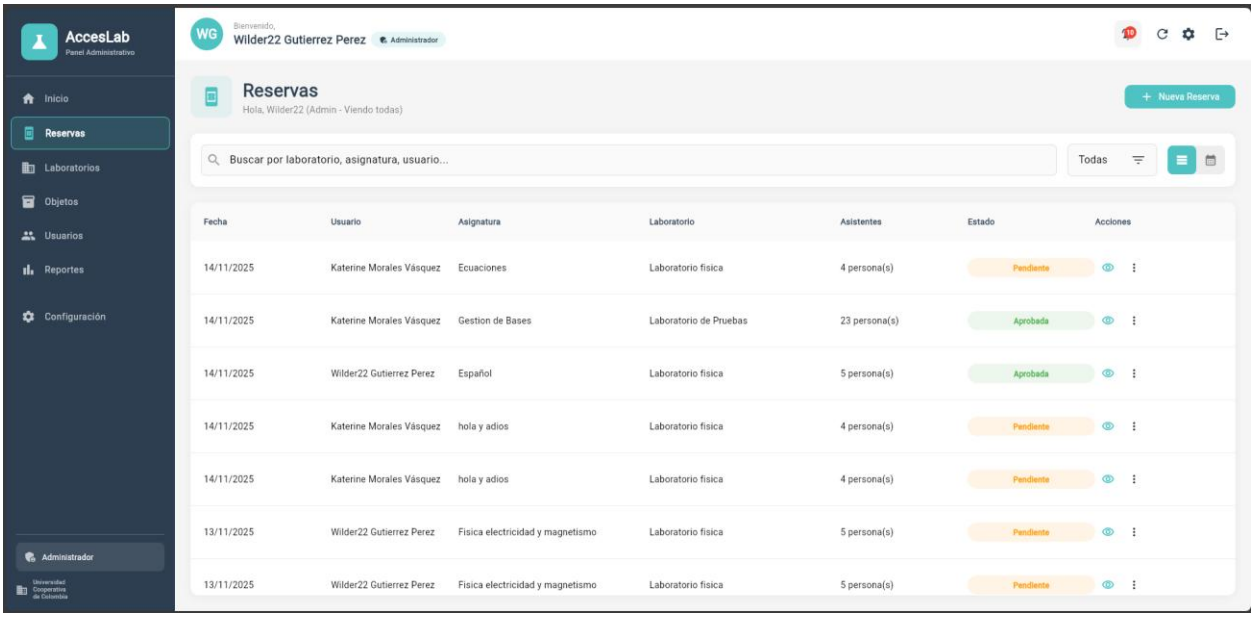
Ilustración 5. Panel de laboratorios disponibles



Fuente: Elaborado por los autores

Nota. En esta figura el usuario puede ver los laboratorios disponibles

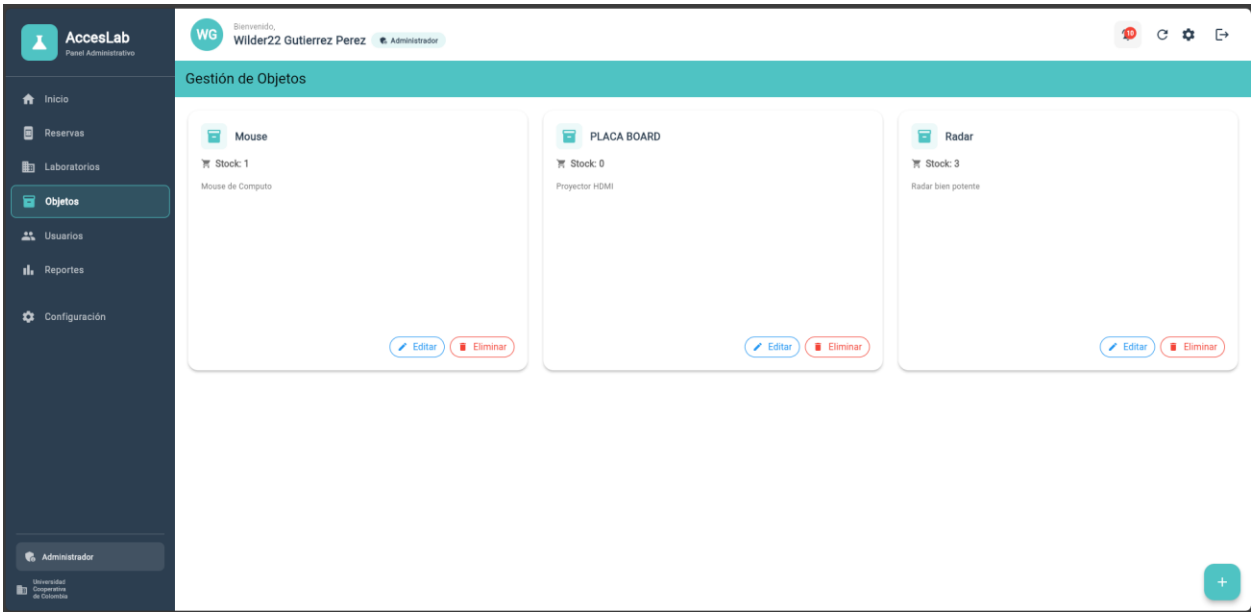
Ilustración 6. Vista del administrador de las reservas activas



Fuente: Elaborado por los autores

Nota. En esta vista el usuario puede ver el historial de todas las reservas hechas

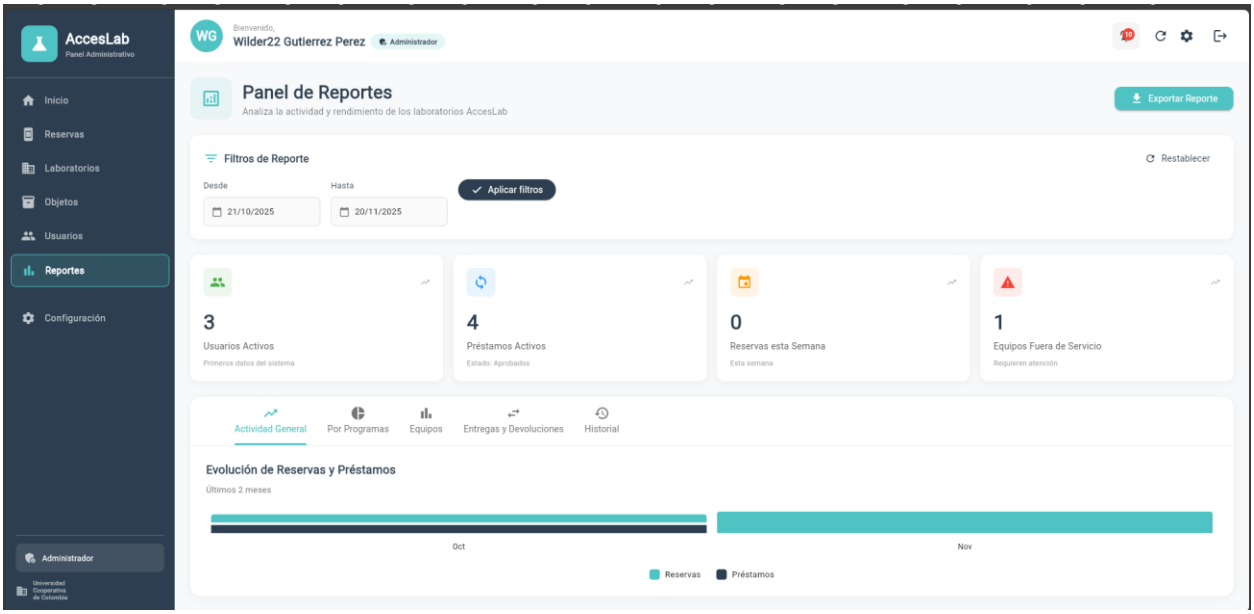
Ilustración 7. Panel de objetos disponibles para los usuarios



Fuente: Elaborado por los autores

Nota. En esta vista el usuario puede ver los objetos disponibles para solicitar

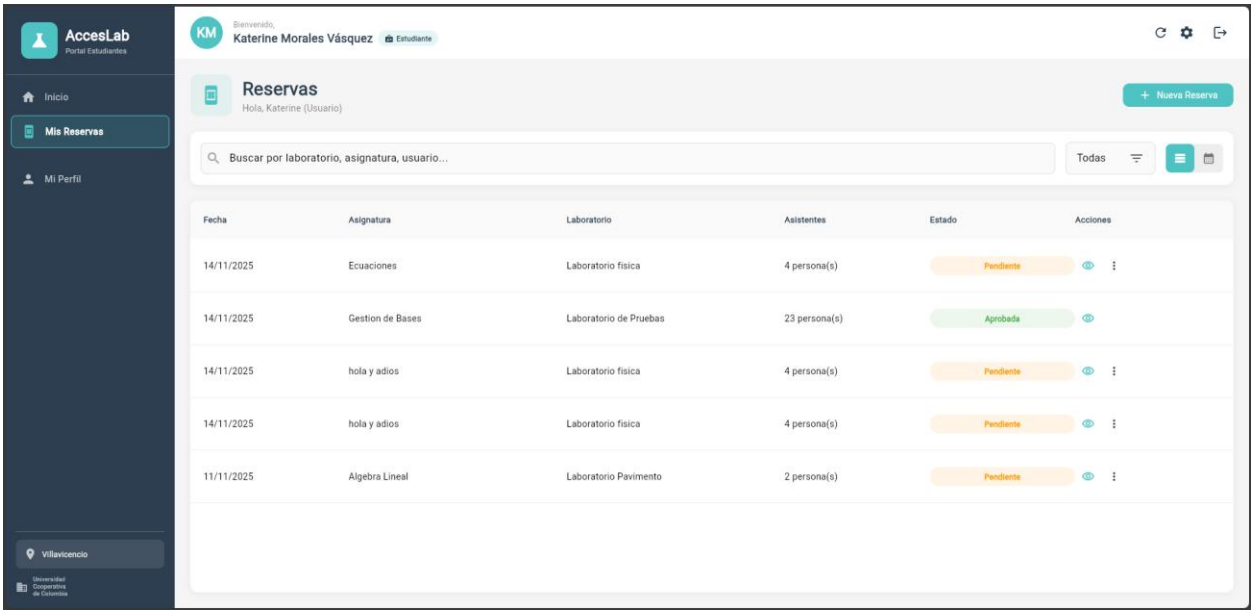
Ilustración 8. Panel de reportes para el administrador



Fuente: Elaborado por los autores

Nota. Esta es la vista del administrador, donde puede generar reportes

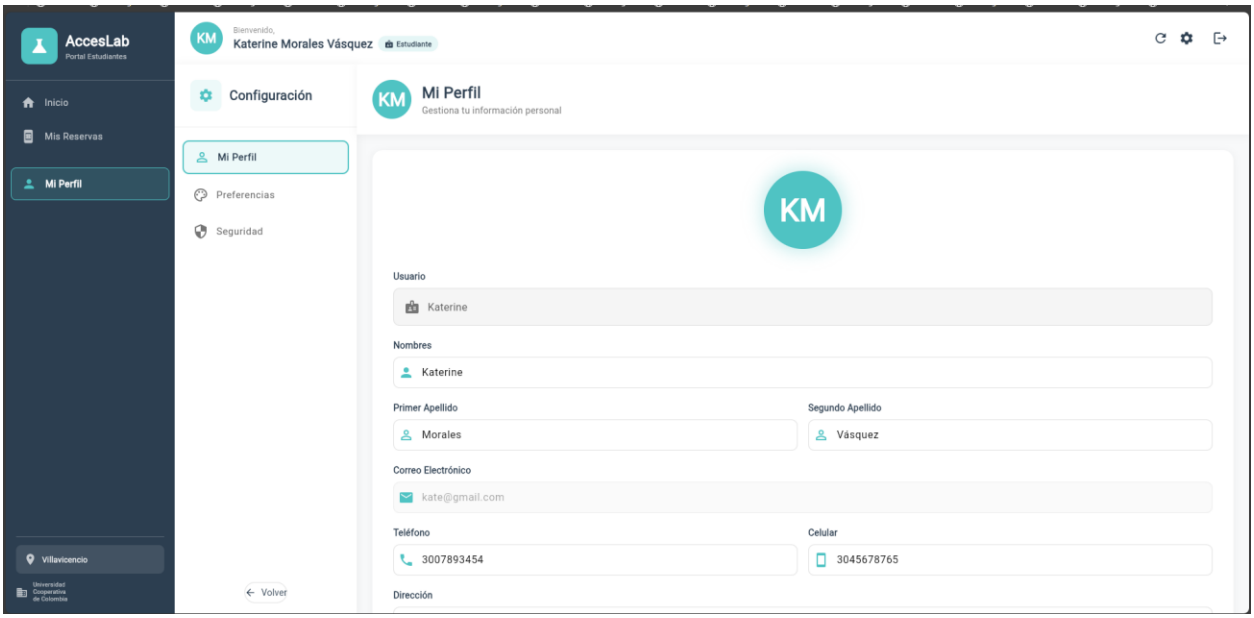
Ilustración 9. Vista de Reservas del Rol usuario



Fuente: Elaborado por los autores

Nota. En esta vista el usuario puede ver sus reservas realizadas

Ilustracion 11. Perfil del usuario



Nota. En esta vista el usuario puede modificar sus credenciales y ver sus datos

5.4 Descripción técnica del sistema

El Sistema Web de Gestión Digital de Reservas de Laboratorios es una aplicación desarrollada para optimizar la administración de los recursos académicos en la Universidad Cooperativa de Colombia, sede Villavicencio. Su arquitectura está basada en un modelo cliente-servidor, donde el frontend (interfaz web) y el backend (lógica de negocio y servicios API) se comunican mediante peticiones HTTP/HTTPS bajo un esquema RESTful.

La solución busca reemplazar el proceso manual actual, permitiendo realizar reservas de laboratorios y equipos de manera eficiente, automatizada y accesible desde cualquier dispositivo con conexión a Internet.

1. Arquitectura General del Sistema

El sistema está conformado por tres capas principales:

1.1. Capa de Presentación (Frontend)

Desarrollada en **Flutter Web**, esta capa proporciona una interfaz de usuario moderna, responsiva e intuitiva que permite la interacción directa con el sistema desde navegadores web.
Principales características:

- Diseño adaptativo (responsive design) para escritorio, Tablet y móvil.
- Formularios interactivos para realizar reservas y registrar usuarios.
- Visualización de la disponibilidad de laboratorios y equipos.
- Integración con el backend mediante peticiones REST API en formato JSON.

1.2. Capa de Lógica y Servicios (Backend)

Implementada en Django REST Framework (Python), esta capa gestiona la lógica de negocio, la autenticación de usuarios, la validación de datos y la conexión con la base de datos Oracle.
Funciones principales:

- Exposición de endpoints RESTful para las operaciones CRUD de usuarios, laboratorios, reservas y equipos.

- Control de permisos y roles (administrador, estudiante/docente).
- Gestión de autenticación segura mediante tokens JWT (JSON Web Tokens)

1.3. **Capa de Datos (Base de Datos Oracle)**

La información se almacenará en una base de datos Oracle Database, estructurada bajo un modelo relacional con integridad referencial entre las tablas principales:

- Usuarios
- Laboratorios-Equipos / Objetos
- Reservas
- Solicitudes

3. Requisitos del Cliente

- Navegador moderno (Google Chrome, Edge, Firefox, Safari)
- Conexión a Internet estable
- Resolución mínima recomendada: 1366x768

4. **Funcionamiento General del Sistema**

- Inicio de sesión: El usuario se autentica en la plataforma mediante su cuenta institucional.
- Interacción con la API: Flutter envía solicitudes HTTP al backend de Django REST, que procesa la información y accede a la base de datos Oracle.
- Validaciones automáticas: Los procedimientos PL/SQL verifican la disponibilidad de los laboratorios y equipos antes de confirmar la reserva.
- Gestión de reservas: Los usuarios pueden crear, modificar o cancelar reservas de forma instantánea.
- Reportes y estadísticas: El administrador puede generar reportes automáticos del uso de recursos por semestre y visualizar el porcentaje de ocupación de los laboratorios.

6 Análisis y discusión

El desarrollo del sistema AccessLab surge como respuesta a una necesidad real identificada en la Universidad Cooperativa de Colombia sede Villavicencio: la optimización del proceso de gestión y reserva de laboratorios académicos. El sistema actual, basado en formularios físicos y trámites presenciales, representa un proceso tedioso, ineficiente y propenso a errores que afecta tanto a estudiantes como a docentes y personal administrativo.

Cumplimiento de Objetivos

El sistema logró implementar exitosamente las funcionalidades principales establecidas en los objetivos:

- Operaciones CRUD completas para usuarios, laboratorios y equipos
- Automatización con PL/SQL para validación de horarios y prevención de conflictos
- Módulo de reportes que calcula ocupación y uso de equipos por semestre
- Interfaz intuitiva que simplifica la interacción con el sistema

Estos logros representan una digitalización efectiva del proceso de reservas, eliminando la burocracia del sistema anterior.

Desafíos Técnicos Críticos

Conexión con la Base de Datos

El mayor obstáculo técnico fue establecer una conexión estable y confiable con la base de datos. Los problemas incluyeron:

- Configuración compleja de credenciales y permisos
- Gestión del pool de conexiones para evitar agotamiento de recursos
- Timeouts intermitentes que requerían reconexión automática
- Manejo de transacciones en operaciones complejas que involucraban múltiples tablas

Estos problemas consumieron tiempo considerable de desarrollo, requiriendo investigación profunda y múltiples sesiones de depuración para lograr estabilidad.

Integración Frontend-Backend

La comunicación entre frontend y backend presentó dificultades inesperadas:

- Problemas de CORS que bloqueaban peticiones entre capas
- Inconsistencias en serialización JSON entre datos enviados y recibidos
- Sincronización asíncrona de operaciones CRUD encadenadas
- Manejo de estados de carga, éxito y error en la interfaz

Esta integración requirió constante comunicación entre miembros del equipo y pruebas exhaustivas de cada endpoint.

Ampliación a Sistema Multicampus

Durante el desarrollo, la docente sugirió transformar el sistema de solución local a plataforma multicampus. Este cambio de alcance tardío generó:

- Reestructuración completa del modelo de datos incorporando la dimensión "sede" en todas las entidades
- Lógica de negocio más compleja con validaciones segmentadas por campus
- Modificaciones significativas en la interfaz para selección y filtrado por sede
- Jerarquía de permisos multinivel distinguiendo administradores globales y locales
- Reportes segmentados con análisis consolidados e individuales

Esta ampliación no contemplada originalmente implicó esfuerzo adicional considerable y ajustes en los cronogramas del proyecto.

Estrategias de Solución

Para superar estos desafíos, el equipo implementó:

- Investigación intensiva de documentación técnica y mejores prácticas
- Desarrollo modular con pruebas incrementales

- Refactorización continua del código cuando fue necesario
- Comunicación constante para resolver bloqueos técnicos
- Gestión controlada de cambios para evaluar viabilidad de nuevos requerimientos

Resultados Finales

A pesar de las dificultades, AccessLab cumplió sus objetivos principales:

1. Digitalización completa del proceso de reservas
2. Sistema multicampus funcional capaz de gestionar múltiples sedes
3. Conexión estable entre todos los componentes del sistema
4. Automatización efectiva que previene conflictos de horarios
5. Reducción significativa de tiempo y errores en el proceso de reservas

Lecciones Aprendidas

El desarrollo evidenció la importancia de:

- Mayor tiempo de diseño arquitectónico antes de la implementación
- Definir claramente el alcance desde el inicio del proyecto
- Realizar integración continua desde etapas tempranas
- Documentar problemas y soluciones para referencia futura

7 Conclusiones

Se logró desarrollar exitosamente el sistema AccessLab para la gestión digital de reservas de laboratorios en la Universidad Cooperativa de Colombia, cumpliendo con el objetivo general de optimizar la administración de recursos académicos mediante la automatización de procesos. El sistema implementado elimina las deficiencias del método tradicional basado en trámites físicos, reduciendo significativamente los tiempos de gestión, minimizando errores en las asignaciones y proporcionando visibilidad en tiempo real sobre la disponibilidad de laboratorios y equipos.

La implementación de operaciones CRUD para usuarios, laboratorios y equipos, junto con la integración de herramientas PL/SQL para validación automática de horarios y prevención de conflictos, garantizó un control estructurado y confiable de los recursos disponibles. El módulo de reportes desarrollado proporciona información precisa sobre la ocupación de laboratorios y el cálculo de uso de equipos por semestre, facilitando la toma de decisiones informadas por parte de los administradores. Estos componentes técnicos aseguran la integridad de los datos y la eficiencia operativa del sistema.

El diseño de una interfaz intuitiva y moderna, desarrollada en Flutter Web, facilita la interacción fluida de estudiantes, docentes y administradores con el sistema desde cualquier dispositivo con conexión a internet. A pesar de los desafíos técnicos encontrados, especialmente en la conexión con la base de datos y la ampliación del alcance hacia un sistema multicampus, AccessLab representa una solución integral que moderniza la gestión de recursos académicos, mejora significativamente la experiencia del usuario y establece las bases para futuras expansiones y mejoras en la plataforma.

8 Referencias

- PIÑEIRO GOMEZ, J. O. S. E. (2013). *Bases de datos relacionales y modelado de datos*. Ediciones Paraninfo, SA.

- Wang, Y., Wei, Z., Cao, J., & Liu, Z. (2019, April). Research and implementation of big data technology laboratory equipment reservation management system. In *IOP Conference Series: Earth and Environmental Science* (Vol. 252, No. 4, p. 042072). IOP Publishing.
- Enriquez, O. Y., & del Busto, H. G. (2011). Mapeo Objeto/Relacional (ORM). *Revista Telem@ tica. Vol*, *10*(3), 1-7.
- *Oracle Database explicado: así funciona la base de datos Oracle*. (2022, 16 febrero). IONOS Digital Guide. https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/oracle-database/#:~:text=Oracle%20Database%20es%20un%20sistema,solo%20pueden%20acceder%20administradores%20autorizados.
- *What is Django? - Django Software Explained - AWS*. (s. f.). Amazon Web Services, Inc. https://aws-amazon-com.translate.goog/what-is/django/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=sge
- Zazo Millán, C. (2019). *Migración de aplicaciones Android hacia Flutter, un framework para desarrollo de apps multiplataforma* (Doctoral dissertation, Universitat Politècnica de València).
- Tech, M. (s. f.). *Mentores Tech - Acelera tu Carrera en Tecnología*. Mentores Tech. https://www.mentorestech.com/resource-guide/arquitectura-de-software/question/que-es-la-arquitectura-de-capas-layered-architecture
- Goodwin, M. (2025, 11 abril). API. *IBM*. https://www.ibm.com/es-es/think/topics/api#:~:text=Una%20API%2C%20o%20interfaz%20de,intercambiar%20datos%2C%20caracter%C3%ADsticas%20y%20funcionalidades.
- *Db2 for Linux, UNIX and Windows*. (s. f.). https://www.ibm.com/docs/es/db2/12.1.x?topic=support-functions-plsql
- Moisset, D. (s. f.). *Descripción: Indices (Crear - Información) (Oracle)*. https://www.tutorialesprogramacionya.com/oracleya/temarios/descripcion.php?cod=203&punto=1&inicio=

9 **Anexos**