UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

ReservApp

Aplicación para la Gestión de Reservas

Presentado por Ahmad Mareie Pascual
en Universidad de Burgos — 16 de septiembre de 2025
Tutores: D. José Manuel Aroca Fernández
y Dr. Jesús Manuel Maudes Raedo

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. José Manuel Aroca Fernández y D. Jesús Manuel Maudes Raedo, profesores del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Exponen:

Que el alumno D. Ahmad Mareie Pascual, con DNI 23008380P, ha realizado el Trabajo Final de Grado en Ingeniería Informática titulado "ReservApp - Aplicación para la Gestión de Reservas".

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de los que suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 16 de septiembre de 2025

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. José Manuel Aroca Fernández	D. Jesús Manuel Maudes Raedo

Resumen

En mi lugar de trabajo, contamos con varias salas de reuniones que son utilizadas diariamente por diferentes equipos para coordinar proyectos, realizar presentaciones y llevar a cabo sesiones de trabajo. Sin embargo, la falta de un sistema para gestionar la reserva de las mismas ha generado diversos problemas organizativos. Actualmente, las reservas se realizan de manera informal, lo que ocasiona situaciones como la concurrencia simultánea o solapada de una misma sala por diferentes grupos, la falta de visibilidad sobre su disponibilidad y la dificultad para coordinar cambios o cancelaciones. Estos inconvenientes generan pérdida de tiempo, confusión entre los empleados y retrasos en la planificación de reuniones.

Ante esta problemática, me ha surgido la ocasión de desarrollar una aplicación web que permita gestionar las reservas de salas de reuniones de manera eficiente y organizada. Esta aplicación nos proporcionará una plataforma centralizada donde se podrá consultar la disponibilidad de las salas en tiempo real, realizar reservas de forma rápida y sencilla, y gestionar cancelaciones o modificaciones sin generar conflictos. Además, se implementarán restricciones para evitar solapamientos en las reservas, asegurando que cada sala esté disponible exclusivamente para un grupo en un horario determinado.

Descriptores

Gestión de Reservas, Planificación de Recursos, Tecnologías web, Aplicación web, Spring Boot, Thymeleaf.

Abstract

At my workplace, we have several meeting rooms that are used daily by different teams to coordinate projects, give presentations, and conduct work sessions. However, the lack of an efficient system to manage their reservation has led to various organizational problems. Currently, bookings are made informally, resulting in situations such as simultaneous occupation of the same room by different groups, lack of visibility on availability, and difficulty in coordinating changes or cancellations. These inconveniences generate time loss, confusion among employees, and delays in meeting planning.

Faced with this issue, the need has arisen to develop a web application that allows for efficient and organized management of meeting room reservations. This application will provide us with a centralized platform where real-time room availability can be checked, reservations can be made quickly and easily, and cancellations or modifications can be managed without generating conflicts. Additionally, restrictions will be implemented to prevent overlapping reservations, ensuring that each room is available exclusively for one group at a specific time.

Keywords

Reservation Management, Resource Planning, Web Technologies, Web Application, Spring Boot, Thymeleaf.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

En el actual entorno empresarial, la gestión eficiente de los espacios de trabajo es necesaria para alcanzar una organización adecuada y un uso eficiente de los recursos de los que dispone. En muchas de las empresas, las salas de reuniones son espacios que se utilizan para la realización de presentaciones, la coordinación de equipos o las distintas tareas que pueden hacerse en equipo. Sin embargo, la falta de un sistema adecuado para la reserva de dichas salas puede generar problemas como la ocupación indebida, la falta de visibilidad sobre su disponibilidad o la dificultad para gestionar cambios o cancelaciones.

En mi lugar de trabajo, este problema es bastante habitual, y es que actualmente la reserva de salas de reunión se realiza de manera informal, lo que provoca distintos tipos de conflictos y confusiones entre los empleados, como la pérdida de tiempo a la hora de planificar una reunión. Es por ello que se propone abordar esta problemática mediante el desarrollo de una aplicación web que facilite la gestión de reservas de dichas salas de una manera eficiente, organizada y accesible para todos los empleados.

La implementación de soluciones software para la gestión de dichos espacios se alinea con los principios de la gestión sostenible de instalaciones. Tal y como dice C.S. Okoro en [25] en su revisión sobre gestión sostenible de instalaciones, la gestión adecuada de las instalaciones no solo contribuye al ahorro de costes, sino que también mejora la eficiencia y moral de los empleados. En este escenario, un sistema de reservas de salas no solo optimiza el uso de los espacios disponibles, sino que también reduce el desperdicio de recursos energéticos que conllevan dichas instalaciones cuando se reservan pero no se utilizan, mientras que la experiencia laboral de los empleados resulta positiva cuando se elimina la incertidumbre y los conflictos en la planificación de reuniones.

Esta aplicación proporcionará una plataforma centralizada en la que los usuarios podrán consultar la disponibilidad de las salas en tiempo real, realizar reservas de manera sencilla y gestionar modificaciones o cancelaciones sin generar conflictos. Además, se implementarán restricciones que evitarán reservas superpuestas o solapadas, garantizando un uso óptimo de los espacios.

El objetivo principal de este Trabajo de Fin de Grado es diseñar e implementar una solución tecnológica que optimice la gestión de las salas de reuniones, mejorando la eficiencia en la organización interna de la empresa. A lo largo de este documento, se detallará el proceso de desarrollo de la aplicación, incluyendo el análisis del problema, el diseño de la solución, la implementación y la evaluación del sistema propuesto.

1.1. Estructura de la memoria

La memoria sigue la siguiente estructura:

- **Introducción:** breve descripción del proyecto, estructura de la memoria y listado de materiales adjuntos.
- **Objetivos del proyecto:** descripción de los objetivos que persigue el proyecto.
- **Conceptos teóricos:** breve explicación de los conceptos necesarios para la realización del proyecto.
- **Técnicas y herramientas:** descripción de las metodologías y herramientas que han sido utilizadas para llevar a cabo el proyecto.
- **Aspectos relevantes del desarrollo:** aspectos a destacar a lo largo de la realización del proyecto.
- **Trabajos relacionados:** resumen de trabajos y proyectos ya realizados en el campo del proyecto en curso.
- **Conclusiones y líneas de trabajo futuras:** conclusiones obtenidas al finalizar el proyecto y posibles ideas de continuidad.

Acompañando a esta memoria, se adjuntan los anexos que se detallan a continuación:

- **Plan del proyecto software:** presenta la planificación temporal y el estudio de viabilidad asociados al proyecto.

- **Especificación de requisitos del software:** se documenta la fase de análisis, que abarca los objetivos generales, el catálogo de requisitos del sistema, y la definición de requisitos funcionales y no funcionales.
- **Especificación de diseño:** se centra en la fase de diseño, cubriendo el alcance del software, el diseño de la base de datos, el diseño procedimental y la arquitectura general.
- **Manual del programador:** recurso en el que se resumen los aspectos técnicos más relevantes del código fuente, como su organización, procesos de compilación, instalación, ejecución y procedimientos de prueba.
- **Manual de usuario:** guía completa diseñada para facilitar el uso correcto de la aplicación por parte del usuario final.

1.2. Recursos disponibles

Los siguientes recursos están accesibles a través de Internet:

- Dirección del repositorio del proyecto: https://github.com/AhmadMarPas/TFG-ReservApp/.
- Dirección de la documentación del proyecto: https://github.com/AhmadMarPas/TFG-ReservApp/tree/main/documentaci%C3%B3n/.
- Dirección de las métricas del proyecto: https://sonarcloud.io/project/overview?id=ReservApp/.
- Dirección de la web de Reservas usada en el desarrollo (ATENCIÓN: Es posible que tarde entre 2 y 3 minutos en mostrarse la página inicial porque la aplicación se repliega cuando no se utiliza): https://reservapp-1u8c.onrender.com/login/.
- Dirección del proyecto en Zube: https://zube.io/ahmadmarpas/tfg/w/reservapp/sprints/.
- Dirección del video con descripción del proyecto: https://github.com/AhmadMarPas/TFG-ReservApp/blob/main/documentaci%C3%B3n/presentaci%C3%B3n/reservapp_descripcion.mp4/.
- Dirección del video con demostración funcional de la aplicación: https://github.com/AhmadMarPas/TFG-ReservApp/blob/main/documentaci%C3%B3n/presentaci%C3%B3n/reservapp_demostracion.mp4.

2. Objetivos del proyecto

En las siguientes secciones se detallan los diferentes objetivos que han promovido o motivado la realización del actual proyecto.

2.1. Objetivos generales

Este proyecto busca resolver un problema común en las oficinas: la gestión de las reservas de salas. Por ello, se ha buscado la creación de una solución que no solo mejore la eficiencia en el uso de los espacios, sino que también facilite la vida de los empleados. Con una aplicación web intuitiva, cualquier persona podrá reservar y administrar salas de manera ágil y sin complicaciones.

Con este proyecto no solo se busca solucionar un problema específico dentro de la empresa, sino que se busca la aplicación de un caso práctico en el que entran en juego los conocimientos adquiridos a lo largo de la formación académica para la implementación de una aplicación web para la gestión de recursos compartidos. A lo largo de este desarrollo, se explorarán diferentes herramientas y metodologías para garantizar que la solución final sea escalable, segura y fácil de usar.

2.2. Objetivos técnicos

La propuesta de este TFG es la del desarrollo de una aplicación web de tipo cliente-servidor [10] que se implementará utilizando, como principales *frameworks* [12], *Spring Boot* [33] y *Thymeleaf* [36], permitiendo la creación de una interfaz dinámica y robusta.

La seguridad fue una prioridad, integrando *Spring Security* [35] de modo que se garantiza que solo los usuarios autorizados tengan acceso a funcionalidades específicas, protegiendo de esta forma la integridad del sistema.

Para gestionar la evolución del código y su versionado, se empleó *Git* como sistema de control de versiones distribuido, en combinación con un repositorio en *GitHub*. Esto permitió un seguimiento detallado de los cambios realizados.

La estructura de datos fue cuidadosamente diseñada, optando por un modelo relacional normalizado que asegura la coherencia y eficiencia en el manejo de la información. Además, la aplicación siguió la arquitectura MVC (Modelo-Vista-Controlador) [11], favoreciendo de esta forma la modularidad y el mantenimiento del código.

Para el proceso de construcción del software, la gestión de dependencias y la compilación, se utilizó Maven como herramienta de automatización. Además, se implementaron herramientas de integración continua, dispuestas por *GitHub Actions* [13] e integradas directamente con el repositorio para garantizar un control de calidad constante y automatizado.

Finalmente, en el proyecto se adoptó el marco de trabajo con metodología ágil Scrum, utilizando *Zube* [19] como herramienta de gestión, lo que permitió una planificación flexible y adaptable a los cambios, entregando valor de manera iterativa. Además, se implementaron *tests* unitarios y de integración robustos para garantizar la calidad y el correcto funcionamiento de cada componente y del sistema en su conjunto.

2.3. Objetivos personales

El proyecto es una apuesta personal por definir una aplicación que pueda tener salida en el entorno laboral, buscando dar servicio a negocios de tipo servicio granular y personalizado a usuarios, y que pueda suponer una salida laboral para mí en el futuro como apuesta de emprendimiento.

Además, con la finalización del TFG, termino también el Grado y es que, tras no haber completado la Ingeniería Técnica en Informática de Gestión en su momento por no haber realizado el Proyecto Fin de Carrera, la realización de este TFG supone la culminación de mi formación académica como ingeniero.

3. Conceptos teóricos

En este capítulo se describe cómo se establece la base del proyecto, demostrando que la propuesta no solo resuelve un problema práctico, sino que también está fundamentada en principios bien establecidos.

3.1. Sistemas de Gestión de Reservas

El sistema de gestión de reservas es una herramienta que facilita la asignación de recursos, como salas de reuniones. Aunque éste fue el punto de partida, la aplicación ha sido diseñada para ser adaptable; se puede utilizar en diferentes sectores, como restaurantes, servicios y para la gestión de citas.

En el contexto de este TFG, para que la experiencia de reserva fuera lo más intuitiva posible, se ha introducido el concepto de **"slots"** o intervalos de tiempo. De modo que se han diferenciado los espacios de reserva en dos tipos para que el usuario pueda elegir el que mejor se adapte a su necesidad:

- **Duración variable**: Es la opción que ofrece flexibilidad, permitiendo al usuario elegir la hora de inicio y fin de su reserva.
- **Duración fija**: Para estos casos, la aplicación muestra una serie de horarios preestablecidos (los "slots"), facilitando al usuario la selección de un intervalo de tiempo ya definido, agilizando con ello el proceso.

Otro concepto a tener en cuenta es el de aforo. Los valores de aforo no hacen referencia a la capacidad de una sala, que para eso ya está el atributo de capacidad, sino a la cantidad de reservas que pueden concurrir al mismo tiempo. El aforo está orientado a los establecimiento que se rigen por "slots" de modo que si un establecimiento tiene de aforo 2, significa que permite que se realicen 2 reservas para el mismo periodo de tiempo.

3.2. Arquitectura de Aplicaciones Web

Patrones de Arquitectura

Para el desarrollo de la aplicación se ha aplicado el conocido patrón Modelo-Vista-Controlador (MVC) [11], consiguiendo una solución estructurada y desacoplada, y para ello se ha aprovechado las características del ecosistema Spring obteniendo una aplicación web sólida y fácil de mantener, separando mediante capas la lógica de negocio, la de presentación y la de control de la aplicación:

- **Modelo**: Gestiona los datos y la lógica de negocio. Implementado en Spring Boot con acceso a la base de datos MySQL.
- **Vista**: Representa la interfaz de usuario, creada con Thymeleaf.
- **Controlador**: Maneja las interacciones del usuario y actualiza el modelo o la vista según corresponda.

Esta arquitectura modular facilita la escalabilidad, el mantenimiento y la reutilización de código.

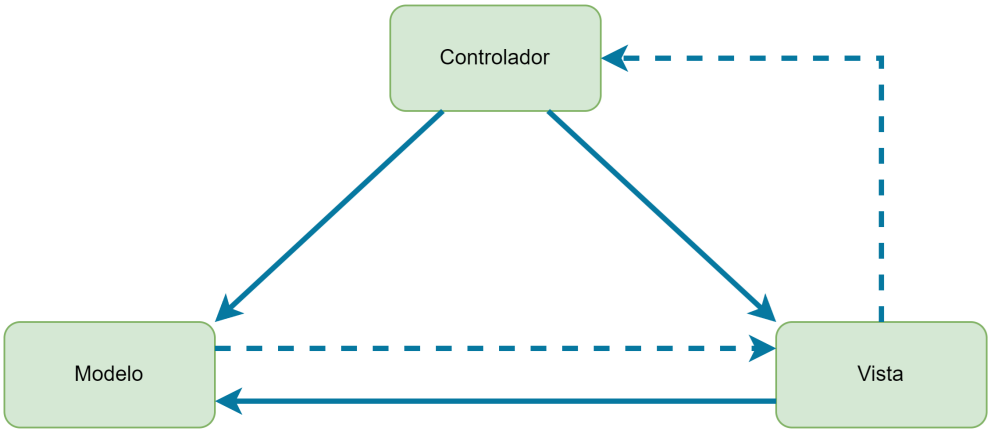

Figura 3.1: Patrón modelo MVC.

Arquitectura Cliente-Servidor

La arquitectura de la aplicación web sigue un modelo cliente-servidor [10], donde:

- **Cliente o *frontend***: Representado por el navegador web del usuario, que interactúa con la aplicación a través de una interfaz construida con Thymeleaf.
- **Servidor o *backend***: Implementado con Java y Spring Boot, que procesa las solicitudes del cliente, accede a la base de datos MySQL y devuelve las respuestas correspondientes.

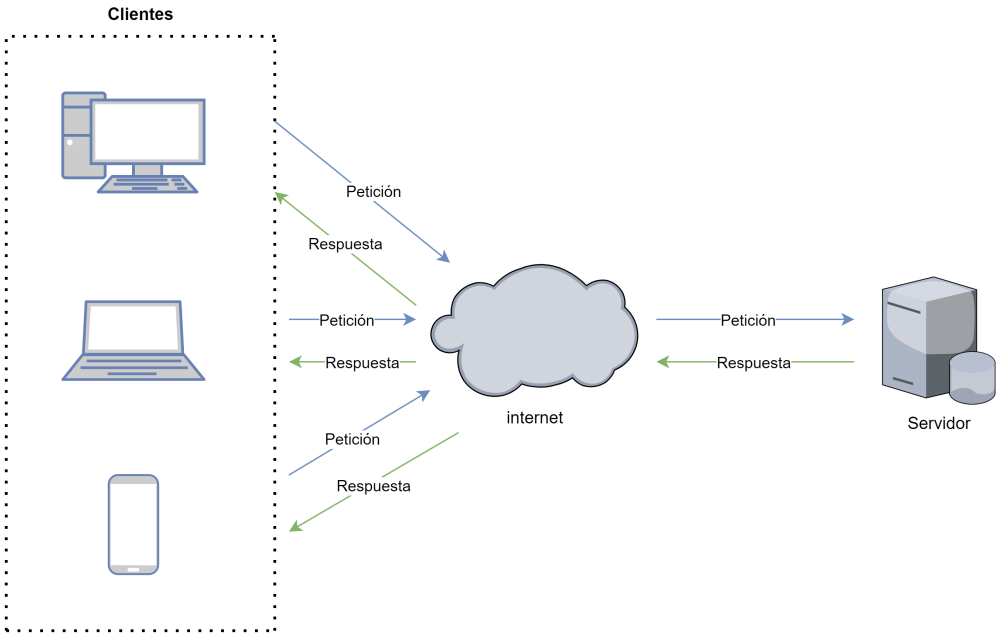

Figura 3.2: Arquitectura Cliente-Servidor.

3.3. Patrones de diseño

El sistema ha sido diseñado siguiendo principios de arquitectura en capas, separación de responsabilidades y buenas prácticas de desarrollo. A continuación, se describen algunos de los patrones utilizados.

Patrón Modelo-Vista-Controlador (MVC)

Para el desarrollo de la aplicación web de gestión de reservas con Spring Boot, se siguió el patrón MVC por su separación clara de responsabilidades que facilita el mantenimiento y escalabilidad del código al dividir la lógica de negocio (Modelo), la presentación (Vista) y el control de flujo (Controlador). Su integración nativa con Spring Boot mediante las anotaciones que ofrece el *framework* simplifica significativamente el desarrollo. Además, ofrece flexibilidad para cambios futuros como modificar la interfaz de usuario sin afectar la lógica de negocio, o modificar la lógica de negocio sin afectar a la capa visual o al modelo de datos.

Arquitectura por Capas

De alguna forma, o bien como consecuencia o bien como complementación al punto anterior, el patrón MVC favorece la construcción de los diferentes elementos separándose en diferentes capas, donde cada capa (presentación, lógica de negocio, acceso a datos) se alinea naturalmente con los componentes MVC, creando una estructura coherente y bien organizada. Esta estructura proporciona alta cohesión y bajo acoplamiento, permitiendo que los cambios en una capa (como modificar la base de datos) no afecten a las demás, mientras que MVC organiza la interacción usuario-sistema dentro de cada capa. Además, esta separación facilita la implementación de principios SOLID [3] posibilitando el *testing* independiente de cada capa, donde se puede probar la lógica de negocio sin depender de la interfaz de usuario o de la base de datos. Esta combinación entre arquitectura por capas y MVC es una buena práctica que conduce a un diseño maduro y profesional que cumple con los estándares para el desarrollo de aplicaciones.

Patrón Repository

Al disponer Spring Boot de una integración nativa con Spring Data JPA, se permite la creación de repositorios automáticamente mediante interfaces, reduciendo significativamente el código. Este patrón abstrae la lógica de acceso a los datos de la capa de negocio, permitiendo cambiar la implementación de la capa de persistencia sin afectar el resto de la aplicación y facilitando el *testing* mediante *mocks*. Además, Spring Boot proporciona funcionalidades avanzadas como *query methods* automáticos y paginación. El uso de este patrón garantiza el cumplimiento del principio de inversión de dependencias, creando una arquitectura limpia y mantenible.

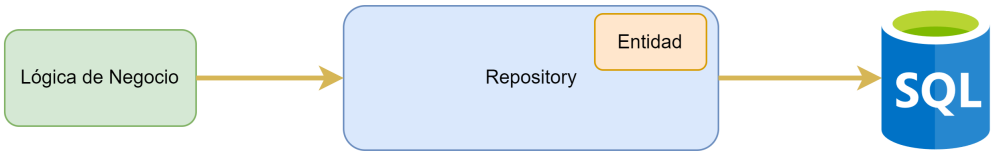

Figura 3.3: Patrón Repository.

Patrón Service Layer

Gracias a la utilización del patrón de diseño MVC comentado anteriormente, la implementación conduce casi de forma intuitiva o directa al uso del patrón *service layer* que actúa como intermediario entre los *Controllers* y los *Repositories*, encapsulando toda la lógica de negocio y las reglas de validación en una capa dedicada que mantiene los controladores enfocados únicamente al manejo de peticiones HTTP. Su integración con Spring Boot mediante *@Service* facilita la inyección de dependencias y la gestión transaccional con *@Transactional*, permitiendo la reutilización de la lógica de negocio desde diferentes controladores o incluso APIs REST y servicios web. Este patrón refuerza la separación de responsabilidades del MVC, donde el Controller delega las operaciones complejas en el *Service*, que a su vez utiliza los *Repositories* para la persistencia, creando una arquitectura limpia, testeable y que cumple con los principios de buenas prácticas del diseño de software.

Patrón Data Transfer Object (DTO)

Con el uso del patrón DTO se consigue transferir datos entre capas sin exponer la estructura interna de las entidades JPA, evitando problemas como la serialización de relaciones *lazy* y mejorando la seguridad al controlar qué información se envía al cliente. Los DTOs proporcionan flexibilidad en la representación de datos, permitiendo combinar información de múltiples entidades o mostrar vistas específicas según el contexto. Por ello, se facilita la evolución independiente de la API y el modelo de datos, donde se pueden modificar las entidades de la base de datos sin afectar los contratos de la API, y mejorando el rendimiento al reducir la cantidad de datos transferidos.

4. Técnicas y herramientas

4.1. Metodologías

Scrum

Para gestionar el proyecto, se utilizó Scrum, que es un enfoque ágil que permite trabajar por etapas cortas (llamadas *sprints*). Así, se avanza paso a paso, revisando y mejorando el producto en cada fase. Esto ayuda a adaptarse rápidamente a los cambios y a mantener el trabajo bien organizado.

4.2. Gestión del proyecto

Control de versiones

- Herramientas consideradas: Git y Subversion.
- Opción elegida: Git.

En este proyecto, se decidió utilizar Git porque es el sistema de control de versiones más utilizado y, por experiencia, el que mejor se adapta a cualquier tipo de desarrollo. Lo que ha resultado más útil es que permite crear ramas para probar nuevas funcionalidades sin tocar el código principal; es decir, si algo sale mal, no afecta a lo que ya funciona. Otra ventaja es que, al ser distribuido, en un equipo con varios integrantes, cada uno tiene una copia completa del proyecto en su máquina, permitiendo seguir trabajando incluso cuando haya problemas con la conexión a Internet. Además, como cliente de Git, se ha utilizado TortoiseGit [37] para todo lo referente a la documentación del proyecto.

Repositorio

- Herramientas consideradas: GitHub, Bitbucket y GitLab.
- Opción elegida: GitHub.

En este proyecto, se optó por usar GitHub porque, además de basarse en Git, ofrece herramientas extra que han facilitado mucho el trabajo. Por un lado, la integración con CI/CD [17] permitió automatizar pruebas y despliegues, algo que el tutor pidió demostrar como parte de las buenas prácticas. Por ejemplo, cada vez que se subía código a la rama principal, se ejecutaban automáticamente los *tests*, lo que ayudó a detectar errores antes de la entrega. También ofrece el uso de los *issues* de GitHub para organizar las tareas: se asignaban etiquetas según el tipo de trabajo (*bug*, mejora, documentación, etc.). Así, de un simple vistazo siempre se puede saber en qué se estaba trabajando en cada momento. Otra cosa que resultó útil fue la opción de marcar versiones importantes con *tags*, sobre todo cuando se terminaba una fase del proyecto y se quería guardar un "punto de control" antes de seguir avanzando.

Integración continua y análisis de calidad

GitHub Actions por su integración nativa con el repositorio permite automatizar el *pipeline* de CI/CD directamente desde GitHub. Su facilidad de configuración mediante archivos YAML [18] permite habilitar la ejecución de diferentes *workflows* para la compilación y empaquetado del proyecto, así como la comunicación con SonarCloud para el análisis estático del código para identificar bugs, vulnerabilidades y *code smells* automáticamente. Su integración con GitHub Actions para análisis continuo en cada *commit*, y el hecho de proporcionar métricas objetivas de calidad (cobertura de código, complejidad ciclomática, duplicación de código, etc...) enriquecen significativamente la documentación técnica del proyecto, ofreciendo un enfoque profesional de cara a la calidad del software.

Documentación

Para documentar el proyecto, se utilizó LaTeX a través de la plataforma Overleaf [28]. Esta combinación fue propuesta por los tutores, ya que permite generar documentos académicos con un formato profesional y una tipografía de alta calidad, aspectos esenciales para cumplir con los requisitos de presentación. Al estar basado en la nube, Overleaf facilita el trabajo colaborativo en tiempo real desde cualquier dispositivo, lo que simplificó las revisiones

con los tutores. Además, la gestión automática de referencias bibliográficas y la numeración de secciones redujeron significativamente los errores manuales, asegurando que el documento final cumpliera con los estándares académicos establecidos.

4.3. Tecnologías Utilizadas

Java y Spring Boot

Java es un lenguaje de programación orientado a objetos ampliamente utilizado en el desarrollo de aplicaciones empresariales. Spring Boot [33] es un *framework* [12] basado en Spring [32] que simplifica la configuración y el desarrollo de aplicaciones web. Spring Boot proporciona funcionalidades como:

- Inyección de dependencias (DI)
- Gestión de transacciones
- Conectividad con bases de datos
- Seguridad integrada

Spring Data JPA [34]

Simplifica la capa de acceso a datos, facilitando la interacción con bases de datos relacionales a través de mapeo objeto-relacional (ORM).

Hibernate [8]

Es el responsable del mapeo de las clases Java a tablas en una base de datos relacional, permitiendo a los desarrolladores trabajar con objetos en lugar de tener que escribir consultas SQL manualmente.

Spring Security [35]

Proporciona medidas para la autenticación y autorización de modo que añade una capa de protección a la aplicación.

Lombok [4]

Es una librería que, mediante anotaciones, ayuda a reducir el código repetitivo en Java como *getters*, *setters* o constructores.

Thymeleaf [36]

Se trata de un motor de plantillas para Java que permite crear vistas HTML dinámicas en el servidor. Se integra con Spring.

MySQL [27]

Es un sistema de gestión de bases de datos relacional de código abierto que ofrece alta fiabilidad, integridad de datos y compatibilidad con transacciones ACID [40].

4.4. Librerías

JUnit

JUnit [21] es considerado el *framework* de *testing* por excelencia, situándose como un estándar de facto para los *tests* en Java. Viene integrado nativamente con Spring Boot Test, facilitando la escritura de tests unitarios y de integración sin tener que realizar una configuración adicional. Su sintaxis intuitiva con anotaciones permite crear tests legibles y mantenibles, posibilitando el testing de controladores y servicios de forma aislada. Además, JUnit proporciona reporting detallado de resultados y se integra perfectamente con herramientas de CI/CD como GitHub Actions y análisis de cobertura con SonarCloud, permitiendo obtener métricas de cobertura de la calidad del software.

Mockito

Para complementar los *tests* unitarios implementados con JUnit, se ha utilizado la librería Mockito [24] por su capacidad de crear *mocks* y *stubs* de dependencias que permiten aislar completamente los elementos que se están *testeando*, resultando especialmente útil para testear servicios sin depender de *repositories* reales o bases de datos. Su integración con JUnit mediante anotaciones como *@Mock* y *@InjectMocks* simplifica la configuración de tests, mientras que su sintaxis intuitiva con métodos como *when().thenReturn()* hace que los *tests* sean legibles y entendibles. Con el uso de Mockito se ha posibilitado el testing de las diferentes capas que componen la arquitectura, permitiendo verificar interacciones entre componentes (*verify()*) y simular diferentes escenarios de error o éxito sin la complejidad de configurar un entorno completo, lo que resulta práctico para demostrar buenas prácticas de testing en un proyecto.

4.5. Desarrollo Web

Thymeleaf

Por su integración nativa y oficial con Spring Boot [33], se ha utilizado Thymeleaf [36] ya que elimina configuraciones adicionales y proporciona soporte completo para el patrón MVC mediante resolución automática de plantillas y *binding* de modelos. Su sintaxis natural en HTML permite que las plantillas sean visualizables en navegadores sin procesamiento, facilitando el desarrollo colaborativo con diseñadores, mientras que sus expresiones Spring EL se integran a la perfección con los objetos del modelo pasados desde los controladores. Además, Thymeleaf ofrece características específicas para aplicaciones web como validación de formularios integrada con Spring Validation, internacionalización automática, y fragmentos reutilizables, creando una solución completa que demuestra un uso adecuado del ecosistema Spring para el desarrollo de aplicaciones web.

Bootstrap

Como complementación de Thymeleaf [36], se ha utilizado Bootstrap [6] por su integración sencilla con sus plantillas o los recursos estáticos de Spring Boot, permitiendo aplicar clases CSS directamente en los elementos HTML, así como crear interfaces responsivas sin una configuración compleja. Su sistema de componentes predefinidos (formularios, tablas, modales) se adapta a la perfección a las funcionalidades utilizadas en la aplicación, mientras que la gestión de recursos estáticos de Spring Boot permite servir Bootstrap de forma optimizada y cacheable. Además, Bootstrap proporciona consistencia visual y experiencia de usuario sin requerir conocimientos avanzados de CSS, permitiendo que el proyecto se enfoque en la lógica de negocio y la arquitectura del *backend* manteniendo una interfaz moderna y funcional.

JavaScript

Se ha utilizado JavaScript [7] por su capacidad para añadir interactividad dinámica en aquellos cometidos en los que Bootstrap no puede proporcionar por sí solo, como validación de formularios en tiempo real, calendarios interactivos para selección de fechas de reserva, y componentes dinámicos que mejoren la experiencia de usuario sin recargar la página. Su integración natural con Thymeleaf permite generar código JavaScript dinámico con datos del servidor, mientras que se complementa perfectamente con los componentes de Bootstrap añadiendo funcionalidad a modales, *dropdowns*, y formularios

mediante *event listeners* y manipulación del DOM[1]. Además, JavaScript permite comunicación asíncrona con el *backend* mediante AJAX [1] para funcionalidades como verificación de disponibilidad de reservas en tiempo real, creando una aplicación web moderna e interactiva.

CSS

Se ha utilizado un CSS personalizado para permitir su adaptación y extensión de los estilos de Bootstrap sin modificar el *framework* base, consiguiendo crear una identidad visual propia para la aplicación mediante variables CSS personalizadas y una sobrescritura selectiva de clases Bootstrap. Su integración con Thymeleaf permite aplicar estilos condicionales y crear hojas de estilo específicas para diferentes vistas de la aplicación de reservas. Además, CSS permite optimizar la experiencia de usuario con animaciones o transiciones que complementen la *responsividad* de Bootstrap, a la vez que facilita la personalización específica de elementos visuales que Bootstrap no cubre.

4.6. Entorno de desarrollo integrado (IDE)

- Herramientas consideradas: Visual Studio code, Eclipse y IntelliJ IDEA.
- Opción elegida: Eclipse.

Para la codificación de la aplicación, se ha utilizado Eclipse [9] como IDE de desarrollo por su integración nativa con el ecosistema Java empresarial y su excelente soporte para proyectos Maven/Gradle, *debugging* avanzado, y herramientas específicas de Spring como Spring Tool Suite (STS) que facilitan el desarrollo con autocompletado inteligente y configuración automática. Su gestión robusta de proyectos grandes con refactoring automático, navegación de código eficiente, e integración con sistemas de control de versiones como Git lo hacen adecuado para la organización y mantenibilidad del código a largo plazo. Además, Eclipse ofrece plugins especializados para tecnologías del proyecto como Thymeleaf, JUnit, asistentes a la codificación como GitHub Copilot [14] y herramientas de análisis de código, mientras que por su naturaleza gratuita y *open-source* lo convierten en una opción accesible para cualquier desarrollador para su uso como herramienta profesional estándar de la industria Java sin costes adicionales.

[1]DOM es la sigla de document object model

5. Aspectos relevantes del desarrollo del proyecto

5.1. Ciclo de vida y metodología de desarrollo

Para el desarrollo de la aplicación de gestión de reserva se adoptó un ciclo de vida iterativo-incremental basado en Scrum como marco de trabajo para metodologías ágiles, estructurando en *sprints* de dos semanas con el fin de obtener una evolución controlada y adaptativa del producto. Aunque la aplicación de Scrum no fue del todo rigurosa a todas las directrices que establece, principalmente debido a que no se trataba de un equipo de desarrollo al uso, ya que sólo estaba formado por una persona. A pesar de ello, sí se ha aplicado la filosofía que promueve Scrum, como es el desarrollo incremental, revisiones de hitos alcanzados, adaptación al cambio, planificación de los siguientes sprints, etc... Esta decisión se justificó por la naturaleza académica del proyecto, donde los requisitos iniciales podían evolucionar conforme se avanzaba en el desarrollo de la aplicación y se identificaban nuevas necesidades o modificaciones durante las reuniones con los tutores.

La gestión de riesgos técnicos se planificó desde el primer *sprint*, identificando como principales amenazas los conflictos de dependencias entre Spring Boot y otras librerías, así como posibles problemas de rendimiento en consultas concurrentes para la disponibilidad de salas. Para mitigar estos riesgos, se estableció un entorno de desarrollo con perfiles específicos (desarrollo, testing, producción).

Para la evolución de requisitos inicialmente se planificó un sistema básico de reservas, pero durante el desarrollo se identificaron otras necesidades como la de implementar funcionalidades para las notificaciones por correo, el registro de auditoría para identificar quién creaba o modificaba la información y cuándo se producían. Una decisión importante que surgió durante el desarrollo fue la implementación de borrado lógico en lugar de eliminación física de registros, lo que implicó rediseñar el modelo de datos para incluir el campo válido y modificar todas las consultas JPA para el filtrado de registros válidos. Esta decisión, aunque incrementó la complejidad inicial, proporcionó trazabilidad completa de las operaciones y capacidad de recuperación de datos.

Otro cambio de diseño relevante fue la flexibilización del sistema de duración de reservas, evolucionando desde un modelo rígido, de duración libre, hacia la posibilidad de establecer "slots" predefinidos, consiguiendo un sistema híbrido que soporta tanto duraciones libres como intervalos fijos configurables por sala. Esta funcionalidad requirió implementar una lógica compleja de validación de solapamientos que considerara ambos escenarios, desarrollando soluciones específicas para detectar conflictos entre reservas de duración libre y aquellas que siguen "slots" horarios. El diseño modular por capas y el uso de patrones como *Service Layer* permitieron incorporar estos cambios sin afectar significativamente la arquitectura base, demostrando la flexibilidad de la metodología ágil para adaptarse a requisitos emergentes en proyectos como el actual.

Otro hecho importante que se dio durante el desarrollo fue que surgieron nuevas necesidades que requirieron ajustes en el diseño significativos. Esta circunstancia ocurrió tras una revisión con el tutor, donde se identificó que el modelo de reservas unipersonales era limitante para el contexto inicialmente pensado.

Esta retroalimentación condujo a una redefinición relevante del dominio de negocio: las reservas evolucionaron hacia un sistema de convocatorias colaborativas donde el usuario creador podía invitar a otros participantes. Esta modificación implicó cambios en el diseño importantes, incluyendo la reestructuración del modelo de datos para soportar relaciones del tipo muchos-a-muchos entre usuarios y reservas, y el desarrollo de un sistema de notificaciones por correo electrónico que alertara automáticamente a los convocados.

La implementación del sistema de notificaciones requirió la integración de Spring Mail [31] con configuración SMTP [38], la creación de plantillas de correo personalizadas con Thymeleaf, y la gestión asíncrona de envíos para

evitar bloqueos en la experiencia de usuario. Además, se implementó una lógica de negocio adecuada para determinar cuándo enviar notificaciones (creación de convocatoria, modificaciones, cancelaciones).

Esta evolución de requisitos, aunque incrementó significativamente la complejidad del proyecto, demostró la flexibilidad de la metodología ágil para adaptarse a necesidades emergentes y la robustez de la arquitectura por capas que permitió incorporar estas funcionalidades sin afectar los módulos ya desarrollados. El resultado fue una aplicación mucho más alineada con las nuevas necesidades sugeridas, transformando un simple sistema de reservas en una plataforma colaborativa de gestión de reuniones.

5.2. Decisiones arquitectónicas

Una de las decisiones más significativas durante las primeras fases del proyecto fue la elección de la tecnología para la capa de presentación. Inicialmente se contempló utilizar JSF (JavaServer Faces) [26] con PrimeFaces [20] como *framework* interfaz de usuario, motivada principalmente por la experiencia en ello y por su rico conjunto de componentes predefinidos y su capacidad para crear interfaces web complejas con mínimo código JavaScript. PrimeFaces ofrece componentes avanzados como calendarios interactivos, tablas con filtrado automático y diálogos modales que parecían ideales para las funcionalidades de gestión de reservas. Sin embargo, tras una evaluación técnica e investigación sobre diferentes opciones para la capa de presentación, se decidió utilizar Thymeleaf por los siguientes factores:

- La integración nativa con Spring Boot eliminaba configuraciones complejas de JSF y proporcionaba soporte automático para el patrón MVC mediante resolución de plantillas y *binding* de modelos.
- Aunque no se tenía experiencia previa con ninguna de las dos tecnologías, Thymeleaf presentó una curva de aprendizaje más suave al trabajar directamente con HTML válido, permitiendo que las plantillas fueran visualizables en navegadores sin tener que realizar el procesamiento del servidor, facilitando de esta manera el desarrollo y debug.

La decisión final se basó en que Thymeleaf se alineaba mejor con los objetivos académicos del TFG al demostrar un uso profesional del ecosistema Spring, mientras que JSF habría requerido configuraciones adicionales y conocimiento específico de ciclos de vida de componentes que podrían haber

desviado el foco del aprendizaje hacia aspectos menos relevantes para el dominio del problema. Esta elección resultó acertada, permitiendo concentrar los esfuerzos en la lógica de negocio y la arquitectura del sistema de reservas.

5.3. Aspectos técnicos específicos del stack

Una de las implementaciones técnicas más relevantes del proyecto fue la configuración de *workflows* automatizados de CI/CD que garantizaran la calidad del código y la entrega continua de la aplicación. Para ello se desarrollaron dos *workflows* principales en GitHub Actions que se ejecutan automáticamente en cada *push* y *pull request* al repositorio.

Los primeros tres *jobs* corresponden a la preparación de la estructura básica para el *pipeline*, la preparación del entorno para Java y Maven, y la compilación y empaquetado de los artefactos necesarios. Estos *jobs* incluyen la descarga automática de dependencias Maven, la compilación del código fuente, la generación del archivo JAR [39] y su posterior almacenamiento como artefacto en GitHub. La configuración incluye optimizaciones como el *caching* de dependencias Maven para reducir los tiempos de *build* y la paralelización de tareas donde sea posible.

Los otros tres *jobs* corresponden a la configuración para la validación, análisis y reporte de la integración continua. Esta parte resultó ser la implementación más compleja, ya que requirió la configuración de un contenedor Docker con MySQL dedicado para las pruebas de acceso a datos. Esta decisión técnica implicó parametrizar una instancia de MySQL como servicio de GitHub Actions con variables de entorno específicas, así como configurar los *health checks* correspondientes para garantizar que la base de datos estuviera completamente operativa antes de ejecutar los *tests* de integración. Una vez finalizado el análisis y con los artefactos generados, el sistema genera un informe final que resume el estado de la ejecución en el que se indica si la construcción del proyecto y el análisis de calidad han sido exitosos o, si por el contrario, han fallado en alguna etapa. Así, estos tres jobs en conjunto aseguran que el código no solo se compila correctamente, sino que además cumple con los estándares de calidad y generando la información sobre el proceso queda registrada y accesible.

El *workflow* orquesta, en esta última fase, la ejecución secuencial de *tests* unitarios (que no requieren base de datos) y *tests* de integración (que utilizan la instancia MySQL), la generación de reportes de cobertura unificados con JaCoCo [29], y la transmisión de métricas hacia SonarCloud para el análisis estático. La configuración requirió la gestión de *secrets* de GitHub

para las claves de autenticación de SonarCloud, así como la configuración específica de paths de exclusión para evitar el análisis de código generado automáticamente.

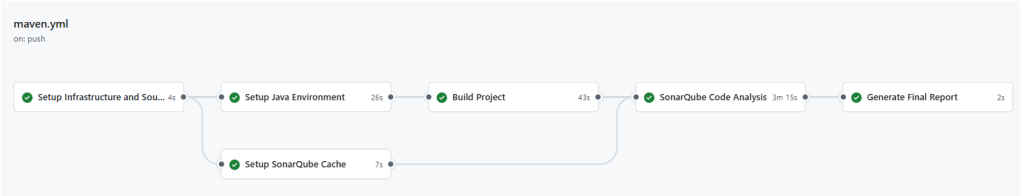

Figura 5.1: Secuencia con las distintas tareas del Workflow.

Un desafío técnico significativo fue sincronizar los reportes de cobertura de JaCoCo con los requisitos de SonarCloud, que implicó configurar correctamente los paths relativos en el archivo "sonar-project.properties" y asegurar que los reportes XML se generaran en las ubicaciones exactas esperadas por SonarCloud. Esta configuración permite obtener métricas objetivas de calidad como cobertura de código, complejidad ciclomática, duplicación de código y detección automática de *code smells*, proporcionando retroalimentación continua sobre la salud técnica del proyecto.

6. Trabajos relacionados

El diseño de sistemas para gestionar reservas es un campo que ha evolucionado mucho en los últimos años, especialmente gracias al crecimiento de internet y a que las empresas han digitalizado sus gestiones. En este apartado se presenta un análisis de las soluciones más relevantes en el campo de la gestión de reservas de recursos, sistemas de planificación colaborativa y aplicaciones web empresariales que han servido como referencia y contexto para el desarrollo del presente proyecto.

6.1. Soluciones comerciales de gestión de reservas

Microsoft Bookings y Microsoft 365

Microsoft Bookings [23] representa uno de los sistemas de reservas más extendidos en el ámbito empresarial, integrado nativamente en el ecosistema Microsoft 365 [22]. Esta aplicación ha destacado por su capacidad de sincronización automática con Outlook y su integración con Microsoft Teams para reuniones virtuales. Sin embargo, presenta limitaciones significativas en términos de personalización y flexibilidad, especialmente en organizaciones con necesidades específicas en gestión de espacios físicos. La arquitectura de Microsoft Bookings se rige por un modelo SaaS (Software as a Service) [2] que, aunque por un lado reduce la complejidad de mantenimiento, por otro lado limita las posibilidades de personalización y control sobre los datos. En contraposición a esta solución, el enfoque adoptado en el presente TFG permite una mayor flexibilidad de configuración y la posibilidad de adaptarse a requisitos específicos de la organización.

Google Workspace y Calendar

El sistema de reservas de Google Workspace [16], centrado en Google Calendar [15], destaca por su simplicidad de uso y la excelente integración que establece con otros servicios de Google, pero presenta limitaciones en la gestión granular de recursos y la falta de funcionalidades específicas para la gestión de espacios físicos. Una característica destacable de Google Calendar es su API robusta que permite integraciones personalizadas. Este aspecto ha influido en la decisión de diseñar la aplicación del presente TFG con una arquitectura modular que facilite futuras integraciones con sistemas externos.

Robin y sistemas especializados

Entre las soluciones actualmente consideradas de nueva generación de sistemas especializados en la gestión de espacios de trabajo se encuentra Robin [30], que se hizo especialmente popular tras la pandemia de COVID-19. Este tipo de sistemas incorporan funcionalidades avanzadas como análisis de ocupación, integración con sistemas IoT y gestión de protocolos sanitarios. Aunque Robin ofrece funcionalidades avanzadas, el modelo de licenciamiento que conlleva por usuario y su enfoque hacia grandes organizaciones lo hace menos accesible para pequeñas y medianas empresas. Esta limitación no se da en la aplicación desarrollada, ofreciendo una solución propia que se puede adaptar a diferentes tamaños de organización sin restricciones de licenciamiento.

6.2. Enfoques académicos y de investigación

Optimización de recursos compartidos

En el ámbito académico, se han desarrollado investigaciones sobre algoritmos de optimización que buscan asignar de forma eficiente espacios compartidos, aprovechando patrones de uso histórico y predicciones de demanda. Entre estos enfoques destaca la idea de la “reserva inteligente”, capaz de sugerir horarios y espacios de manera automática según la disponibilidad de los participantes y el historial de uso. Aunque se trata de soluciones técnicamente avanzadas, su puesta en práctica requiere disponer de grandes volúmenes de datos y capacidades de *machine learning* que superan el alcance de la actual propuesta. Aun así, los principios que persiguen optimizar la experiencia del usuario y reducir conflictos han servido como base para diseñar las funciones de validación incluidas en el actual proyecto.

Arquitecturas de sistemas colaborativos

La investigación sobre arquitecturas escalables para sistemas de colaboración empresarial ofrece una base sólida para diseñar aplicaciones web capaces de atender a muchos usuarios al mismo tiempo. Enfoques como la arquitectura de microservicios o el uso de patrones como CQRS (Command Query Responsibility Segregation) aportan ideas útiles para gestionar la consistencia de los datos en sistemas de reservas. Si bien las arquitecturas distribuidas pueden ser más complejas de lo que un sistema básico requiere, sus principios —como la separación de responsabilidades y la gestión eficiente de la concurrencia— han inspirado la elección de una arquitectura por capas, con una clara división entre las operaciones de lectura y escritura.

Estudios de usabilidad en sistemas empresariales

En los sistemas de gestión empresarial, la experiencia del usuario es importante para que las personas adopten la herramienta y se sientan cómodas usándola. Factores como que el proceso de reserva sea sencillo, que la disponibilidad se muestre de forma clara y que sea fácil hacer cambios sin perderse, pueden marcar la diferencia. Teniendo esto en cuenta, en este proyecto se han incorporado mejoras como calendarios visuales para ver la disponibilidad de un vistazo, formularios que dan respuesta inmediata a las acciones del usuario y flujos de trabajo más simples para las tareas que se realizan con mayor frecuencia.

6.3. Posicionamiento del proyecto desarrollado

Frente a las soluciones analizadas, el presente proyecto se posiciona como una alternativa que combina las siguientes características que se muestran la siguiente tabla comparativa:

Característica	**Microsoft Booking**	**Google Worskpace**	**Robin**	***ReservApp***
Control y flexibilidad	Sí	Sí	Sí	Sí
Arquitectura moderna	Sí	Sí	No	Sí
Enfoque híbrido	No	No	No	Sí
Escalabilidad progresiva	Sí	Sí	Sí	Sí
Gestión colaborativa	Sí	Sí	Sí	Sí
Arquitectura educativa	No	No	No	Sí
Metodología adaptada	No	No	No	Sí
Licenciamiento gratuito	No	Sí	No	Sí

Tabla 6.1: Comparativa de aplicaciones

Limitaciones y contexto

Es importante reconocer las limitaciones del proyecto actual:

- **Alcance funcional**: Enfocado a necesidades básicas de gestión de reservas.
- **Funcionalidades avanzadas**: Ausencia de características como IA predictiva o análisis avanzado.
- **Integración**: Limitaciones en conectividad con sistemas empresariales complejos.

7. Conclusiones y Líneas de trabajo futuras

El desarrollo de la aplicación web de gestión de reservas de salas de reuniones ha constituido una experiencia completa que ha permitido abordar un problema real del entorno empresarial mediante la aplicación de tecnologías y metodologías modernas de desarrollo de software. En este apartado se presentan las conclusiones obtenidas tras la finalización del proyecto, tanto desde la perspectiva de los resultados funcionales alcanzados como desde el punto de vista técnico y metodológico. Asimismo, se identifican las líneas de trabajo futuras que podrían dar continuidad y evolución al sistema desarrollado.

7.1. Conclusiones relacionadas con los resultados del proyecto

Cumplimiento de objetivos funcionales

El proyecto ha logrado cumplir satisfactoriamente los objetivos planteados inicialmente, materializándose en una aplicación web completamente funcional que resuelve eficazmente los problemas de gestión de reservas identificados en el entorno laboral. La centralización de la gestión de reservas se ha conseguido mediante una plataforma web accesible que elimina la informalidad previa en la asignación de salas, proporcionando visibilidad completa sobre la disponibilidad de recursos y evitando los conflictos de solapamiento que anteriormente generaban pérdida de tiempo y confusión entre los empleados.

La evolución del sistema hacia un modelo colaborativo ha superado las expectativas iniciales. Lo que comenzó como un sistema básico de reservas individuales evolucionó hacia una plataforma de gestión de convocatorias que permite la invitación de participantes y la notificación automática por correo electrónico. Esta funcionalidad ha demostrado ser especialmente valiosa en el contexto laboral, ya que integra la gestión de espacios con la coordinación de equipos, proporcionando una solución más completa y alineada con las necesidades de la empresa.

La flexibilidad en la gestión temporal mediante el sistema híbrido que soporta tanto duraciones libres como slots predefinidos ha demostrado ser una decisión acertada. Esta característica permite que el sistema se adapte a diferentes tipos de reuniones y políticas organizacionales, desde reuniones rápidas de coordinación hasta sesiones de trabajo extensas, manteniendo siempre la coherencia en la prevención de conflictos.

7.2. Conclusiones técnicas

Arquitectura y patrones de diseño

La utilización del patrón MVC combinado con arquitectura por capas ha demostrado ser una decisión arquitectónica sólida que ha facilitado tanto el desarrollo como el mantenimiento del sistema. La separación clara de responsabilidades ha permitido evolucionar diferentes aspectos de la aplicación de forma independiente, como se evidenció durante la incorporación de funcionalidades colaborativas sin afectar a la lógica de presentación o el acceso a datos.

La implementación de patrones empresariales como *Repository*, *Service Layer* y DTO ha proporcionado una base de código robusta y mantenible. El patrón *Repository*, en particular, ha facilitado significativamente el testing mediante la creación de *mocks*, mientras que el *Service Layer* ha centralizado la lógica de negocio, simplificando la gestión de transacciones y validaciones complejas.

La decisión de utilizar borrado lógico en lugar de eliminación física de registros ha demostrado ser una solución interesante para mantener la integridad y trazabilidad de los datos. Esta aproximación ha proporcionado capacidades de auditoría que son interesantes en un entorno empresarial, permitiendo la recuperación de datos que ha resultado valiosa durante las fases de *testing* y validación.

Tecnologías y herramientas

La elección de Spring Boot como *framework* principal ha demostrado ser acertada, proporcionando un ecosistema completo y bien integrado que ha simplificado significativamente el desarrollo. La configuración automática, la gestión de dependencias y la integración nativa con herramientas como Spring Security y Spring Data JPA han permitido concentrar los esfuerzos en la lógica de negocio específica del problema.

El cambio de JSF con PrimeFaces hacia Thymeleaf resultó ser una decisión apropiada para el éxito del proyecto. Aunque inicialmente representó un retraso en el cronograma, la integración óptima con Spring Boot y la menor complejidad de configuración compensaron ampliamente esta inversión de tiempo. La capacidad de crear plantillas HTML válidas que pueden visualizarse sin el procesamiento por parte del servidor facilitó considerablemente el desarrollo y *debug* de la interfaz de usuario.

La configuración de CI/CD con GitHub Actions y SonarCloud ha proporcionado un nivel de profesionalización al proyecto que excede las expectativas típicas de un TFG. La automatización de *tests*, análisis de calidad y despliegue ha garantizado la consistencia del código y ha proporcionado métricas objetivas de calidad que han sido enriquecedoras para la validación técnica del proyecto.

Metodología de desarrollo

La gestión mediante Scrum adaptado al contexto académico ha demostrado ser efectiva para gestionar la complejidad y evolución del proyecto. La organización en *sprints* alineados con las revisiones académicas permitió una gestión equilibrada entre el desarrollo funcional y los requisitos de documentación. La flexibilidad inherente a las metodologías ágiles fue apropiada para incorporar cambios significativos como la evolución hacia el modelo colaborativo de reservas.

La gestión de riesgos técnicos mediante la identificación temprana de problemas potenciales y la implementación de estrategias de mitigación (como los perfiles específicos para *testing* y la configuración de servicios MySQL en CI/CD) evitó bloqueos significativos en el desarrollo y mantuvo el proyecto dentro de los plazos académicos establecidos.

7.3. Análisis crítico y áreas de mejora

Limitaciones identificadas

A pesar de los resultados positivos obtenidos, el análisis retrospectivo permite identificar varias áreas de mejora que podrían ser abordadas en futuras iteraciones. La interfaz de usuario, aunque funcional e intuitiva, podría beneficiarse de un diseño más moderno y responsive que aproveche mejor las capacidades de los dispositivos móviles. La dependencia de navegadores web para todas las interacciones limita la experiencia del usuario en contextos donde el acceso rápido desde dispositivos móviles podría ser preferible.

La gestión de concurrencia en reservas simultáneas, aunque funcional, podría ser más sofisticada. El sistema actual previene efectivamente los conflictos de solapamiento, pero no optimiza la experiencia del usuario cuando múltiples personas intentan reservar recursos en horarios contiguos. Un sistema de sugerencias inteligentes podría mejorar significativamente esta situación.

El sistema de notificaciones, aunque cumple su función básica, carece de personalización avanzada. Los usuarios no pueden configurar sus preferencias de notificación o elegir diferentes canales de comunicación según el tipo de convocatoria, lo que limita su adaptabilidad a diferentes estilos de trabajo y preferencias organizacionales.

Aspectos técnicos a mejorar

Desde el punto de vista técnico, la arquitectura monolítica actual, aunque apropiada para el alcance del proyecto, podría constituir una limitación para la escalabilidad futura en organizaciones grandes. La separación en microservicios podría proporcionar mejor escalabilidad y mantenibilidad a largo plazo.

La gestión de sesiones y autenticación podría beneficiarse de mecanismos más avanzados como autenticación multifactor o integración con sistemas de identidad corporativos (LDAP, Active Directory) [5]. Actualmente, el sistema depende de autenticación básica con credenciales propias, lo que puede no ser suficiente para organizaciones con políticas de seguridad estrictas.

7.4. Líneas de trabajo futuras

Mejoras funcionales inmediatas

Aplicación móvil nativa: El desarrollo de una aplicación móvil complementaria podría mejorar significativamente la accesibilidad del sistema. Una app nativa permitiría notificaciones *push*, acceso *offline* a reservas personales y funcionalidades específicas como *check-in* automáticas basadas en geolocalización.

Dashboard analítico: La implementación de un dashboard con métricas de uso proporcionaría información valiosa sobre patrones de utilización de salas, permitiendo optimizaciones de recursos y planificación de espacios. Métricas como ocupación promedio, patrones temporales de uso y análisis de demanda podrían ayudar a la toma de decisiones sobre la gestión de instalaciones.

Sistema de aprobaciones: Para organizaciones con estructuras jerárquicas, un sistema de *workflow* de aprobaciones podría añadir control administrativo sobre ciertas reservas o recursos especiales, permitiendo diferentes niveles de autorización según el tipo de usuario y/o el recurso solicitado.

Evoluciones tecnológicas

Integración con sistemas de calendario: La sincronización bidireccional con sistemas como Outlook, Google Calendar o calendarios corporativos eliminaría la duplicidad en la gestión de eventos y mejoraría significativamente la adopción del sistema.

API REST pública: El desarrollo de una API REST completa facilitaría integraciones con otros sistemas empresariales como ERPs, sistemas de RRHH o plataformas de videoconferencia, ampliando el ecosistema de funcionalidades disponibles.

Inteligencia artificial: La implementación de algoritmos de *machine learning* podría proporcionar funcionalidades avanzadas como predicción de demanda, sugerencias inteligentes de horarios basadas en patrones históricos y optimización automática de asignación de recursos.

Migración hacia microservicios: Una evolución natural del proyecto sería la separación en microservicios especializados (gestión de usuarios, reservas, notificaciones, análisis) que permitiría una escalabilidad independiente y mejor mantenibilidad en entornos empresariales grandes.

Containerización y orquestación: La implementación con Docker y Kubernetes proporcionaría capacidades de despliegue más robustas y escalables, facilitando la adopción en diferentes entornos de infraestructura.

Reservas de recursos diversos: La expansión más allá de salas de reunión hacia equipamiento técnico, espacios de trabajo flexibles, vehículos corporativos o instalaciones deportivas convertiría el sistema en una plataforma integral de gestión de recursos organizacionales.

7.5. Conclusiones finales

El desarrollo de este Trabajo de Fin de Grado ha cumplido satisfactoriamente tanto los objetivos académicos como los funcionales planteados inicialmente. La aplicación resultante constituye una solución robusta y escalable que aborda efectivamente los problemas de gestión de reservas identificados en el entorno laboral.

Desde el punto de vista académico, el proyecto ha permitido la aplicación práctica de conocimientos teóricos en un contexto real, demostrando la viabilidad de tecnologías y metodologías modernas de desarrollo de software. La experiencia adquirida en la gestión de proyectos ágiles, implementación de arquitecturas empresariales y configuración de *pipelines* de CI/CD constituye una base sólida para el desarrollo profesional.

La identificación de líneas de trabajo futuras evidencia el potencial de evolución del proyecto hacia una solución empresarial completa, con posibilidades reales de comercialización y escalabilidad. El fundamento técnico sólido establecido facilita estas evoluciones futuras y posiciona el proyecto como una base viable para emprendimientos tecnológicos en el sector de gestión de recursos empresariales.

En definitiva, este TFG ha demostrado que es posible desarrollar soluciones tecnológicas profesionales que combinen rigor académico con aplicabilidad práctica, estableciendo una solución valiosa tanto para futuros proyectos académicos como para iniciativas empresariales en el ámbito de la digitalización de procesos organizacionales.

Bibliografía

[1] AJAX. Ajax introduction. https://www.w3schools.com/xml/ajax_intro.asp/, 2025. [En línea; Consultado 07-agosto-2025].

[2] Amazon. ¿qué es el software como servicio (saas)? https://aws.amazon.com/es/what-is/saas/, 2025. [En línea; Consultado 09-agosto-2025].

[3] Arsys. Principios solid en programación orientada a objetos. https://www.arsys.es/blog/principios-solid-en-la-programacion-orientada-a-objetos/, 2024. [En línea; Consultado 09-agosto-2025].

[4] The Project Lombok Authors. Project lombok. https://projectlombok.org/, 2025. [En línea; Consultado 03-agosto-2025].

[5] Ciberseguridad. Ldap: qué es, cómo funciona, usos y riesgos de seguridad. https://ciberseguridad.com/guias/prevencion-proteccion/ldap/, 2025. [En línea; Consultado 10-agosto-2025].

[6] Geeks for Geeks. What is bootstrap? https://www.geeksforgeeks.org/bootstrap/what-is-bootstrap/, 2025. [En línea; Consultado 16-septiembre-2025].

[7] Geeks for Geeks. What is javascript? https://www.geeksforgeeks.org/javascript/what-is-javascript/, 2025. [En línea; Consultado 16-septiembre-2025].

[8] Commonhaus Foundation. Hibernate. https://hibernate.org/, 2025. [En línea; Consultado 03-agosto-2025].

[9] Eclipse Foundation. What is eclipse? https://www.eclipse.org/home/whatis/, 2025. [En línea; Consultado 16-septiembre-2025].

[10] GeeksForGeeks. Client-Server Model. https://www.geeksforgeeks.org/system-design/client-server-model/, 2025. [En línea; Consultado 03-agosto-2025].

[11] GeeksForGeeks. Model-View-Controller(MVC) architecture for Node applications. https://www.geeksforgeeks.org/node-js/model-view-controllermvc-architecture-for-node-applications/, 2025. [En línea; Consultado 04-septiembre-2025].

[12] GeeksForGeeks. What is a Framework? https://www.geeksforgeeks.org/blogs/what-is-a-framework/, 2025. [En línea; Consultado 03-agosto-2025].

[13] GitHub. Github actions documentation. https://docs.github.com/en/actions/, 2025. [En línea; Consultado 04-septiembre-2025].

[14] GitHub. Github copilot. https://marketplace.eclipse.org/content/github-copilot/, 2025. [En línea; Consultado 07-agosto-2025].

[15] Google. Google calendar. https://workspace.google.com/intl/es/products/calendar/, 2025. [En línea; Consultado 09-agosto-2025].

[16] Google. Google workspace. https://workspace.google.com/intl/es/, 2025. [En línea; Consultado 09-agosto-2025].

[17] Red Hat. La integración y la distribución continuas (ci/cd). https://www.redhat.com/es/topics/devops/what-is-ci-cd/, 2025. [En línea; Consultado 10-agosto-2025].

[18] Red Hat. Yaml: qué es y usos. https://www.redhat.com/es/topics/automation/what-is-yaml/, 2025. [En línea; Consultado 09-septiembre-2025].

[19] Pivit Inc. Zube docs. https://zube.io/docs/, 2025. [En línea; Consultado 04-septiembre-2025].

[20] PrimeTek Informatics. Primefaces. https://www.primefaces.org/, 2025. [En línea; Consultado 09-agosto-2025].

[21] Junit. Junit. https://junit.org/, 2025. [En línea; Consultado 16-septiembre-2025].

[22] Microsoft. Microsoft 365. https://learn.microsoft.com/es-es/microsoft-365/?view=o365-worldwide/, 2025. [En línea; Consultado 09-agosto-2025].

[23] Microsoft. Microsoft bookings. https://learn.microsoft.com/es-es/microsoft-365/bookings/bookings-overview?view=o365-worldwide/, 2025. [En línea; Consultado 09-agosto-2025].

[24] Mockito. mockito. https://site.mockito.org/, 2025. [En línea; Consultado 16-septiembre-2025].

[25] Chioma Sylvia Okoro. Sustainable facilities management in the built environment: A mixed-method review. *Sustainability*, 15(4), 2023.

[26] Oracle. Javaserver faces technology. https://www.oracle.com/java/technologies/javaserverfaces.html/, 2025. [En línea; Consultado 09-agosto-2025].

[27] Oracle. Mysql. https://www.mysql.com/, 2025. [En línea; Consultado 03-agosto-2025].

[28] OverLeaf. OverLeaf, Online LaTeX Editor. https://www.overleaf.com/, 2025. [En línea; Accedido 06-agosto-2025].

[29] Platzi. Jacoco y la cobertura de pruebas en el código. https://platzi.com/tutoriales/1503-testing-java/3841-jacoco-y-la-cobertura-de-pruebas-en-el-codigo/, 2025. [En línea; Consultado 09-agosto-2025].

[30] Robin. The ai platform for workplace operations. https://robinpowered.com/, 2025. [En línea; Consultado 09-agosto-2025].

[31] VMware Tanzu. Sending email. https://docs.spring.io/spring-boot/reference/io/email.html/, 2025. [En línea; Consultado 09-agosto-2025].

[32] VMware Tanzu. Spring. https://spring.io/, 2025. [En línea; Consultado 03-agosto-2025].

[33] VMware Tanzu. Spring boot. https://spring.io/projects/spring-boot/, 2025. [En línea; Consultado 03-agosto-2025].

[34] VMware Tanzu. Spring data jpa. https://spring.io/projects/spring-data-jpa/, 2025. [En línea; Consultado 03-agosto-2025].

[35] VMware Tanzu. Spring security. https://spring.io/projects/spring-security/, 2025. [En línea; Consultado 03-agosto-2025].

[36] Thymeleaf. Thymeleaf. https://www.thymeleaf.org/, 2024. [En línea; Consultado 03-agosto-2025].

[37] TortoiseGit. The power of git in a windows shell. https://tortoisegit.org/, 2025. [En línea; Consultado 07-agosto-2025].

[38] Wikipedia. Protocolo para transferencia simple de correo. https://es.wikipedia.org/wiki/Protocolo_para_transferencia_simple_de_correo/, 2025. [En línea; Consultado 09-agosto-2025].

[39] Cecilio Álvarez Caules. Java war (el concepto de web archive). https://www.arquitecturajava.com/modulos-de-java-i-empaquetamiento-jar/, 2013. [En línea; Consultado 14-septiembre-2025].

[40] Cecilio Álvarez Caules. Transacciones acid y sus propiedades. https://www.arquitecturajava.com/transacciones-acid-y-sus-propiedades/, 2021. [En línea; Consultado 09-agosto-2025].