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 — 9 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, 9 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 entorno empresarial actual, la gestión eficiente de los espacios de trabajo es necesaria para garantizar una organización adecuada y un uso eficiente de los recursos disponibles. En muchas empresas, las salas de reuniones son espacios que se utilizan para la coordinación de equipos, la realización de presentaciones y la toma de decisiones estratégicas. Sin embargo, la falta de un sistema apropiado para la reserva de estas salas puede generar problemas como la ocupación simultánea, la falta de visibilidad sobre su disponibilidad y la dificultad para gestionar cambios o cancelaciones.

En mi lugar de trabajo, este problema es recurrente, ya que actualmente la asignación de salas de reuniones se realiza de manera informal, lo que provoca confusión entre los empleados, pérdida de tiempo y conflictos en la planificación de reuniones. Para abordar esta problemática, se propone el desarrollo de una aplicación web que facilite la gestión de reservas de salas de reuniones de manera eficiente, organizada y accesible para todos los empleados.

La implementación de soluciones software para la gestión de espacios se alinea con los principios de la gestión sostenible de instalaciones. Como señalan [Autor et al., 2023] [20] en su revisión sobre gestión sostenible de instalaciones, las instalaciones adecuadamente gestionadas no solo contribuyen al ahorro de costos, sino que también mejoran la moral y eficiencia de los empleados. En este contexto, un sistema de reservas de salas de reuniones no solo optimiza el uso de los espacios físicos disponibles, sino que también reduce el desperdicio de recursos energéticos asociados a salas reservadas pero no utilizadas, mientras mejora la experiencia laboral de los empleados al eliminar incertidumbres y 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 [27].
- Dirección de la documentación del proyecto [26].
- Dirección de las métricas del proyecto [25].
- Dirección de la web de Reservas usada en el desarrollo [24].
- Dirección del video con descripción del proyecto [29].
- Dirección del video con demostración funcional de la aplicación [28].

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

El objetivo de este Trabajo de Fin de Grado es diseñar e implementar una solución tecnológica que mejore la gestión de las reservas de salas en el entorno laboral, facilitando la coordinación entre los empleados y mejorando la eficiencia en el uso de estos espacios. Para ello, se desarrollará una aplicación web con una interfaz intuitiva y accesible, que permitirá a los usuarios registrar y administrar reservas de manera ágil y sin ambigüedades.

Este proyecto no solo busca solucionar un problema específico dentro de mi empresa, sino que también servirá como un caso práctico de aplicación de tecnologías web en la gestión de recursos compartidos. A lo largo del desarrollo, se explorarán diferentes herramientas y metodologías para garantizar que la solución final sea escalable, segura y fácil de utilizar.

2.2. Objetivos técnicos

La propuesta implica el desarrollo de una aplicación web de tipo cliente-servidor [7] que se construirá utilizando, como principales *frameworks* [9], *Spring Boot* [34] y *Thymeleaf* [37], lo que permitirá crear una interfaz dinámica y robusta.

La seguridad será una prioridad máxima, integrando *Spring Security* [36] para garantizar que solo los usuarios autorizados tengan acceso a funcionalidades específicas, protegiendo así la integridad del sistema.

Para gestionar la evolución del código y el versionado del mismo, se empleará *Git* como sistema de control de versiones distribuido, en conjunto con un repositorio en *GitHub*. Esto permitirá un seguimiento detallado de los cambios realizados.

La estructura de datos será cuidadosamente diseñada, optando por un modelo relacional normalizado para asegurar la coherencia y eficiencia en el manejo de la información. Además, la aplicación seguirá la arquitectura MVC (Modelo-Vista-Controlador) [8], lo cual favorecerá la modularidad y facilitará el mantenimiento del código.

Para el proceso de construcción del software, la gestión de dependencias y la compilación, se utilizará Maven como herramienta de automatización. Además, se implementarán herramientas de integración continua, como *GitHub Actions* [10], integradas directamente con el repositorio para garantizar un control de calidad constante y automatizado.

Finalmente, el proyecto adoptará la metodología ágil Scrum, utilizando *Zube* [16] como herramienta de gestión, lo que permitirá una planificación flexible y adaptable a los cambios, entregando valor de manera iterativa. Además, se implementarán *tests* unitarios y de integración exhaustivos 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 este 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.

Los sistemas de reservas buscan optimizar la utilización de los recursos disponibles y reducir los conflictos derivados de dobles reservas y mejorar la eficiencia operativa. Un buen sistema debe incluir funcionalidades como la gestión de disponibilidad en tiempo real, la modificación de reservas, la integración con calendarios empresariales y la autenticación de usuarios.

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, permitiéndos 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.

3.2. 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 [34] es un *framework* [9] basado en Spring [33] 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 [35]

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

Hibernate [6]

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 [36]

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 [37]

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

MySQL [22]

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 [41].

3.3. Arquitectura de Aplicaciones Web

Patrones de Arquitectura

La aplicación sigue el patrón Modelo-Vista-Controlador (MVC) [8], de forma estructurada y desacoplada, aprovechando al máximo las características del ecosistema de Spring para crear 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.

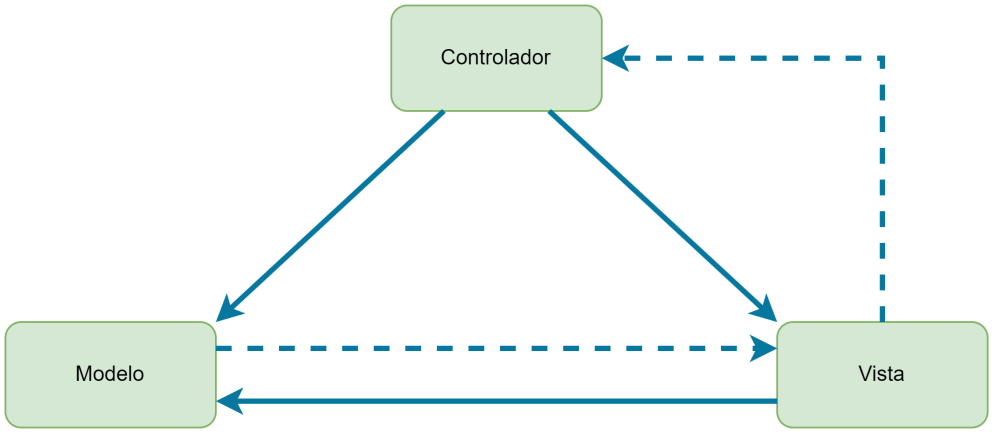

Figura 3.1: Patrón modelo MVC.

Arquitectura Cliente-Servidor

La arquitectura de la aplicación web sigue un modelo cliente-servidor [7], 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.

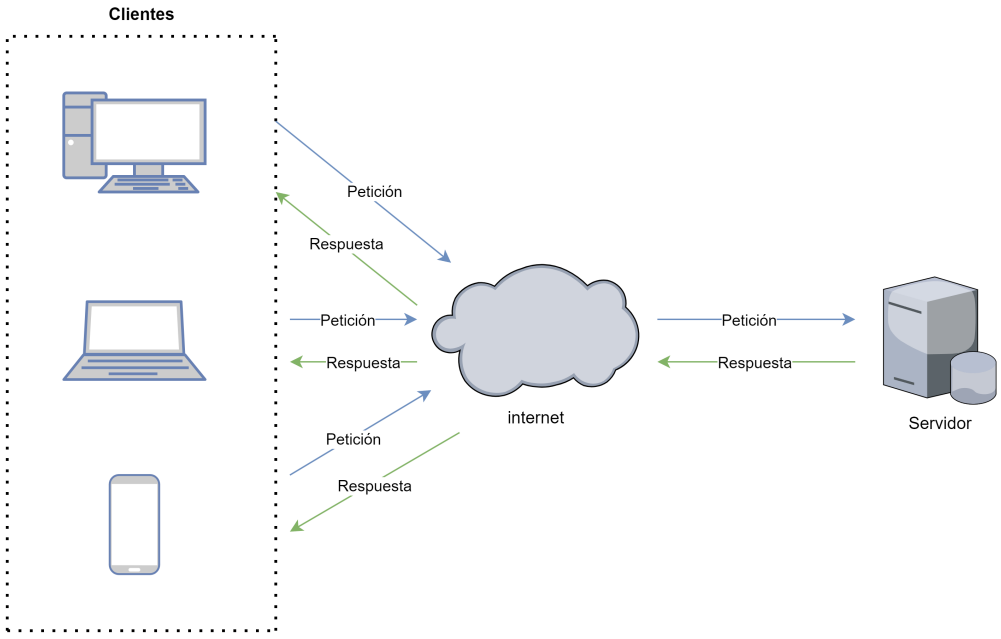

Figura 3.2: Arquitectura Cliente-Servidor.

4. Técnicas y herramientas

4.1. Metodologías

Scrum

Como marco de trabajo para la gestión del proyecto, se ha aplicado la metodología ágil Scrum. Scrum permite abordar el desarrollo de forma iterativa e incremental, facilitando la adaptación a los cambios y la mejora continua del producto. Mediante la planificación en *sprints* de corta duración, se ha podido llevar a cabo la organización del trabajo de manera efectiva, asegurando que cada fase del proyecto se revisa y ajusta constantemente.

4.2. Gestión del proyecto

Repositorio

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

GitHub trabaja sobre un repositorio Git que lleva un control de versiones robusto y que permite documentar todo el proceso de desarrollo del proyecto académico, además ofrece la posibilidad de integración nativa con herramientas de CI/CD [14] facilitando la demostración de las buenas prácticas de desarrollo llevadas a cabo en el actual trabajo. Además, GitHub ofrece la posibilidad de trabajar con *issues* para organizar las diferentes tareas, pudiendo organizarlas por etiquetas o prioridades, así como la opción de *taggear* las diferentes versiones o *releases* que se alcanzan durante el desarrollo del proyecto.

Control de versiones

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

Git es un sistema de control de versiones estándar en la industria que ha demostrado robustez y flexibilidad en todo tipo de desarrollos. Su capacidad de ramificación y fusión eficiente permite experimentar con diferentes funcionalidades sin afectar el código principal, y por su naturaleza distribuida garantiza múltiples respaldos del proyecto, permitiendo trabajar offline, aspecto muy útil para el desarrollo de cualquier tipo de proyecto. Además, como cliente de Git, se ha utilizado TortoiseGit [38] para todo lo referente a la documentación del proyecto.

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 [15] 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 la documentación del proyecto se ha elegido LaTeX desde la plataforma Overleaf [23] por su capacidad de generar documentos académicos de una notable calidad con formatos consistentes y alta calidad tipográfica. La colaboración en tiempo real en la nube permite trabajar desde cualquier dispositivo y compartir fácilmente con los tutores para las revisiones. Por otro lado, la gestión automática de referencias bibliográficas y numeración elimina errores manuales y garantiza el cumplimiento de los estándares académicos.

4.3. 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 la aplicación web de gestión de reservas con Spring Boot, se utilizó 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 combinación 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, facilita la implementación de principios SOLID [3] y permite el *testing* independiente de cada capa, donde se puede probar la lógica de negocio sin depender de la interfaz de usuario o la base de datos. Esta sinergia entre arquitectura por capas y MVC es una buena práctica que conduce a un diseño maduro y profesional que cumple con los estándares de la industria 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 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, mientras que el uso de este patrón garantiza el cumplimiento del principio de inversión de dependencias, creando una arquitectura limpia y mantenible.

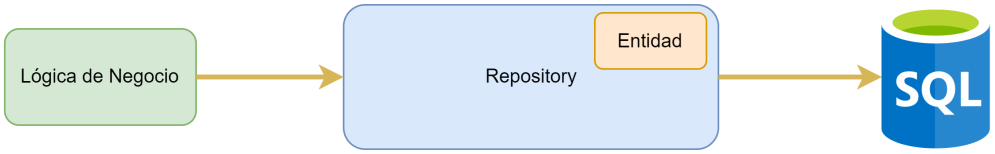

Figura 4.1: Patrón Repository.

Patrón Service Layer

Gracias a la utilización del patrón de diseño MVC comentado anteriormente, conduce casi de forma intuitiva o directa al uso del patrón *service layer* actuando 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 ligeros y 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*, mientras que permite reutilizar 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 al 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. Además, facilitan la evolución independiente de la API y el modelo de datos, donde se pueden

modificar las entidades de base de datos sin afectar los contratos de la API, y mejoran el rendimiento al reducir la cantidad de datos transferidos.

4.4. Librerías

JUnit

JUnit es considerado el *framework* de *testing* estándar de facto en Java que 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 por su capacidad de crear *mocks* y *stubs* de dependencias que permiten aislar completamente los elementos bajo prueba, 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 la capa de servicios en la arquitectura por capas, 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, se ha utilizado Thymeleaf 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 perfectamente 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, se ha utilizado Bootstrap 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 perfectamente 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 por su capacidad para añadir interactividad dinámica para 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 decidido el uso de un CSS personalizado por su capacidad de adaptación y extensión de los estilos de Bootstrap sin modificar el *framework* base, permitiendo crear una identidad visual propia para la aplicación mediante variables CSS personalizadas y una sobreescritura 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, mientras 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 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 [11] 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 costos 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 reservas, se adoptó un ciclo de vida iterativo-incremental basado en la metodología ágil Scrum, estructurado en *sprints* de una semana que permitió una evolución controlada y adaptativa del producto. Aunque el uso de scrum no ha sido del rigurosamente estricto a todas las directrices establecidas por el marco de trabajo, principalmente debido a que no se trataba de un equipo de desarrollo al uso, sí que 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 como las librerías de testing, así como posibles problemas de rendimiento en consultas concurrentes de 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 identificó la necesidad de

implementar funcionalidades como notificaciones por correo, el registro de auditoría para identificar quién creaba o modificaba la información y cuándo se producía. 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 filtrar registros válidos. Esta decisión, aunque incrementó la complejidad inicial, proporcionó trazabilidad completa de las operaciones y capacidad de recuperación de datos.

Otro cambio arquitectónico 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 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 emergieron nuevas necesidades que requirieron ajustes arquitectónicos significativos. Esta circunstancia ocurrió tras una revisión con el tutor, donde se identificó que el modelo de reservas unipersonales era limitante para el contexto incialmente 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 arquitectónicos importantes, incluyendo la reestructuración del modelo de datos para soportar relaciones 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 [32] con configuración SMTP [39], 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) [21] con PrimeFaces [17] como *framework* de interfaz de usuario, motivado 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 migrar hacia 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 procesamiento del servidor, lo cual facilitaba 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. Se desarrollaron dos *workflows* principales en GitHub Actions que se ejecutan automáticamente en cada *push* y *pull request* al repositorio.

El primer *workflow* de compilación y empaquetado se configuró para ejecutarse en múltiples versiones de Java, garantizando la compatibilidad multiplataforma. Este *workflow* incluye la descarga automática de dependencias Maven, la compilación del código fuente, la generación del archivo WAR [40] desplegable y su posterior almacenamiento como artefacto de GitHub. La configuración incluye optimizaciones como el *caching* de dependencias Maven para reducir los tiempos de *build* y la paralelización de tareas donde fue posible.

El segundo *workflow* de *testing* y análisis de calidad representa la implementación más compleja, ya que requirió la configuración de un servicio 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. La configuración incluyó la creación de un perfil específico de Spring Boot (test) con propiedades de conexión que apuntaran al servicio MySQL del workflow, diferenciándolo del perfil de desarrollo que utiliza H2 en memoria.

El *workflow* orquesta 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 [30], 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.

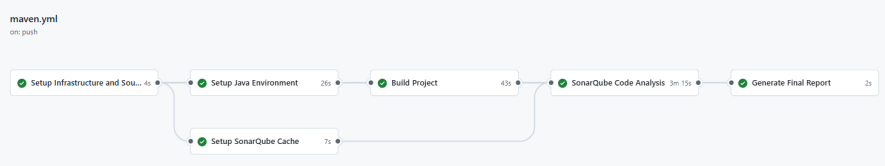

Figura 5.1: Secuencia con las distintas tareas del Workflow.

Un desafío técnico significativo fue sincronizar los reportes de cobertura de JaCoCo con los requisitos de SonarCloud mientras se gestionaban múltiples fuentes de datos (H2 para tests unitarios, MySQL para tests de integración), lo 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 desarrollo de sistemas de gestión de reservas ha sido un área de investigación y desarrollo activo durante las últimas décadas, especialmente con el auge de las tecnologías web y la digitalización de procesos empresariales. 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 [19] representa uno de los sistemas de reservas más extendidos en el ámbito empresarial, integrado nativamente en el ecosistema Microsoft 365 [18]. Este sistema destaca 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 de gestión de espacios físicos. La arquitectura de Microsoft Bookings sigue un modelo SaaS (Software as a Service) [2] que, aunque reduce la complejidad de mantenimiento, limita las posibilidades de personalización y control sobre los datos. En contraste, el enfoque adoptado en el presente TFG permite una mayor flexibilidad arquitectónica y la posibilidad de adaptarse a requisitos específicos de la organización.

Google Workspace y Calendar

El sistema de reservas de Google Workspace [13], centrado en Google Calendar [12], destaca la simplicidad de uso y la excelente integración 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 notable de Google Calendar es su API robusta que permite integraciones personalizadas, aspecto que 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 [31], especialmente populares tras la pandemia de COVID-19. Estos sistemas incorporan funcionalidades avanzadas como análisis de ocupación, integración con sistemas IoT y gestión de protocolos sanitarios. Aunque Robin ofrece funcionalidades avanzadas, su modelo de licenciamiento 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 pudiera adaptarse 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 sistemas más sencillos. 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 este 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 clave para que las personas realmente 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. Frameworks y metodologías de desarrollo

Spring Boot en aplicaciones empresariales

La adopción de Spring Boot en el desarrollo de aplicaciones web empresariales ha demostrado ventajas significativas en términos de productividad y mantenibilidad. Habiendo analizado diferentes modelos arquitectónicos, se muestra que la combinación de Spring Boot con arquitecturas por capas resulta en sistemas más robustos y fáciles de mantener. Estas conclusiones han servido de apoyo para las decisiones arquitectónicas adoptadas en el presente TFG, especialmente en cuanto al uso de patrones como *Repository*, *Service Layer* y la integración con Spring Security para la gestión de autenticación y autorización.

Metodologías ágiles en contextos académicos

La aplicación de metodologías ágiles en el contexto específico de proyectos académicos ha demostrado beneficios en la gestión de restricciones temporales y requisitos evolutivos. Los enfoques de "Scrum Académico" adaptan los principios ágiles a las restricciones específicas de proyectos educativos, incluyendo la gestión de revisiones con los tutores y la documentación académica requerida. Esta metodología ha sido influyente en la planificación del presente proyecto, especialmente en la definición de *sprints* alineados con las revisiones académicas y la gestión del *backlog* considerando tanto objetivos funcionales como requisitos de documentación.

6.4. Posicionamiento del proyecto desarrollado

Frente a las soluciones analizadas, el presente proyecto se posiciona como una alternativa que combina las siguientes características:

- **Control y flexibilidad**: Al ser una solución de código abierto, permite personalización completa sin restricciones comerciales.
- **Arquitectura moderna**: Utiliza tecnologías actuales y patrones de diseño consolidados.
- **Enfoque híbrido**: Combina simplicidad de uso con funcionalidades avanzadas según sea necesario.
- **Escalabilidad progresiva**: Diseño que permite evolución gradual según crecen las necesidades
- **Sistema híbrido de duración**: Flexibilidad para manejar tanto reservas libres como slots predefinidos.
- **Gestión colaborativa**: Funcionalidades de convocatoria con notificaciones integradas.
- **Arquitectura educativa**: Implementación que sirve como caso de estudio para futuros desarrollos.
- **Metodología adaptada**: Aplicación práctica de Scrum en contexto académico.

Limitaciones y contexto

Es importante reconocer las limitaciones del proyecto actual:

- **Alcance funcional**: Enfocado en 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 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 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 reales 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 adopción 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 la lógica de presentación o 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 eficazmente 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 y ha permitido implementar funcionalidades de recuperación de datos que han resultado especialmente valiosas 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 dominio.

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 procesamiento del servidor facilitó considerablemente el desarrollo y *debug* de la interfaz de usuario.

La implementación 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 aplicación de 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 similares 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* automático basado 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 informar decisiones de 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 o recurso solicitado.

Evoluciones tecnológicas

Integración con sistemas de calendario: La sincronización bidireccional con sistemas como Outlook, Google Calendar o calendarios corporativos eliminaría la duplicidad de 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, expandiendo 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 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 futuro.

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

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

[8] 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].

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

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

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

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

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

[14] 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].

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

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

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

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

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

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

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

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

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

[24] Ahmad Mareie Pascual. Acceso a la aplicación de ReservApp. https://PENDIENTE/, 2025. [En línea; Accedido 04-agosto-2025].

[25] Ahmad Mareie Pascual. Métricas de ReservApp en SonarCloud. https://sonarcloud.io/project/overview?id=ReservApp/, 2025. [En línea; Accedido 04-agosto-2025].

[26] Ahmad Mareie Pascual. Repositorio de la documentación de ReservApp en GitHub. https://github.com/AhmadMarPas/TFG-ReservApp/tree/main/documentaci%C3%B3n/, 2025. [En línea; Accedido 04-agosto-2025].

[27] Ahmad Mareie Pascual. Repositorio de ReservApp en GitHub. https://github.com/AhmadMarPas/TFG-ReservApp/, 2025. [En línea; Accedido 04-agosto-2025].

[28] Ahmad Mareie Pascual. Video con la demostración de ReservApp en GitHub. https://github.com/AhmadMarPas/TFG-ReservApp/PENDIENTE, 2025. [En línea; Accedido 04-agosto-2025].

[29] Ahmad Mareie Pascual. Video con la descripción de ReservApp en GitHub. https://github.com/AhmadMarPas/TFG-ReservApp/PENDIENTE, 2025. [En línea; Accedido 04-agosto-2025].

[30] 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].

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

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

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

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

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

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

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

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

[39] 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].

[40] Cecilio Álvarez Caules. Java war (el concepto de web archive). https://www.arquitecturajava.com/modulos-de-java-ii-war/, 2013. [En línea; Consultado 09-agosto-2025].

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