

Práctica Final de Curso:

Desarrollo de una Aplicación
Java con Scrum y MySQL

Joel Eguren Guijarro

Contenido

1. Introducción.....	3
2. Historias de usuario.....	4
2.1. HU1 – Alta sala.....	4
2.2. HU2 – Baja sala.....	5
2.3. HU3 – Modificación sala.....	6
2.4. HU4 – Listar salas.....	7
2.5. HU5 – Alta empleado.....	8
2.7. HU7 – Modificación empleado.....	10
2.8. HU8 – Listar empleados.....	11
2.9. HU9 – Reservar una sala.....	12
1.10. HU10 – Cancelar reserva.....	13
3. Sprint 1.....	14
3.1. Objetivos y planificación.....	14
3.2. Reflexión.....	15
4. Sprint 2.....	16
4.1. Objetivos y planificación.....	16
4.2. Reflexión.....	17
5. Retrospectiva final.....	18

1. Introducción

En este documento se presenta la planificación Scrum relacionada con el proyecto Java para la gestión de reservas de salas de reuniones en una empresa.

El objetivo principal es demostrar los conocimientos adquiridos durante el curso y en las dos semanas de formación en Minsait.

Las tecnologías usadas son:

- **Maven**, para la construcción del proyecto Java y la gestión de tests JUnit.
- **Java**, para la lógica del programa.
- **MySQL**, para la gestión de la base de datos.
- **DBeaver**, para la administración de la base de datos.
- **Trello**, para el tablero de tarjetas Scrum.
- **Git/GitHub**, para el control de versiones y el repositorio.

Este proyecto se desarrolla en un total de cuatro días. El primer día se ha dedicado a la planificación de tareas e historias de usuario en Trello, siguiendo la metodología Scrum. El desarrollo se ha dividido en dos sprints: dos días para el Sprint 1 y un día para el Sprint 2.

Se ha fijado un límite de *Work In Progress (WIP)* de 2 tarjetas cuando se realizan tareas técnicas sencillas, o de 1 tarjeta en el caso de historias de usuario. Esto permitirá optimizar el flujo de trabajo, porque nos centraremos en menos tareas a la vez.

[\(Enlace al tablero Scrum en Trello\)](#)

2. Historias de usuario

A continuación, se presentan todas las historias de usuario comprendidas en el proyecto usando Trello, el orden de las tarjetas se indica en cada Sprint.

2.1. HU1 – Alta sala

Product Backlog

[HU1] Alta sala

+ Añadir Fechas Checklist Miembros Adjunto

Etiquetas

Maven y Java Git/GitHub MySQL +

Descripción Editar

Como administrador,
Quiero dar de alta una nueva sala de reuniones con su nombre, capacidad y recursos disponibles,
Para poder gestionar los espacios disponibles en la empresa.

Estimación: 0,2 horas

☒ Criterios de aceptación Eliminar

0%

- ☐ El sistema permite crear una sala nueva con nombre, capacidad y recursos
- ☐ No se permite guardar una sala con capacidad negativa o vacía
- ☐ No se permite guardar una sala que ya existe

Añade un elemento

☒ Checklist Eliminar

0%

- ☐ Hacer método boolean addRoom(Room room) en RoomDAO
- ☐ Añadir método boolean addRoom(Room room) en MeetingRoomManager

Añade un elemento

Comentarios y Actividad Mostrar detalles

Escribe un comentario...

Eguren Guijarro, Joel ha añadido esta tarjeta a Product Backlog
24 jun 2025, 22:34

2.2. HU2 – Baja sala

Product Backlog ▾

🔊

📎

⋮

✕

○ [HU2] Baja sala

+ Añadir

🕒 Fechas

📋 Checklist

👤 Miembros

📎 Adjunto

Etiquetas

Maven y Java

Git/GitHub

MySQL

+

☰ Descripción

Editar

Como administrador,

Quiero dar de baja una sala de reuniones existente,

Para mantener actualizada la lista de salas disponibles.

Estimación: 0,3 horas

☑ Criterios de aceptación

Eliminar

0%

☐ La sala debe existir para ser eliminada

☐ No se eliminará una sala con reservas activas

Añade un elemento

☑ Checklist

Eliminar

0%

☐ Añadir método boolean deleteRoom(int roomId) en RoomDAO

☐ Añadir método boolean deleteRoom(int roomId) en MeetingRoomManager

Añade un elemento

Comentarios y Actividad

Mostrar detalles

Escribe un comentario...

Ej

Eguren Guijarro, Joel ha añadido esta tarjeta a Product Backlog

24 jun 2025, 23:29

2.3. HU3 – Modificación sala

In Progress

[HU3] Modificación sala

+ Añadir

🕒 Fechas

📋 Checklist

👤 Miembros

📎 Adjunto

Etiquetas

Maven y Java

Git/GitHub

MySQL

+

☰ Descripción

Editar

Como administrador,

Quiero modificar una sala de reuniones existente,

Para mantener actualizada las salas con la información correcta.

Estimación: 0,5 horas

☒ Criterios de aceptación

Eliminar

0%

☐ Se debe verificar la existencia de la sala

☐ Se debe permitir modificar cualquiera de los campos de la sala excepto el room_id

☐ Los campos a modificar tendrán que ser válidos

Añade un elemento

☒ Checklist

Eliminar

0%

☐ Crear clase POJO Room

☐ Crear clase RoomDAO

☐ Añadir método boolean updateRoom(Room room) en RoomDAO

☐ Añadir método modifyRoom(Room room) en MeetingRoomManager

☐ Validar campos antes de la modificación

Añade un elemento

Comentarios y Actividad

Mostrar detalles

Escribe un comentario...

Eguren Guijarro, Joel

 ha añadido esta tarjeta a Product Backlog
24 jun 2025, 23:34

2.4. HU4 – Listar salas

Product Backlog

[HU4] Listar salas

+ Añadir

🕒 Fechas

📋 Checklist

👤 Miembros

📎 Adjunto

Etiquetas

Maven y Java

Git/GitHub

MySQL

+

☰ Descripción

Editar

Como administrador,

Quiero ver el listado completo de las salas,

Para consultar sus características

Estimación: 0,3 horas

☒ Criterios de aceptación

Eliminar

0%

☐ Se deben de mostrar todas las salas con sus atributos

☐ Si no hay salas se debe mostrar un mensaje informativo

☐ Se debe mostrar por orden de inserción

Añade un elemento

☒ Checklist

Eliminar

0%

☐ Crear método `LinkedHashSet<Room> getAllRooms()` en `RoomDAO`

☐ Añadir método `LinkedHashSet<Room> getAllRooms()` en `MeetingRoomManager`

Añade un elemento

Comentarios y Actividad

Mostrar detalles

Escribe un comentario...

EJ

Eguren Guijarro, Joel

ha añadido esta tarjeta a Product Backlog

24 jun 2025, 23:41

2.5. HU5 – Alta empleado

Product Backlog

[HU5] Alta empleado

+ Añadir

🕒 Fechas

📋 Checklist

👤 Miembros

📎 Adjunto

Etiquetas

Maven y Java

Git/GitHub

MySQL

+

☰ Descripción

Editar

Como administrador,

Quiero dar de alta un nuevo empleado con sus atributos,

Para que pueda realizar reservas de salas de reuniones.

Estimación: 0,3 horas

☒ Criterios de aceptación

Eliminar

0%

☐ Se debe permitir registrar un nuevo empleado con DNI, nombre, email y departamento.

☐ Los campos deben ser válidos

☐ El empleado no debe de existir en la base de datos

Añade un elemento

☒ Checklist

Eliminar

0%

☐ Crear método addEmployee(Employee employee) en EmployeeDAO

☐ Añadir método boolean addEmployee() en MeetingRoomManager

Añade un elemento

Comentarios y Actividad

Mostrar detalles

Escribe un comentario...

Ej

Eguren Guijarro, Joel ha añadido esta tarjeta a Product Backlog

25 jun 2025, 0:12

2.6. HU6 – Baja empleado

Product Backlog

[HU6] Baja empleado

+ Añadir

🕒 Fechas

☑ Checklist

👤 Miembros

📎 Adjunto

Etiquetas

Maven y Java

Git/GitHub

MySQL

+

☰ Descripción

Editar

Como administrador,

Quiero dar de baja a un empleado existente,

Para que no pueda realizar reservas de salas de reuniones.

Estimación: 0,2 horas

☑ Criterios de aceptación

Eliminar

0%

☐ El sistema debe permitir eliminar un empleado existente por su DNI

☐ Si el empleado no existe, debe mostrarse un mensaje claro.

☐ No se debe permitir eliminar si el empleado tiene reservas asociadas

Añade un elemento

☑ Checklist

Eliminar

0%

☐ Crear método boolean deleteEmployeeById(String dni) en EmployeeDAO

☐ Añadir método boolean deleteEmployee(String dni) en MeetingRoomManager.

Añade un elemento

Comentarios y Actividad

Mostrar detalles

Escribe un comentario...

Eguren Guijarro, Joel

 ha añadido esta tarjeta a Product Backlog
25 jun 2025, 0:20

2.7. HU7 – Modificación empleado

In Progress

[HU7] Modificación empleado

+ Añadir

🕒 Fechas

📋 Checklist

👤 Miembros

📎 Adjunto

Etiquetas

Maven y Java

Git/GitHub

MySQL

+

☰ Descripción

Editar

Como administrador,

Quiero poder modificar los atributos del empleado excepto el DNI

Para mantener la información actualizada en caso de cambios

Estimación: 0,4 horas

☒ Criterios de aceptación

Eliminar

0%

☐ Se debe permitir modificar uno o varios campos de un empleado

☐ Los atributos a modificar deben ser válidos

Añade un elemento

☒ Checklist

Eliminar

0%

☐ Crear clase POJO Employee

☐ Crear clase EmployeeDAO

☐ Crear método boolean updateEmployee(Employee employee) en EmployeeDAO

☐ Añadir método modifyEmployee() en MeetingRoomManager

Añade un elemento

Comentarios y Actividad

Mostrar detalles

Escribe un comentario...

Eguren Guijarro, Joel

 ha añadido esta tarjeta a Product Backlog
25 jun 2025, 0:25

2.8. HU8 – Listar empleados

Product Backlog

[HU8] Listar empleados

+ Añadir

🕒 Fechas

📧 Checklist

👤 Miembros

📎 Adjunto

Etiquetas

Testing JUnit

Maven y Java

Git/GitHub

MySQL

+

☰ Descripción

Editar

Como

usuario del sistema,

Quiero

ver un listado de todos los empleados registrados,

Para

consultar sus atributos.

Estimación:

0,3 horas

☑ Criterios de aceptación

Eliminar

0%

☐ El sistema debe mostrar todos los empleados con sus atributos

☐ Si no hay empleados, debe mostrar un mensaje informativo claro.

☐ El listado debe estar ordenado por departamento y nombre

Añade un elemento

☑ Checklist

Eliminar

0%

☐ Crear método `TreeSet<Employee> getAllEmployees()` en `EmployeeDAO`

☐ Añadir método `TreeSet<Employee> getAllEmployees()` en `MeetingRoomManager`

☐ Añadir test

Añade un elemento

Comentarios y Actividad

Mostrar detalles

Escribe un comentario...

Ej

Eguren Guijarro, Joel

 ha añadido esta tarjeta a Product Backlog
25 jun 2025, 0:29

2.9. HU9 – Reservar una sala

In Progress

[HU9] Reservar una sala

+ Añadir

📅 Fechas

📋 Checklist

👤 Miembros

📎 Adjunto

Etiquetas

Testing JUnit

Maven y Java

Git/GitHub

MySQL

+

Descripción

Editar

Como empleado,
Quiero reservar una sala para una fecha y franja horaria concreta,
Para asegurarme de que la sala estará disponible en ese momento para mis reuniones.

Estimación: 0,6 horas

👍 Criterios de aceptación

Eliminar

0%

☐ El sistema debe permitir seleccionar empleado, sala, fecha, hora de inicio y hora de fin.

☐ No se deben permitir reservas dobles, evitar solapamiento

☐ Debe mostrarse un mensaje claro en caso de conflicto o error

Añade un elemento

👍 Checklist

Eliminar

0%

☐ Crear clase POJO Reservation y package model

☐ Crear clase ReservationDAO en package model

☐ Crear clase MeetingRoomManager en package controller

☐ Crear el método String addReservation(Reservation reservation) en ReservationDAO

☐ Añadir el método String addReservation(Reservation reservation) en MeetingRoomManager

☐ Añadir test unitario para reserva válida y para reserva con conflicto

Comentarios y Actividad

Mostrar detalles

Escribe un comentario...

Ej

Eguren Guijarro, Joel ha añadido esta tarjeta a Product Backlog

hace 12 horas

1.10. HU10 – Cancelar reserva

Product Backlog

[HU10] Cancelar reserva

+ Añadir

📅 Fechas

📋 Checklist

👤 Miembros

📎 Adjunto

Etiquetas

Testing JUnit

Maven y Java

Git/GitHub

MySQL

+

Descripción

Editar

Como empleado

Quiero cancelar una reserva existente

Para liberar la sala si ya no la necesito, siempre que falte más de 1 día para la reserva.

Estimación: 0,5 horas

👑 Criterios de aceptación

Eliminar

0%

☐ El sistema debe permitir cancelar una reserva por su identificador

☐ No se podrá cancelar una reserva si quedan menos de 24 horas para el inicio

☐ Debe mostrarse un mensaje claro si la cancelación no es posible

Añade un elemento

👑 Checklist

Eliminar

0%

☐ Crear método cancelReservation(String reservationId) en ReservationDAO

☐ Añadir método cancelReservation(String reservationId) en MeetingRoomManager.

☐ Añadir test unitario para cancelar reserva válida y para reserva con conflicto

Añade un elemento

Comentarios y Actividad

Mostrar detalles

Escribe un comentario...

Eguren Guijarro, Joel

ha añadido esta tarjeta a Product Backlog

hace 10 horas

Práctica Final de Curso

13

3. Sprint 1

3.1. Objetivos y planificación

El Sprint 1 comprenderá el día dos y tres. Para este Sprint se han elegido las tarjetas e historias de usuario con mayor estimación, para dar un alto valor al proyecto teniendo en cuenta las dependencias técnicas. *(La estimación está en la descripción de cada tarjeta).*

Ordenados, los objetivos de este Sprint son:

- Diseñar modelo ER
- Crear script SQL y base de datos
- Inicializar repositorio Git/GitHub
- Configurar entorno para pruebas unitarias
- Crear y configurar proyecto Maven
- Configurar conexión a base de datos
- [HU9 - Reservar una sala](#)
- [HU10 - Cancelar reserva](#)
- [HU3 - Modificación sala](#)
- [HU7 - Modificación empleado](#)
- [HU4 - Listar salas](#)

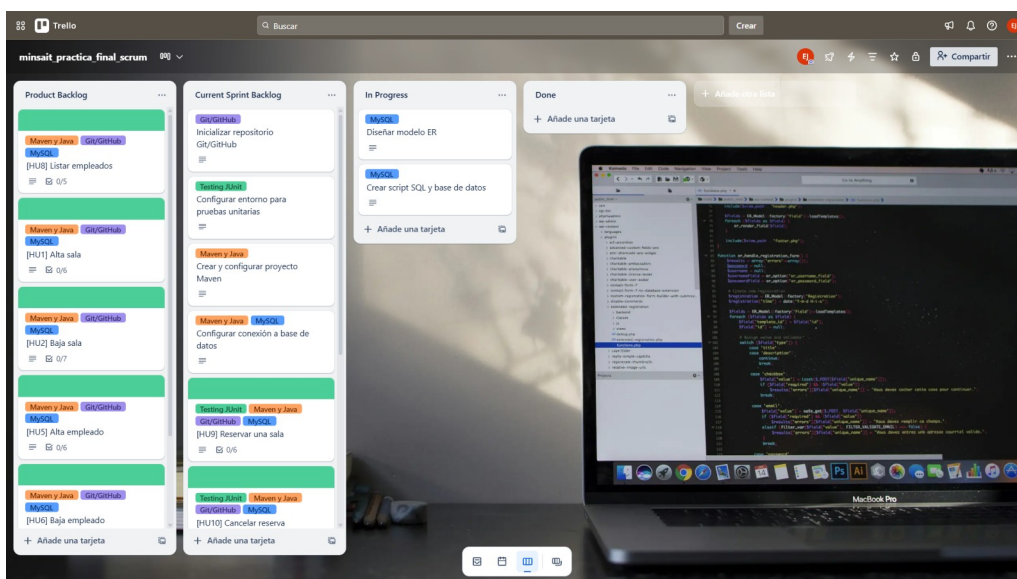


Figura 1. Situación actual en Trello

3.2. Reflexión

En este Sprint se han logrado realizar todas las tarjetas planificadas, excepto algún objetivo de la tarjeta *HU9 – Reservar una sala*, porque se ha necesitado crear tres tareas técnicas para verificar que una reserva es válida:

- En la clase **EmployeeDAO** crear el método *Optional<Employee> getEmployeeByDNI(String dni)*: para verificar que el empleado existe.
- En la clase **RoomDAO** crear el método *Optional<Room> getRoomById(int roomId)*: para verificar que la sala existe.
- En la clase **ReservationDAO** crear el método *boolean canReservate(Reservation reservation)*: para verificar que la fecha de la reserva es correcta y no solape con otra.

Se ha controlado correctamente el límite *Work In Progress* (WIP), lo cual ha facilitado centrarse en pocas tareas a la vez y mejorar la eficiencia en el desarrollo. Además, se pudo absorber y completar satisfactoriamente la [*HU8 – Listar Empleados*](#).

Como mejora, se podrían optimizar la estimación de las tarjetas para evitar tiempos muertos entre sprints y controlar mejor posibles tareas técnicas que podrían surgir.

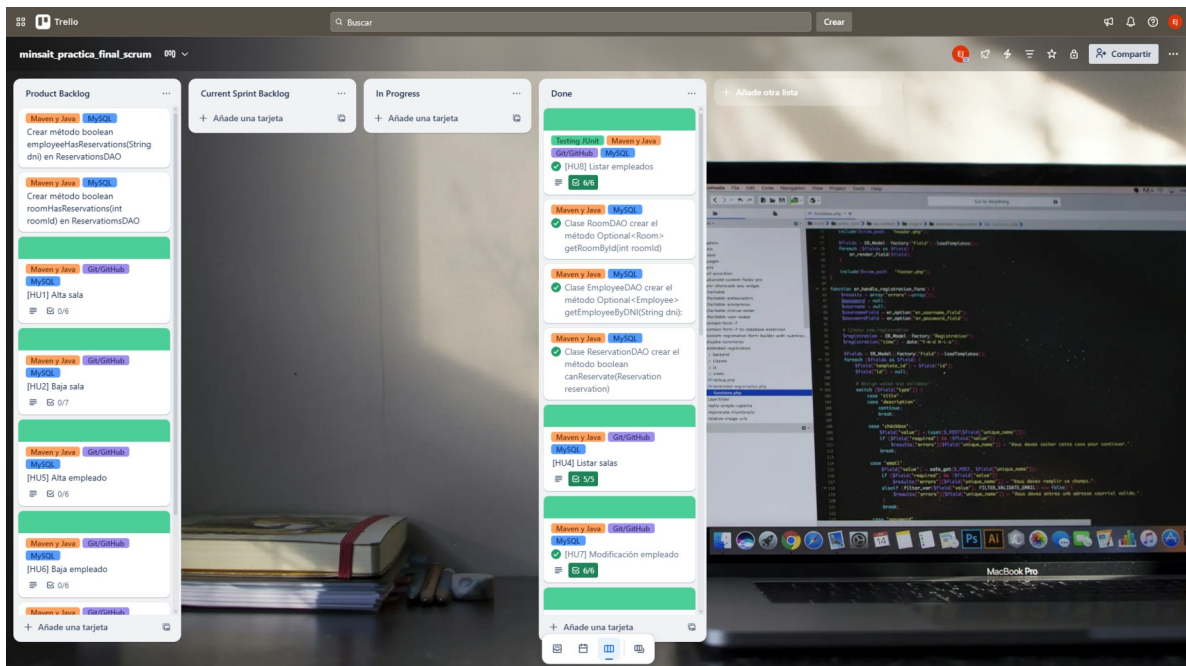


Figura 2. Situación final Sprint en Trello

4. Sprint 2

4.1. Objetivos y planificación

El Sprint 2 se ha planificado para el día cuatro del desarrollo. Para este Sprint se han elegido las tarjetas e historias restantes ordenadas de mayor a menor estimación y teniendo en cuenta las dependencias técnicas. *(La estimación está en la descripción de cada tarjeta).*

Ordenados, los objetivos de este Sprint son:

- Crear método *boolean employeeHasReservations(String dni)* en ReservationsDAO
- Crear método *boolean roomHasReservations(int roomId)* en ReservationsDAO
- [\[HU1\] - Alta sala](#)
- [\[HU2\] - Baja sala](#)
- [\[HU5\] - Alta empleado](#)
- [\[HU6\] - Baja empleado](#)
- Añadir tests faltantes de verificación
- Crear programa principal Main
- Crear README.md

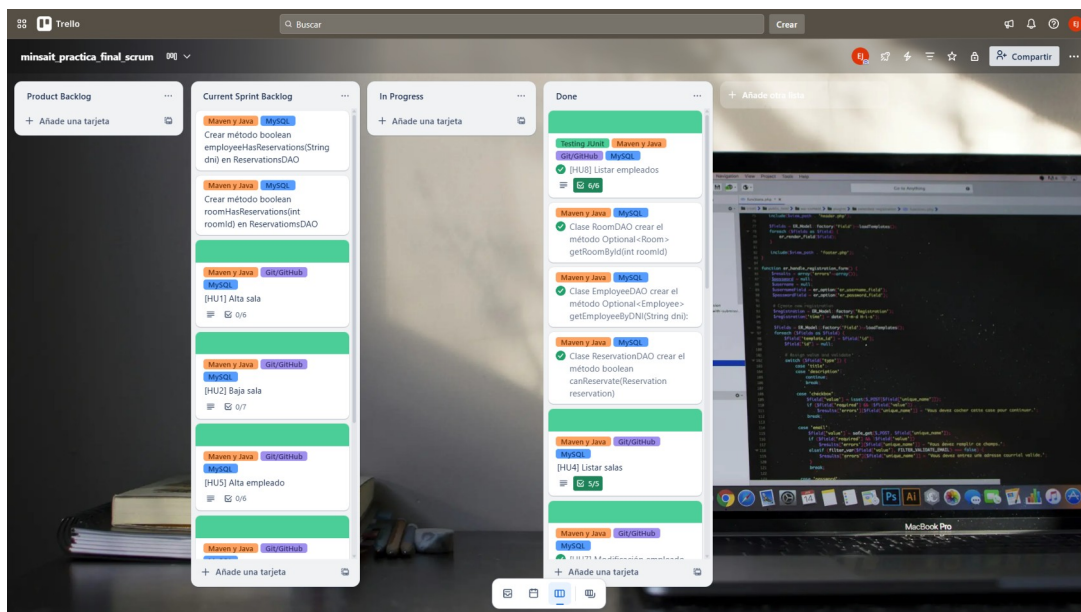


Figura 3. Situación actual en Trello

4.2. Reflexión

En este Sprint se han logrado realizar todas las tarjetas planificadas satisfactoriamente, pero se ha necesitado crear dos tareas técnicas para verificar si una sala es válida y un empleado es válido:

- En la clase **MeetingRoomManager** crear el método *boolean isValidRoom(Room room)*: para verificar que la sala apta para ser introducida en la BD.
- En la clase **MeetingRoomManager** crear el método *boolean isValidEmployee(Employee employee)*: para verificar que el empleado es apto para ser introducido en la BD.

También se ha controlado correctamente el límite *Work In Progress* (WIP), lo cual ha facilitado centrarse en pocas tareas a la vez y mejorar la eficiencia en el desarrollo.

Como mejora, se podría haber dividido la clase **MeetingRoomMain** en varias clases auxiliares, con el objetivo de repartir mejor las responsabilidades y favorecer una mayor modularidad y mantenibilidad del código.

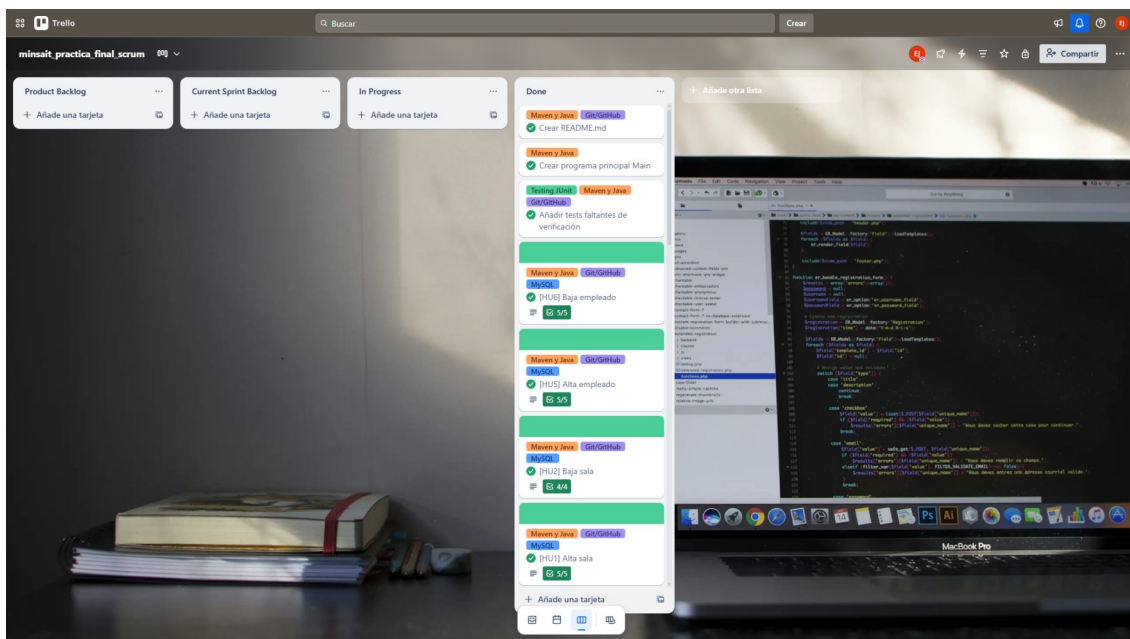


Figura 4. Situación final Sprint en Trello

5. Retrospectiva final

Los objetivos de este proyecto han sido completados como se esperaba. Se han logrado alcanzar todos los objetivos de los sprints, incluyendo las historias de usuario, las tareas técnicas que se planificaron y alguna tarea que ha surgido durante el desarrollo.

Durante el trascurso de este proyecto he reforzado habilidades relacionadas con la importancia de una buena planificación y organización, así como de mantener un flujo de trabajo constante. A nivel técnico he mejorado mi conocimiento en las tecnologías trabajadas como Java, MySQL, Maven y JUnit. He procurado aplicar principios de *clean code*, para mejorar la legibilidad y mantenibilidad del código y he documentado cada commit en Git de forma clara y coherente.

Como posibles dificultades durante el desarrollo, he enfrentado la gestión y creación de todas las tarjetas y historias de usuario, decidiendo su estimación y prioridad. Además, he diseñado toda la estructura de código aplicando MVC y cumpliendo con los requisitos del enunciado. Otra dificultad podría ser el corto plazo de entrega, pero finalmente se ha podido cumplir.

En resumen, este proyecto ha sido una experiencia muy enriquecedora que me ha permitido conocer y aplicar de manera práctica la metodología Scrum y mejorar en las tecnologías usadas.