FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Informe Final - 1C 2025

Grupo 8

Ingeniería de Software I - Cátedra Montaldo

Nombre	Padrón
Abril Giordano Hoyo	108760
Dalmiro Vilaplana	109875
Juan Martín de la Cruz	109588
Lucas Ruiz Abelleira	109674
Mateo Alvarez	108666
Tomas Caporaletti	108598
Emanuel Nicolas Sagman	103439
Rami Aleid	110465

Índice

1. Introducción **3**
1. Modelo de Vistas **4**
2.1. Vista Lógica 4
2.2. Vista de Componentes 4
2.3. Vista de Procesos 5
2.4. Vista de Despliegue 5
2.5. Vista de Escenarios 6
2. Atributos de Calidad **8**
2.1. Seguridad 8
2.2. Rendimiento 8
2.3. Usabilidad 8
3. Desarrollo del Proyecto **8**
3.1. Gitlab 9
3.2. Jira 9
4.1. Sprint 1 10
5.2. Sprint 2 11
5.3. Sprint 3 11
5.4. Sprint 4 12
5.5. Historias Faltantes 13
5. Velocidad del Equipo **14**
6. Decisiones Importantes **15**
6.1. Documentación de la API 15
6.2. Organización de Paquetes y Clases 15
6.3. Validación de Datos 16
6.3. Persistencia de datos 16
7. API **17**
8. Conclusiones **17**

1. Introducción

El presente informe documenta el desarrollo de una aplicación web orientada a la gestión y reserva de canchas de fútbol 5, así como a la organización de partidos abiertos y cerrados, equipos y torneos. El sistema busca resolver una problemática común en el ámbito deportivo amateur: la dificultad para coordinar partidos, reunir jugadores, encontrar canchas disponibles y gestionar inscripciones de forma eficiente y centralizada.

La solución está pensada para cubrir las necesidades de dos tipos principales de usuarios:

- **Jugadores**: pueden registrarse, visualizar canchas disponibles en su zona, inscribirse a partidos abiertos, crear equipos y participar en torneos.
  Además, según la funcionalidad que utilicen, pueden desempeñar dos roles:
  - Como **jugadores**, se inscriben y participan en partidos.
  - Como **organizadores**, pueden crear partidos (abiertos o cerrados), reservar canchas y gestionar equipos o torneos.
- **Administradores de canchas**: pueden publicar canchas con sus características, gestionar la disponibilidad horaria, monitorear reservas y controlar el uso de sus instalaciones.

El sistema permite crear partidos abiertos —donde jugadores individuales se suman hasta alcanzar un cupo— o partidos cerrados entre equipos preformados. Los usuarios pueden visualizar los partidos disponibles, inscribirse o darse de baja, mientras que los organizadores pueden formar equipos automáticamente o de forma manual una vez confirmado el evento. Además, se pueden organizar torneos, inscribir equipos y llevar el seguimiento del fixture, resultados y estadísticas en tiempo real.

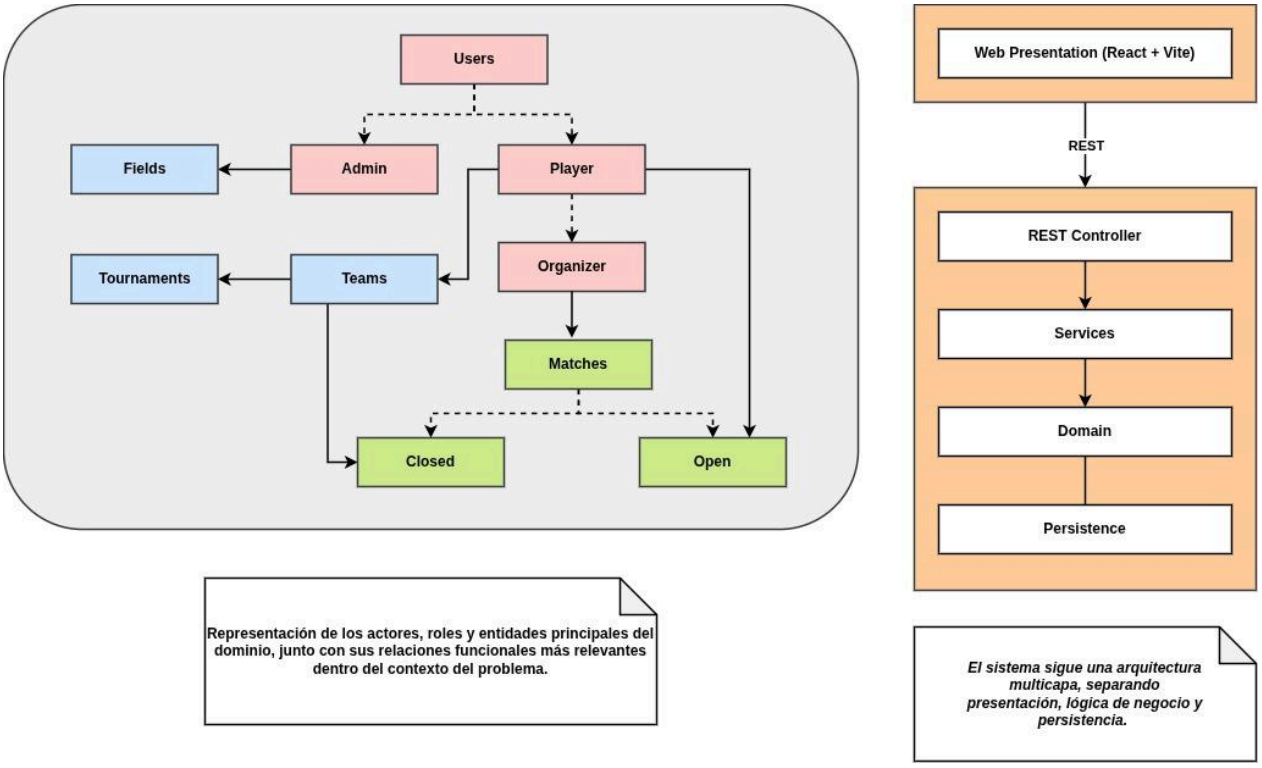
Para garantizar una experiencia fluida y segura, se contemplaron funcionalidades como validación de email al registrarse, autenticación mediante JWT, recuperación de contraseña, historial de partidos, sistema de comentarios y valoración de canchas, y filtros avanzados de búsqueda.

En términos técnicos, se desarrolló una API RESTful, con persistencia en base de datos y cobertura de pruebas unitarias, y una interfaz de usuario sencilla que permite interactuar con las principales funcionalidades.

1. Modelo de Vistas

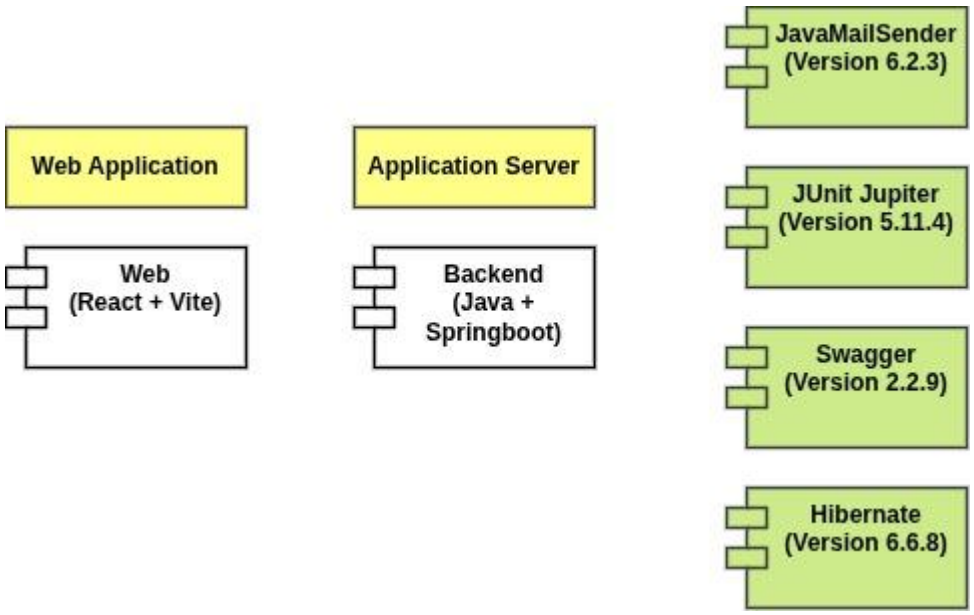
2.1. Vista Lógica

Representa la estructura funcional del sistema, organizada en capas (presentación, lógica, dominio y persistencia). Incluye las entidades principales como usuarios, partidos, canchas, equipos y torneos, y sus relaciones clave. Permite comprender la organización conceptual sin detallar clases específicas.



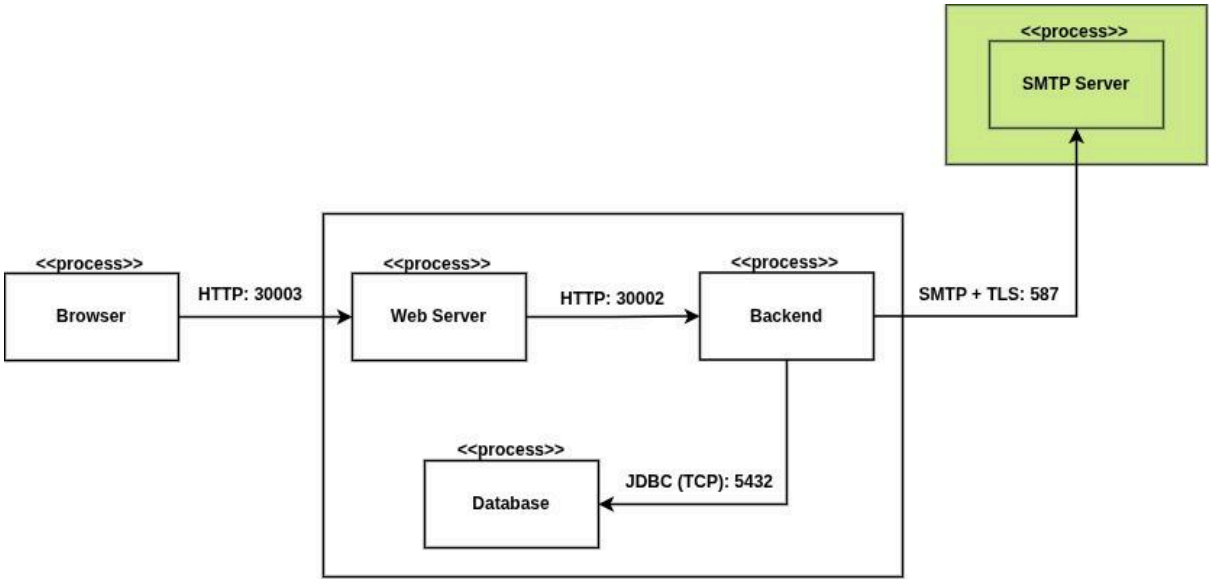
2.2. Vista de Componentes

Expone la estructura del software desde el punto de vista del desarrollo. Incluye los módulos internos y las librerías externas utilizadas (Hibernate, Swagger, JavaMailSender, JUnit). Permite visualizar dependencias y organización modular del código.



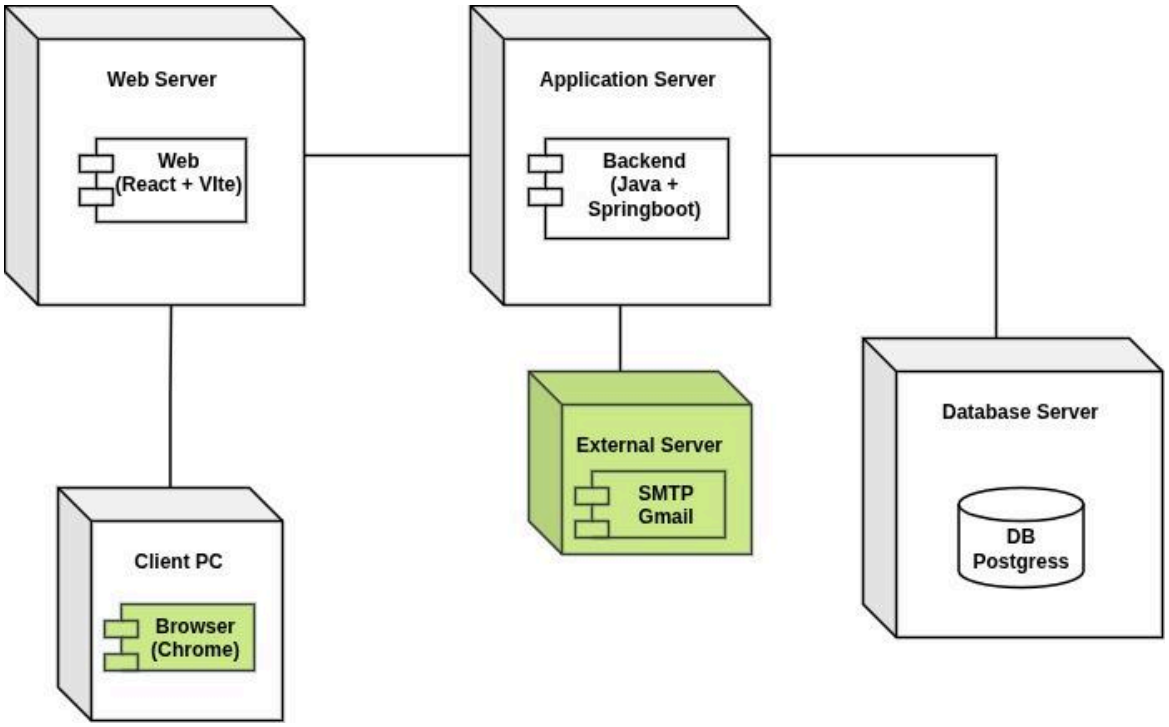
2.3. Vista de Procesos

Describe la interacción entre los procesos activos del sistema y los protocolos utilizados (HTTP, JDBC, SMTP). Muestra cómo se comunican el frontend, backend, base de datos y servicios externos. Refleja la dinámica del sistema y su comportamiento en tiempo de ejecución.



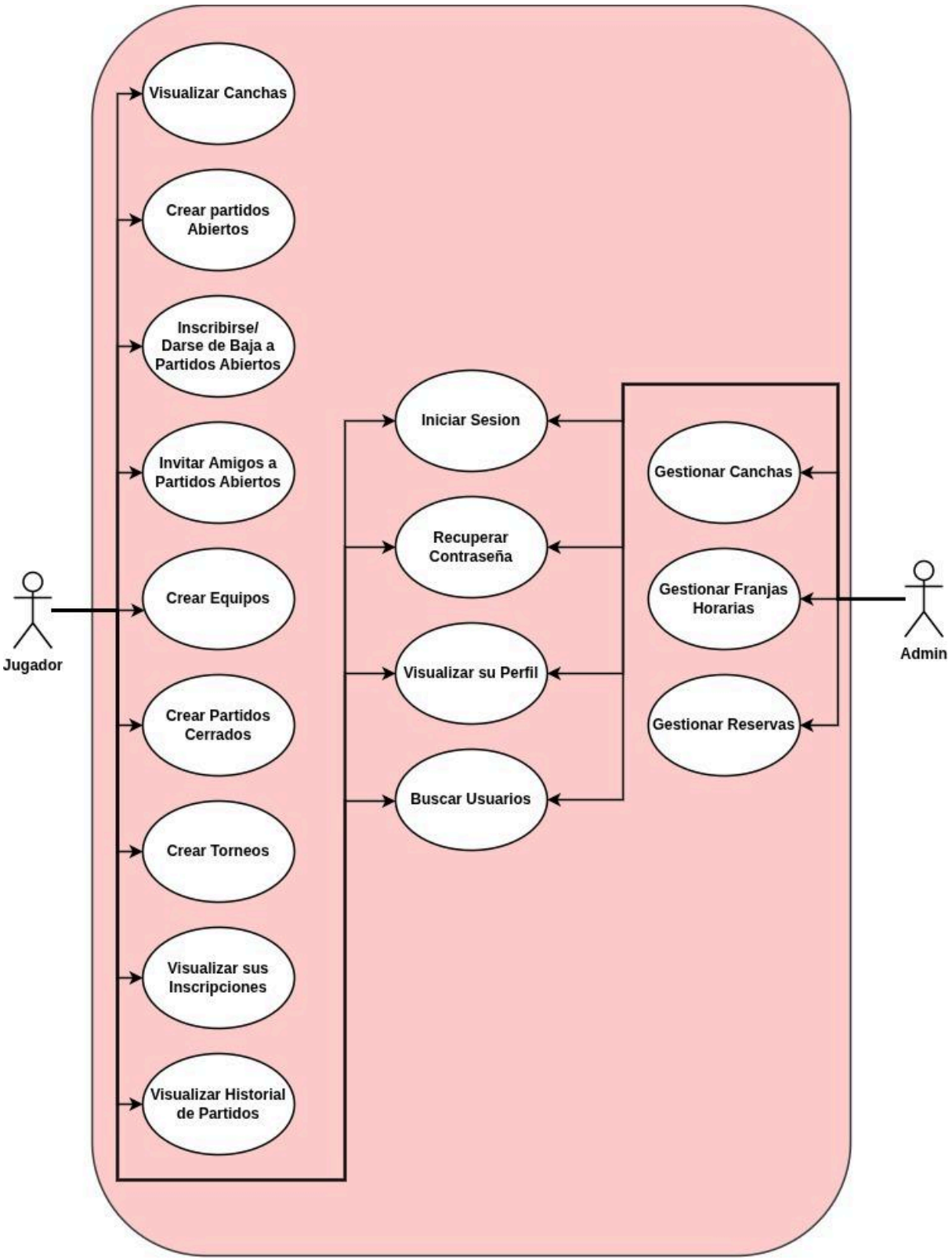
2.4. Vista de Despliegue

Muestra cómo se distribuyen los componentes sobre la infraestructura real, utilizando Docker. Se representan los servidores web, backend, base de datos y servicios externos. Refleja la configuración de red y despliegue del sistema.



2.5. Vista de Escenarios

El diagrama de escenarios representa los principales casos de uso del sistema desde la perspectiva de los diferentes actores. Se muestran las funcionalidades disponibles tanto para los jugadores (partidos, equipos, torneos) como para los administradores de canchas (gestión de disponibilidad y reservas). Este escenario valida la arquitectura definida, demostrando cómo las funcionalidades del sistema son alcanzadas a través de las interacciones entre los usuarios y los módulos desarrollados.


Visualizar Canchas
Crear partidos Abiertos
Inscribirse/ Darse de Baja a Partidos Abiertos
Invitar Amigos a Partidos Abiertos
Crear Equipos
Crear Partidos Cerrados
Crear Torneos
Visualizar sus Inscripciones
Visualizar Historial de Partidos
Iniciar Sesion
Recuperar Contraseña
Visualizar su Perfil
Buscar Usuarios
Gestionar Canchas
Gestionar Franjas Horarias
Gestionar Reservas
Jugador
Admin

2. Atributos de Calidad

Durante el diseño e implementación del sistema se priorizaron tres atributos de calidad fundamentales: **seguridad**, **rendimiento** y **usabilidad**. Esta elección se basó tanto en los requerimientos funcionales y no funcionales de la consigna como en las decisiones tomadas a lo largo del desarrollo.

2.1. Seguridad

La seguridad en nuestro sistema fue un aspecto prioritario debido a la gestión de datos sensibles y roles con permisos diferenciados. Se implementó autenticación con JWT para proteger las sesiones, validación de email para confirmar identidades y un mecanismo seguro de recuperación de contraseña. Además, el acceso a funcionalidades específicas está controlado según el tipo de usuario.

2.2. Rendimiento

Se buscó garantizar una experiencia fluida, especialmente en operaciones críticas como el registro, la creación de partidos o la inscripción. Se diseñó el sistema para que tareas no críticas, como el envío de emails, se ejecuten de forma asíncrona, evitando demoras innecesarias en la interfaz del usuario.

2.3. Usabilidad

El frontend fue diseñado con un enfoque claro en la simplicidad y accesibilidad. Se implementaron vistas limpias, formularios intuitivos y flujos guiados que permiten a los usuarios registrarse, crear partidos, inscribirse o gestionar canchas sin necesidad de capacitación previa, mejorando así la adopción y experiencia de uso.

3. Desarrollo del Proyecto

El desarrollo del sistema se organizó de forma colaborativa utilizando herramientas de gestión modernas como **GitLab** para el control de versiones y gestión del código fuente, y **Jira** para la planificación y seguimiento de tareas. Esta combinación permitió llevar adelante el trabajo de manera ordenada, ágil y trazable, respetando la metodología de desarrollo basada en sprints.

3.1. Gitlab

El repositorio del proyecto se alojó en **GitLab**, donde se definió una rama principal *dev* como base de desarrollo continuo. A partir de esa rama, se generaban ramas individuales para cada funcionalidad específica, nombradas según la subtarea correspondiente del Jira. Esta organización facilitó la trazabilidad entre código y planificación. En algunos casos se trabajó de forma colaborativa, con más de un integrante del equipo contribuyendo en la misma funcionalidad.

Finalizado el trabajo en una rama, se generaba un *"merge request" como* requisito previo para fusionarla en la rama *dev*. Al culminar el desarrollo el proyecto tras el cuarto sprint, se realizó un merge final de *dev* hacia *master*, consolidando la versión estable o "productiva" del sistema.

3.2. Jira

La planificación y gestión del trabajo se realizó utilizando **Jira**, siguiendo una estructura jerárquica organizada en tres niveles:

- **Épicas**: grandes bloques funcionales del sistema (gestión de usuarios, gestión de canchas, gestión de partidos y reservas, gestión de perfil y equipos y gestión de torneos).
- **Historias de Usuario**: funcionalidades concretas desde el punto de vista del usuario, derivadas de las épicas.
- **Subtareas**: acciones específicas necesarias para completar cada historia (como desarrollar un formulario, implementar un endpoint o diseñar una vista o componente específico).

Cada subtarea fue priorizada, estimada en *story points*, y se asignó al menos una persona responsable por su implementación. Esta estructura permitió dividir el trabajo de forma clara, mantener visibilidad del progreso del sprint y facilitar la coordinación del equipo.

4. Sprints

El proyecto se desarrolló a lo largo de cuatro sprints semanales, siguiendo una metodología ágil. Cada sprint tuvo objetivos definidos, permitiendo avanzar de forma incremental en la implementación de funcionalidades. Al finalizar cada uno, se realizaba una revisión del trabajo completado, se ajustaban prioridades y se planificaba el siguiente ciclo, asegurando una entrega continua y evolutiva del sistema.

4.1. Sprint 1

Durante el primer sprint el objetivo principal fue inicializar el sistema y avanzar con algunas funcionalidades básicas iniciales, tanto del módulo de usuarios como de la gestión de canchas. En ese marco, se planificaron las siguientes tareas:

- **FUT-102**: Inicializar la aplicación y configurar la base de datos.
- **FUT-7**: Registro de usuario.
- **FUT-13**: Inicio de sesión.
- **FUT-42**: Creación de canchas.
- **FUT-44**: Eliminación de canchas.

De todas ellas, sólo se completó en su totalidad la tarea **FUT-102** (Inicialización del proyecto y configuración de la base de datos). Las restantes funcionalidades se avanzaron en gran medida, pero ninguna de ellas fue finalizada dentro del sprint y, en consecuencia, se incluyeron en los siguientes sprints. En este contexto, de la retrospectiva podemos remarcar los siguientes puntos.

START	STOP	CONTINUE
Enfocar el esfuerzo del equipo en completar tareas específicas antes de comenzar nuevas, evitando acumular historias parcialmente desarrolladas.	Subestimar el tiempo necesario para tareas aparentemente simples, como formularios o validaciones, sin contemplar sus detalles.	Coordinar antes del inicio de cada sprint quién se encargará de cada tarea, para organizarnos mejor y avanzar de forma más ordenada.

5.2. Sprint 2

En el segundo sprint se buscó continuar con el desarrollo de las funcionalidades básicas asociadas a usuarios y canchas, y comenzar con el módulo de partidos abiertos, de forma tal que se decidió avanzar con la siguientes tareas:

- **FUT-7**: Registro de Usuario (*continuación*)
- **FUT-13**: Inicio de Sesión (*continuación*)
- **FUT-42**: Creación de Canchas (*continuación*)
- **FUT-26**: Visualización de Canchas Disponibles
- **FUT-30**: Creación de Partidos Abiertos
- **FUT-31**: Visualización de Partidos Abiertos

Durante este sprint se completaron correctamente las funcionalidades vinculadas al inicio de sesión, al registro de usuarios, a la visualización de canchas y a la creación de canchas. Por otro lado, las tareas relacionadas con la visualización y creación de partidos abiertos (**FUT-30** y **FUT-31**) fueron parcialmente desarrolladas, pero no lograron ser finalizadas en su totalidad dentro de este sprint. De la retrospectiva, podemos destacar:

START	STOP	CONTINUE
Mejorar la comunicación durante el sprint para detectar a tiempo obstáculos y asegurar la finalización de las historias priorizadas.	Crear tareas que dependan simultáneamente de frontend y backend sin una planificación clara, ya que eso genera bloqueos y retrasa el desarrollo.	Avanzar con funcionalidades clave del MVP que aporten valor real al sistema y permitan consolidar los módulos principales.

5.3. Sprint 3

En el tercer sprint se abordaron funcionalidades más avanzadas del sistema, incluyendo mejoras en la experiencia del usuario, gestión de partidos, interacción entre usuarios y administración de canchas. Se planificaron las siguientes tareas:

- **FUT-30**: Creación de Partidos Abiertos (*continuación*)
- **FUT-44**: Eliminación de Canchas (*continuación*)
- **FUT-17**: Recuperación de Contraseña
- **FUT-18**: Invitación a Usuarios No Registrados

- **FUT-32**: Baja de Partidos
- **FUT-33**: Organización de Jugadores
- **FUT-41**: Creación de Equipos
- **FUT-43**: Edición de Canchas
- **FUT-45**: Visualización del Historial de Perfil
- **FUT-113**: Visualización de Detalles de Canchas
- **FUT-120**: Creación de Partidos Cerrados

Durante este sprint se completaron correctamente la mayoría de las funcionalidades planificadas. Sin embargo, tres tareas quedaron incompletas: **FUT-45** (Visualización del historial de perfil), **FUT-43** (Edición de canchas) y **FUT-32** (Baja de partidos), que fueron planificadas para el sprint siguiente. De este sprint, decidimos remarcar los siguientes aspectos:

START	STOP	CONTINUE
Realizar revisiones de avance a mitad del sprint para detectar tareas demoradas y redistribuir el esfuerzo de forma más efectiva.	Dedicar tiempo excesivo a detalles no críticos de las tareas más importantes, en lugar de enfocarse en cumplir claramente lo definido por la historia de usuario.	Comunicar al equipo las decisiones y soluciones implementadas en funcionalidades complejas, para mantener una visión compartida y coherente del sistema.

5.4. Sprint 4

En el cuarto y último sprint se abordaron principalmente las funcionalidades relacionadas con torneos, gestión avanzada de canchas y ajustes generales del sistema. También se incluyeron tareas pendientes de sprints anteriores para consolidar una versión funcional estable. Las tareas planificadas fueron:

- **FUT-32**: Baja de Partidos (*continuación*)
- **FUT-43**: Edición de Canchas (*continuación*)
- **FUT-45**: Visualización del Historial de Perfil (*continuación*)
- **FUT-46**: Visualización de Torneos
- **FUT-47**: Creación de Torneos
- **FUT-48**: Gestionar Horarios de Canchas
- **FUT-49**: Visualizar y Editar mis Torneos

- **FUT-50**: Eliminación de Torneos
- **FUT-51**: Inscripción de Equipos a Torneos
- **FUT-55**: Administrar Reservas de Canchas
- **FUT-141**, **FUT-142**, **FUT-143**, **FUT-150**, **FUT-156** y **FUT-159**: Correcciones y mejoras generales (homepage por rol, edición de equipos, validación de email, visualización de partidos desde canchas, control de usuarios activos, etc.)

Este sprint permitió completar la mayoría de las tareas relacionadas con la funcionalidad de torneos, así como cerrar las pendientes heredadas del sprint anterior. Además, se aplicaron mejoras generales y correcciones sobre funcionalidades ya implementadas, apuntando a consolidar una versión estable del sistema de cara al cierre del desarrollo.

Finalmente, de la última retro nos parece relevante mencionar:

START	STOP	CONTINUE
Registrar el avance de cada tarea en Jira con mayor frecuencia, para tener visibilidad clara y actualizada del progreso del sprint.	Dejar las validaciones funcionales para el último momento, ya que esto reduce el margen para detectar y corregir errores a tiempo.	Mantener una buena organización en la gestión de merges, resolviendo conflictos de forma colaborativa.

5.5. Historias Faltantes

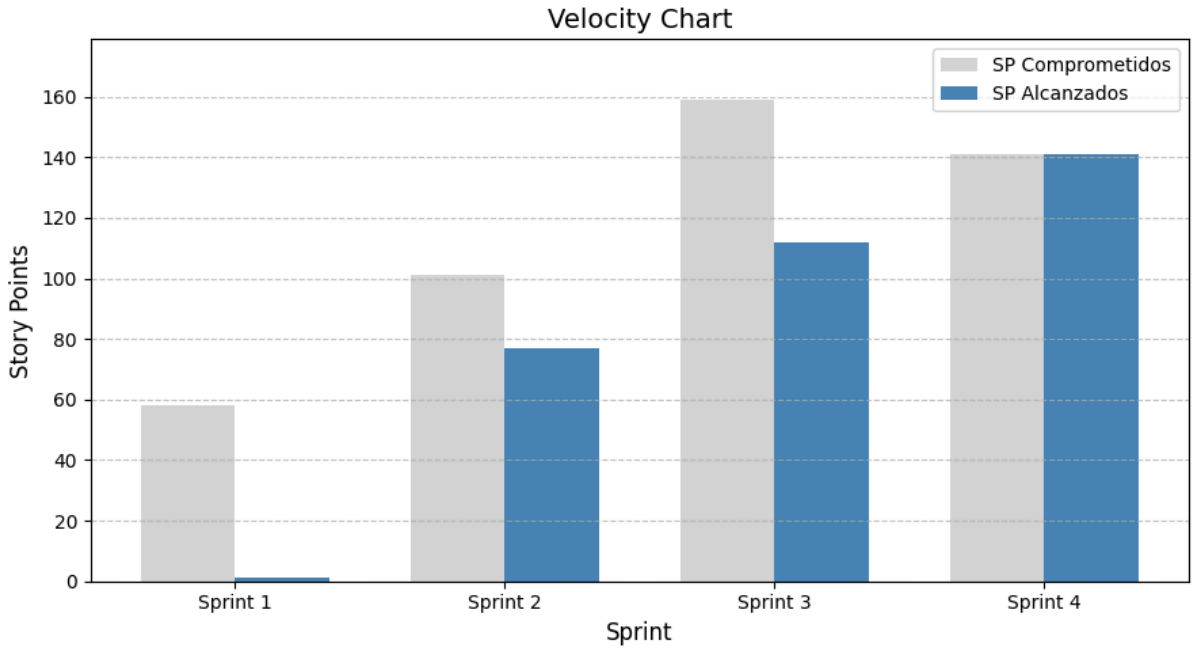
Al cierre del Sprint 4, quedaron dos tareas en el backlog que no llegaron a implementarse:

- **FUT-171**: Gestionar Fixture y Rondas del Torneo
- **FUT-172**: Visualizar Progreso y Estadísticas del Torneo

Ambas funcionalidades corresponden a etapas avanzadas del módulo de torneos, y se priorizaron por debajo de otras tareas esenciales para lograr una versión estable y funcional del sistema. Podrían retomarse en futuras iteraciones.

5. Velocidad del Equipo

La velocidad del equipo se midió en función de los *story points* (SP) completados por Sprint, permitiendo observar la evolución del ritmo de trabajo a lo largo del desarrollo.



Se puede observar claramente que a medida que el equipo se fue adaptando a la dinámica de trabajo, se logró aumentar progresivamente la cantidad de puntos alcanzados por sprint. Esta métrica refleja tanto el grado de familiaridad con el sistema como la mejora en la coordinación y distribución de tareas.

Además, a partir del segundo sprint se incorporaron nuevos integrantes al equipo, lo cual permitió distribuir mejor las responsabilidades y avanzar de forma más eficiente en las funcionalidades planificadas.

El Sprint 3 marca un punto de inflexión: si bien los SP comprometidos se incrementaron notablemente, el equipo logró también aumentar de forma considerable los puntos efectivamente entregados, reduciendo la distancia con el compromiso inicial. Este comportamiento indica una mejor calibración de las expectativas y una mayor eficiencia en la ejecución de las tareas asignadas.

Si analizamos los cuatro sprints y calculamos un promedio general, podemos estimar que la velocidad del equipo fue de aproximadamente 83 *story points* por sprint. Este valor

promedio se encuentra cercano a la carga de trabajo efectiva del segundo sprint, punto a partir del cual el equipo logró sostener una dinámica de entrega más estable.

6. Decisiones Importantes

A lo largo del desarrollo de la aplicación, se tomaron diversas decisiones de diseño y arquitectura que impactaron en la organización del código, la mantenibilidad del sistema y la facilidad de integración entre sus distintos componentes. A continuación se detallan algunas de las más relevantes.

6.1. Documentación de la API

Para permitir una integración eficiente entre el frontend y el backend, se decidió exponer toda la documentación de la API a través de Swagger (OpenAPI). Gracias a esta herramienta, los desarrolladores del frontend pudieron consultar de manera clara y centralizada todos los endpoints disponibles, junto con sus métodos HTTP, parámetros requeridos, tipos de datos esperados y respuestas posibles. Esta decisión mejoró significativamente la comunicación entre equipos, facilitó las pruebas de la API y redujo la posibilidad de errores de integración. Además, permitió disponer de una interfaz interactiva para validar rápidamente el funcionamiento de los servicios expuestos.

6.2. Organización de Paquetes y Clases

En cuanto a la organización de paquetes, se optó modularizar por feature o entidad, lo cual significa que cada módulo funcional del sistema (como usuarios, canchas, partidos, torneos, equipos, etc.) tiene su propio paquete independiente, que contiene todas las clases necesarias para su funcionamiento, tales como controladores, servicios, repositorios, modelos y otros componentes relacionados.

Por ejemplo, la funcionalidad de gestión de usuarios se encuentra completamente contenida en el paquete *user*, mientras que la gestión de canchas reside en el paquete *fields*, y así sucesivamente con *matches*, *teams, tournaments* e *images*. Además, existen paquetes comunes como *common* y *config* para elementos compartidos o de configuración global.

Esta decisión permite una mayor modularidad y cohesión, facilitando el mantenimiento, la escalabilidad del proyecto y el trabajo en equipo, ya que distintos integrantes pueden trabajar en diferentes *features* sin interferir en el código de otros módulos.

Además de reducir el riesgo de dependencias cruzadas innecesarias entre funcionalidades no relacionadas, esta estructura prepara al proyecto para una posible evolución futura hacia una arquitectura de microservicios, dado que cada feature está razonablemente encapsulada en su propio espacio lógico.

6.3. Validación de Datos

Para garantizar la integridad y validez de los datos recibidos por el sistema, se decidió utilizar las anotaciones de validación provistas por "*Jakarta Bean Validation"* (integrada en Spring Boot), tales como `@Valid`, `@NotNull`, `@Size`, entre otras. Esto permite definir reglas de validación directamente en los DTOs o entidades, centralizando la lógica de verificación y evitando la necesidad de implementar validaciones manuales en los controladores. Esta decisión contribuye a reducir errores, mejorar la robustez de la API y simplificar el mantenimiento de las reglas de negocio.

6.3. Persistencia de datos

En cuanto a la capa de persistencia, se utilizó JPA *(Java Persistence API)* junto con *Hibernate* como proveedor *ORM*, dado que esta tecnología ya formaba parte del template inicial provisto para el trabajo práctico. A lo largo del desarrollo, el equipo adoptó esta herramienta, aprendiendo su funcionamiento y aprovechando sus ventajas para mapear entidades Java a tablas relacionales de PostgreSQL sin necesidad de escribir consultas SQL nativas de manera manual.

El uso de JPA permite gestionar transacciones, definir relaciones entre entidades y realizar operaciones CRUD (Create, Read, Update, Delete) de forma simplificada, lo que facilitó el desarrollo de la aplicación. Además, esta tecnología aporta portabilidad, permitiendo que el código pueda adaptarse a otros motores de bases de datos relacionales en caso de ser necesario en el futuro.

7. API

La plataforma expone una API RESTful desarrollada con *Spring Boot*, la cual permite interactuar con todas las funcionalidades del sistema: gestión de usuarios, reservas de canchas, organización de partidos y torneos, entre otras.

Como mencionamos anteriormente, toda la documentación de los endpoints disponibles se encuentra generada automáticamente mediante *Swagger*, lo que facilita su exploración y prueba desde una interfaz gráfica intuitiva. Para acceder a la documentación completa de la API y probar sus endpoints de manera interactiva:

1. Asegúrese de tener el entorno levantado con Docker:
   `docker compose up --build`
2. Ingrese a la siguiente URL desde su navegador:
   `http://localhost:30002/swagger-ui/index.html#`

Desde esta interfaz es posible visualizar todos los endpoints disponibles (GET, POST, PUT, DELETE), consultar los parámetros requeridos y las posibles respuestas que puede devolver cada uno, así como ejecutar pruebas directamente desde el navegador de manera interactiva.

La API fue diseñada siguiendo los principios RESTful, asegurando que las respuestas estén estandarizadas mediante el uso de códigos HTTP adecuados para cada situación. Los endpoints se encuentran organizados de manera lógica según los recursos principales del sistema, tales como usuarios, canchas, partidos y torneos. Además, los mecanismos de autenticación y autorización, en caso de ser necesarios, están gestionados mediante las herramientas provistas por Spring Security, tal como se detalla en la sección de seguridad del proyecto.

8. Conclusiones

El desarrollo del sistema de gestión y reserva de fútbol 5 representó una experiencia integral de aplicación de conceptos clave de ingeniería de software, tanto en lo técnico como en lo organizacional. A lo largo de los cuatro sprints, el equipo trabajó de forma incremental

sobre los distintos módulos del sistema, priorizando funcionalidades esenciales para los usuarios del mismo.

Es importante destacar que, especialmente en las primeras etapas del proyecto, el equipo enfrentó ciertos desafíos relacionados con la adaptación al entorno de trabajo propuesto. Inicialmente, costó familiarizarse con la estructura base del template entregado, las herramientas utilizadas (como Spring Boot y JPA) y la dinámica de trabajo con historias de usuario divididas en tareas concretas. Esto hizo que en los primeros sprints tomáramos una cantidad reducida de historias de usuario, priorizando entender el flujo de trabajo del equipo, la integración entre las distintas partes del sistema y la correcta asignación de responsabilidades.

Sin embargo, con el correr de los sprints, logramos adaptarnos mejor a la metodología, repartir las tareas de manera más eficiente y ganar ritmo en el desarrollo. Esta mejora continua se reflejó en la posibilidad de abordar y finalizar varias historias de usuario importantes en los últimos sprints, consolidando así una versión funcional estable que cumple con los objetivos centrales del trabajo práctico.

Más allá de los resultados técnicos alcanzados, este proyecto permitió al equipo ejercitar habilidades fundamentales como el trabajo colaborativo, la comunicación constante, la adaptación al cambio y la búsqueda de mejoras en la forma de encarar las tareas. Si bien algunas funcionalidades complementarias quedaron pendientes de implementación, el balance general del proyecto es positivo, habiendo logrado entregar una solución que resuelve las necesidades principales planteadas en la consigna inicial.