.UBAfiuba
FACULTAD DE INGENIERÍA

Ingeniería de Software
(TA046_C2) Curso 2

Trabajo Práctico
Gestión y Reserva de Fútbol 5

24 de junio de 2025

Nicolás Grüner
110835

Martín del Castillo
106296

Melina Retamozo
110065

Alejandro Peña
106828

Lucas Villarrubia
108230

Alejo Sendra
107716

Victoria Zubieta
110769

1. Introducción

En muchas ciudades existe una comunidad creciente de personas interesadas en jugar al fútbol 5 de manera recreativa. Algunas veces se organizan con amigos, otras se anotan en grupos abiertos, o forman equipos estables que buscan participar en torneos. Sin embargo, organizar estos encuentros no siempre resulta sencillo: encontrar canchas disponibles, coordinar horarios, armar equipos, confirmar quiénes asisten y quiénes no, entre otras tareas logísticas.

A su vez, los dueños de canchas necesitan una forma eficiente de gestionar las reservas, evitar solapamientos y, eventualmente, fomentar el uso de sus instalaciones.

Como equipo de desarrollo, fuimos contactados por personas interesadas en resolver esta problemática, aunque sin una especificación detallada de requerimientos. Por lo tanto, fue necesario llevar a cabo un proceso de relevamiento para identificar las necesidades reales de los distintos tipos de usuarios.

El objetivo de este trabajo práctico es diseñar y especificar una plataforma de organización de partidos y torneos de fútbol 5, considerando los distintos perfiles de usuario: jugadores, organizadores, dueños de canchas y administradores.

En esta etapa del trabajo, nos enfocamos en llevar adelante el desarrollo de la aplicación, mediante el diseño e implementación de una arquitectura basada en **API REST**, y en la construcción de un **frontend funcional** que permita visualizar e interactuar con la plataforma. Esto no solo permitió materializar los requerimientos levantados, sino también contar con una primera versión navegable del sistema.

2. Entendiendo el problema

Para llevar a cabo el desarrollo de este sitio web se plasmaron diversas *Historias de Usuario* como punto de partida. Luego de la primera etapa de este proyecto, se llevó a cabo un análisis profundo del problema a abordar:

- ¿Qué queremos lograr?
- ¿A quiénes nos dirigimos?
- ¿Cuáles son las dificultades actuales al momento de organizar un partido?
- ¿Qué agilizaría el uso de esta aplicación?

Todas aquellas preguntas fueron respondidas en la etapa de ***Product Discovery***. Como equipo de trabajo, nos planteamos una lluvia de ideas lo que concluyó en la elaboración de diversas hipótesis que condujeron a una idea más general de la cual partir.

A partir de allí, se elaboraron encuestas y entrevistas con el fin de conocer a potenciales clientes de la plataforma a desarrollar: amigos que se juntan a jugar al fútbol, ligas amateurs y los propios propietarios de los establecimientos. Como consecuencia, se realizaron distintos mapas de empatía como un posible reflejo de los usuarios a tratar.

Asimismo, como una síntesis del problema, se tomaron declaraciones del mismo desde las perspectivas de los usuarios, y se pensó que diversos conflictos pueden llegar a tener al momento de organizar un partido. Del mismo modo, se tomaron declaraciones de los dueños de las canchas como punto de vista administrativo. Algunas propuestas:

- Facundo necesita una forma de encontrar gente para los partidos que estén dispuestos a jugar en horarios similares a los suyos. De esta forma, puede jugar más partidos en su tiempo libre.
- Jorge, dueño de múltiples canchas dispersas en varios distritos, necesita administrar mejor el manejo de sus reservas.
- Franco, jugador amateur, busca gente de su mismo nivel futbolístico.

Todas estas ideas y/o declaraciones fueron tomadas dado que, como desarrolladores, queríamos implementar una plataforma que satisfaga los posibles inconvenientes para mejorar la vivencia del usuario dentro del sitio. A continuación, se detallan algunas de las funcionalidades clave contempladas en el diseño e implementación de la plataforma:

- **Perfiles diferenciados:** Se contemplan registros con distintos roles: jugadores y dueños de cancha.
- **Filtros de búsqueda:** Los usuarios pueden buscar canchas según criterios como ubicación, horarios disponibles y nivel de juego.
- **Gestión de reservas:** El organizador tiene control sobre la reserva (invitar, expulsar jugadores) y puede iniciar una búsqueda de participantes si no completa los cupos. Los participantes pueden unirse, salirse o aceptar invitaciones.
- **Reservas mínimas garantizadas:** Se permite confirmar una reserva cuando se alcanza una cantidad mínima de jugadores, garantizando la realización del partido.
- **Funcionalidades para el dueño de cancha:** Puede gestionar horarios disponibles, crear torneos y cancelar reservas cuando sea necesario. También puede emitir recordatorios de pago.

Del mismo modo que las funcionalidades claves, se plantearon las mejoras que presentarían en los usuarios abordar dicha implementación:

- **Documentación de reseñas:** Se permite obtener las reseñas del establecimiento donde se desea jugar.
- **Mejor manejo de reservas:** Agendar las reservas de los usuarios.
- **Ranking:** Los equipos poseen un nivel de habilidad.

2.1. Historias de Usuario

Como se mencionó anteriormente, para abordar este proyecto nos basamos **historias de usuario** ya elaboradas. Las mismas son una técnica comúnmente utilizada en metodologías ágiles para representar requerimientos funcionales desde la perspectiva del usuario final. Su principal objetivo es capturar qué necesita un usuario, para qué lo necesita y cuál es el valor agregado que obtiene al utilizar el sistema.

Una historia de usuario típica sigue la siguiente estructura:

Como [tipo de usuario], *quiero* [funcionalidad], *para* [objetivo].

Funcionalidad y utilidad

- **Claridad en los requerimientos:** Permiten expresar necesidades de forma sencilla y comprensible, sin tecnicismos innecesarios.
- **Enfoque centrado en el usuario:** Ayudan a mantener el desarrollo orientado al valor real que el sistema aporta a sus distintos usuarios.
- **Priorización y planificación:** Facilitan la estimación del esfuerzo de implementación, la organización de tareas y la definición de prioridades.
- **Base para pruebas:** Cada historia puede complementarse con criterios de aceptación que indican cuándo se considera cumplida.
- **Facilitan la comunicación:** Sirven como herramienta de diálogo entre desarrolladores, usuarios y stakeholders, evitando documentación extensa.

2.2. Diseño preliminar

A continuación, se presenta el modelo arquitectónico diseñado para la plataforma de gestión y reserva de partidos de fútbol 5. Este modelo permite comprender cómo se estructuran los distintos componentes del sistema, cómo interactúan entre sí y cómo se distribuyen las responsabilidades a lo largo de la aplicación.

El diagrama incluye múltiples vistas arquitectónicas, organizadas de acuerdo con las buenas prácticas en diseño de software:

- **Vista Lógica:** describe la organización interna del backend según una arquitectura por capas. Se destacan los controladores REST, los servicios de negocio, las entidades del dominio y el acceso a datos. Además, se incluye el manejador global de excepciones y la capa de presentación web desarrollada con React, TypeScript y Vite.

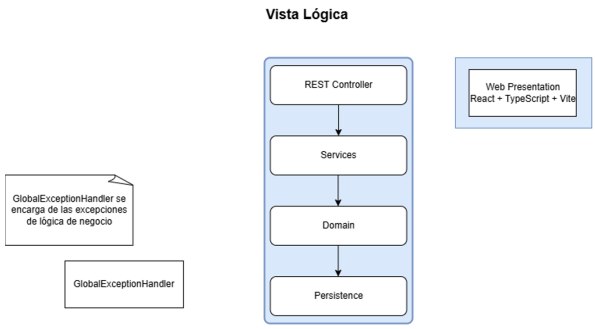

Figura 1: Vista Lógica

- **Vista de Escenarios (Casos de Uso):** ilustra las principales funcionalidades que ofrece el sistema, diferenciando claramente entre los actores *jugador* y *dueño de cancha*. Entre ellas se incluyen tareas como crear partidos, gestionar reservas o inscribirse a torneos.

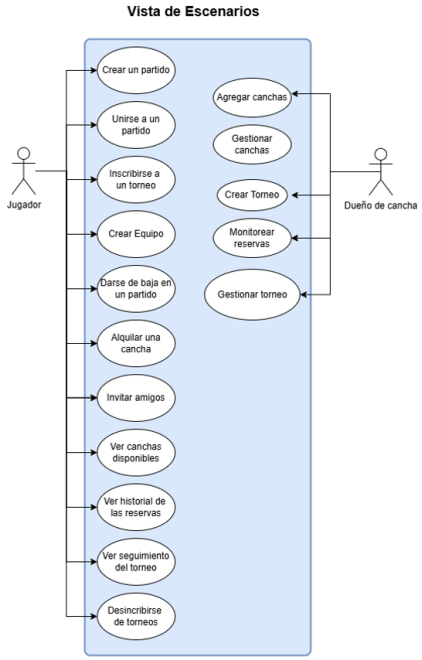

Figura 2: Vista de Escenarios

- **Vista de Componentes:** presenta la estructura de los principales módulos del sistema, incluyendo la aplicación web, el backend y las librerías externas utilizadas (Spring, JWT, JUnit, Postgres driver, entre otras).

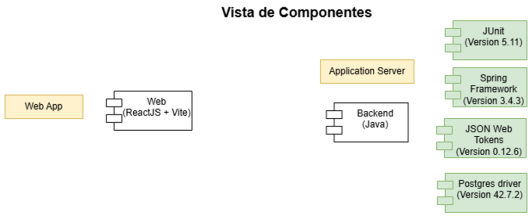

Figura 3: Vista de Componentes

- **Vista de Despliegue:** muestra cómo se distribuyen físicamente los distintos elementos de la arquitectura sobre los servidores. Se evidencia la interacción entre el cliente, el servidor web, el servidor de aplicación y el servidor de base de datos, junto con el servicio de envío de correos *Mailpit*.

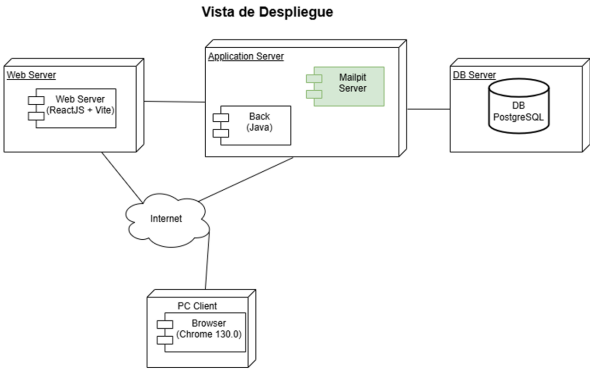

Figura 4: Vista de Despliegue

- **Vista de Procesos:** detalla cómo se comunican los distintos procesos de la aplicación a través de conexiones TCP, explicando el flujo entre navegador, backend, servidor de correo y base de datos.

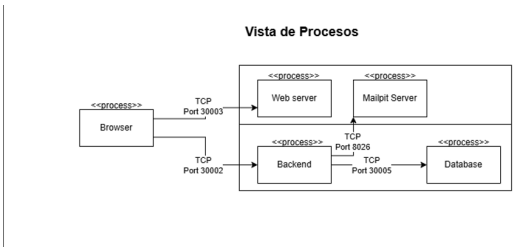

Figura 5: Vista de Procesos

Este enfoque permite visualizar de forma clara la arquitectura general del sistema, facilitando tanto su comprensión como su mantenimiento y evolución futura.

Para ver el modelo en su totalidad, ver el siguiente enlace: Modelo 4+1

3. Atributos de Calidad

Para garantizar una experiencia satisfactoria y segura para los usuarios, se definieron los siguientes atributos de calidad como prioritarios durante el desarrollo de la aplicación:

- **Usabilidad**: Se priorizó la usabilidad mediante una interfaz intuitiva, con campos claramente identificados y una navegación sencilla. Esto permite que los usuarios puedan organizar y participar en partidos de forma ágil, sin necesidad de conocimientos técnicos previos. Un criterio de aceptación podría ser lanzar una encuesta en la que se mida la satisfacción del usuario con el producto. El 85 % de los usuarios debería estar conforme con la aplicación.
- **Seguridad**: La seguridad se abordó mediante la implementación de autenticación basada en tokens JWT, asegurando que solo usuarios autenticados puedan acceder a recursos protegidos. Además, se controlan los permisos para evitar accesos indebidos a datos o funcionalidades que no correspondan al perfil del usuario.
- **Disponibilidad**: Se buscó asegurar que la aplicación esté disponible de forma continua, minimizando posibles tiempos de inactividad. La aplicación debería estar disponible las 24 horas del día, con algunos días de mantenimiento si fuera lanzada.

Metodología de trabajo

Para ejecutar este proyecto, se estableció un cronograma de *sprints* semanales donde se desarrollaban múltiples tareas, *tasks*, adaptadas de las historias de usuario proporcionadas por la cátedra. Además, en cada *sprint*, el grupo discutía sobre lo que aún estaba pendiente, qué mejoras implementar y se buscaba orientación con el tutor a cargo. Asimismo, para la gestión del flujo de trabajo y el desarrollo colaborativo de la aplicación, se utilizaron las herramientas *Jira* y *GitLab*.

En *Jira* se definieron las tareas a realizar, junto con su estimación de esfuerzo en puntos y la planificación de las distintas *sprints*. Esto permitió organizar de forma ágil y estructurada el avance del equipo, facilitando el seguimiento de los objetivos y responsabilidades individuales.

Por otra parte, se creó un repositorio remoto en *GitLab*, utilizando como base un proyecto inicial trabajado en clase. A partir de este repositorio, se crearon distintas ramas para que cada integrante del equipo pudiera desarrollar una funcionalidad específica de manera paralela. Esto permitió mantener un flujo de trabajo ordenado y aprovechar las ventajas del control de versiones para integrar progresivamente los aportes de todos los miembros.

3.1. Estimación historias de usuario

Como primer objetivo de este proyecto, se debio segmentar las historias de usuarios brindadas en tareas, *tasks*, para hacerlas lo más concisas y atómicas.

A modo de ejemplo se muestra la siguiente historia:

Inicio de Sesión

- Como usuario registrado, quiero iniciar sesión en la plataforma, para acceder a mi cuenta y utilizar las funcionalidades disponibles.

Criterios de Aceptación:

- La página de inicio de sesión debe tener campos para email y contraseña.
- El sistema debe validar que el email y la contraseña sean ingresados.
- El backend debe verificar las credenciales contra los datos almacenados.
- El sistema debe verificar que la cuenta del usuario esté activa y el email validado.
- Si las credenciales son incorrectas o la cuenta no está activa/validada, se debe mostrar un mensaje de error apropiado.
- Si las credenciales son correctas y la cuenta está activa, se debe generar un token de sesión seguro (e.g., JWT).

Esta historia fue dividida en varias tareas técnicas específicas, respetando los criterios de aceptación previstos:

- **ITD-139: Iniciar sesión (Back) – Redirigir al panel principal**
  Estimación: 1 punto
  Sprint: 1
- **ITD-140: Iniciar sesión (Back) – Campos obligatorios**
  Estimación: 2 puntos
  Sprint: 1
- **ITD-141: Iniciar sesión (API) – Validación de campos**
  Estimación: 2 puntos
  Sprint: 1
- **ITD-142: Iniciar sesión (Back) – Validar credenciales en base de datos**
  Estimación: 4 puntos
  Sprint: 1
- **ITD-143: Iniciar sesión (Back) – Verificar cuenta activada**
  Estimación: 3 puntos
  Sprint: 2
- **ITD-144: Iniciar sesión (Front) – Redirección al panel principal**
  Estimación: 2 puntos
  Sprint: 2
- **ITD-183: Iniciar sesión (Front) – Pantalla con campos de ingreso**
  Estimación: 4 puntos
  Sprint: 1
- **ITD-186: Iniciar sesión (Front) – Mensajes de error y validación visual**
  Estimación: 4 puntos
  Sprint: 2

Esta división permitió asignar responsabilidades claras, planificar en sprints y distribuir el esfuerzo entre los miembros del equipo. De esta forma, cada componente (backend, frontend, validaciones y redirecciones) fue desarrolla de forma paralela, asegurando la entrega incremental y funcional del módulo de autenticación.

Por otra parte, las tareas fueron subdivididas en *Backend, Frontend y API*. De esta manera, se fue construyendo la plataforma teniendo foco en los tres sectores a desarrollar.

3.1.1. Planning Poker

Como puede observarse en la sección interior, cada una de las tareas ejecutadas tienen una estimación por puntos. Luego de realizar la segmentación de las historias de usuarios, como equipo nos enfocamos en discutir y analizar cauanto tiempo llevaría su implemntación, es decir, cuanros recursos serían gastados.

Para ello, fue de gran ayuda el método de *Planning Poker*, abordado previamente en distintas tareas en clase. Mediante el uso de tarjetas, se fueron estimando las dstintas tareas y exponiendo ideas sobre como desarrollarlas y su costo.

En casos de discrepancia o difrencias en los votos, se buscaba como grupo llegar a un acuerdo y establecer un valor medio de estimación.

3.2. Sprints

Para este proyecto, se establecieron *4 sprints* semanales donde se establecían que tareas abordar. Finalizada cada *sprint,* se realizaba un análisis de lo alcanzado hasta el momento y se conversaba con el tutor a cargo sobre los avances próximos.

3.2.1. Sprint 1

En el primer *sprint*, se abordó todo lo relacionado con la creación de las entidades principales involucradas en la plataforma a desarrollar, así como también la implementación de las primeras funcionalidades básicas que servirían de base para el resto del sistema.

Entre las tareas realizadas en esta etapa inicial, se destacan las siguientes:

- Creación de Usuario
- Creación de Equipos
- Creación de Canchas
- Persisitencia en base de datos
- Base del front
- Regsitro y Creación de usuario

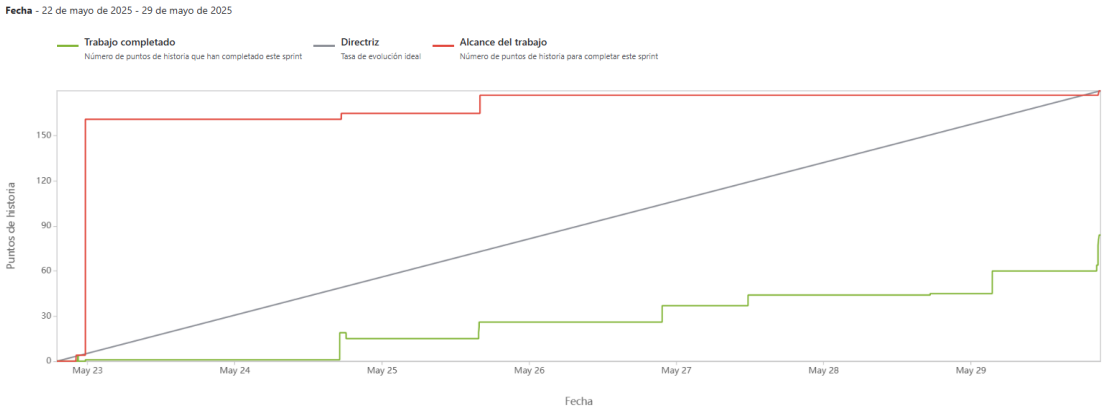

Figura 6: Gráfica del trabajo completado en el Sprint 1

En este sprint, al ser el primero, se buscó como equipo evaluar el flujo de trabajo, y establecer ciertos criterios comunes para las reuniones futuras. De acuerdo al gráfico presentado puede observarse que no se lograron concretar todas las tareas del sprint, las cuales quedaron pendientes para el próximo.

3.2.2. Sprint 2

Para el segundo *sprint*, se retomaron las tareas que habían quedado pendientes en el *sprint* anterior. Al mismo tiempo, se identificaron y resolvieron diversos inconvenientes surgidos durante el desarrollo, lo que permitió refinar tanto el diseño como la lógica de ciertas funcionalidades.

Durante esta etapa, se comenzó a trabajar sobre funcionalidades clave asociadas a las entidades ya definidas, avanzando en su implementación.

Entre las tareas concretadas en este sprint, se destacan las siguientes:

- Visualización de canchas (API) - Obtener un listado de cada cancha
- Inscribirse a Partido Abierto (Back) - Los participantes y el contador se actualizan
- Inscribirse a Partido Abierto (API) - No poder inscribirse si el partido está lleno o comenzado
- Editar Canchas (API) - Procesar edición de las canchas existentes

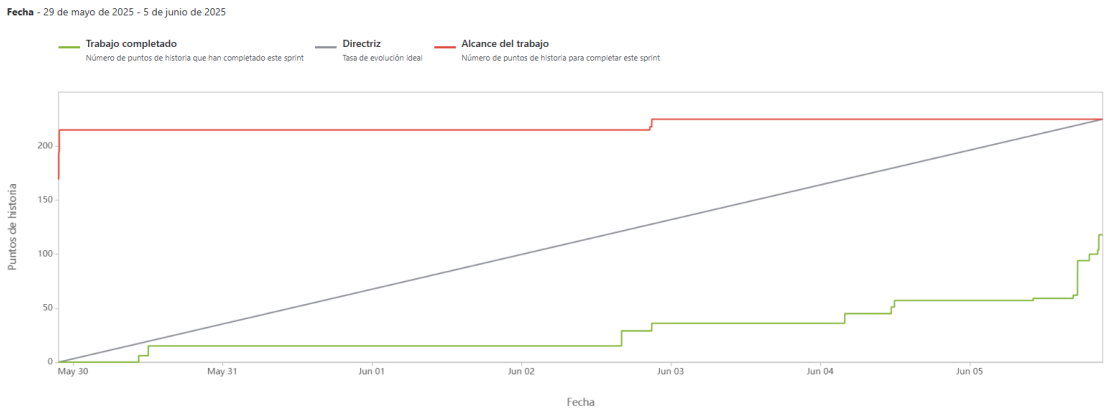

Figura 7: Gráfica del trabajo completado en el Sprint 2

Al igual que en el sprint anterior, no fueron alcanzados los puntos de historia preestablecidos. Sin embargo, se pudieron mejorar ciertos criterios para seguir adelante.

3.2.3. Sprint 3

Para este sprint, la mayoría de los objetivos relacionados con el *backend* y la construcción de la *API* ya se encontraban cumplidos. Las funcionalidades centrales del sistema, así como los endpoints principales, habían sido desarrollados e integrados satisfactoriamente.

Sin embargo, aún quedaban pendientes varias tareas vinculadas al desarrollo del *frontend*, especialmente aquellas relacionadas con la presentación visual y la interacción del usuario con el sistema.

En esta sprint se cumplió:

- Recuperación de contraseña (Front) - Pantalla de recuperación de contraseña
- Visualización de canchas (Front) - Mostrar jugadores faltantes de cada partido
- Organizar jugadores en Partido Abierto (Back) - Dar opciones para poder asignar jugadores manualmente a los equipos
- Darse de baja del partido (Back) - Validar que no sea el organizador quien se da de baja
- Eliminar Canchas (Back) - Verificar que no existan reservas futuras confirmadas

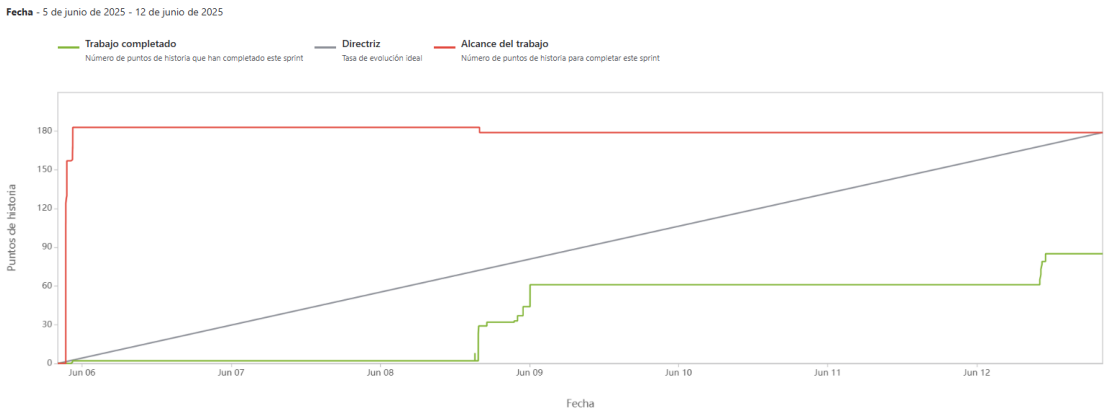

Figura 8: Gráfica del trabajo completado en el Sprint 3

Como se mencionó anteriormente, en este sprint se buscó abordar de forma integral todos los puntos que habían sido planteados previamente para el *backend* y la *API*, completando las funcionalidades pendientes y asegurando su correcto funcionamiento dentro del sistema.

Además, se lograron avances en el desarrollo del *frontend*, con la implementación de componentes visuales clave y su integración con los servicios del backend. También se comenzaron a ejecutar ciertas pruebas unitarias sobre los módulos desarrollados, con el fin de validar el comportamiento esperado de las funcionalidades y garantizar mayor estabilidad en el producto final.

3.2.4. Sprint 4

Durante la última semana del proyecto, correspondiente al sprint final, se avanzó en la finalización de tareas pendientes y en la consolidación del trabajo realizado. Se completaron detalles técnicos que habían quedado abiertos en las entidades ya implementadas, asegurando su correcto funcionamiento y coherencia con el resto del sistema.

Uno de los principales focos del sprint fue el desarrollo y ajuste del *Frontend*. Se trabajó intensamente en la construcción de interfaces amigables, asegurando la correcta integración con las funcionalidades del backend previamente desarrolladas. Además, se priorizó que la experiencia de usuario sea clara, funcional y coherente con los requerimientos relevados.

Paralelamente, se implementaron **tests unitarios** sobre los componentes centrales de la aplicación, con el objetivo de validar el comportamiento esperado de cada módulo y garantizar mayor robustez en el sistema final.

Otra actividad clave fue la *refactorización del código*. Se dedicó tiempo a revisar los distintos módulos del proyecto para mejorar su legibilidad, eliminar fragmentos de código innecesarios, comentarios obsoletos, nombres confusos y redundancias. Esto permitió dejar un código más limpio, mantenible y profesional.

Finalmente, se reorganizó el repositorio remoto, integrando los últimos avances de cada integrante del equipo. Se consolidaron las ramas principales, resolviendo conflictos y asegurando una estructura clara del proyecto en su versión final.

En conjunto, este sprint representó una etapa de cierre técnico, pulido visual y alineación colaborativa para entregar una versión funcional, estable y presentable de la aplicación.

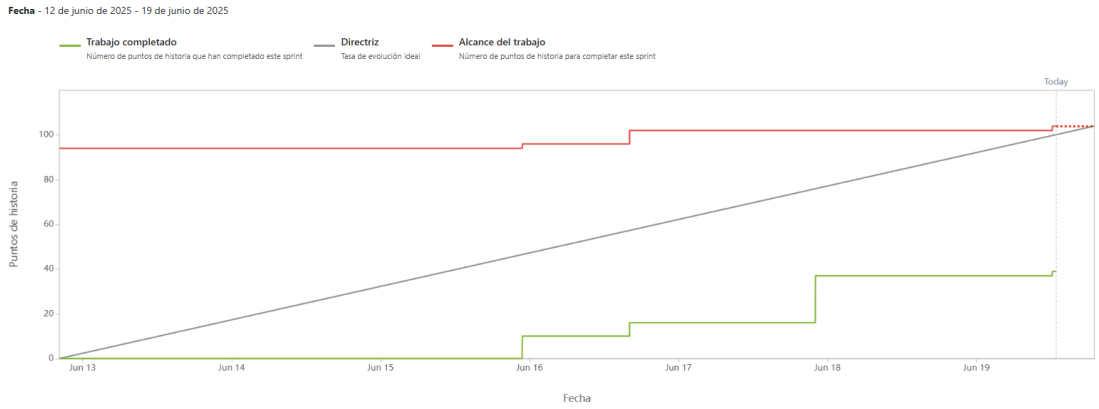

Figura 9: Gráfica del trabajo completado en el Sprint 4

3.3. Flujo de trabajo por sprint

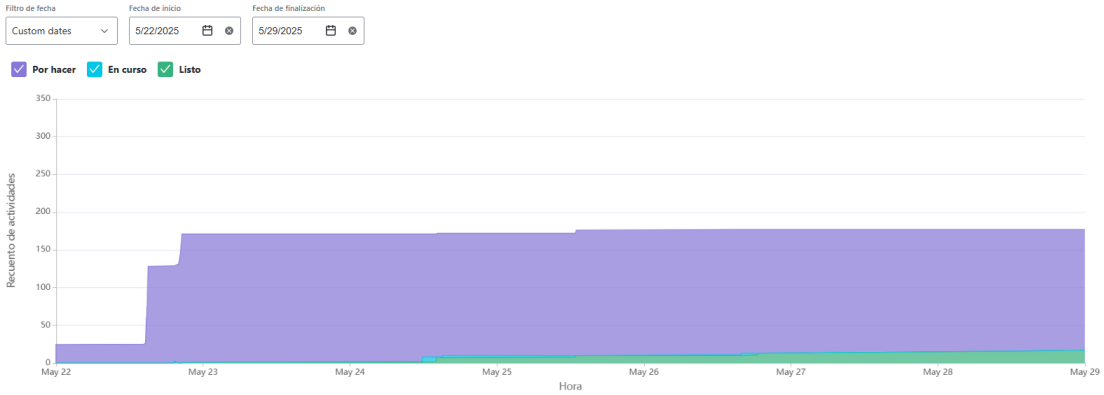

Figura 10: Flujo acumulado Sprint 1

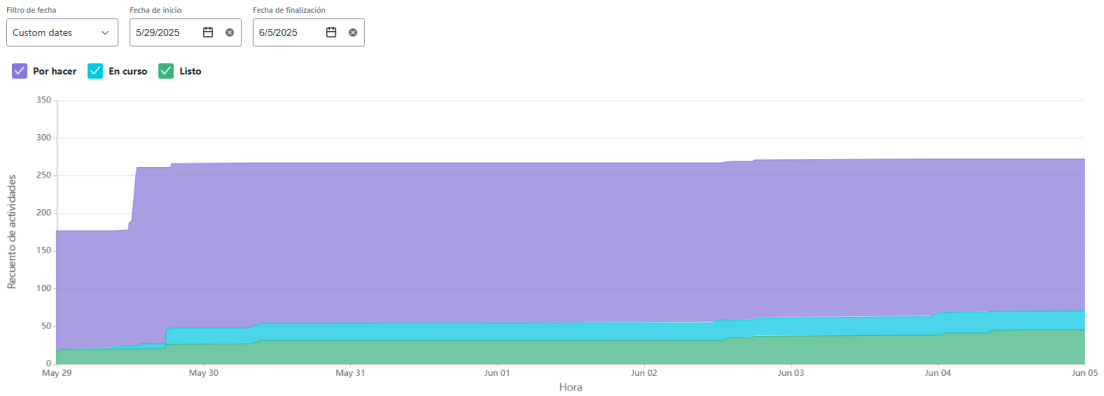

Figura 11: Flujo acumulado Sprint 2

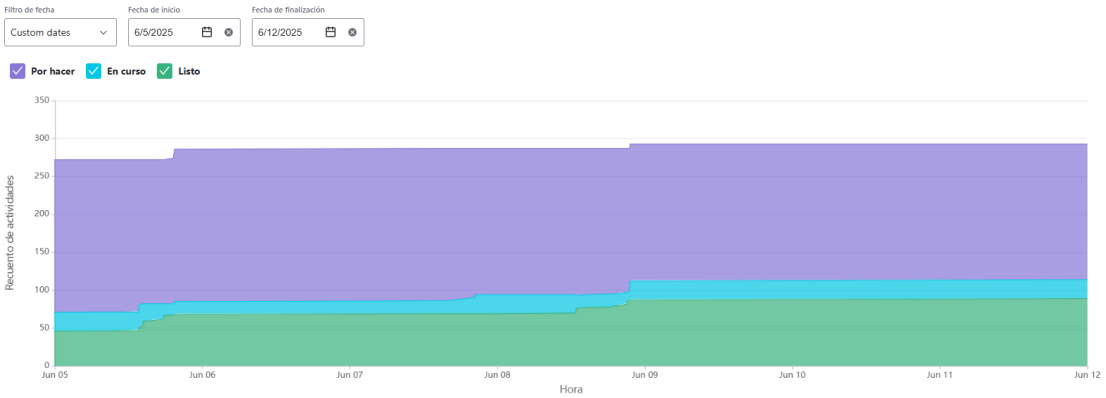

Figura 12: Flujo acumulado Sprint 3

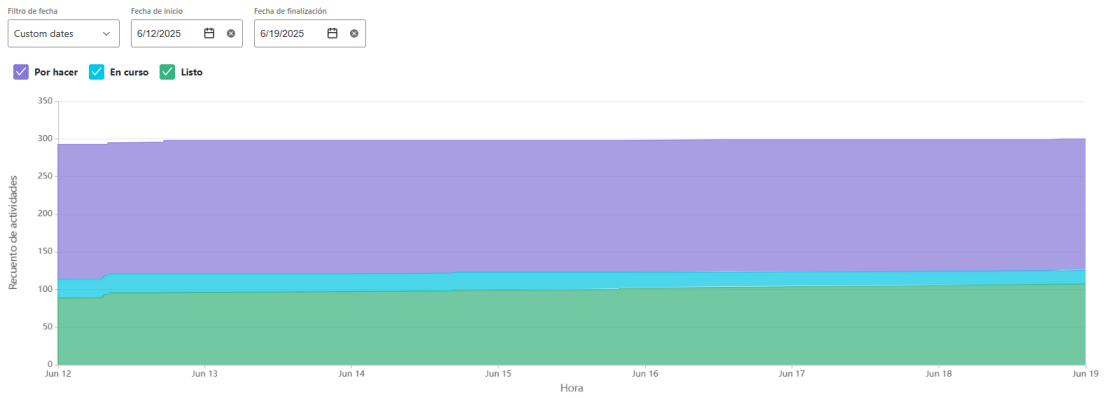

Figura 13: Flujo acumulado Sprint 4

Los gráficos propuestos evidencian como fue el flujo de trabajo a lo largo de cada sprint. Si se realiza una comparativa entre el Sprint 1 y el Sprint 4, se puede observar el avance de las tareas cumplidas. Sin embargo, en algunos sprints se identifican áreas con acumulación prolongada de tareas en curso, lo que refleja dificultades en la distribución del trabajo, tiempos de implementación o integración como equipo.

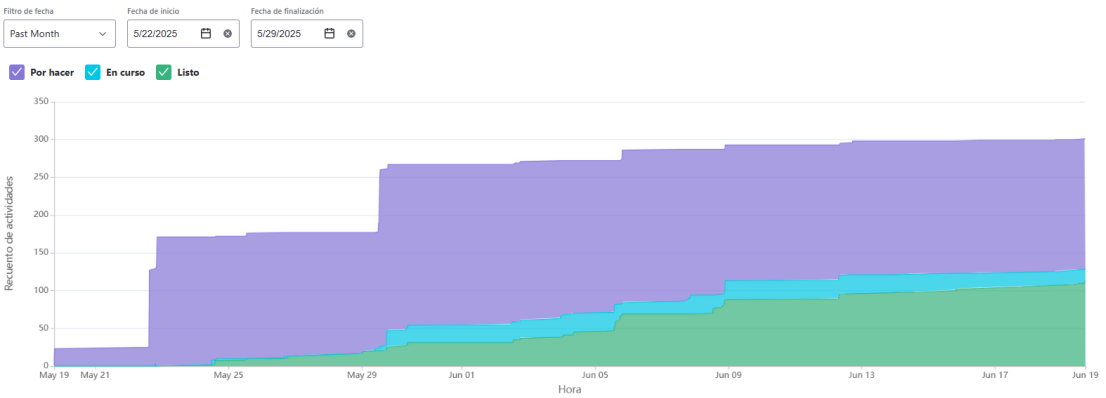

Figura 14: Flujo acumulado a lo largo del proyecto

En conjunto, los gráficos reflejan un progreso constante en el desarrollo del proyecto, aunque

también revelan una deficiencia por parte del grupo en la finalización de ciertas tareas.

4. Validación del Sistema: Tests Realizados

Durante las etapas finales del desarrollo se llevó a cabo un conjunto de pruebas destinadas a validar las funcionalidades implementadas tanto en el backend como en el frontend. El objetivo principal de estos tests fue garantizar la estabilidad, consistencia y correcto comportamiento del sistema ante distintos escenarios de uso.

Se desarrollaron principalmente tests unitarios para comprobar la lógica interna de los servicios y componentes clave, así como también validaciones de entidades para asegurar que los datos cumplieran con las restricciones definidas. Estas pruebas se realizaron empleando frameworks como JUnit y el sistema de validación de Bean Validation (Jakarta).

Las pruebas abarcan los siguientes aspectos:

- **Validación de campos obligatorios**: se verificó que las entidades no aceptan valores nulos o vacíos en atributos esenciales como nombre, email, contraseña o zona.
- **Pruebas sobre DTOs**: se testearon los distintos DTOs (como *UserProfileDTO*, *TeamCreateDTO*, *FieldDTO*, entre otros) para asegurar que la conversión desde las entidades fuera correcta y coherente.
- **Lógica de negocio**: se evaluó el comportamiento de métodos relevantes, como la lógica de asignación de jugadores, la creación y edición de equipos, y las reglas para reservas o partidos.
- **Cobertura de casos extremos**: se incluyeron pruebas negativas, como intentos de crear usuarios con emails duplicados o de eliminar capitanes de equipos, para validar el manejo de errores y excepciones.

Estas pruebas fueron fundamentales para detectar errores de forma temprana y asegurar que las funcionalidades desarrolladas se comportan según lo esperado.

5. Análisis de lo desarrollado

Durante el desarrollo del proyecto, se procuró implementar todos los objetivos establecidos en cada uno de los sprints planificados. No obstante, no se logró cumplir con la totalidad de los requerimientos. Como equipo, enfrentamos dificultades principalmente en el desarrollo del frontend de la plataforma y en la coordinación de tareas, lo cual derivó en la imposibilidad de implementar ciertos puntos asignados.

Si bien se realizaron reuniones para discutir la implementación de algunos ítems, se evidenció una fuerte individualidad a la hora de abordar las tareas, lo que dificultó el trabajo colaborativo. Además, durante el proceso se produjo la baja de un integrante del equipo, lo que impidió finalizar la parte del sistema que tenía a su cargo.

Por otro lado, el uso de herramientas como Jira y GitLab no presentó inconvenientes. Cada integrante trabajó en su propia rama, agregando funcionalidades que luego se fusionaban con la rama principal de desarrollo. La planificación y estimación de historias en Jira también se realizó sin problemas, buscando siempre priorizar los aspectos esenciales para el desarrollo de la plataforma.

Lamentablemente, no se logró entregar el producto que originalmente habíamos proyectado a partir de los diagramas e ideas iniciales. Si bien se intentó cumplir con los requisitos establecidos, el código como en la lógica de implementación podría mejorarse. Sin embargo, a pesar de las dificultades, se buscó avanzar tanto en el desarrollo del backend como en la construcción de la API, con el objetivo de presentar una base sólida que funcione.

Por otra parte, desde el frontend se intentó cubrir las funcionalidades principales, tales como el *inicio de sesión*, la *registración de usuarios*, la *validación del correo electrónico*, el *restablecimiento de la contraseña*, y la *creación de partidos, equipos y canchas.*

6. Anexo: Refactorizaciones y funcionalidades pendientes

- **Front**
  - UI e integración de la entidad Torneo en la aplicación: formulario de creación y edición, otras acciones predefinidas para el manejo de la entidad, interacción con los endpoints correspondientes.
  - Visualización de una Cancha no alcanzada: falta de muestra de imágenes y campo `images` faltante en formularios de creación y edición.
  - Manejo de canchas: no se muestran los calendarios de una cancha con sus fechas disponibles. Tampoco se pueden crear partidos (reservaciones) dentro de una cancha en específico (automatización de un formulario de creación con ID de cancha).
  - Distribuciones en partidos: falta la implementación de frontend de la entidad secundaria `MatchOrganizer`, elección de distintos criterios para organizar jugadores en un partido incluyendo el modo manual.
  - Historial de partidos: no se muestran los partidos cerrados en la página de perfil de un usuario.
  - Edición de jugadores en partidos y equipos: UIs de "agregar jugador" y "eliminar jugador" incompletas, falta la implementación de presentar a los jugadores en una lista editable básica con checkboxes para eliminar fácilmente; y un componente de barra de búsqueda para encontrar jugadores existentes en el recurso de Usuarios de la base de datos.
  - Componente de subir archivos (fotos: `.jpg`, `.png`, etc.): sin terminar y sin incluir en ninguna página. Previsto para formularios de creación de ítems y para registro de usuario.
  - Acciones de usuario ADMIN y escalabilidad a otros roles de usuario: sin diferenciación ni acciones diferidas según tipo de rol de usuario.
  - Performance y eficiencia de aplicación: algunos hooks de navegación no permiten a la aplicación refrescarse correctamente para mostrar la información correcta en pantalla. En algunas ocasiones se requiere un refresh manual. Falta un refactor de usabilidad a la UI integral de la aplicación para mejorar su rendimiento y eficacia.
- **Tournaments** (relevada en la branch `tournaments`): La entidad Torneo (prevista en la visión general inicial del proyecto y en el enunciado) no cuenta con su integración en el modelo de clases y entidades previsto.
  - No posee un recurso creado como tal en la base de datos (ninguna tabla ni asociaciones).
  - Tiene atributos incompletos y no es funcional con el resto de clases y servicios con las que debería relacionarse.
  - No se implementaron los endpoints necesarios para la manipulación del recurso desde la API.
- **Teams**: Al recurso de Equipos le falta un endpoint de obtención (GET) según el usuario actual de la API (recibido por headers).
- **API - Estandarización**: No se estandarizó la estructura de requests y responses para una integración escalable entre front y back. Detalles en documentación presentada en Swagger.

7. Anexo: Notas - correcciones 19/06/25

- La clase y entidad Usuario no cuentan con diferenciación de roles de usuario ni con acciones específicas de autoridad.
- El frontend guarda el token de inicio de sesión de un usuario para persistir la autenticación a algunos recursos que la requieren en la aplicación. Esto se ejecuta en el archivo `"frontend/src/features/users/usersAPI.js"`, en la función `loginUser`; se guarda el token de login en el `localStorage` del navegador y se agrega a los headers por default (como "Bearer") de `axios` (librería predeterminada para las llamadas a la API desde la aplicación).