Desarrollo de una aplicación web para la gestión de las selecciones deportivas de la Universidad Industrial de Santander

Anderson Andrés González Cortes y Efraín José Blanco Duarte

Trabajo de grado para optar al título de Ingenieros de Sistemas

Director

Urbano Eliécer Gómez Prada,

Doctor en Tecnología Educativa

Tutor

Fabio Andelfo Villafrades González,

Doctor en Ciencias de la Cultura Física y Deportes

Entidad Interesada

Departamento de Educación Física y Deportes

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2026

Tabla de Contenido

Lista de figuras

Resumen

Título: Desarrollo de una aplicación web para la gestión de las selecciones deportivas de la Universidad Industrial de Santander*

Autor: Anderson Andrés González Cortes, Efraín José Blanco Duarte**

Palabras Clave: Arquitectura hexagonal, Angular, Spring Boot, JWT, Scrum, aplicación web, gestión deportiva, CQRS.

Descripción: El presente trabajo de grado describe el diseño e implementación de una aplicación web orientada a la gestión de las selecciones deportivas de la Universidad Industrial de Santander. La propuesta surge como respuesta a las limitaciones existentes en los procesos administrativos actuales, caracterizados por el manejo manual y descentralizado de la información relacionada con jugadores, entrenadores, eventos, publicaciones y logros deportivos.

La solución desarrollada se fundamenta en una arquitectura hexagonal complementada con los enfoques de vertical slicing y CQRS, permitiendo una separación clara de responsabilidades y favoreciendo la escalabilidad, mantenibilidad y modularidad del sistema. El frontend fue implementado mediante Angular y Tailwind CSS, utilizando componentes standalone y mecanismos reactivos basados en signals para la gestión eficiente del estado de la aplicación. El backend fue desarrollado con Spring Boot y Spring Security, incorporando autenticación y autorización mediante JSON Web Tokens (JWT). Asimismo, la persistencia de datos se gestionó con MySQL y el entorno de despliegue fue soportado mediante contenedores Docker.

El desarrollo del aplicativo se realizó bajo la metodología ágil Scrum, permitiendo iteraciones continuas y validaciones periódicas con los usuarios involucrados en el proceso. Como resultado, se obtuvo una plataforma centralizada que permite administrar selecciones deportivas, integrantes, horarios, eventos, publicaciones, logros y recursos multimedia, optimizando el acceso a la información y fortaleciendo los procesos de gestión deportiva institucional.

---

* Trabajo de Grado

** Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Urbano Eliécer Gómez Prada. Doctor en Tecnología Educativa. Tutor: Fabio Andelfo Villafrades González. Doctor en Ciencias de la Cultura Física y Deportes

Abstract

Title: Development of a Web Application for the Management of the Sports Teams of Universidad Industrial de Santander [*]

Author: Anderson Andrés González Cortes, Efraín José Blanco Duarte[**]

Key Words: Hexagonal architecture, Angular, Spring Boot, JWT, Scrum, web application, sports management, CQRS.

Description: This undergraduate thesis presents the design and implementation of a web application focused on the management of the sports teams of Universidad Industrial de Santander. The proposal emerged as a response to the limitations identified in the current administrative processes, which are characterized by the manual and decentralized management of information related to players, coaches, events, publications, and sports achievements.

The developed solution is based on a hexagonal architecture complemented by vertical slicing and CQRS approaches, enabling a clear separation of responsibilities while improving the scalability, maintainability, and modularity of the system. The frontend was implemented using Angular and Tailwind CSS, incorporating standalone components and reactive state management mechanisms based on signals. The backend was developed using Spring Boot and Spring Security, integrating authentication and authorization through JSON Web Tokens (JWT). In addition, data persistence was managed using MySQL, while the deployment environment was supported through Docker containerization technologies.

The application was developed following the Scrum agile methodology, allowing continuous iterations and periodic validation with the users involved in the process. As a result, a centralized platform capable of managing sports teams, members, schedules, events, publications, achievements, and multimedia resources was obtained, optimizing information accessibility and strengthening institutional sports management processes.

---

[*] Degree Work

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Urbano Eliécer Gómez Prada. Doctor en Tecnología Educativa. Tutor: Fabio Andelfo Villafrades González. Doctor en Ciencias de la Cultura Física y Deportes.

Introducción

El avance tecnológico ha permitido la digitalización de múltiples procesos en distintos ámbitos, mejorando la eficiencia y optimización en la gestión de la información. En el contexto deportivo universitario, la implementación de herramientas tecnológicas resulta clave para facilitar la administración de selecciones deportivas, ofrecer una mejor organización de eventos y optimizar la toma de decisiones basada en datos precisos y accesibles.

En este sentido, el presente proyecto tiene como objetivo el desarrollo de una aplicación web que permita gestionar de manera eficiente la información de las selecciones deportivas de la Universidad Industrial de Santander. Esta herramienta brindará funcionalidades como la administración de jugadores y la comunicación efectiva de noticias y logros.

Para la implementación de la solución, se adoptará una arquitectura hexagonal que permitirá una mayor escalabilidad y mantenibilidad del software. El desarrollo se realizará utilizando tecnologías modernas como Angular para el frontend, Spring Boot para el backend y MySQL como base de datos. Además, se empleará la metodología ágil Scrum, fomentando una entrega iterativa e incremental, con validaciones continuas por parte de los usuarios finales.

Este documento detalla el marco metodológico y técnico del proyecto, proporcionando un cronograma de trabajo estructurado y describiendo los recursos tecnológicos y humanos necesarios para su implementación. A través de esta iniciativa, se busca modernizar la gestión deportiva universitaria y optimizar los procesos administrativos asociados a las selecciones deportivas.

1 Planteamiento del Problema

La Universidad Industrial de Santander tiene una destacada trayectoria en el desarrollo del deporte y en la formación integral de sus estudiantes a través de diversas selecciones deportivas. “Los deportes representan un componente fundamental en la vida universitaria al fomentar valores como el trabajo en equipo, la disciplina, el esfuerzo y el liderazgo” (Universidad de los Andes, 2024).

No obstante, a pesar de esta tradición, actualmente la gestión de la información relacionada con las selecciones deportivas se realiza de manera manual y descentralizada, lo que genera diversos problemas que afectan la eficiencia y calidad en la administración de las actividades deportivas, según Fabio Villafrades director del Departamento Educación Física y Deportes, tutor de este proyecto (2025), quien fue entrevistado en el marco de esta investigación.

La situación actual se caracteriza por el uso de registros físicos y herramientas dispersas, como archivos en papel, hojas de cálculo no integradas y sistemas de almacenamiento personales de los entrenadores y administrativos, según lo mencionado por Fabio Villafrades (2025), la falta de este aplicativo genera inconvenientes significativos, tales como:

- Dificultad en el acceso a la información: Por ejemplo, cuando un entrenador necesita conocer el historial deportivo de un jugador, debe consultar múltiples fuentes o documentos físicos que no siempre están actualizados o disponibles. Esta búsqueda consume tiempo considerable y limita la toma de decisiones en tiempo real.
- Duplicidad y pérdida de información: Al no existir un sistema centralizado, los mismos datos pueden registrarse varias veces en diferentes formatos, lo que genera inconsistencias y errores en la información. Además, los registros en papel son vulnerables a pérdidas, deterioro o extravío.

- Falta de análisis y seguimiento sistemático: La ausencia de una herramienta digital limita la capacidad de analizar datos. Por ejemplo, evaluar la historia de los equipos a lo largo del tiempo resulta complicado porque los datos históricos no están organizados ni accesibles de forma centralizada.
- Ineficiencia en la comunicación de eventos: La aplicación permitirá comunicar fechas y registrar la participación de los jugadores a competencias o entrenamientos de las selecciones deportivas.

Para mitigar estos inconvenientes, se desarrollará una aplicación web de gestión que automatice y mejore los procesos administrativos de las selecciones. El cual permitirá registrar y consultar la información de manera organizada, integrando en una sola plataforma datos sobre jugadores, entrenadores, calendarios de entrenamientos y competencias. Al digitalizar estos procesos, se reducirá significativamente el tiempo de búsqueda y gestión de la información, facilitando la toma de decisiones oportunas y precisas.

La implementación de esta herramienta tecnológica busca una mayor capacidad de análisis y seguimiento del desempeño deportivo. Permitirá realizar evaluaciones sistemáticas de los jugadores y equipos, y mantener un historial actualizado y accesible. De este modo, se potenciará la comunicación de las actividades deportivas y se fomentará un ambiente más estructurado que responda a las necesidades actuales de la comunidad universitaria.

2 Objetivos

2.1 Objetivo General

Desarrollar una aplicación web para la gestión por parte de docentes y entrenadores de los equipos, jugadores, eventos, y logros de las selecciones deportivas de la Universidad Industrial de Santander.

2.2 Objetivos Específicos

1. Definir los requerimientos para la gestión de los datos personales y atléticos de los integrantes de las selecciones deportivas, la historia deportiva de la universidad incluyendo logros y eventos, la comunicación de noticias, próximos eventos y reconocimientos.
2. Diseñar las especificaciones funcionales y técnicas de la aplicación abarcando su alcance, capacidades de almacenamiento y mecanismos de visualización, según las necesidades del proyecto.
3. Implementar la aplicación web siguiendo los requerimientos definidos y el diseño realizado, que integre los componentes de backend, frontend y base de datos.
4. Elaborar un plan de pruebas involucrando una muestra representativa de los usuarios para la validación de la funcionalidad, usabilidad y cumplimiento de los requisitos establecidos.

3 Marco de Referencia

El marco de referencia que requiere este proyecto es presentado a continuación, el cual se desagregó en:

3.1 Marco contextual

Las definiciones y conceptos en las que se contextualiza el proyecto se presentan a continuación.

3.1.1 Selecciones deportivas

Las selecciones deportivas son equipos conformados por estudiantes, docentes o personal administrativo que representan a la institución en competencias deportivas a nivel local, regional o nacional. Estas selecciones son organizadas y dirigidas por entrenadores designados y cumplen con procesos de inscripción, entrenamiento y participación en eventos oficiales. (Universidad Industrial de Santander, 2014)

3.1.2 Historia y misión de la UIS en el ámbito deportivo

La Universidad Industrial de Santander fue creada el 9 de diciembre de 1947 y comenzó sus labores el 1 de marzo de 1948. Desde su fundación, ha estado comprometida con la formación integral de sus estudiantes, incorporando el deporte como una herramienta esencial para el desarrollo físico y mental. La universidad reconoce la importancia del deporte en la educación superior y ha promovido activamente actividades deportivas a lo largo de su historia. (Universidad Industrial de Santander, 2021)

3.1.3 Políticas institucionales sobre el fomento del deporte y la recreación

La Universidad Industrial de Santander cuenta con una política específica para grupos culturales y selecciones deportivas, cuyo objetivo es fomentar las actividades culturales y

deportivas mediante el apoyo y la promoción integral de las actividades debidamente programadas. Esta política busca fortalecer la participación estudiantil en actividades que complementen su formación académica y contribuyan a su bienestar general. (Universidad Industrial de Santander, 2014)

3.1.4 Importancia del deporte en la formación integral de los estudiantes

El deporte juega un papel fundamental en la formación integral de los estudiantes, ya que contribuye al desarrollo de habilidades físicas, mentales y sociales. En el contexto educativo, la práctica deportiva promueve la salud, el trabajo en equipo, la disciplina y el liderazgo, aspectos esenciales para el crecimiento personal y profesional de los jóvenes. (Gobernación de Santander, 2023)

3.2 Marco tecnológico

Las tecnologías y conceptos que se van a usar en este proyecto se presentan a continuación.

3.2.1 Angular

Angular es un framework de desarrollo frontend basado en TypeScript, diseñado para la creación de aplicaciones web dinámicas y escalables. Angular proporciona un modelo basado en componentes que permite la reutilización de código, modularidad y separación de responsabilidades. Su arquitectura permite la integración con APIs REST, lo que facilita la conexión con servicios backend. Además, cuenta con herramientas avanzadas como Angular CLI para la gestión de proyectos, enrutamiento dinámico, inyección de dependencias y pruebas unitarias, optimizando el desarrollo de aplicaciones interactivas y eficientes. (Team, 2024)

3.2.2 Tailwind CSS

Tailwind CSS es un framework de diseño basado en clases de utilidad que permite la creación de interfaces de usuario altamente personalizadas sin necesidad de escribir CSS adicional. Tailwind CSS facilita el desarrollo de componentes reutilizables y ofrece consistencia visual en la aplicación. Su enfoque "utility-first" permite la creación rápida de estilos, mejorando la productividad de los desarrolladores al evitar la necesidad de escribir hojas de estilo personalizadas. Además, ofrece compatibilidad con herramientas de optimización que reducen el tamaño del CSS final, mejorando el rendimiento de la aplicación. (Labs, 2024)

3.2.3 Señales en Angular

Las señales en Angular fueron introducidas en las versiones más recientes de Angular, las señales son un mecanismo para gestionar el estado de los componentes de manera reactiva y eficiente. A diferencia de otros enfoques de detección de cambios, las señales optimizan la actualización de la interfaz de usuario al ejecutar solo las modificaciones necesarias, lo que mejora el rendimiento de la aplicación. Este modelo simplifica la gestión del estado y la comunicación entre componentes, reduciendo la necesidad de soluciones externas en ciertos escenarios. (Team, 2024)

3.2.4 Spring Boot

Spring Boot es un framework de desarrollo backend en Java que simplifica la creación de aplicaciones empresariales al proporcionar una estructura preconfigurada y una amplia integración con herramientas del ecosistema Java. Spring Boot permite la creación de APIs REST de manera eficiente, incorporando características como la gestión de dependencias con Spring Boot Starter, seguridad integrada con Spring Security y soporte para bases de datos

mediante Spring Data JPA. Su arquitectura modular permite escalabilidad y facilita la implementación de microservicios. (Software, 2024)

3.2.5 Json Web Token (JWT)

Json Web Token es un estándar de autenticación que permite la transmisión segura de información entre sistemas mediante la generación de tokens en formato JSON. En este proyecto, JWT se utilizará para la autenticación y autorización de usuarios, eliminando la necesidad de gestionar sesiones en el servidor. Cada token contiene información cifrada sobre el usuario y sus permisos, lo que provee un acceso seguro a los recursos protegidos de la aplicación. (Jones, Bradley, & Sakimura, 2015)

3.2.6 Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés)

API es una interfaz que define la forma en que diferentes componentes de software interactúan entre sí. En este proyecto, se utilizarán APIs REST para la comunicación entre el frontend y backend, lo que permitirá una integración flexible y desacoplada. Las APIs seguirán principios REST, ofreciendo operaciones eficientes para la manipulación de datos, con soporte para peticiones HTTP como GET, POST, PUT y DELETE. (Richardson, 2020)

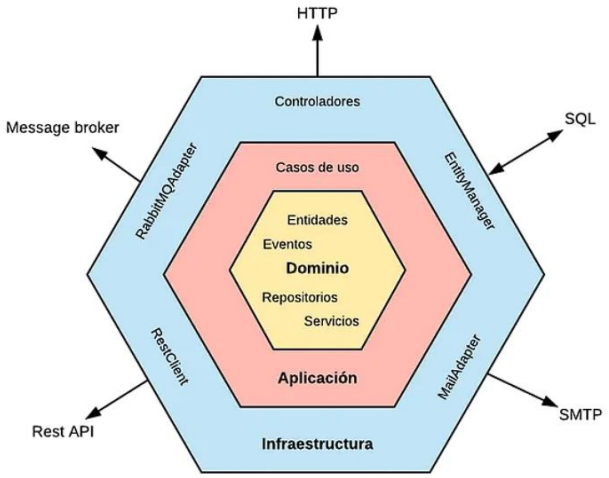
3.2.7 Arquitectura Hexagonal

La Arquitectura Hexagonal es un patrón de diseño de software que promueve la separación entre la lógica de negocio y las interfaces externas, como bases de datos, interfaces gráficas y servicios web. Su implementación en este proyecto permitirá que los componentes del aplicativo sean modulares, facilitando la mantenibilidad y escalabilidad. La arquitectura hexagonal se basa en el uso de puertos y adaptadores para permitir la independencia entre las capas de la aplicación, promoviendo un diseño flexible que facilita la prueba y evolución de la

aplicación, en la Figura 1 se presenta un diagrama de la arquitectura hexagonal que ilustra la organización de sus componentes y las interacciones entre ellos. (Buckley, 2021)

Figura 1

Arquitectura hexagonal



3.2.8 MySQL

MYSQL es un sistema de gestión de bases de datos relacional que ofrece almacenamiento seguro, consultas eficientes y herramientas avanzadas para la manipulación de datos. MySQL soporta integridad referencial, transacciones y escalabilidad, lo que lo convierte en una opción robusta para gestionar la información como por ejemplo de jugadores, equipos y eventos deportivos. (Oracle, 2024)

3.2.9 Docker

Docker es una plataforma de virtualización a nivel de sistema operativo que permite empaquetar aplicaciones y sus dependencias en unidades llamadas contenedores. Estos contenedores garantizan que el software se ejecute de manera consistente en diferentes entornos, eliminando problemas asociados a configuraciones específicas de cada sistema.

En el contexto de aplicaciones web, Docker facilita el despliegue al permitir encapsular componentes como el frontend, el backend y la base de datos en contenedores independientes, los cuales pueden ser gestionados y ejecutados de forma aislada pero coordinada.

El uso de contenedores aporta beneficios como la portabilidad, la escalabilidad y la reproducibilidad del entorno de ejecución. Además, permite simplificar la configuración de infraestructura mediante herramientas como Docker Compose, que facilita la orquestación de múltiples servicios.

3.2.10 Metodología Scrum

Scrum es un marco de trabajo ágil que permite la gestión eficiente de proyectos mediante iteraciones llamadas sprints. Scrum facilita la entrega incremental de funcionalidades, permitiendo la validación continua del producto con los usuarios. La metodología incluye roles definidos como Scrum Master, Product Owner y equipo de desarrollo, así como eventos clave como reuniones diarias (Daily Scrum), planificación de sprint y retrospectivas. (Rubin, 2012) Su aplicación en este proyecto proveerá una ejecución organizada y adaptable a los cambios de requerimientos, maximizando la eficiencia del equipo de desarrollo (Schwaber, 2020).

4 Metodología

La metodología por implementar en este proyecto se basa en el cumplimiento de los objetivos establecidos, permitiendo el desarrollo estructurado de cada fase. Para ello, se seguirán una serie de actividades organizadas en un enfoque incremental y basado en la metodología ágil Scrum, lo que permitirá un desarrollo iterativo con validaciones constantes por parte de los usuarios finales.

Las etapas establecidas para la realización de este proyecto son presentadas a continuación, cada etapa desglosa sus actividades.

4.1 Definición y Análisis de Requerimientos

- Identificación de requerimientos funcionales y no funcionales del software.
- Levantar información con entrenadores y administrativos.
- Analizar los procesos actuales de gestión de selecciones deportivas.

4.2 Diseño del Sistema

- Definir la arquitectura de la aplicación basada en la arquitectura hexagonal.
- Diseñar el modelo entidad-relación para la base de datos.
- Diseñar prototipos de la interfaz de usuario mediante mockups.
- Definir estándares de desarrollo y tecnologías a emplear.

4.3 Implementación

- Implementar el backend con Spring Boot.
    - Implementar la API REST para la comunicación de la información de la base de datos y lógica de negocio.
    - Implementar la autenticación con JWT.
- Implementar el frontend con Angular.
    - Implementar de vistas responsivas.
    - Integrar con la API del backend.
- Configurar la base de datos en MySQL.

4.4 Pruebas y Validación

- Elaborar y ejecutar el plan de pruebas.
    - Pruebas unitarias e integración.
    - Pruebas de usabilidad con entrenadores y administradores.

- Identificar y corregir errores.
- Validar el software.

A lo largo de estas etapas, el desarrollo seguirá un enfoque ágil con iteraciones basadas en la planificación de sprint, ofreciendo flexibilidad y adaptación a los requerimientos en evolución del proyecto.

5 Resultados

Los resultados se presentan en función de los objetivos específicos, reflejando el avance técnico del aplicativo desde su definición de requerimientos hasta las pruebas y la validación final. Cada apartado muestra los entregables generados, las evidencias que los respaldan y los ajustes recomendados para fortalecer la solución.

5.1 Definición de requerimientos del aplicativo de selecciones deportivas.

5.1.1 Requerimientos Funcionales (RF)

RF#1 – Gestión de Usuarios y Roles: El aplicativo debe permitir el registro, autenticación y gestión de usuarios con diferentes roles (administrador, entrenador y jugador). Cada rol tendrá permisos específicos sobre las funcionalidades del aplicativo.

RF#2 – Gestión de Selecciones Deportivas: El aplicativo debe permitir registrar, editar, consultar y eliminar selecciones deportivas. Cada selección debe contener información como deporte asociado, nombre, tipo (masculino, femenino o mixto), espacio deportivo y visibilidad. Además, debe permitir asignar integrantes y un entrenador responsable.

RF#3 – Gestión de Jugadores (Integrantes): El aplicativo debe permitir registrar, actualizar, consultar y eliminar la información de los integrantes de las selecciones deportivas, incluyendo datos personales y deportivos como nombre, apellidos, fecha de nacimiento, posición, dorsal, estatura, peso, rol y selección a la que pertenecen.

RF#4 – Gestión de Publicaciones (Noticias y Eventos): El aplicativo debe permitir crear, editar, eliminar y consultar publicaciones clasificadas como noticias o eventos. Cada publicación debe incluir información como título, descripción, lugar, fecha, duración, visibilidad y relación con una o varias selecciones.

RF#5 – Gestión de Horarios por Selección: El aplicativo debe permitir registrar y consultar los horarios de entrenamiento o competencia asociados a cada selección deportiva, incluyendo día y hora.

RF#6 – Gestión de Logros por Selección: El aplicativo debe permitir registrar y consultar los logros deportivos de cada selección, incluyendo nombre del logro, competencia y año.

RF#7 – Gestión de Galería de Fotos: El aplicativo debe permitir cargar, almacenar y asociar fotografías a selecciones deportivas o publicaciones, incluyendo información relacionada como temporada.

RF#8 – Portal Público de Consulta: El aplicativo debe permitir que usuarios no autenticados consulten información pública como selecciones deportivas, noticias y eventos visibles.

5.1.2 Requerimientos No Funcionales (RNF)

RNF#1 – Seguridad: El aplicativo debe garantizar el acceso controlado a las funcionalidades según el rol del usuario, protegiendo la información y evitando accesos no autorizados.

RNF#2 – Usabilidad y Accesibilidad: El aplicativo debe ofrecer una interfaz intuitiva, fácil de usar y accesible desde distintos dispositivos, garantizando una navegación clara.

RNF#3 – Rendimiento y Escalabilidad: El aplicativo debe ser capaz de manejar múltiples consultas de usuarios sin afectar significativamente los tiempos de respuesta, permitiendo su crecimiento futuro.

5.1.3 Requerimientos Técnicos (RT)

RT#1 – Tecnologías del aplicativo: El aplicativo debe estar basado en tecnologías web modernas que permitan su desarrollo, mantenimiento y escalabilidad.

RT#2 – Arquitectura del aplicativo: El aplicativo debe implementar una arquitectura que permita la separación de responsabilidades, facilitando la mantenibilidad, escalabilidad y evolución del software.

5.2 Diseño funcional y técnico de la aplicación

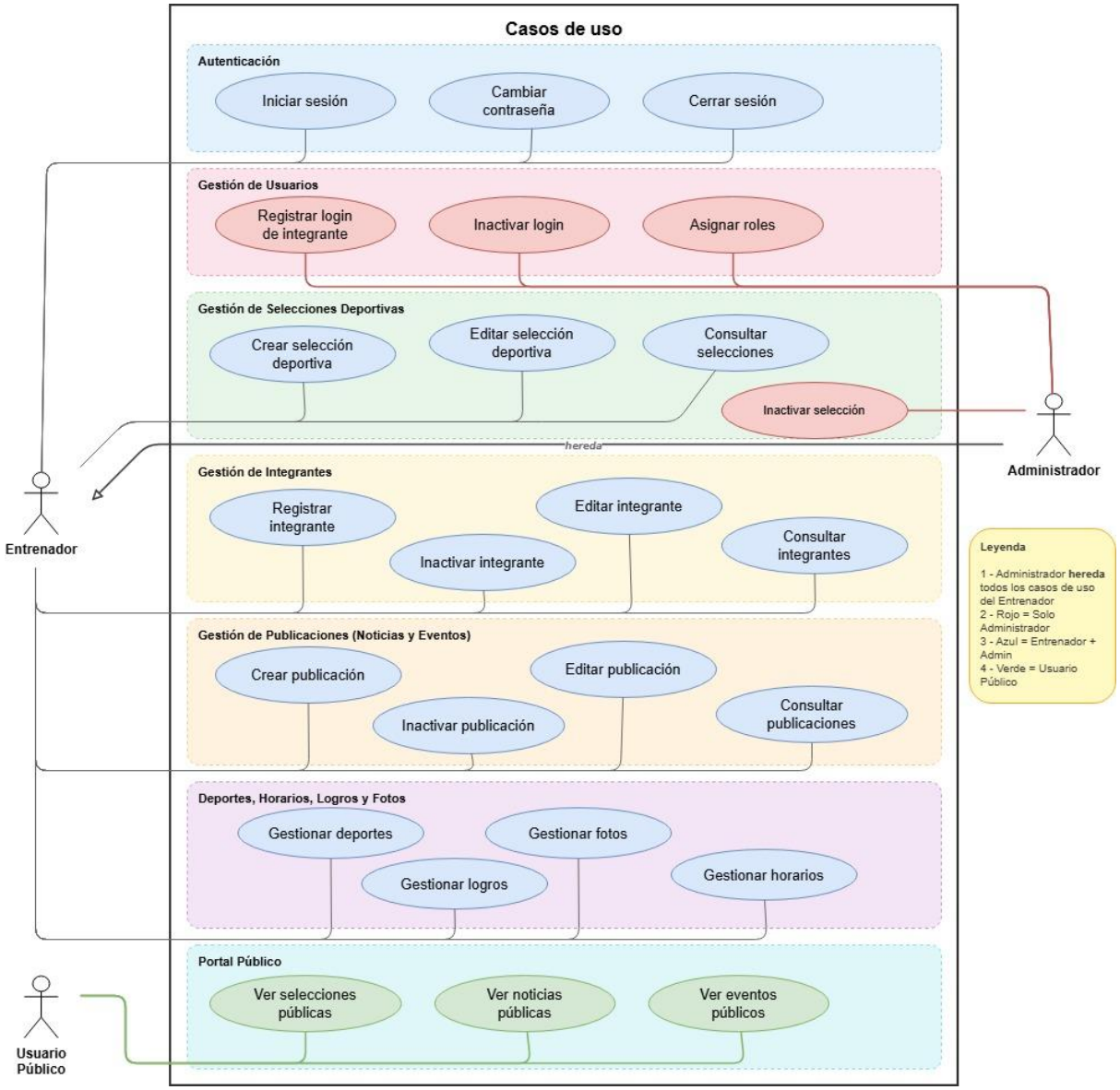
El diseño arquitectónico de la aplicación se desarrolló en cumplimiento del Objetivo Específico 2, cuyo propósito fue definir las especificaciones funcionales y técnicas de la plataforma "Selecciones Deportivas UIS", abarcando su alcance, organización interna, almacenamiento de datos y mecanismos de visualización.

El resultado fue un aplicativo estructurado bajo principios de arquitectura hexagonal, complementada con un enfoque de vertical slicing por módulo funcional, el uso de CQRS (Command–Query Responsibility Segregation) para la separación lógica de responsabilidades, y una base de datos relacional unificada en MySQL.

Como punto de partida para el diseño, se presenta el diagrama de casos de uso del aplicativo (Figura 2), el cual consolida los requerimientos funcionales definidos en la sección anterior e identifica los tres actores principales del aplicativo: el Administrador, el Entrenador y el Usuario Público, junto con sus interacciones con las funcionalidades de la plataforma.

Figura 2

Diagrama de casos de uso



Nota: Diagrama realizado en Draw.io

5.2.1 Arquitectura hexagonal

El aplicativo adopta la arquitectura Hexagonal, donde la lógica de negocio se encuentra completamente desacoplada de los mecanismos externos de persistencia, interfaz o comunicación. Esta arquitectura permite que el dominio de la aplicación se mantenga

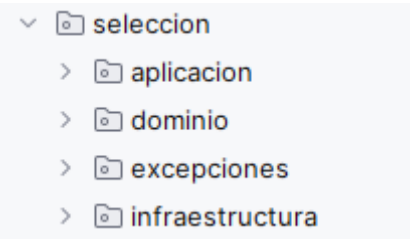
independiente del framework, facilitando la extensibilidad y la prueba unitaria de cada componente, en la Figura 3 se muestra la organización de estas capas dentro del proyecto.

Cada módulo posee tres capas principales:

1. Dominio: define las entidades, objetos de valor y las reglas del negocio.
2. Aplicación: implementa los casos de uso o flujos de la lógica del negocio (commands y queries).
3. Infraestructura: contiene los adaptadores externos, controladores REST, repositorios JPA y configuraciones técnicas (seguridad, persistencia, mapeadores, etc.).

Figura 3

Capas de la arquitectura hexagonal



Nota: Pantallazo tomado de IntelliJ IDEA.

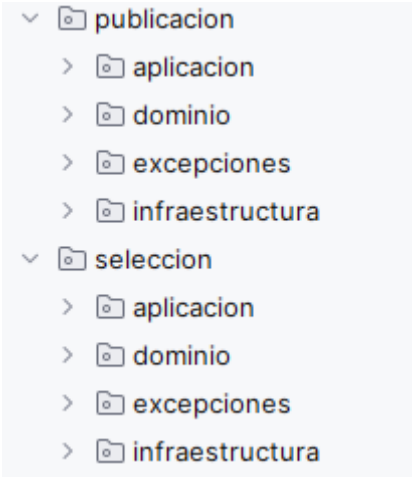
5.2.2 Vertical Slicing

A diferencia del enfoque horizontal tradicional (una carpeta por capa), se adoptó el principio de vertical slicing, que organiza el código por módulos funcionales, en la Figura 4 se muestra cómo cada módulo contiene su propia estructura hexagonal.

Cada módulo (por ejemplo, auth, selecciones, publicaciones, integrantes, fotos, deportes, logros, horarios) contiene internamente sus capas (dominio, aplicación, infraestructura), lo que permite una organización por funcionalidad del negocio y facilita la mantenibilidad y las pruebas.

Figura 4

Módulos que contienen las capas de la arquitectura hexagonal



Nota: Pantallazo tomado de IntelliJ IDEA.

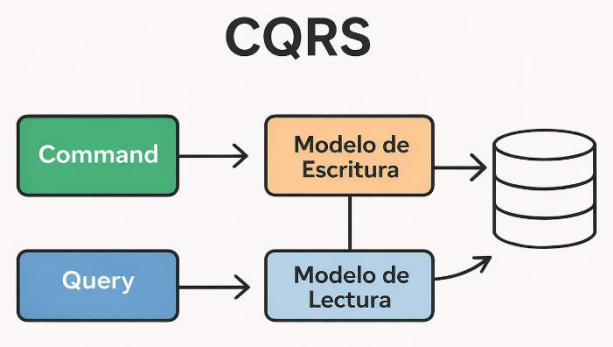
5.2.3 Aplicación del patrón CQRS

El diseño implementa CQRS de manera lógica dentro de cada módulo, separando las operaciones de lectura (Query) y escritura (Command), en la Figura 5 se muestra una ilustración del patrón CQRS.

1. Los servicios Command (SeleccionCommandService, PublicacionCommandService, IntegranteCommandService) se encargan de las operaciones que modifican el estado del aplicativo: creación, actualización o eliminación de entidades.
2. Los servicios Query (SeleccionQueryService, PublicacionQueryService, etc.) realizan consultas optimizadas, aplicando filtros, paginación y proyecciones DTO. Ambos servicios acceden a una única base de datos MySQL, pero mantienen responsabilidades distintas para facilitar el mantenimiento y la claridad del código.

Figura 5

Patrón de diseño CQRS



Nota: Diagrama realizado en Canvas

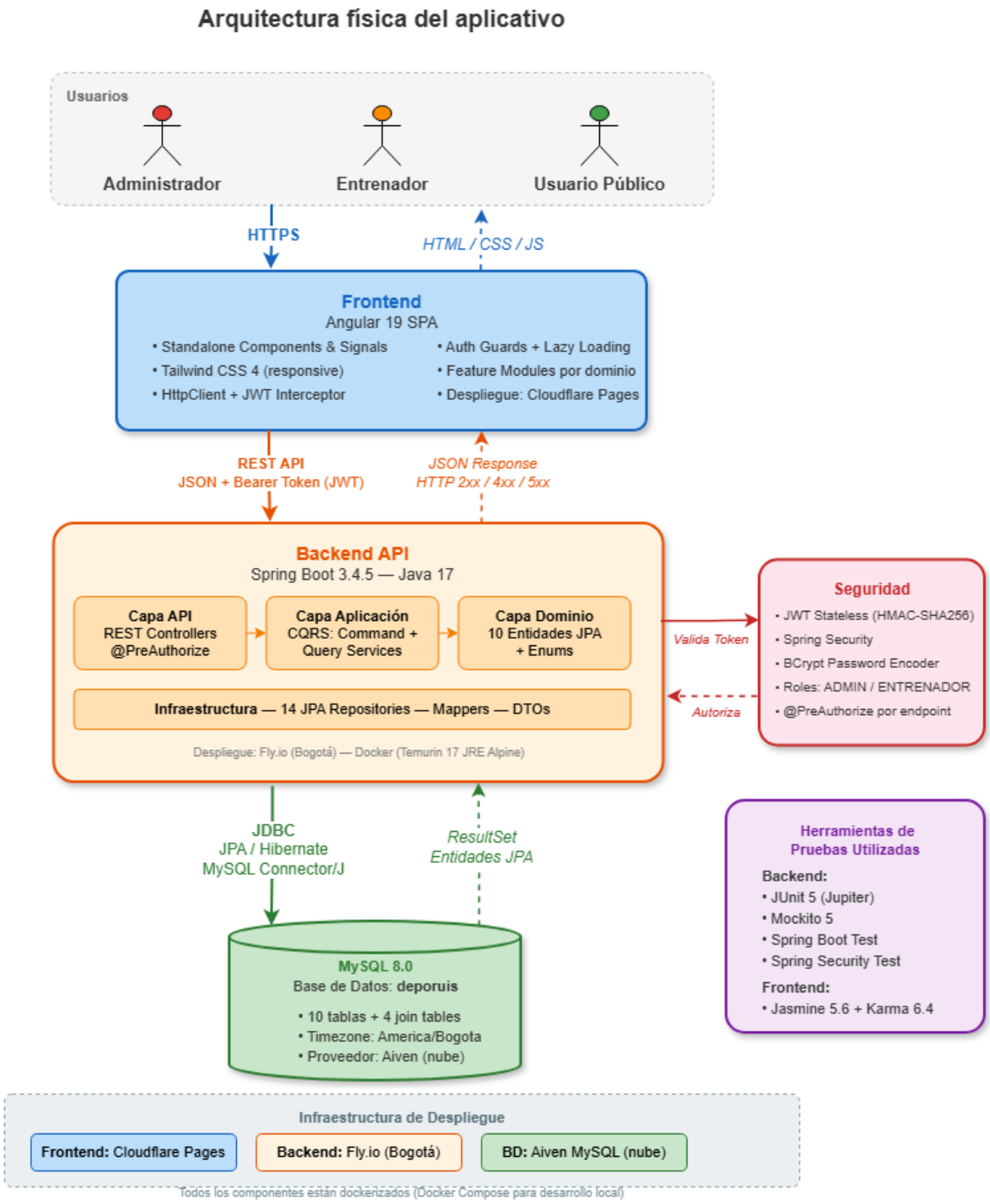
5.2.4 Arquitectura física del aplicativo

El aplicativo se compone de tres contenedores principales, en la Figura 6 se presenta el diagrama de la arquitectura física compuestas por los siguientes tres elementos:

1. Frontend (Angular 19/20): Aplicación SPA modularizada por funcionalidades. Usa componentes *standalone*, signals para manejo de estado y Tailwind CSS 4 para el diseño visual. Cada módulo (auth, selecciones, publicaciones, integrantes) interactúa con el backend mediante peticiones HTTP autenticadas con JWT.
2. Backend (Spring Boot 3.4.x): Núcleo del aplicativo, implementado con arquitectura hexagonal y vertical slicing. Contiene casos de uso distribuidos por módulo y la capa de seguridad basada en Spring Security + JWT.
3. Base de Datos (MySQL 8): Única fuente de persistencia para todos los módulos. Modelo relacional normalizado con claves foráneas y restricciones de integridad referencia.

Figura 6

Diagrama de arquitectura física



Nota: Diagrama realizado en Draw.io

5.2.5 Modelo de datos y persistencia

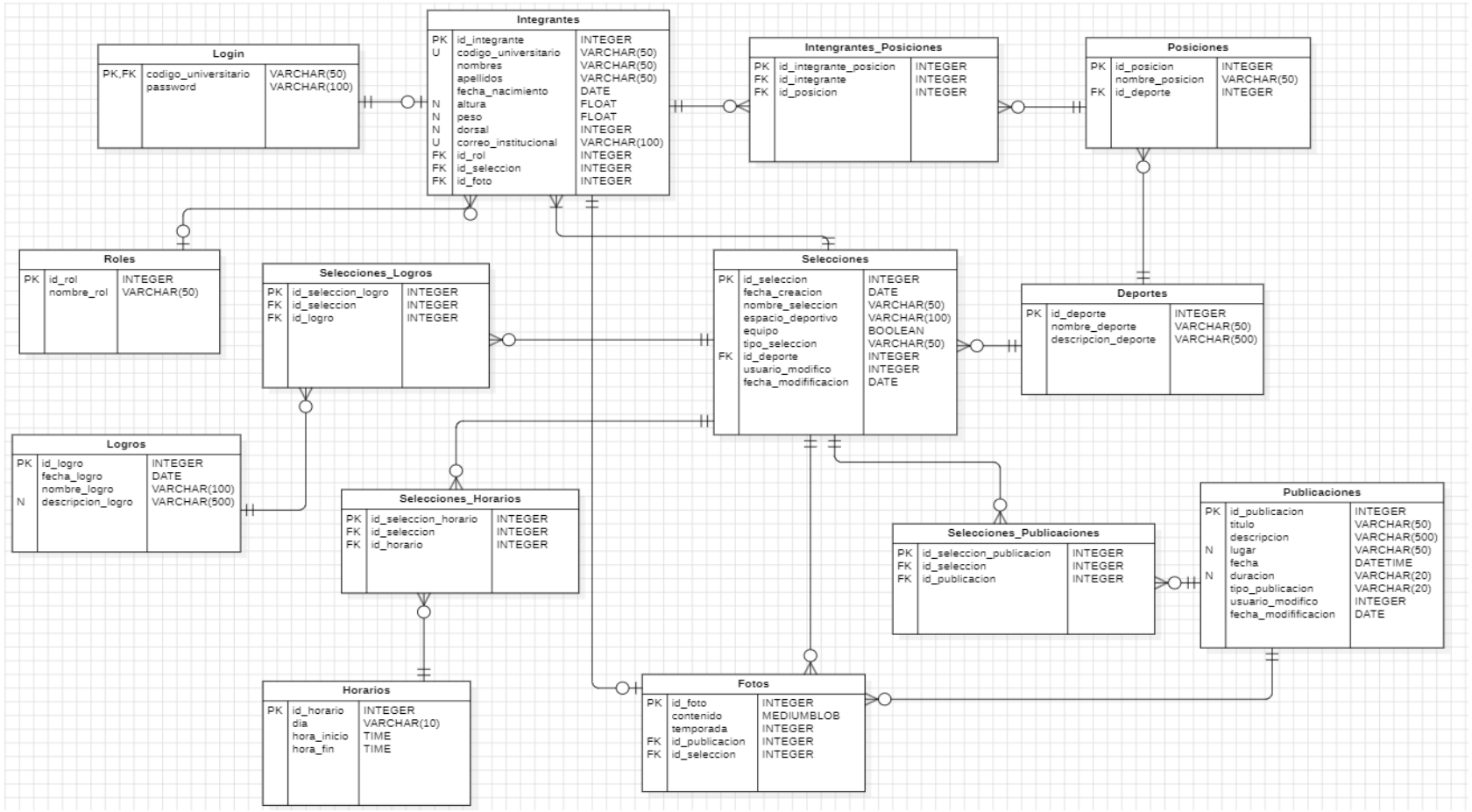
El modelo Entidad–Relación (ERD) fue diseñado como reflejo directo de la estructura modular del dominio, agrupando las entidades del aplicativo en torno a la selección deportiva como eje central. Se priorizó la normalización del esquema para mantener la consistencia de los datos y eliminar redundancia, así como la definición de restricciones de integridad referencial mediante claves foráneas, en la Figura 7 se presenta el diagrama de entidad–relación del aplicativo.

La base de datos contempla relaciones de uno a muchos (1–N) y de muchos a muchos (N–N), estas últimas implementadas mediante tablas intermedias. Las relaciones principales del modelo son:

1. Deporte (1–N) Selección: un deporte agrupa una o más selecciones deportivas.
2. Selección (1–N) Integrante: cada selección contiene múltiples integrantes (jugadores y entrenador).
3. Selección (N–N) Publicación: una selección puede estar asociada a varias publicaciones (noticias o eventos) y viceversa.
4. Selección (N–N) Logro: los logros deportivos pueden vincularse a una o más selecciones.
5. Selección (N–N) Horario: los horarios de entrenamiento pueden ser compartidos entre selecciones.
6. Publicación (1–N) Foto: cada publicación puede contener múltiples fotografías asociadas.

Figura 7

Diagrama entidad relación



Nota: Diagrama realizado en StarUML.

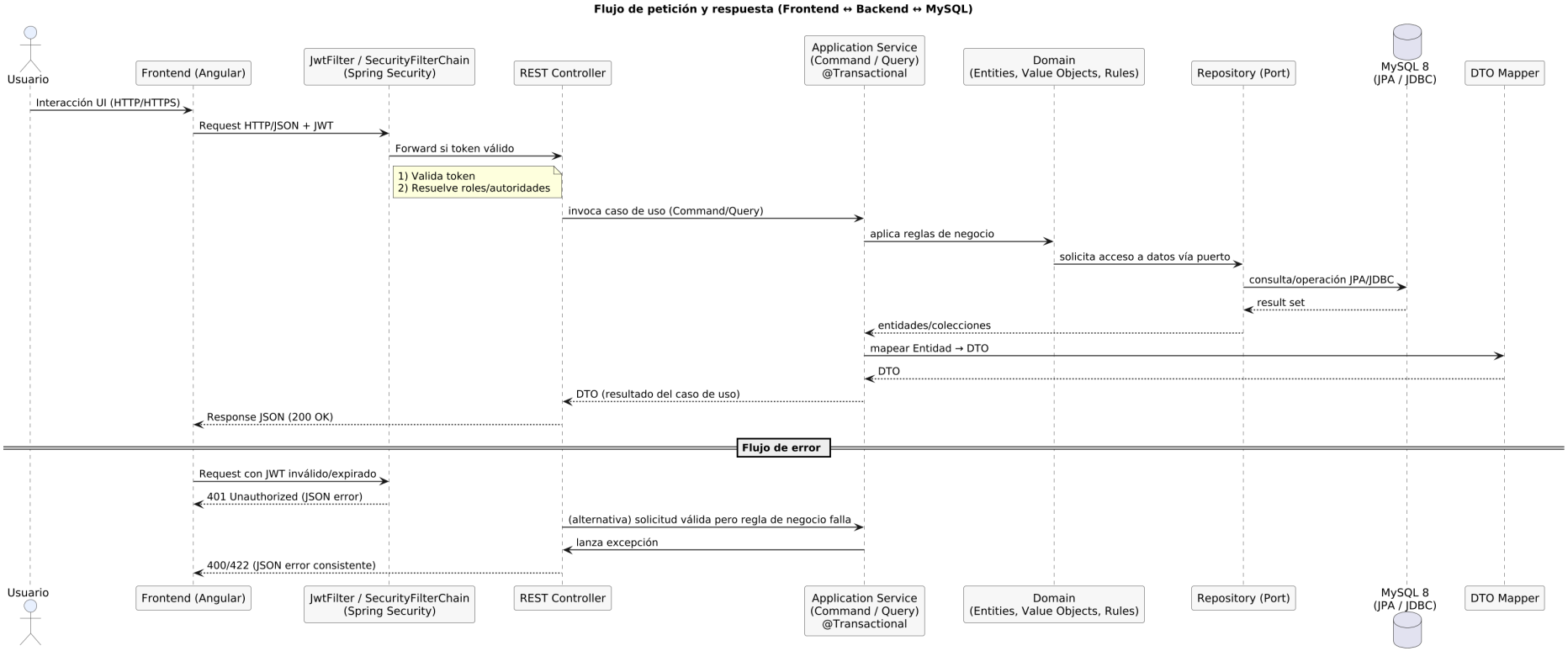
5.2.6 Flujo interno del backend

El flujo interno del backend inicia cuando el controlador REST recibe la solicitud HTTP desde el frontend y, según la operación requerida, invoca un servicio de tipo Command o Query. Este servicio utiliza los puertos definidos en el dominio, los cuales son implementados por los adaptadores de infraestructura mediante el uso de JPA o consultas personalizadas. Una vez procesada la operación, el resultado es transformado y devuelto al frontend en forma de DTO, en la Figura 8 se muestra el flujo de petición y respuesta, incluyendo la validación del token JWT, la ejecución de los casos de uso y la interacción con la base de datos.

En la práctica, este flujo puede observarse al crear una publicación: el frontend en Angular envía una petición POST al endpoint correspondiente, el backend valida el token, ejecuta el método PublicacionCommandService#createPublicacion(), persiste la información en la base de datos MySQL y retorna el DTO generado, tras lo cual el frontend actualiza la vista reflejando los cambios en tiempo real.

Figura 8

Diagrama de secuencia de la arquitectura del backend



Nota: Diagrama realizado en PlantUML

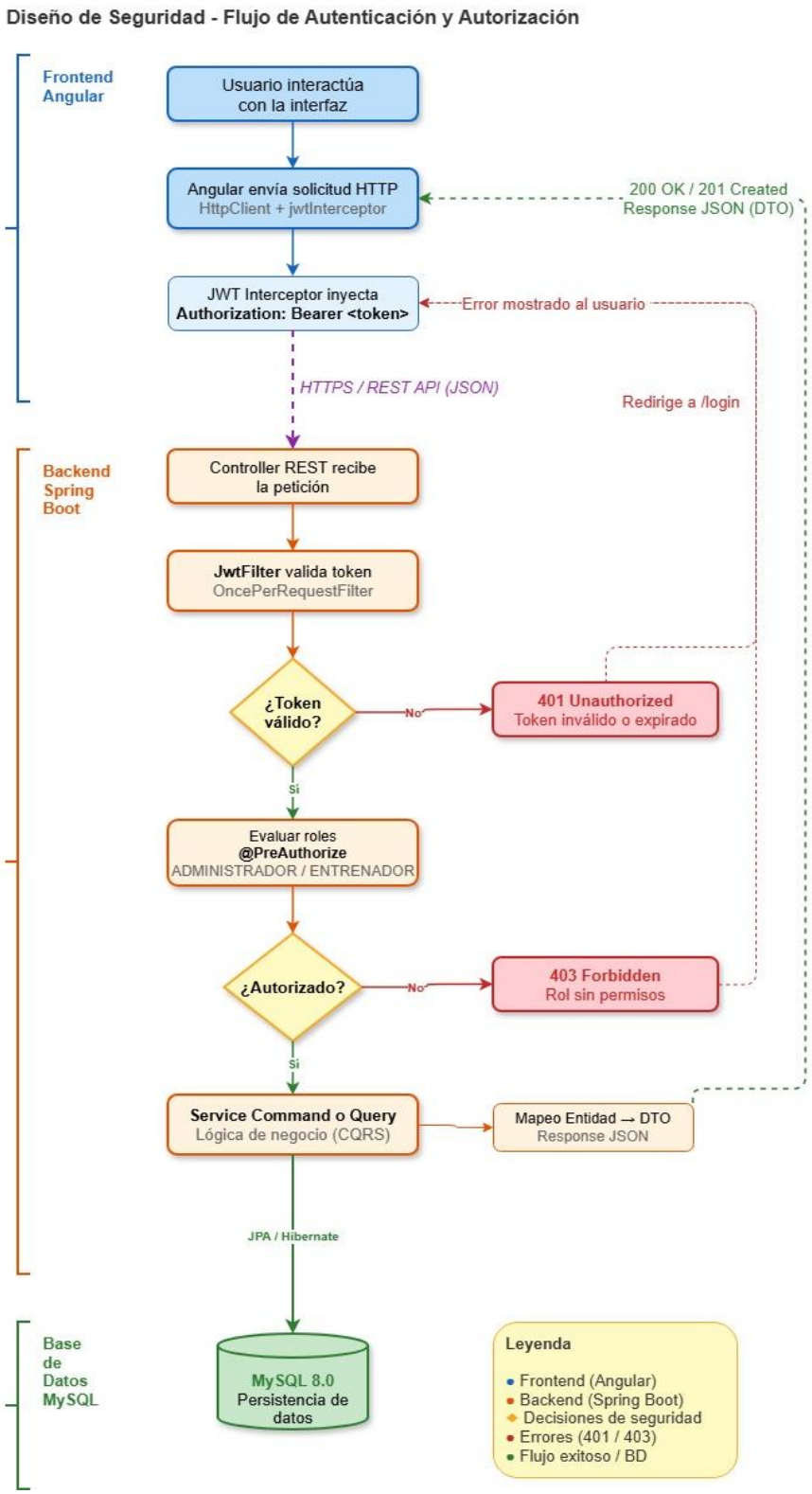
5.2.7 Diseño de seguridad

El componente de autenticación y autorización se implementó en el módulo auth, integrando mecanismos basados en JWT y Spring Security. Este módulo se compone de varios elementos clave, entre los que se encuentran el AuthController, encargado de manejar las solicitudes de inicio de sesión y generar los tokens; el JwtFilter, que intercepta las peticiones HTTP y valida la autenticidad del token; la configuración de SecurityFilterChain, que define las rutas públicas y protegidas del aplicativo; y el BCryptPasswordEncoder, utilizado para el cifrado de contraseñas. Asimismo, se definieron roles de acceso como *administrador* y *entrenador*, los cuales determinan los permisos dentro de la aplicación, en la Figura 9 se muestra el flujo de autenticación y autorización, incluyendo la validación del token y el control de acceso basado en roles.

En la práctica, cuando un usuario ingresa sus credenciales, el aplicativo valida su identidad, genera un token de autenticación y lo redirige al panel correspondiente según su rol; en caso de que el token sea inválido, el aplicativo redirige automáticamente al inicio de sesión.

Figura 9

Diagrama de flujo del proceso de autenticación y autorización con JWT



Nota: Diagrama realizado en Draw.io

5.2.8 Diseño del frontend

El frontend implementa un enfoque modular alineado con la arquitectura del backend, donde cada módulo de Angular encapsula sus componentes, servicios y rutas, manteniendo independencia lógica y facilitando la escalabilidad del aplicativo. En cuanto al diseño de la interfaz de usuario, se definió una paleta de colores basada en la identidad institucional de la UIS, utilizando fondos tipo slate y un color primario de acento. Asimismo, se desarrollaron componentes reutilizables como botones, campos de entrada, modales, paginadores y elementos de carga, lo que contribuye a la consistencia visual de la aplicación. La navegación se encuentra controlada mediante mecanismos como AuthGuard, garantizando el acceso restringido a las rutas protegidas, mientras que el estado de la aplicación es gestionado mediante el uso de signals, lo que permite una mayor reactividad. Finalmente, se empleó Tailwind CSS 4 para la construcción de una interfaz responsiva adaptable a distintos dispositivos.

A continuación, se presentan los mockups de las principales interfaces del aplicativo. Estos mockups permiten visualizar la estructura, distribución de componentes y flujo de interacción del usuario dentro del aplicativo. Cabe resaltar que, al tratarse de prototipos de diseño, las imágenes son únicamente representativas y pueden presentar variaciones respecto a la implementación final, especialmente en aspectos visuales o de detalle, sin afectar la funcionalidad descrita. Las interfaces incluidas se describen a continuación:

1. Figura 10 y 11: Pantalla de inicio del aplicativo, donde se presenta la información general, selecciones destacadas, eventos y noticias recientes.
2. Figura 12: Vista pública de selecciones deportivas, que permite explorar las diferentes selecciones disponibles.
3. Figura 13: Vista pública de noticias, que muestra las publicaciones informativas del aplicativo.

4. Figura 14: Vista pública de eventos deportivos, donde se listan los eventos programados.
5. Figura 15: Pantalla de inicio de sesión, utilizada para la autenticación de usuarios.
6. Figura 16: Dashboard administrativo, donde se visualizan métricas generales y actividad reciente del aplicativo.
7. Figura 17 y 18: Gestión de selecciones deportivas, incluyendo el listado y la creación de nuevas selecciones.
8. Figura 19 y 20: Gestión de eventos deportivos, incluyendo el listado y el formulario de creación.
9. Figura 21 y 22: Gestión de noticias, incluyendo el listado y la creación de publicaciones.
10. Figura 23: Gestión de usuarios del aplicativo, donde se administran los roles y la información de los usuarios.

Figura 10

Mockup pantalla de inicio

Figura 11

Mockup scroll pantalla de inicio

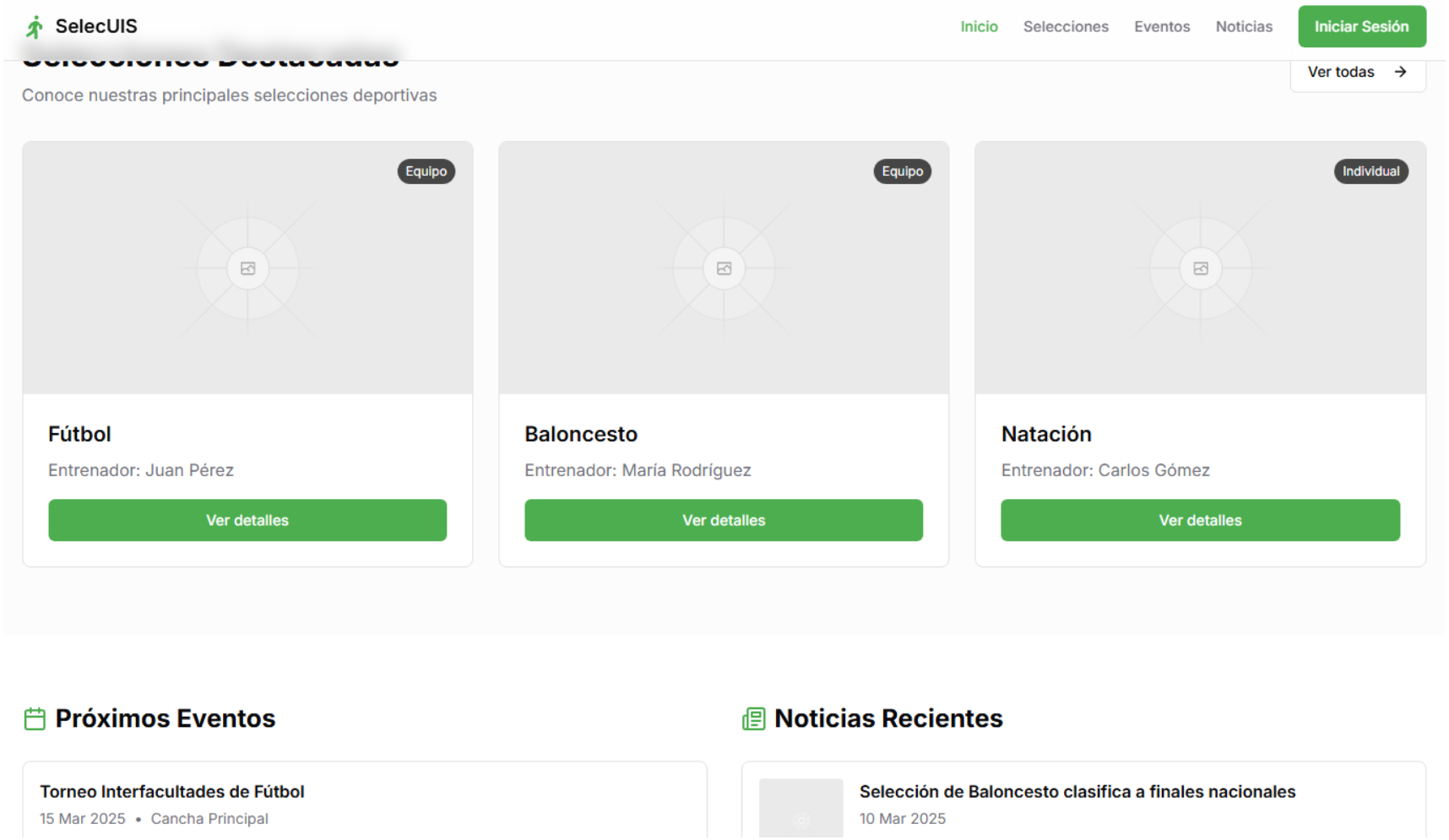


Figura 12

Mockup pantalla pública de selecciones

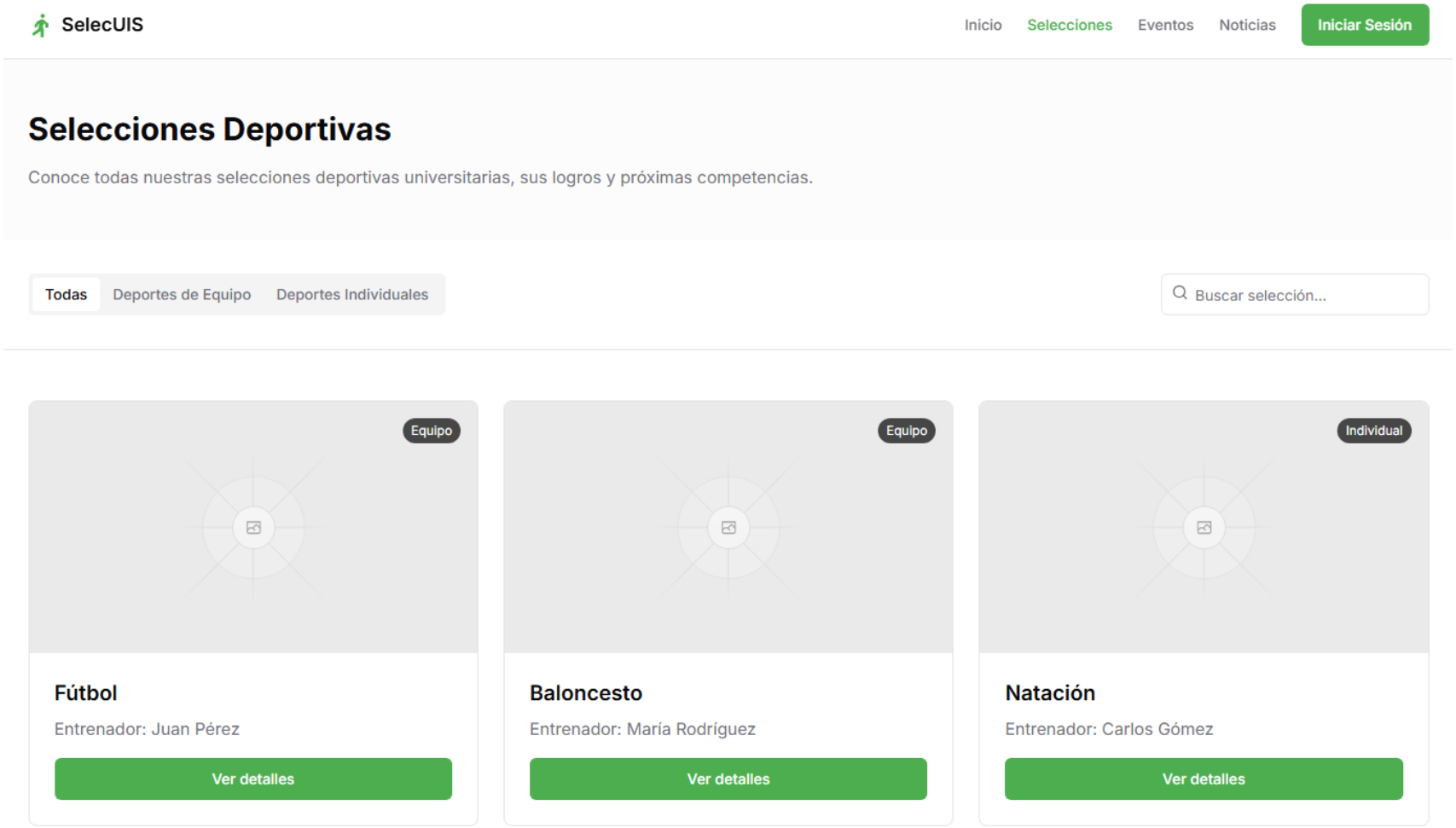


Figura 13

Mockup pantalla pública de noticias

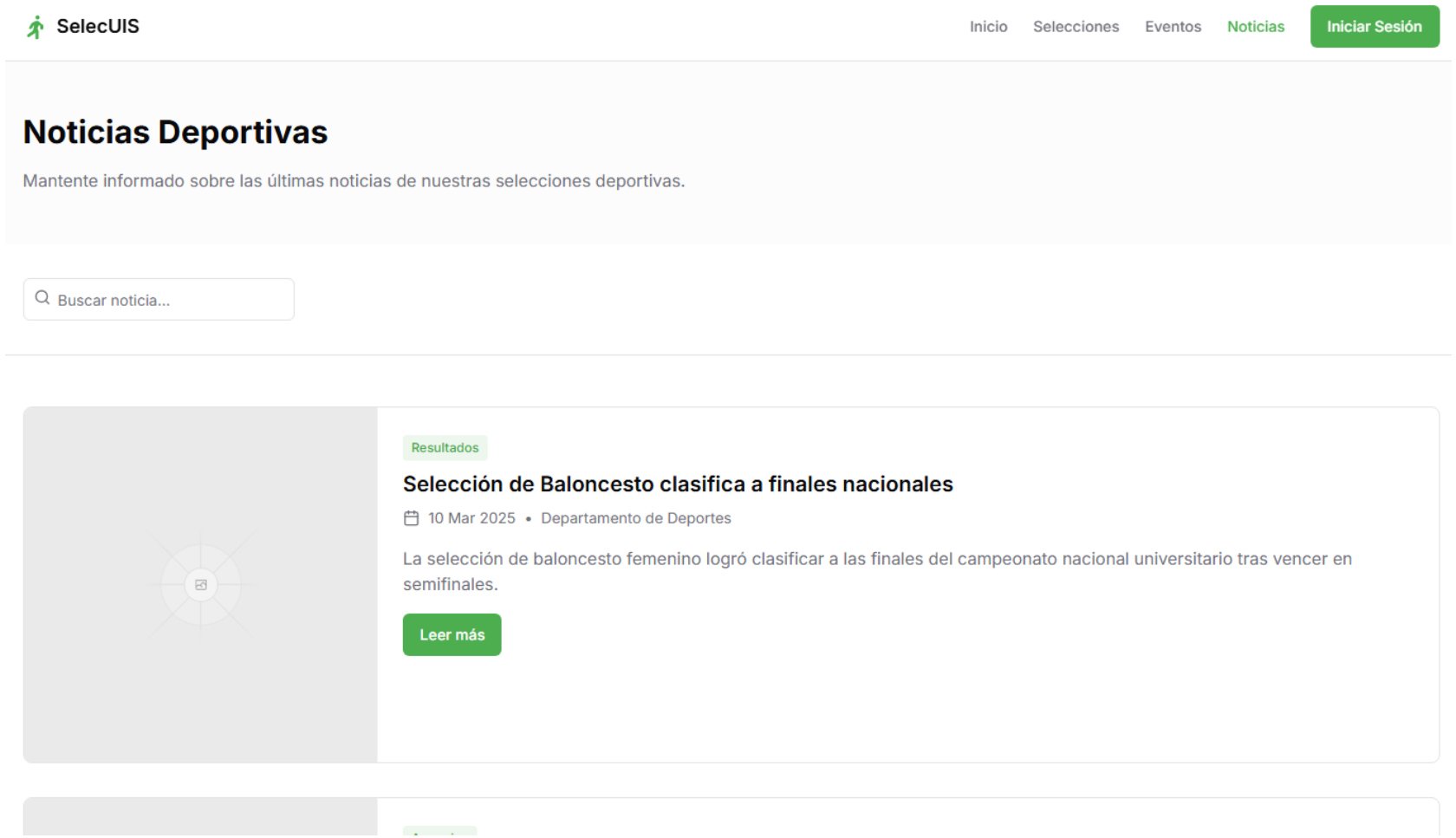


Figura 14

Mockup pantalla pública de eventos

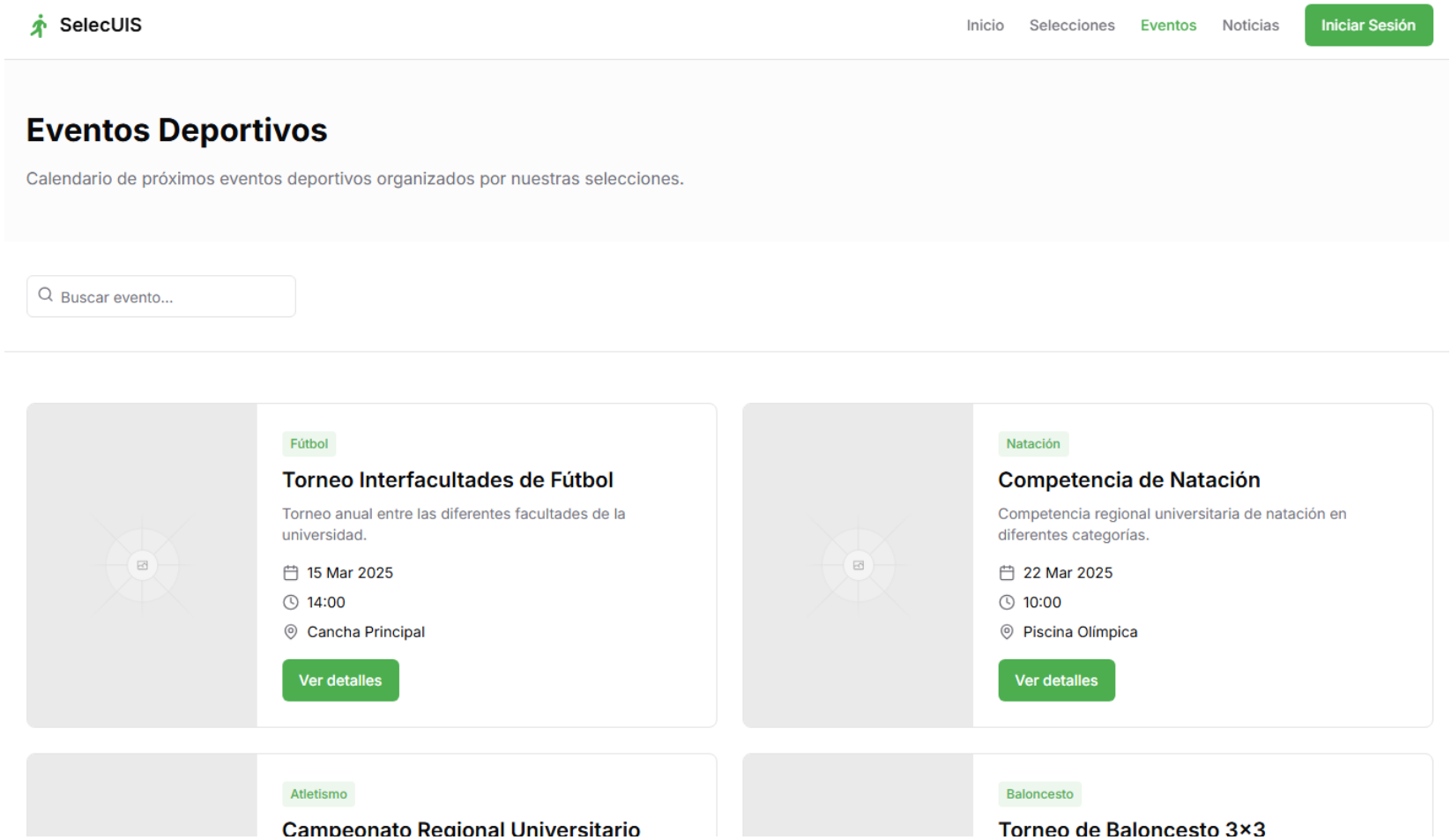


Figura 15

Mockup pantalla de inicio de sesión

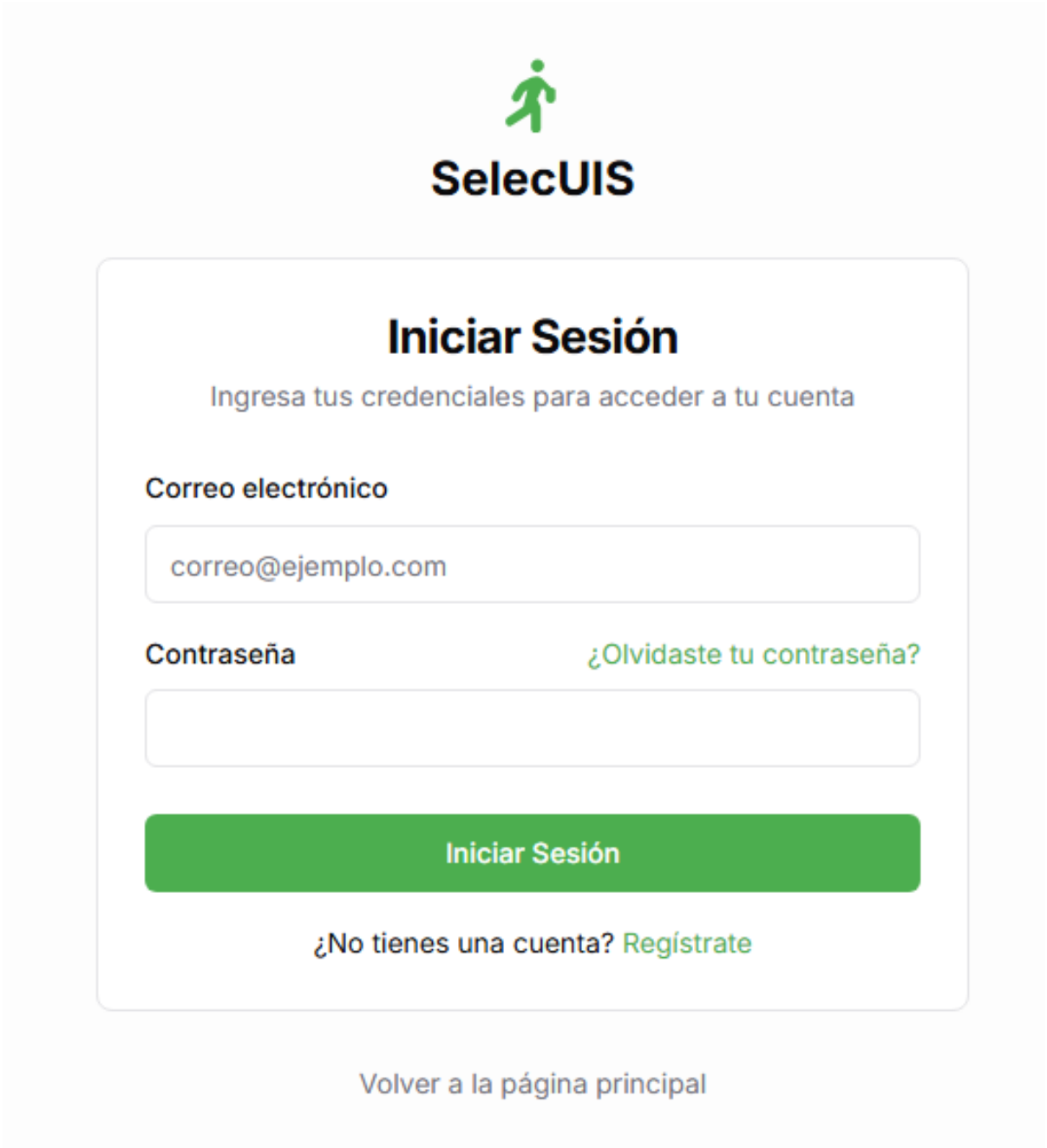


Figura 16

Mockup pantalla del dashboard

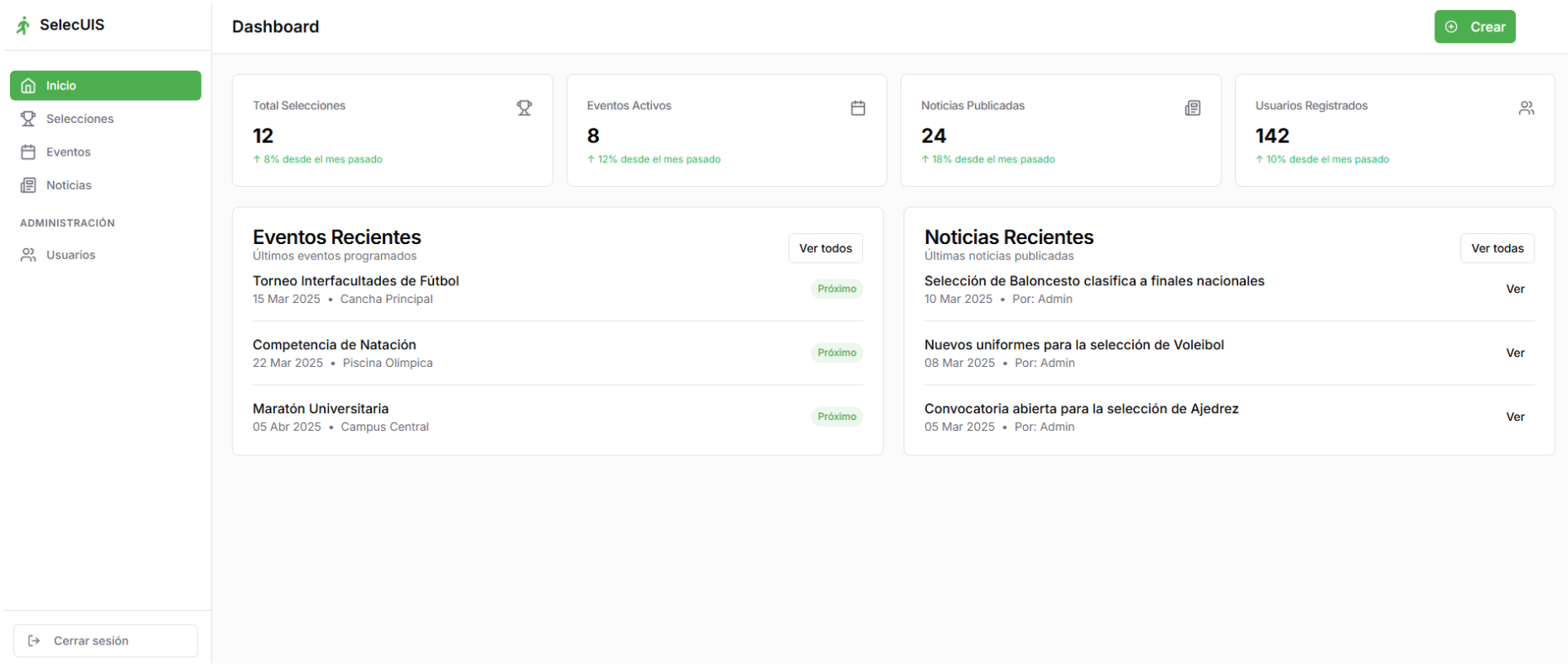


Figura 17

Mockup pantalla gestión de selecciones deportivas

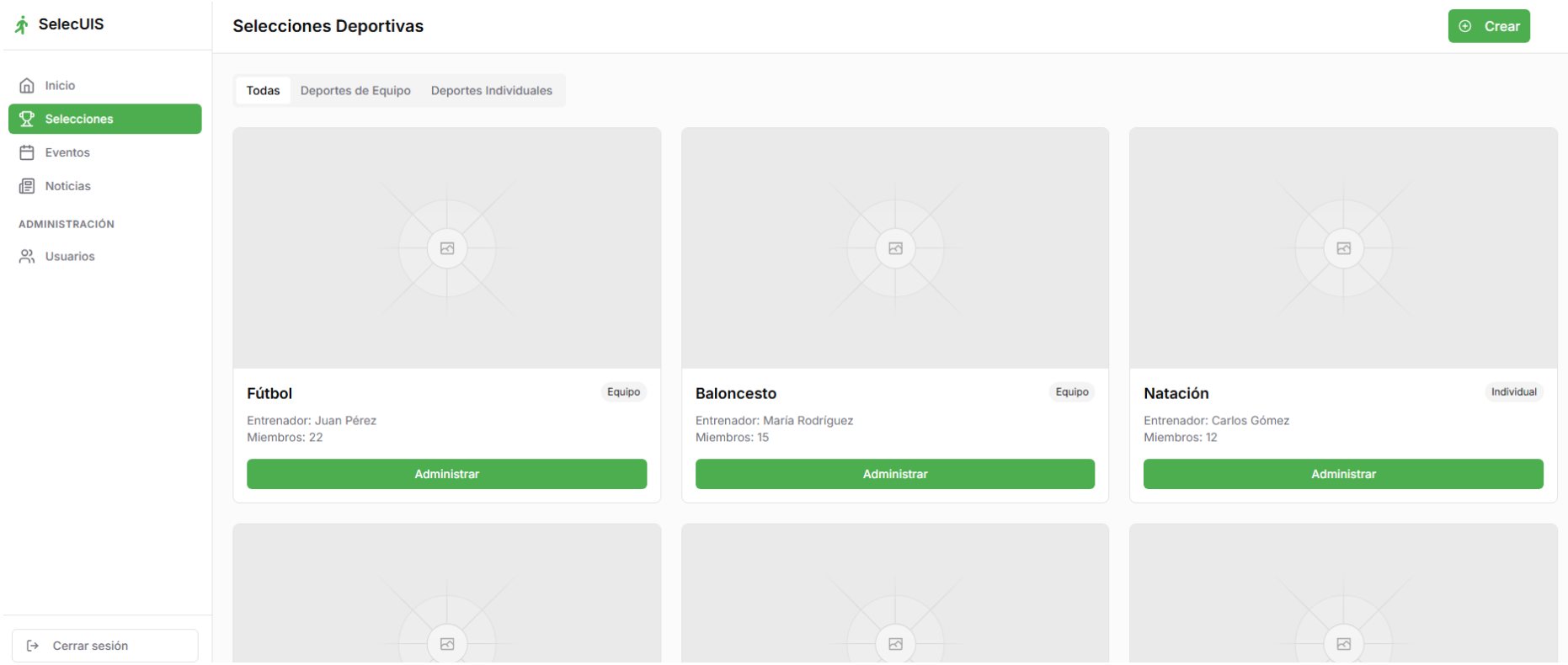


Figura 18

Mockup formulario para crear una selección

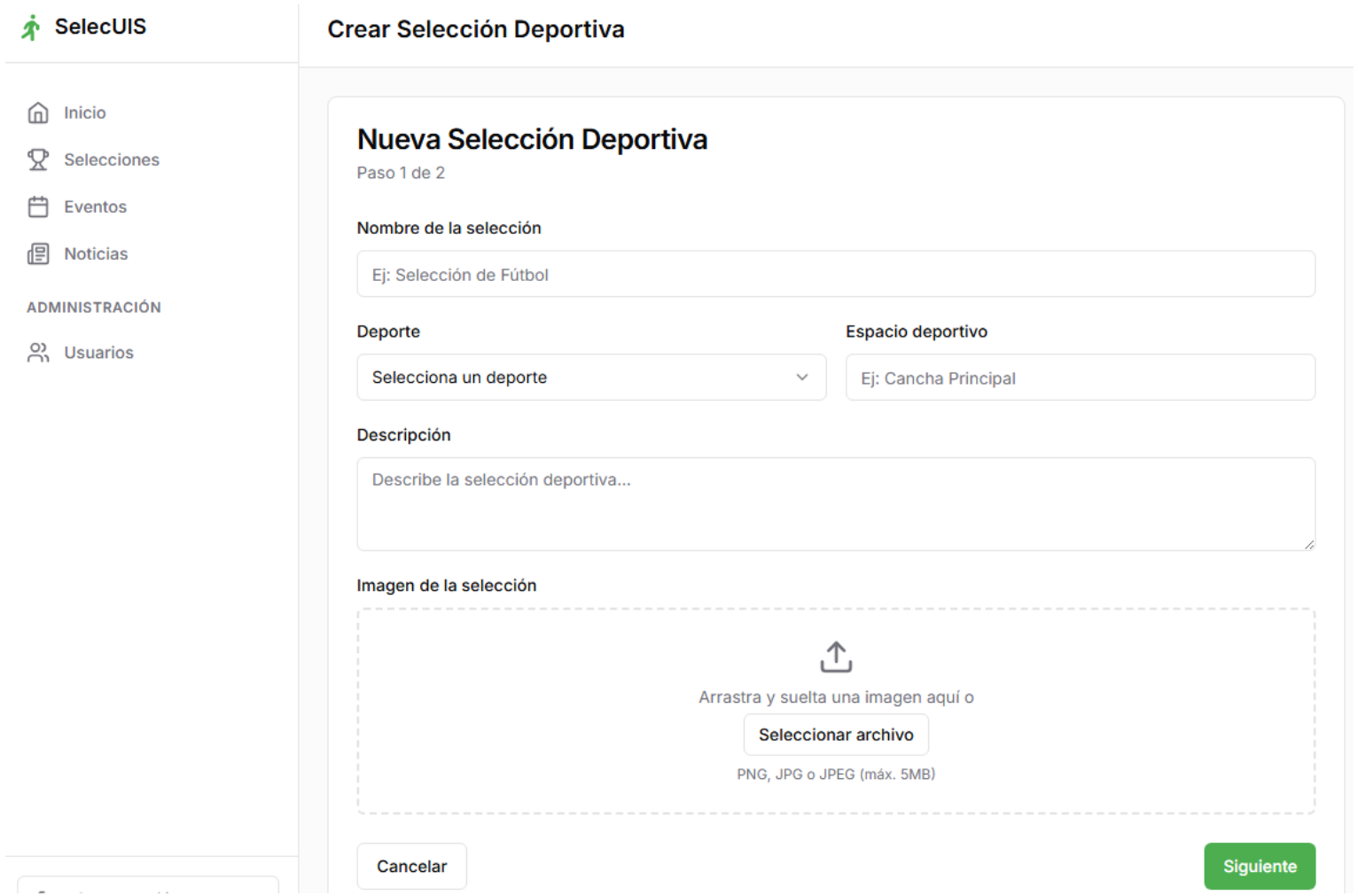


Figura 19

Mockup pantalla gestión de eventos deportivos

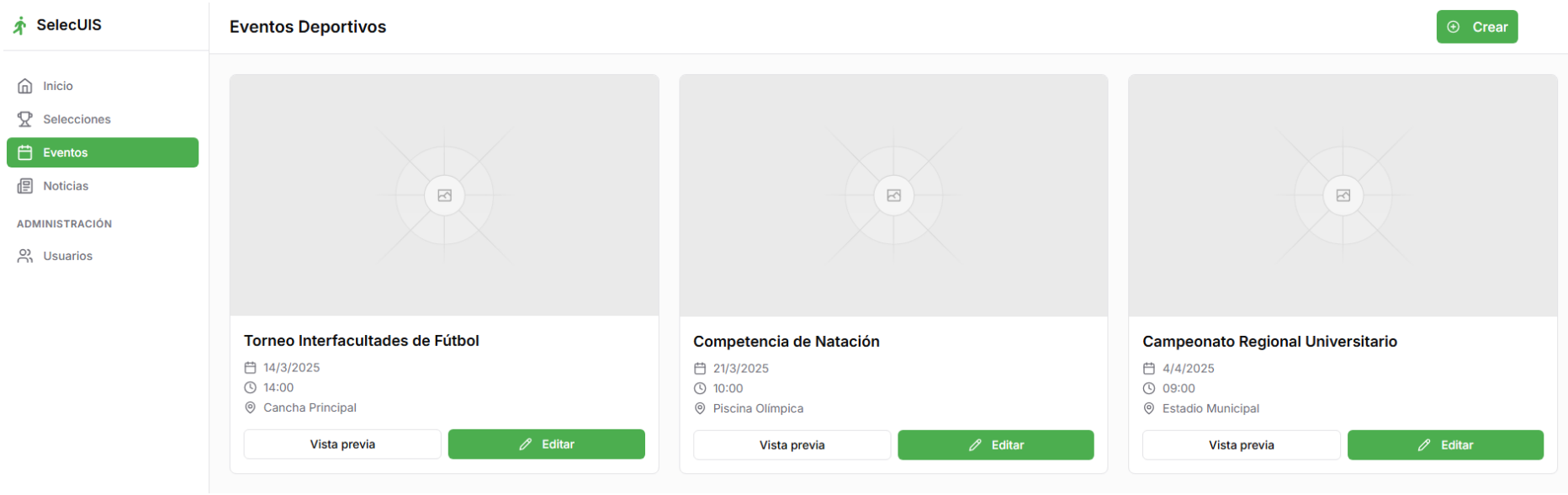


Figura 20

Mockup formulario para crear un evento

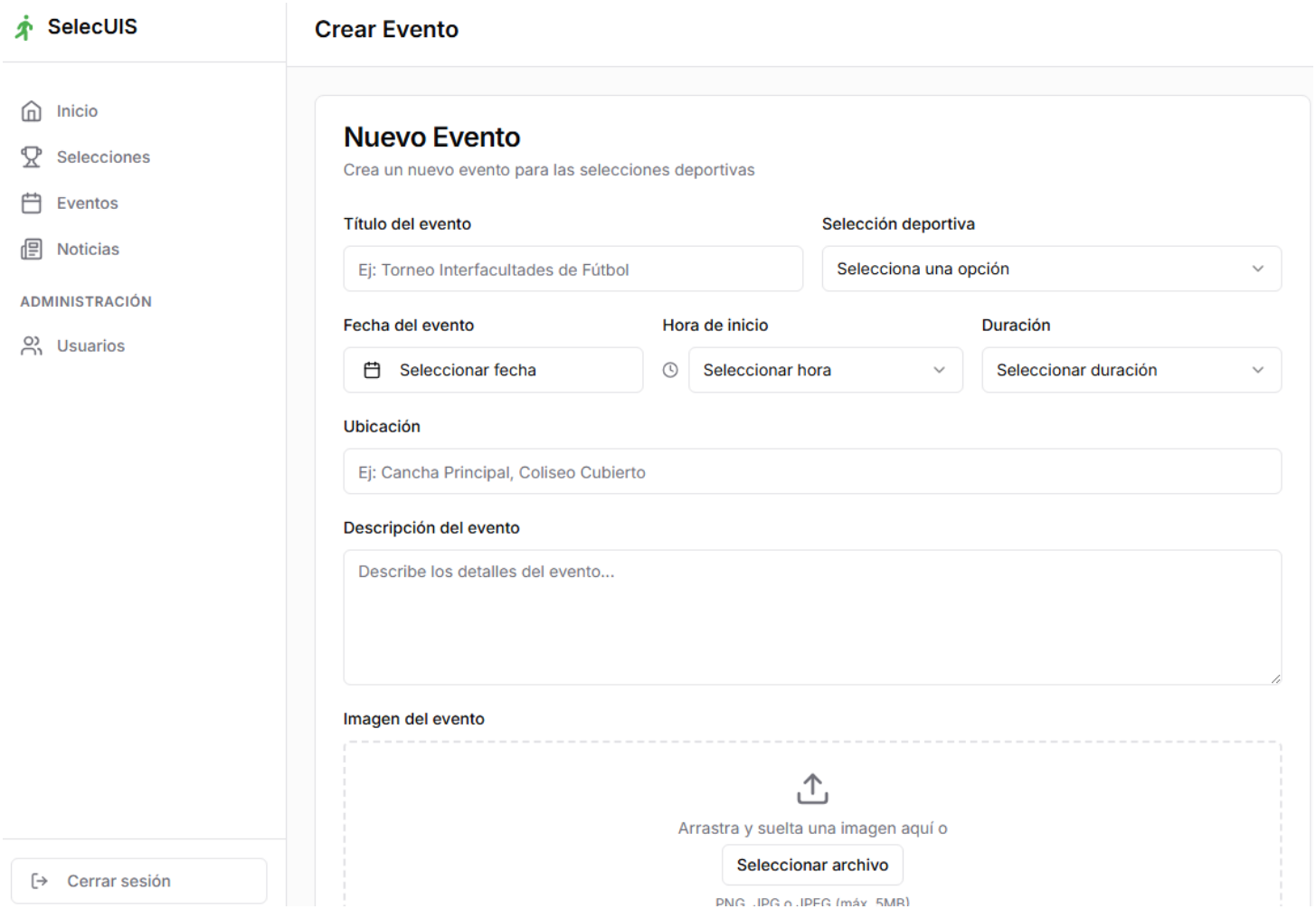


Figura 21

Mockup pantalla gestión de noticias deportivas

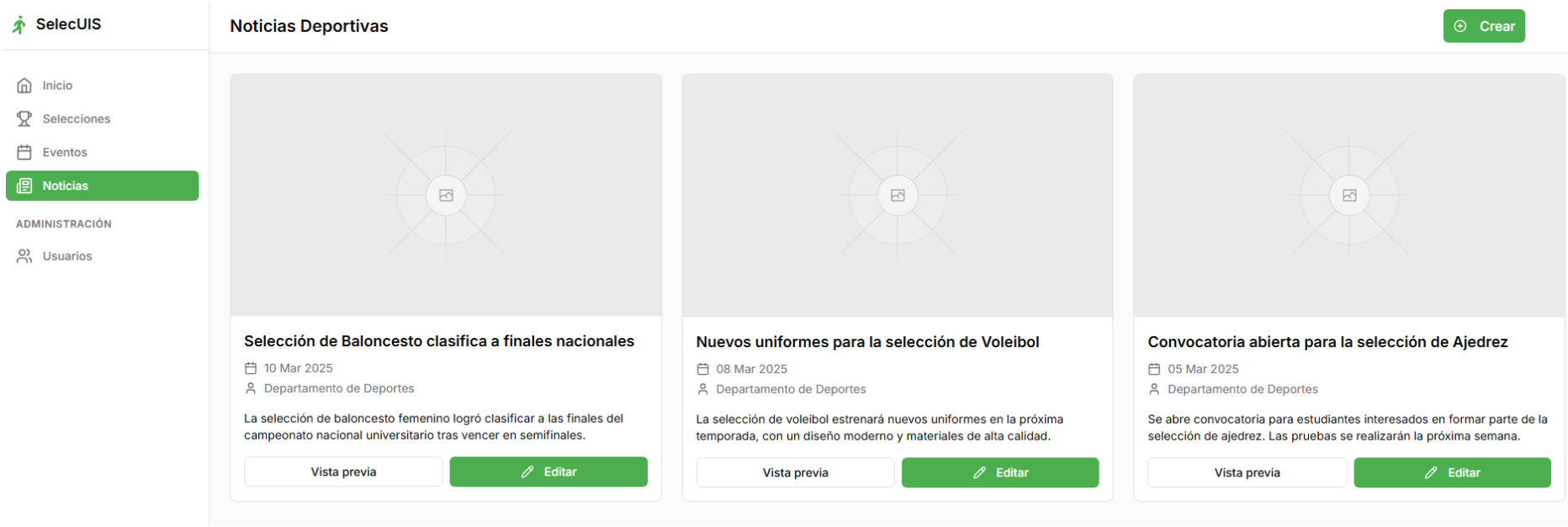


Figura 22

Mockup formulario para crear una noticia

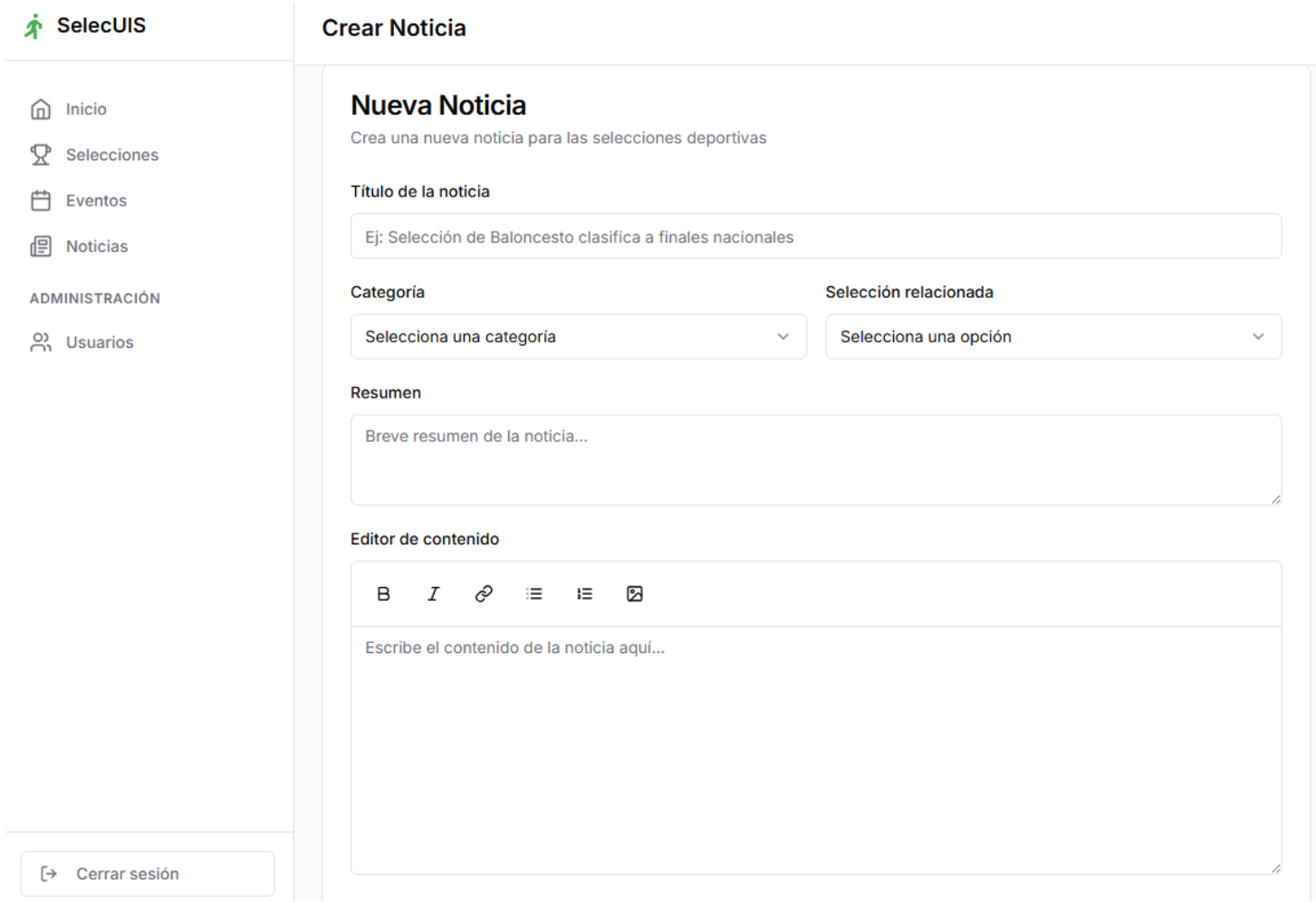
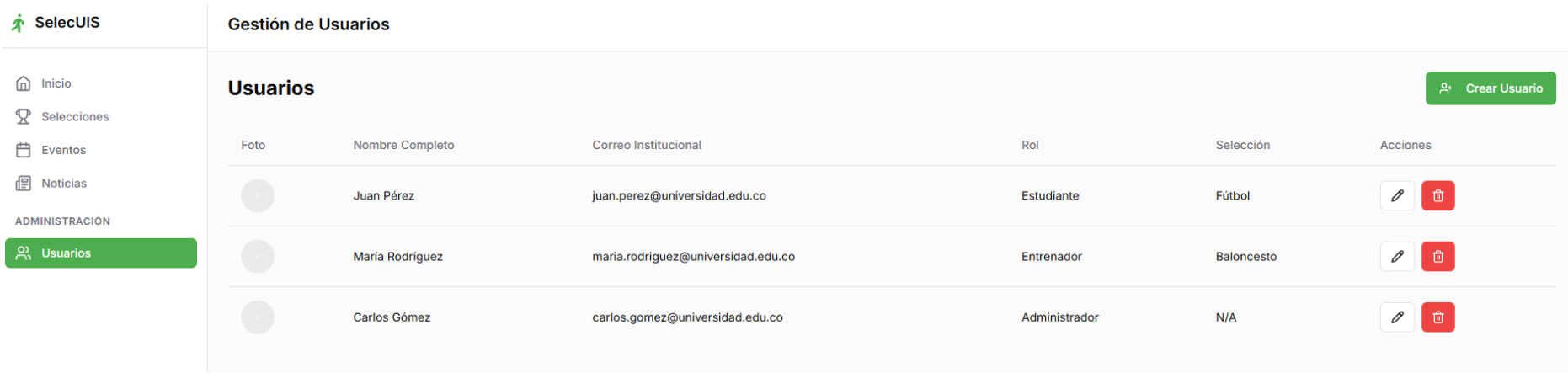


Figura 23

Mockup pantalla de gestión de usuarios

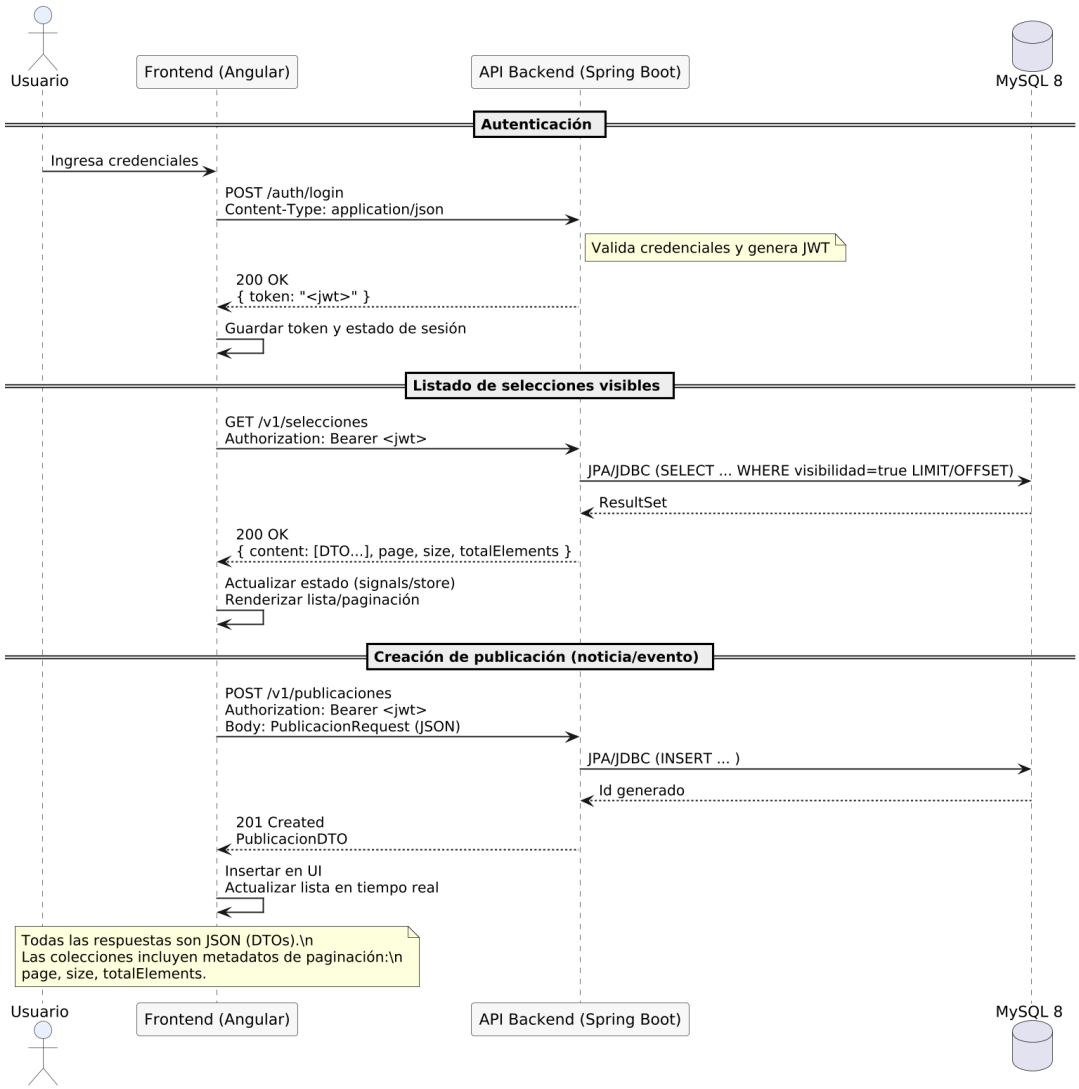


5.2.9 Integración y comunicación

La comunicación entre frontend y backend se realiza mediante API-REST y mensajes en formato JSON, en la Figura 24 se ilustra el flujo de interacción entre el frontend, el backend y la base de datos, incluyendo procesos como la autenticación, la consulta de información y la creación de registros. Ejemplo de endpoint: POST /auth/login – autenticación y generación de token.

Figura 24

Diagrama de secuencia integración y comunicación



Nota: Diagrama realizado en PlantUML

A continuación, en la Figura 25, se presenta la API REST en una tabla de Excel, donde se visualizan los controladores organizados por módulo.

Figura 25

Lista de endpoints por módulo

Nota: Imagen creada de Canvas

5.2.10 Diseño del entorno Docker

Como parte del diseño técnico del aplicativo, se contempló el uso de Docker como herramienta para garantizar la portabilidad, la reproducibilidad del entorno de ejecución y la facilidad de configuración de la aplicación en cualquier máquina o servidor.

El diseño establece que cada componente del aplicativo, frontend (Angular) y backend (Spring Boot), debe empaquetarse en contenedores independientes que se integren con una base de datos MySQL externa. Esta separación permite que cada servicio pueda configurarse y desplegarse de forma autónoma, sin depender de configuraciones específicas del sistema operativo anfitrión. Se definió el uso de un archivo de orquestación (docker-compose.yaml) que centraliza las variables de entorno, redes internas y puertos expuestos de cada servicio. Con este enfoque, cualquier desarrollador puede levantar el aplicativo completo con un único comando, eliminando la necesidad de instalar manualmente dependencias como Java, Node.js o MySQL en su equipo.

La elección de imágenes base ligeras, como Eclipse Temurin 17 JRE Alpine para el backend y Nginx para servir el frontend compilado, responde al objetivo de optimizar el tamaño de los contenedores y reducir los tiempos de construcción y despliegue. Esta decisión de diseño responde al requerimiento técnico RT#2, ya que el uso de Docker complementa la arquitectura del aplicativo al encapsular cada componente de manera independiente, facilitando tanto el desarrollo colaborativo como la migración entre diferentes entornos de ejecución sin modificaciones en el código fuente.

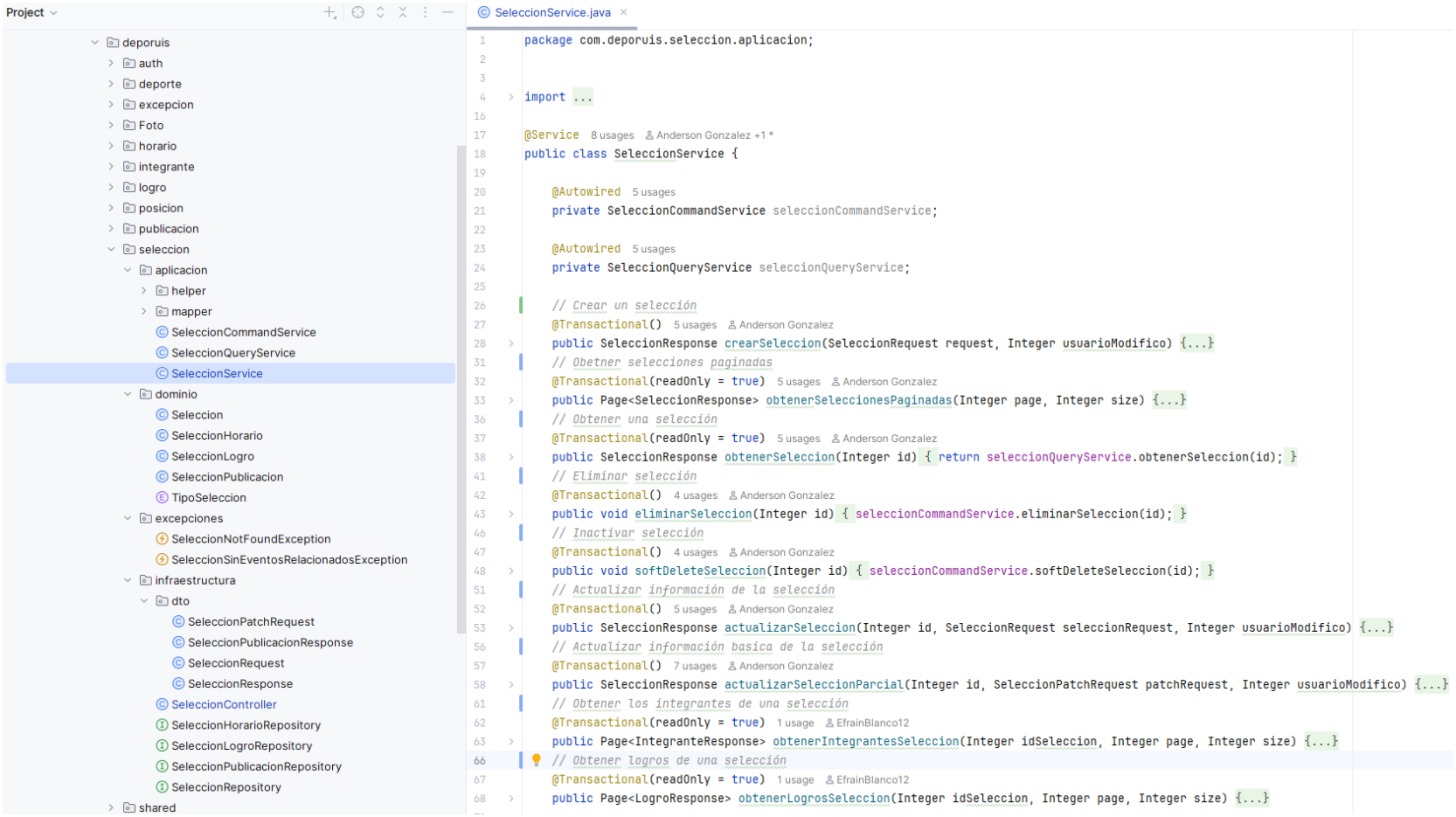
5.3 Implementación e integración de la aplicación web

Este proyecto está organizado por los módulos auth, deporte, foto, horario, integrante, logro, posición, publicación y selección, tal como se presentan en la figura 26. La estructura

del código se presentará en el 6.3.1 siguiendo el orden que se presentaron los requerimientos funcionales, en el 6.3.2 los requerimientos no funcionales, en el 6.3.3 los requerimientos técnicos y en el 6.3.4 el despliegue. Los repositorios del proyecto se encuentran en GitHub: (Backend-Selecciones-Deportivas-UIS, 2025) (Frontend-Selecciones-Deportivas-UIS, 2025) y hay un video de capacitación en: (Capacitación SelecUis, 2026)

Figura 26

Vista general de los módulos



Nota: Pantallazo tomado de IntelliJ IDEA.

La figura 26 es un ejemplo del módulo de selección, en el cual se observan sus capas internas: aplicación, dominio, infraestructura y excepciones. A la derecha se visualiza la clase SeleccionService, encargada de orquestar las operaciones principales del módulo, como crear, consultar, actualizar y eliminar selecciones.

5.3.1 Requerimientos Funcionales (RF)

RF#1 – Gestión de Usuarios y Roles: Permite el registro y autenticación de usuarios con diferentes roles dentro del aplicativo, tales como administrador y entrenador. Los administradores pueden gestionar usuarios, selecciones y asignar roles, mientras que los entrenadores se encargan de administrar la información de su selección y de los integrantes asociados. Adicionalmente, el aplicativo contempla un acceso público que permite a usuarios no autenticados consultar la información visible de las selecciones deportivas, noticias y eventos.

En el backend, este comportamiento se soporta mediante Spring Security y JWT, integrando componentes como el controlador de autenticación y el filtro de validación de tokens, cuya estructura se muestra en la Figura 27. Las contraseñas se almacenan cifradas y la validación de credenciales genera un token que posteriormente es utilizado para autorizar cada solicitud. En el frontend, el servicio de autenticación gestiona el almacenamiento del token y protege las rutas privadas del panel administrativo, tal como se aprecia en la Figura 28. La Figura 29 muestra un ejemplo de consumo del servicio de login en entorno local. Las fuentes de estas figuras son: la 27 corresponde a un pantallazo de IntelliJ IDEA, la 28 a Visual Studio Code y la 29 a Postman.

Figura 27

Estructura backend del módulo de auth

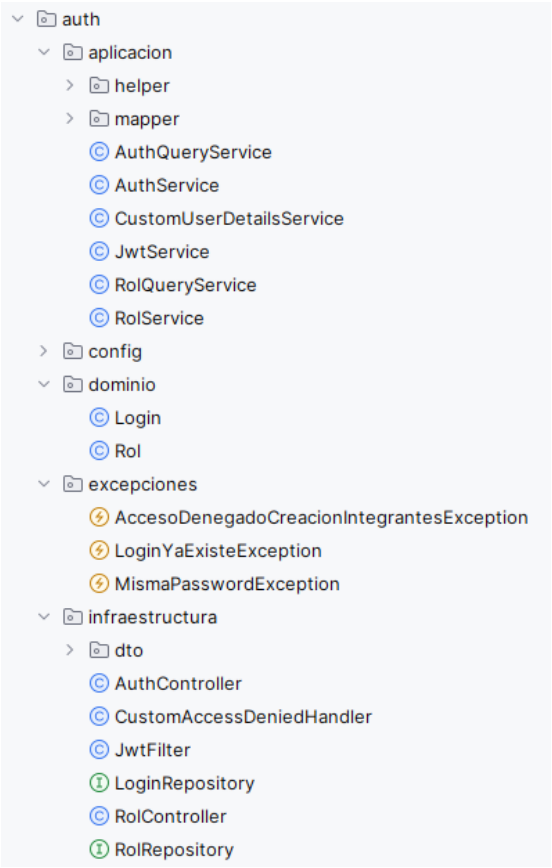


Figura 28

Estructura frontend del módulo de auth

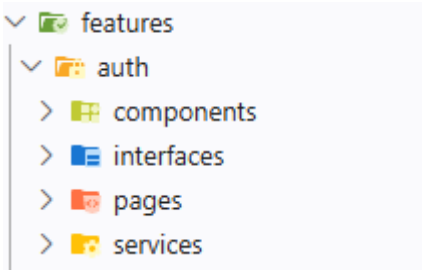
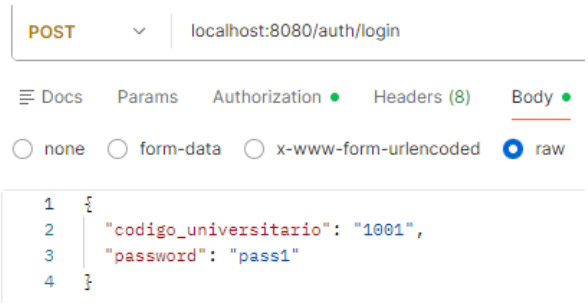


Figura 29

Prueba local del servicio de inicio de sesión



RF#2 – Gestión de Selecciones Deportivas: Permite registrar, editar, listar y eliminar selecciones deportivas. Cada selección tiene un deporte asociado, un nombre, tipo (masculino, femenino o mixto), espacio deportivo y visibilidad. También se permite asignar integrantes y un entrenador responsable. Las fuentes de estas figuras son: la 30 corresponde a un pantallazo de IntelliJ IDEA y la 31 a Postman.

Figura 30

Estructura backend del módulo de selección

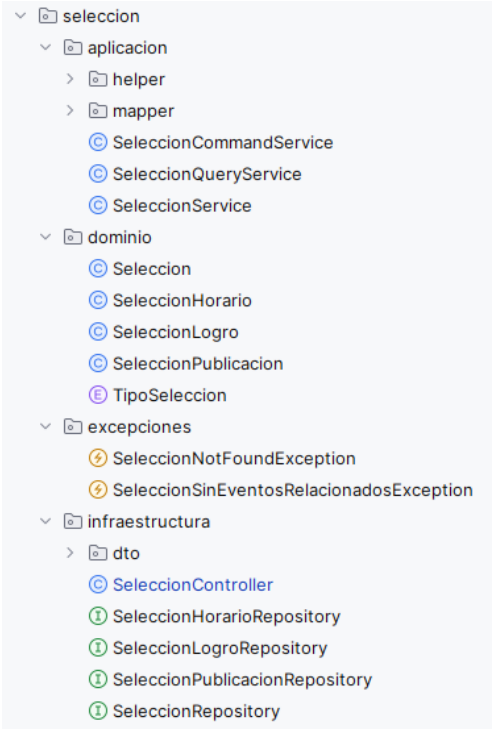
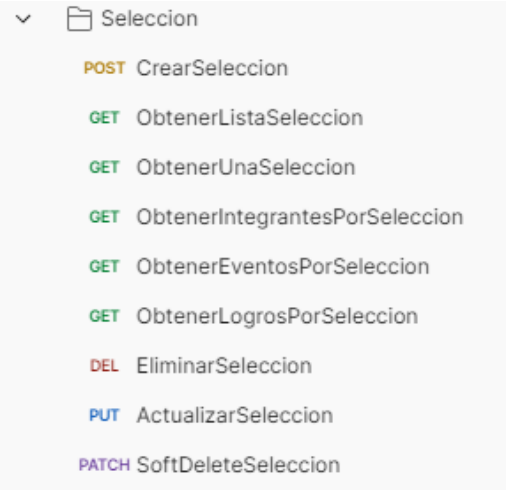


Figura 31

Colección de los servicios de selección



En la estructura del backend, presentada en la Figura 30, puede observarse la organización modular del componente de selección, incluyendo servicios diferenciados para operaciones de consulta y modificación. Las consultas utilizan mecanismos de optimización para cargar relaciones necesarias sin afectar el rendimiento. En el frontend, cuya estructura se muestra en la Figura 31, se desarrollaron vistas específicas para el listado general, el detalle y el formulario de creación o edición, permitiendo una administración coherente desde el panel interno.

En este código se observan distintos endpoints REST (@PostMapping, @GetMapping, @DeleteMapping) que gestionan operaciones como crear, listar, consultar y eliminar selecciones. Además, se evidencia la configuración de seguridad mediante anotaciones como @PreAuthorize, así como la estructura de las solicitudes que posteriormente son consumidas y probadas a través de herramientas como Postman.

Figura 32

Muestra de la clase SelecciónController

```java
public class SeleccionController {

    // Crear Seleeción
    @PostMapping("/crear") 1 usage  Anderson Gonzalez +1
    @PreAuthorize("hasAnyRole('ADMINISTRADOR', 'ENTRENADOR')")
    public ResponseEntity<SeleccionResponse> crearSeleccion(
            @Valid @RequestBody SeleccionRequest seleccionRequest,
            @RequestHeader(value = "usuariomodifico", required = false) Integer usuarioModifico) {
        SeleccionResponse response = seleccionService.crearSeleccion(seleccionRequest, usuarioModifico);
        return ResponseEntity.status(HttpStatus.CREATED).body(response);
    }

    // Obtener Lista Seleccion
    @GetMapping("/lista") 1 usage  Anderson Gonzalez
    public ResponseEntity<Page<SeleccionResponse>> obtenerSeleccionesPaginadas(
            @RequestParam(value = "page", defaultValue = "0") Integer page,
            @RequestParam(value = "size", defaultValue = "3") Integer size
    ) {
        Page<SeleccionResponse> pagina = seleccionService.obtenerSeleccionesPaginadas(page, size);
        return ResponseEntity.ok(pagina);
    }

    // Obtener una Selección
    @GetMapping("/obtener/{id}") 1 usage  Anderson Gonzalez
    public ResponseEntity<SeleccionResponse> obtenerSeleccion(
            @PathVariable Integer id
    ) {
        SeleccionResponse seleccion = seleccionService.obtenerSeleccion(id);
        if (seleccion == null) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
        return ResponseEntity.ok(seleccion);
    }
```

Nota: Pantallazo tomado de IntelliJ IDEA

RF#3 – Gestión de Jugadores (Integrantes): Permite registrar y actualizar los datos personales y deportivos de los integrantes: nombre, apellidos, fecha de nacimiento, posición, dorsal, estatura, peso, rol y selección a la que pertenecen. Las fuentes de estas figuras son: La figura 33 corresponde a un pantallazo de IntelliJ IDEA y la 34 a Postman

Figura 33

Estructura backend del módulo de integrante

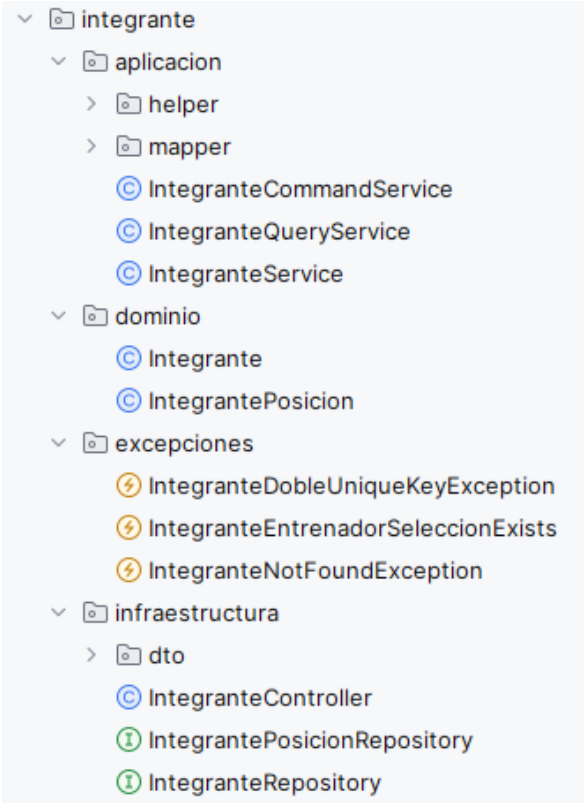
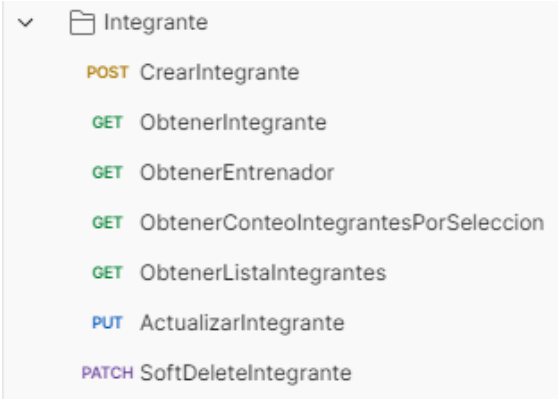


Figura 34

Colección de servicios del módulo de integrante



En la Figura 33, se implementaron servicios diferenciados para operaciones de creación, actualización y consulta, apoyados en repositorios JPA. Además, se definieron restricciones de unicidad sobre campos como el código universitario y el correo institucional, con el fin de evitar duplicidades. La Figura 34 muestra la colección de pruebas realizada para validar los endpoints asociados a este módulo. En el frontend, la gestión se realiza desde el panel administrativo mediante formularios y tablas que permiten editar o consultar la información de cada integrante.

RF#4 – Publicaciones: Noticias y Eventos: Permite crear, editar, eliminar y consultar publicaciones clasificadas como noticia o evento. Cada publicación tiene título, descripción, lugar, fecha, duración, visibilidad y puede estar asociada a una o varias selecciones. Las fuentes de estas figuras son: La 35 corresponde a un pantallazo de IntelliJ IDEA y la 36 a Postman.

Figura 35

Estructura backend del módulo de publicación

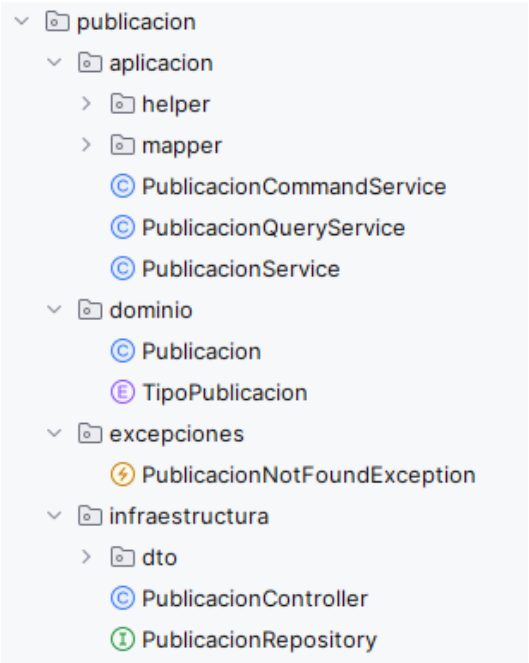
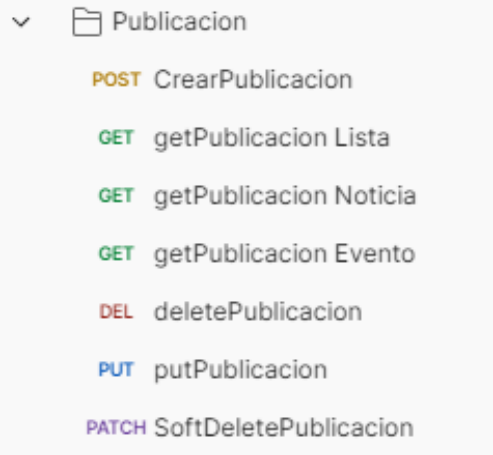


Figura 36

Colección de servicio del módulo de publicación



La Figura 36 presenta las pruebas realizadas sobre estos endpoints. En el frontend se desarrollaron vistas públicas para mostrar noticias y eventos visibles, así como una sección

administrativa para su gestión por parte de usuarios autorizados, en la Figura 35, se evidencia su organización interna bajo el mismo patrón arquitectónico. Se implementaron controladores y servicios que permiten crear, editar, eliminar y consultar publicaciones, incluyendo un método específico para recuperar los eventos más recientes.

RF#5 – Horarios por Selección: Permite registrar y consultar los días y horas de entrenamiento o competencia de una selección, relacionando cada horario con el equipo correspondiente. Las fuentes de estas figuras son: La 37 corresponde a un pantallazo de Postman y la 38 a IntelliJ IDEA.

Figura 37

Colección de servicio del módulo de horario

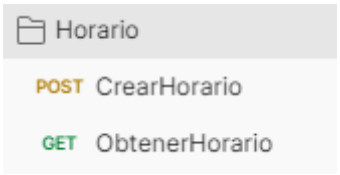
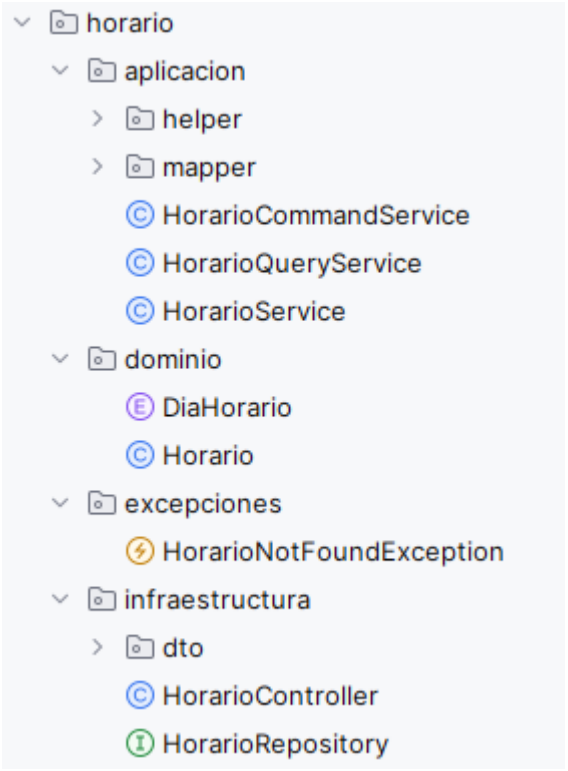


Figura 38

Estructura backend del módulo de horario



En el backend, este comportamiento se modeló mediante las entidades correspondientes y sus relaciones, como se observa en la Figura 38. Los controladores permiten agregar y listar horarios por selección, y la Figura 37 muestra las pruebas realizadas para validar estas operaciones.

En el frontend, los horarios se visualizan dentro del detalle de cada selección, permitiendo consultar de manera clara la programación semanal.

RF#6 – Logros por Selección: Permite registrar los logros deportivos obtenidos por cada selección, incluyendo nombre del logro, competencia y año. Las fuentes de estas figuras son: La 39 corresponde a un pantallazo de Postman y la 40 a IntelliJ IDEA.

Figura 39

Colección de servicio del módulo de logro

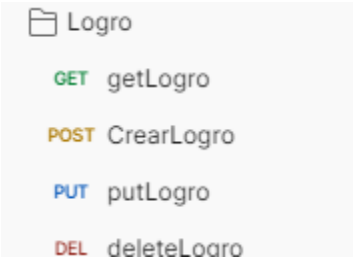
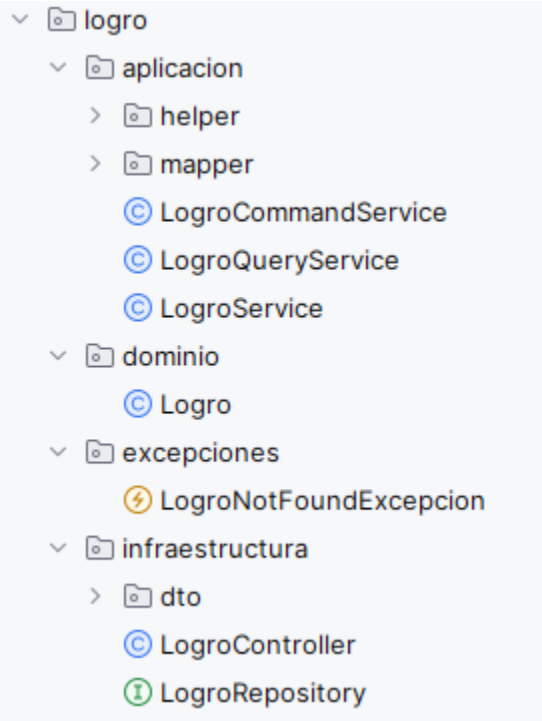


Figura 40

Estructura backend del módulo de logro



La estructura del módulo se presenta en la Figura 40, donde se evidencia la organización interna del componente en el backend. Se implementaron servicios para agregar y consultar logros vinculados a una selección específica, apoyados en una tabla intermedia que gestiona la relación

correspondiente. La Figura 39 muestra la colección de pruebas realizada para validar estas operaciones. En el frontend, los logros se presentan dentro de la vista de detalle de cada selección.

RF#7 – Galería de Fotos: Permite subir y asociar fotografías a selecciones o publicaciones.

Las fotos se almacenan en la base de datos con su metadato de temporada y referencia al elemento al que pertenecen. Las fuentes de estas figuras son: La 41 corresponde a un pantallazo de Postman y la 42 a IntelliJ IDEA.

Figura 41

Colección de servicio del módulo de foto

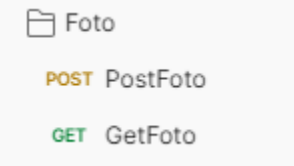
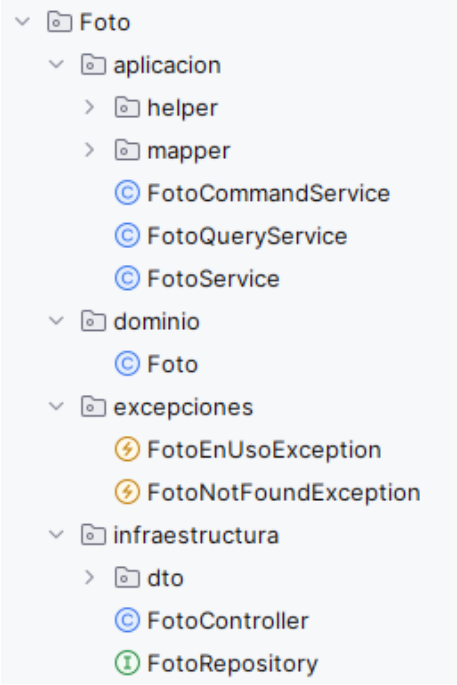


Figura 42

Estructura backend del módulo de foto



La organización del módulo en el backend puede observarse en la Figura 42, donde se identifican los componentes encargados de gestionar las operaciones de carga y consulta. La Figura 41 muestra las pruebas realizadas sobre los endpoints asociados. En el frontend se desarrolló una galería que presenta miniaturas de las imágenes y permite su visualización ampliada, integrándose de forma natural en las vistas de selección o publicación.

RF#8 – Portal Público de Consulta: Permite que cualquier usuario (sin autenticarse) pueda consultar las **selecciones**, sus noticias y próximos eventos visibles.

Figura 43

Clase endpoint whitelist

Nota: Pantallazo tomado de IntelliJ IDEA.

En el backend se definieron endpoints públicos con filtros de visibilidad, mientras que en el frontend se desarrollaron las vistas de inicio, noticias y eventos accesibles sin iniciar sesión. La Figura 43 muestra la configuración de rutas públicas dentro del sistema de seguridad.

5.3.2 Requerimientos No Funcionales (RNF)

RNF#1 – Seguridad: La seguridad del aplicativo se definió como un aspecto transversal a todos los módulos. Para ello se implementó un esquema de autenticación basado en JWT que permite validar cada solicitud sin mantener sesiones activas en el servidor. El backend opera en modo stateless, utilizando un filtro que intercepta las peticiones HTTP y verifica la validez del token antes de permitir el acceso a los recursos protegidos. Las contraseñas se almacenan cifradas, garantizando que la información sensible no se conserve en texto plano. Las **f**uentes de estas figuras son: La 44 corresponde a un pantallazo de Visual Studio Code y la 45 a IntelliJ IDEA.

Figura 44

Estructura frontend del módulo de auth

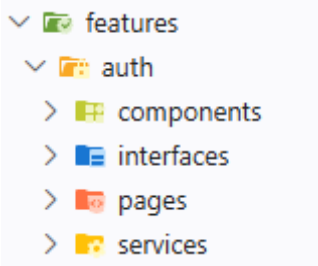


Figura 45

Validador de rol endpoint

```
// Crear Seleeción
@PostMapping("/crear") 1 usage  Anderson Gonzalez +1
@PreAuthorize("hasAnyRole('ADMINISTRADOR', 'ENTRENADOR')")
public ResponseEntity<SeleccionResponse> crearSeleccion(
        @Valid @RequestBody SeleccionRequest seleccionRequest,
        @RequestHeader(value = "usuariomodifico", required = false) Integer usuarioModifico) {
    SeleccionResponse response = seleccionService.crearSeleccion(seleccionRequest, usuarioModifico);
    return ResponseEntity.status(HttpStatus.CREATED).body(response);
}
```

En el frontend, las rutas privadas se encuentran protegidas mediante mecanismos de validación que impiden el acceso si el usuario no cuenta con un token válido. La Figura 44 muestra la estructura del módulo de autenticación en el cliente, mientras que la Figura 45 ilustra el esquema de validación de permisos aplicado a los endpoints. Este diseño controla que cada usuario solo pueda acceder a las funcionalidades correspondientes a su rol.

RNF#2 – Usabilidad y Accesibilidad: La interfaz debe ser responsiva, clara y accesible desde distintos dispositivos, con navegación sencilla y componentes reutilizables.

El uso de Tailwind CSS permitió estructurar estilos de manera uniforme y facilitar ajustes visuales sin afectar la lógica de la aplicación. Elementos como botones, formularios, paginadores

y mensajes de carga fueron diseñados para mantener consistencia en toda la plataforma. El resultado es una interfaz organizada, con navegación intuitiva y separación clara entre las secciones públicas y el panel administrativo.

RNF#3 – Rendimiento y Escalabilidad: El aplicativo fue diseñado considerando la necesidad de manejar múltiples consultas sin afectar el desempeño. En el backend se implementaron estrategias para optimizar el acceso a datos, evitando consultas redundantes y aplicando paginación en los listados más extensos. La Figura 46 muestra un ejemplo de uso de @EntityGraph para mejorar la carga de relaciones en consultas JPA.

Figura 46

Optimización consultas con @EntityGraph

Nota: Pantallazo tomado de IntelliJ IDEA.

Asimismo, se configuraron perfiles diferenciados para entornos de desarrollo y producción, lo que permite adaptar parámetros según el contexto de ejecución. Estas decisiones contribuyen a mantener tiempos de respuesta adecuados y a facilitar futuras ampliaciones de la aplicación.

5.3.3 Requerimientos Técnicos (RT)

RT#1 – Tecnologías Utilizadas: El desarrollo de la aplicación se realizó utilizando un enfoque basado en tecnologías consolidadas dentro del ecosistema empresarial moderno. El frontend fue implementado con Angular en su versión 19.2, empleando TypeScript como lenguaje principal y gestionando dependencias mediante Node.js y npm. La configuración del proyecto se

administra a través de Angular CLI, lo que permite la ejecución en entorno de desarrollo mediante ng serve y la generación de compilaciones optimizadas para producción.

En el backend se utilizó Spring Boot 3.4.5 como framework principal, estructurado como proyecto Maven y configurado sobre Java 17, versión definida explícitamente en el archivo pom.xml. La arquitectura del servidor se apoya en Spring Web para la exposición de servicios REST, Spring Security para el manejo de autenticación y autorización, y Spring Data JPA para la interacción con la base de datos.

La persistencia de la información se gestiona mediante MySQL 8 como sistema de base de datos relacional. La configuración de conexión y perfiles de entorno permite diferenciar ejecución en desarrollo y producción. El proyecto backend se construye y gestiona mediante Maven, lo que facilita la administración de dependencias y el empaquetado del artefacto ejecutable.

RT#2 – Arquitectura: El backend sigue una arquitectura hexagonal dividida en capas: dominio, aplicación e infraestructura, complementada con una distribución Vertical Slicing. El frontend utiliza módulos por funcionalidad (feature modules) y componentes standalone. Las fuentes de estas figuras son: La 47 y 48 corresponden a un pantallazo de IntelliJ IDEA, la 49 y 50 de Visual Studio Code.

Figura 47

Capas basadas en arquitectura hexagonal backend

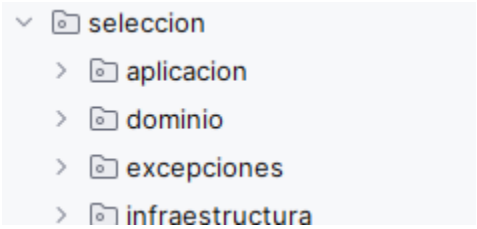


Figura 48

Implementación de vertical slicing backend

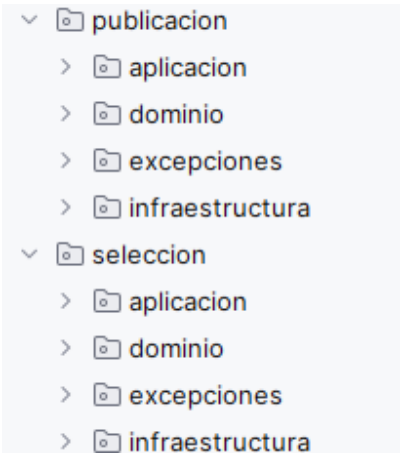


Figura 49

Distribución por funcionalidades frontend

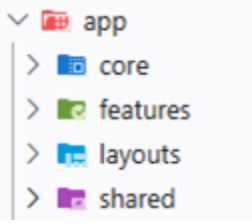
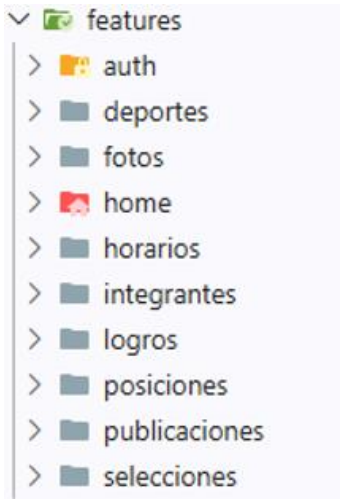


Figura 50

Módulos de funcionalidad frontend



5.3.4 Despliegue de la aplicación

Como parte de la implementación del aplicativo, se llevó a cabo el proceso de despliegue utilizando tecnologías basadas en contenedores, con el objetivo de garantizar un entorno de ejecución consistente y facilitar la portabilidad de la aplicación.

El aplicativo fue dockerizado, permitiendo encapsular sus componentes principales en contenedores independientes. Esta configuración facilita que las dependencias, versiones y parámetros de ejecución se mantengan controlados, evitando inconsistencias entre diferentes entornos. Para la orquestación de los servicios se utilizó Docker Compose, lo que permitió definir y ejecutar los componentes como una única aplicación distribuida, facilitando la comunicación entre ellos mediante redes internas y la exposición de servicios a través de puertos definidos.

En el despliegue del aplicativo, se utilizaron servicios en la nube para la publicación de cada componente:

1. Frontend (Angular): fue desplegado mediante Cloudflare Pages, lo que permitió su distribución global a través de la red CDN de Cloudflare, optimizando los tiempos de carga para los usuarios finales.
2. Backend (Spring Boot): fue desplegado en la plataforma Fly.io, configurado en la región de Bogotá (Colombia) con el fin de reducir la latencia. Este servicio ofrece capacidad de escalado automático y gestión de recursos bajo demanda.
3. Base de datos (MySQL): se implementó mediante un servicio administrado en la nube a través de Aiven, donde se provisionó una instancia lista para su uso. La conexión se configuró directamente en el backend, permitiendo el acceso seguro y la persistencia de la información sin necesidad de administrar la infraestructura.

Finalmente, la dockerización del aplicativo garantiza que la solución no dependa exclusivamente de estas plataformas, ya que puede ser desplegada en cualquier entorno que soporte contenedores. Esto proporciona flexibilidad para futuras migraciones, facilita la replicación del aplicativo en distintos contextos y simplifica su puesta en marcha sin requerir configuraciones adicionales complejas.

5.4 Validación del aplicativo mediante pruebas y ajustes

Con el objetivo de garantizar el correcto funcionamiento del aplicativo desarrollado, se llevaron a cabo diferentes tipos de pruebas orientadas a validar la lógica del negocio, la integración entre componentes, la seguridad del aplicativo y la experiencia de usuario. Estas pruebas permitieron verificar el cumplimiento de los requerimientos definidos y validar la calidad del aplicativo antes de su puesta en funcionamiento.

5.4.1 Pruebas unitarias

En primer lugar, se realizaron pruebas unitarias sobre los servicios del backend, enfocadas en validar la lógica de negocio de manera aislada. Estas pruebas se implementaron utilizando JUnit 5 como framework principal y Mockito para la simulación de dependencias, lo que permitió evaluar el comportamiento de los servicios sin necesidad de interactuar directamente con la base de datos, en la Figura 51 se presenta un ejemplo de una clase de prueba, donde se evidencia el uso de anotaciones para la inyección de dependencias simuladas y la ejecución de casos de prueba.

A través de estas pruebas se verificaron operaciones como la creación, consulta y eliminación de selecciones, validando que cada método respondiera correctamente ante distintos escenarios, incluyendo casos exitosos y condiciones de error. El uso de objetos simulados permitió controlar las entradas y validar las salidas esperadas de forma precisa.

Como evidencia, se pueden observar las clases de prueba en el backend donde se utilizan anotaciones como @Test, @Mock y @InjectMocks, reflejando la validación aislada de los servicios.

Figura 51

Ejemplo de prueba unitaria en el backend utilizando JUnit y Mockito

```
class SeleccionCommandServiceTest {

    @InjectMocks private SeleccionCommandService service;  7 usages
    @Mock private SeleccionRepository seleccionRepository;  15 usages
    @Mock private FotoCommandService fotoCommandService;  7 usages
    @Mock private HorarioCommandService horarioCommandService;  2 usages
    @Mock private SeleccionRelacionService seleccionRelacionService;  7 usages
    @Mock private SeleccionVerificarExistenciaService seleccionVerificarExistenciaService;  22 usages

    private Seleccion seleccion;  55 usages

    @BeforeEach  EfrainBlanco12
    void setup() { seleccion = mock(Seleccion.class); }

    // Eliminar Seleccion
    @Test  EfrainBlanco12
    void eliminarSeleccion_flujoCompleto() {
        when(seleccionVerificarExistenciaService.verificarSeleccion( id: 10)).thenReturn(seleccion);
        doNothing().when(seleccionRelacionService).eliminarRelacionesSeleccion(seleccion);
        doNothing().when(fotoCommandService).eliminarFotosSeleccion(seleccion);
        doNothing().when(seleccionRepository).delete(seleccion);

        service.eliminarSeleccion( id: 10);

        verify(seleccionVerificarExistenciaService).verificarSeleccion( id: 10);
        verify(seleccionRelacionService).eliminarRelacionesSeleccion(seleccion);
        verify(fotoCommandService).eliminarFotosSeleccion(seleccion);
        verify(seleccionRepository).delete(seleccion);
        verifyNoMoreInteractions(seleccionRepository, fotoCommandService, seleccionRelacionService, seleccionVerificarExistenciaService);
    }
```

Nota: Pantallazo tomado de IntelliJ IDEA.

5.4.2 Pruebas de integración

Posteriormente, se realizaron pruebas de integración con el fin de validar la correcta comunicación entre las diferentes capas del aplicativo, incluyendo controladores, servicios y repositorios. Para ello, se emplearon las herramientas proporcionadas por Spring Boot, específicamente Spring Boot Test y MockMvc, las cuales permiten simular solicitudes HTTP y evaluar el comportamiento de los endpoints sin necesidad de desplegar el aplicativo en un entorno externo.

Estas pruebas permitieron verificar el flujo completo de las solicitudes, desde su recepción en los controladores hasta su procesamiento y persistencia, comprobando que las respuestas generadas fueran consistentes y que los códigos de estado HTTP correspondieran a cada operación.

Figura 52

Ejemplo de prueba de integración utilizando MockMvc en Spring Boot

```java
@WebMvcTest(controllers = SeleccionController.class)
@AutoConfigureMockMvc(addFilters = false) // deshabilitar filtros para testing
class SeleccionControllerIT {

    @Autowired private MockMvc mvc;  9 usages
    @Autowired private ObjectMapper om;  5 usages

    @MockitoBean private SeleccionService service;  7 usages
    @MockitoBean private com.deporuis.auth.infraestructura.JwtFilter jwtFilter;  no usages

    private SeleccionRequest buildReq() {...}

    @WithMockUser(roles = {"ADMINISTRADOR"})  EfrainBlanco12 +1
    @Test
    void crear_retorna200_ok() throws Exception {
        SeleccionResponse resp = new SeleccionResponse();
        resp.setIdSeleccion(1);
        resp.setNombreSeleccion("Seleccion UIS");
        resp.setUsuarioModifico(100);

        when(service.crearSeleccion(any(), eq( value: 100))).thenReturn(resp);

        mvc.perform(post( uriTemplate: "/private/seleccion/crear")
                        .with(csrf()) // CSRF requerido
                        .header( name: "usuariomodifico",  ...values: "100")
                        .contentType(MediaType.APPLICATION_JSON)
                        .content(om.writeValueAsString(buildReq())))
                .andExpect(status().isCreated()) // tu controller devuelve 201 CREATED
                .andExpect(jsonPath( expression: "$.usuarioModifico").value( expectedValue: 100));
    }
```

Fuente: Pantallazo tomado de IntelliJ IDEA.

La figura 52 presenta un fragmento de una prueba de integración del controlador de selecciones, en la cual se valida el comportamiento del endpoint de creación mediante el uso de herramientas como MockMvc. En esta prueba se simula una solicitud HTTP tipo POST,

verificando tanto la respuesta del aplicativo como el código de estado retornado, así como la correcta estructura de los datos.

Esta evidencia corresponde a una muestra representativa del conjunto de pruebas de integración implementadas, las cuales cubren distintos escenarios y operaciones del aplicativo, garantizando la correcta interacción entre las capas del backend.

5.4.3 Validación del control de acceso y autorización

Adicionalmente, se validó el correcto funcionamiento del mecanismo de seguridad implementado en la aplicación, basado en autenticación mediante JSON Web Tokens (JWT) y control de acceso por roles mediante Spring Security, en la Figura 53 se muestran ejemplos de endpoints protegidos y públicos, donde se evidencia el uso de la anotación @PreAuthorize para restringir el acceso según roles, así como endpoints sin esta anotación que pueden ser consumidos sin autenticación.

Se comprobó que únicamente los usuarios con los permisos adecuados pudieran acceder a los endpoints protegidos, garantizando la restricción de acceso a funcionalidades críticas. Esta validación se realizó a través de pruebas sobre los controladores, donde se definieron reglas de autorización utilizando anotaciones como @PreAuthorize, así como mediante la verificación del procesamiento del token en cada solicitud. Asimismo, es importante destacar que aquellos endpoints que no cuentan con la anotación @PreAuthorize corresponden a servicios de acceso público, los cuales pueden ser consumidos sin necesidad de autenticación, como es el caso de consultas generales del aplicativo.

Figura 53

Ejemplo de configuración de seguridad en endpoints mediante Spring Security

```
// Crear Seleeción
@PostMapping("/crear") 1 usage  Anderson Gonzalez +1
@PreAuthorize("hasAnyRole('ADMINISTRADOR', 'ENTRENADOR')")
public ResponseEntity<SeleccionResponse> crearSeleccion(
        @Valid @RequestBody SeleccionRequest seleccionRequest,
        @RequestHeader(value = "usuariomodifico", required = false) Integer usuarioModifico) {
    SeleccionResponse response = seleccionService.crearSeleccion(seleccionRequest, usuarioModifico);
    return ResponseEntity.status(HttpStatus.CREATED).body(response);
}

// Obtener Lista Seleccion
@GetMapping("/lista") 1 usage  Anderson Gonzalez
public ResponseEntity<Page<SeleccionResponse>> obtenerSeleccionesPaginadas(
        @RequestParam(value = "page", defaultValue = "0") Integer page,
        @RequestParam(value = "size", defaultValue = "3") Integer size
) {
    Page<SeleccionResponse> pagina = seleccionService.obtenerSeleccionesPaginadas(page, size);
    return ResponseEntity.ok(pagina);
}

// Obtener una Selección
@GetMapping("/obtener/{id}") 1 usage  Anderson Gonzalez
public ResponseEntity<SeleccionResponse> obtenerSeleccion(
        @PathVariable Integer id
) {
    SeleccionResponse seleccion = seleccionService.obtenerSeleccion(id);
    if (seleccion == null) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
    }
    return ResponseEntity.ok(seleccion);
}
```

Nota: Pantallazo tomado de IntelliJ IDEA.

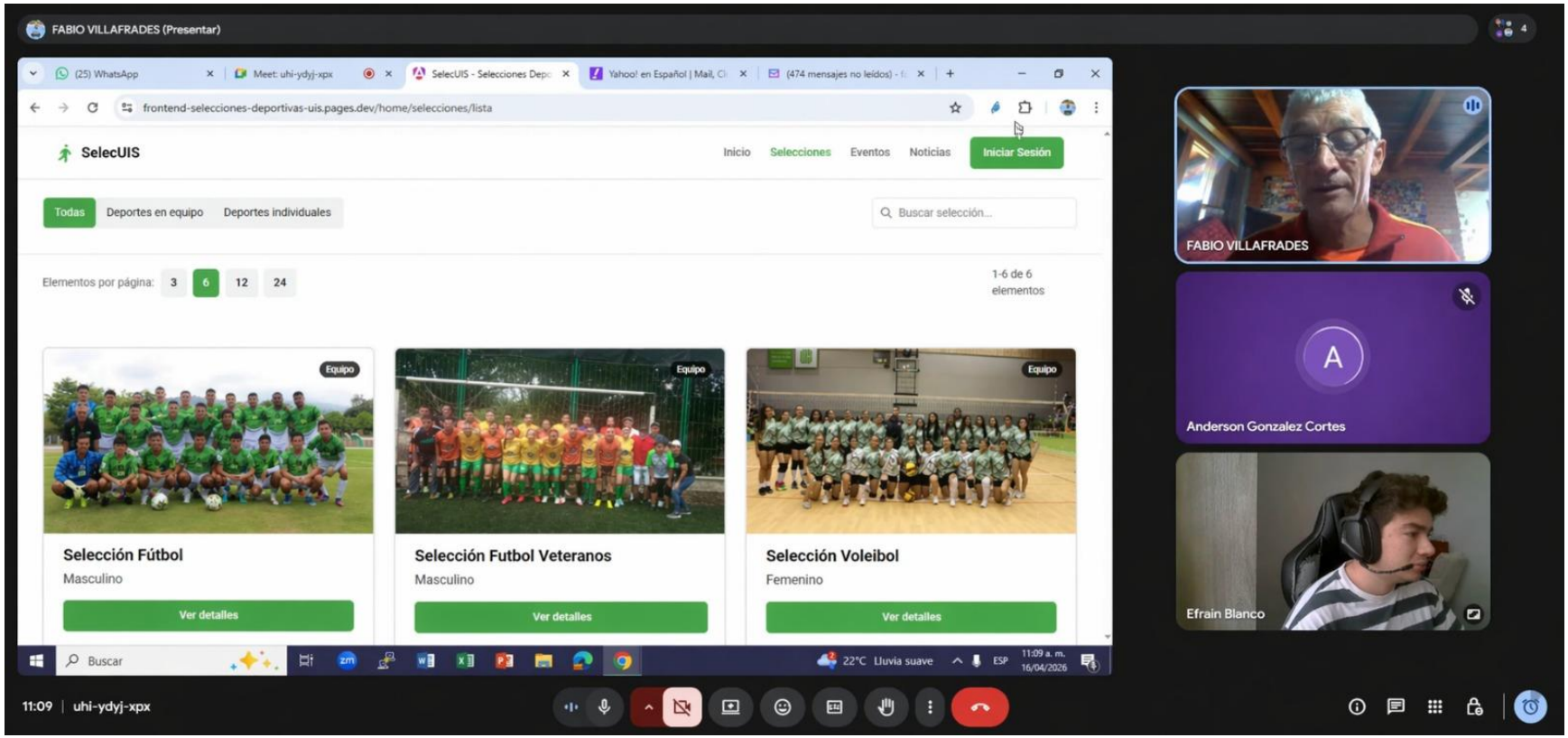
5.4.4 Pruebas de usabilidad

Por otra parte, se llevaron a cabo pruebas de usabilidad utilizando datos de prueba, con el fin de evaluar la facilidad de uso, la navegación y la comprensión de la interfaz de la aplicación. Estas pruebas consistieron en la ejecución de los diferentes flujos funcionales del aplicativo empleando información simulada, lo que permitió validar el comportamiento de la interfaz en escenarios cercanos a su uso real.

De manera complementaria, estas pruebas fueron realizadas por docentes, quienes asumieron el rol de entrenadores dentro del sistema, permitiendo evaluar la aplicación desde una perspectiva cercana al usuario final. Durante la interacción, los entrenadores navegaron por las diferentes funcionalidades disponibles, ejecutando tareas como la gestión de selecciones, integrantes, publicaciones y horarios. En la figura 54 se muestra un ejemplo de una prueba realizada por el profesor Fabio Villafrades, en la cual se navegó por todos los flujos disponibles en el aplicativo.

Figura 54

Prueba de usabilidad entrenador



Nota: Pantallazo tomado de reunión en Google Meet

A partir de estas pruebas, se identificó que las funcionalidades principales del aplicativo son accesibles y facilita la gestión de la información por parte de los usuarios. Asimismo, se observó que los procesos de consulta, registro y actualización de información pueden ser realizados sin dificultad, lo que evidencia un adecuado diseño de la interfaz y una buena experiencia de usuario, alineada con los requerimientos definidos.

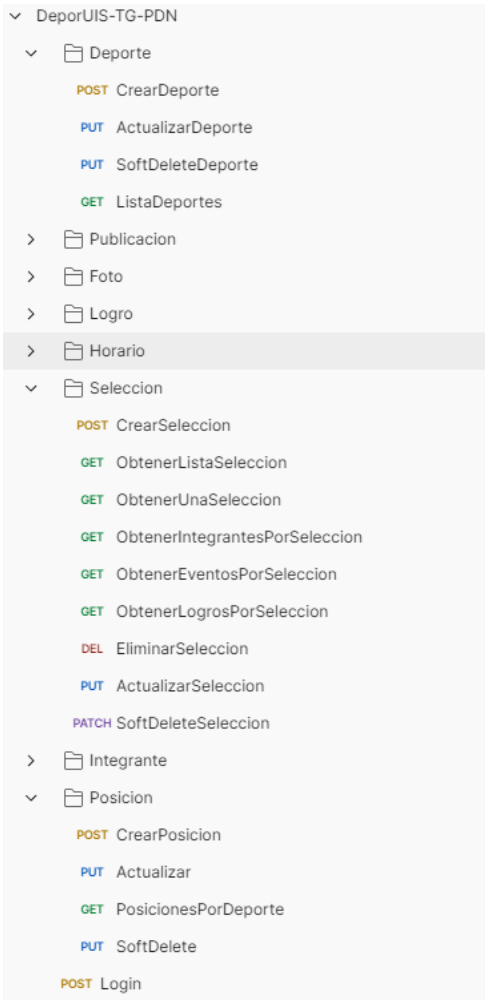
5.4.5 Pruebas de consumo de servicios (API)

Finalmente, se realizaron pruebas de consumo de los servicios expuestos por el backend mediante herramientas externas como Postman, con el objetivo de validar el comportamiento de los endpoints de forma independiente al frontend, en la Figura 55 se muestra la organización de los endpoints probados, agrupados por módulos, así como las operaciones disponibles para cada recurso.

A través de estas pruebas se verificó el correcto envío y recepción de datos en formato JSON, así como la respuesta del aplicativo ante diferentes tipos de solicitudes (creación, consulta y eliminación). Asimismo, se validaron los códigos de estado HTTP retornados por el aplicativo, demostrando que estos reflejen adecuadamente el resultado de cada operación.

Figura 55

Pruebas de endpoints del backend utilizando Postman



Nota: Pantallazo tomado de Postman.

6 Conclusiones

El desarrollo de esta aplicación web permitió transformar un proceso que dependía de hojas de cálculo, grupos de mensajería y comunicación informal en una plataforma centralizada, accesible y confiable. Más allá de la solución técnica, el proyecto evidenció cómo la digitalización

puede devolver tiempo y claridad a quienes gestionan el deporte universitario, permitiéndoles enfocarse en lo que realmente importa: los deportistas y su formación.

Se cumplieron los objetivos planteados: desde el levantamiento de requerimientos con los actores involucrados, hasta el diseño de una arquitectura hexagonal con vertical slicing y CQRS, y la implementación con Angular, Spring Boot y MySQL. Esta combinación arquitectónica logró un aplicativo modular y mantenible, mientras que la seguridad basada en JWT garantizó el acceso controlado a la información sensible de los integrantes.

El proceso de validación con pruebas unitarias, de integración, de seguridad y de usabilidad confirmó que el aplicativo responde de manera correcta ante distintos escenarios y ofrece una experiencia coherente tanto para administradores como para el público general. La dockerización, por su parte, permite la portabilidad y consistencia del despliegue entre entornos.

Adicionalmente, desde la perspectiva del Departamento de Educación Física y Deportes, el aplicativo representa una solución alineada con sus necesidades operativas. Según lo expresado por el profesor Fabio Villafrades, la gestión de la información se realizaba mediante la asignación de un administrador y la delegación de responsabilidades a los entrenadores. En este sentido, la herramienta desarrollada permite formalizar este esquema, facilitando la asignación de roles y garantizando que cada entrenador gestione únicamente la información de su selección, evitando inconsistencias o el uso indebido de los datos. Asimismo, se destaca que su impacto no se limita únicamente a los docentes o entrenadores, sino que se extiende a toda la comunidad universitaria, incluyendo estudiantes que hagan parte o no de una selección, quienes pueden acceder a información actualizada, fortaleciendo la visibilidad y el seguimiento de las actividades deportivas.

En última instancia, este proyecto no solo entrega una herramienta funcional, sino que sienta las bases para que la Universidad Industrial de Santander visibilice sus selecciones deportivas,

preserve su historial de logros y fortalezca el vínculo entre la institución y su comunidad deportiva. La tecnología, en este caso, se pone al servicio del reconocimiento y la gestión del talento deportivo universitario.

7 Trabajo Futuro

Si bien la aplicación cumple con los objetivos definidos para esta práctica, durante el desarrollo y las pruebas de usabilidad se identificaron oportunidades de mejora que enriquecerían la experiencia de uso y ampliarían las capacidades del aplicativo. A continuación, se describen las líneas de trabajo futuro propuestas.

1. Creación contextual de entidades relacionadas: Actualmente, si un usuario necesita registrar una selección deportiva asociada a un deporte que aún no existe en el aplicativo, debe abandonar el formulario, navegar al módulo de deportes, crear el registro y luego regresar al formulario original. Este flujo interrumpe la tarea en curso y afecta la fluidez de la experiencia. Como mejora, se propone implementar un mecanismo de creación en línea que permita, desde el mismo formulario de selección, crear un nuevo deporte mediante un diálogo modal sin perder el contexto ni los datos ya ingresados. Este patrón es aplicable también a otras relaciones, como posiciones o logros.
2. Historial de horarios de entrenamiento: En la versión actual, los horarios de una selección reflejan únicamente su estado vigente. Se propone incorporar un registro histórico que permita conservar los horarios anteriores junto con su período de vigencia. Esto facilitaría el análisis de la evolución de las cargas de entrenamiento a lo largo de los semestres y proporcionaría trazabilidad sobre los cambios realizados.
3. Revisión de la obligatoriedad de imágenes: Durante el uso de la plataforma se identificó la necesidad de evaluar si la carga de imágenes debe ser un requisito obligatorio en todos los

formularios donde actualmente se exige (selecciones, integrantes, publicaciones). En ciertos escenarios, el usuario puede no disponer de una imagen al momento del registro, lo que genera fricción innecesaria. Se plantea flexibilizar este requisito, permitiendo la creación de registros sin imagen y ofreciendo la posibilidad de agregarla o actualizarla posteriormente.

4. Categorización de eventos y publicaciones: Actualmente, las publicaciones se clasifican únicamente como noticia o evento. Se propone extender este modelo mediante la creación de categorías o tipos de eventos personalizables (torneos, jornadas deportivas, entrenamientos abiertos, entre otros), lo que permitiría una organización más granular del contenido, facilitaría el filtrado por parte de los usuarios públicos y brindaría mayor flexibilidad a los administradores para estructurar la comunicación institucional.

Referencias Bibliográficas

Blanco, E., & Gonzalez, A. (2025). *Backend-Selecciones-Deportivas-UIS*. Obtenido de Github: https://github.com/EfrainBlanco12/Backend-Selecciones-Deportivas-UIS

Blanco, E., & Gonzalez, A. (2025). *Frontend-Selecciones-Deportivas-UIS*. Obtenido de Github: https://github.com/EfrainBlanco12/Frontend-Selecciones-Deportivas-UIS

Blanco, E., & Gonzalez, A. (2026). *Capacitación SelecUis*. Obtenido de https://youtu.be/sdRqqaGTyro

Bogard, J. (2017). *Vertical Slice Architecture*. Obtenido de https://www.jimmybogard.com/vertical-slice-architecture/

Buckley, J. (2021). *Hexagonal architecture: Principles and practices for software development.* O'Reilly Media.

Castro Muñoz, M. (2024). *Diseño e implementación de pruebas para una aplicación en Angular. Universidad de Antioquia*. Obtenido de https://hdl.handle.net/10495/40027

Fowler, M. (2011). *CQRS*. Obtenido de https://martinfowler.com/bliki/CQRS.html

Gobernación de Santander. (2023). *Política Pública del Deporte, la Actividad Física y la Recreación en Santander*. Obtenido de https://smi-geoportal.santander.gov.co/smi/docs/PPDeporte.pdf

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT). RFC 7519.* . Obtenido de https://datatracker.ietf.org/doc/html/rfc7519

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT).* Internet Engineering Task Force. Obtenido de https://jwt.io/introduction/

Labs, T. (2024). *Tailwind CSS documentation*. Obtenido de https://tailwindcss.com/docs

Oracle. (2024). *MySQL 8.0 reference manual* . Obtenido de https://dev.mysql.com/doc/refman/8.0/en/

Richardson, L. (2020). *RESTful web services.* O'Reilly Media.

Rivera Florez, J. (2023). *Diseño y cobertura de pruebas de software para el backend. Universidad de Antioquia*. Obtenido de https://hdl.handle.net/10495/34365

Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process.* Addison-Wesley.

Salamanca Calderón, D. (2017). *FilmsApp : Aplicación web con Spring Boot y Angular. Universitat Oberta de Catalunya.* . Obtenido de https://hdl.handle.net/10609/59951

Schwaber, K. &. (2020). *The Scrum guide*. Obtenido de https://scrumguides.org

Software, P. (2024). *Spring Boot reference documentation*. Obtenido de https://docs.spring.io/spring-boot/docs/current/reference/html/

Team, A. (2024). *Angular documentation*. Obtenido de https://angular.io/docs

Universidad Industrial de Santander. (2014). *Política de grupos culturales y selecciones deportivas de la UIS*. Obtenido de https://uis.edu.co/wp-content/uploads/2022/06/2014PoliticaGruposCultDeportivos.pdf

Universidad Industrial de Santander. (2014). *Política de grupos culturales y selecciones deportivas de la UIS*. Obtenido de https://uis.edu.co/wp-content/uploads/2022/06/2014PoliticaGruposCultDeportivos.pdf

Universidad Industrial de Santander. (2021). *Historia de la Universidad Industrial de Santander*. Obtenido de https://uis.edu.co/wp-content/uploads/2022/05/historiaUIS.pdf