Técnicas de diseño

Informe Medallero - Grupo 5

Integrantes

Ignacio Garcia Segovia 101636
Esteban Federico Peña 100102
Ramiro Lozano 98263
Fernando Balmaceda 105525
Tomas Moises Dahab 106855

Índice

Objetivo

El objetivo es crear una plataforma en donde deportistas puedan consultar eventos deportivos y su medallero, y administradores puedan cargar resultados e información de eventos deportivos

Consideraciones

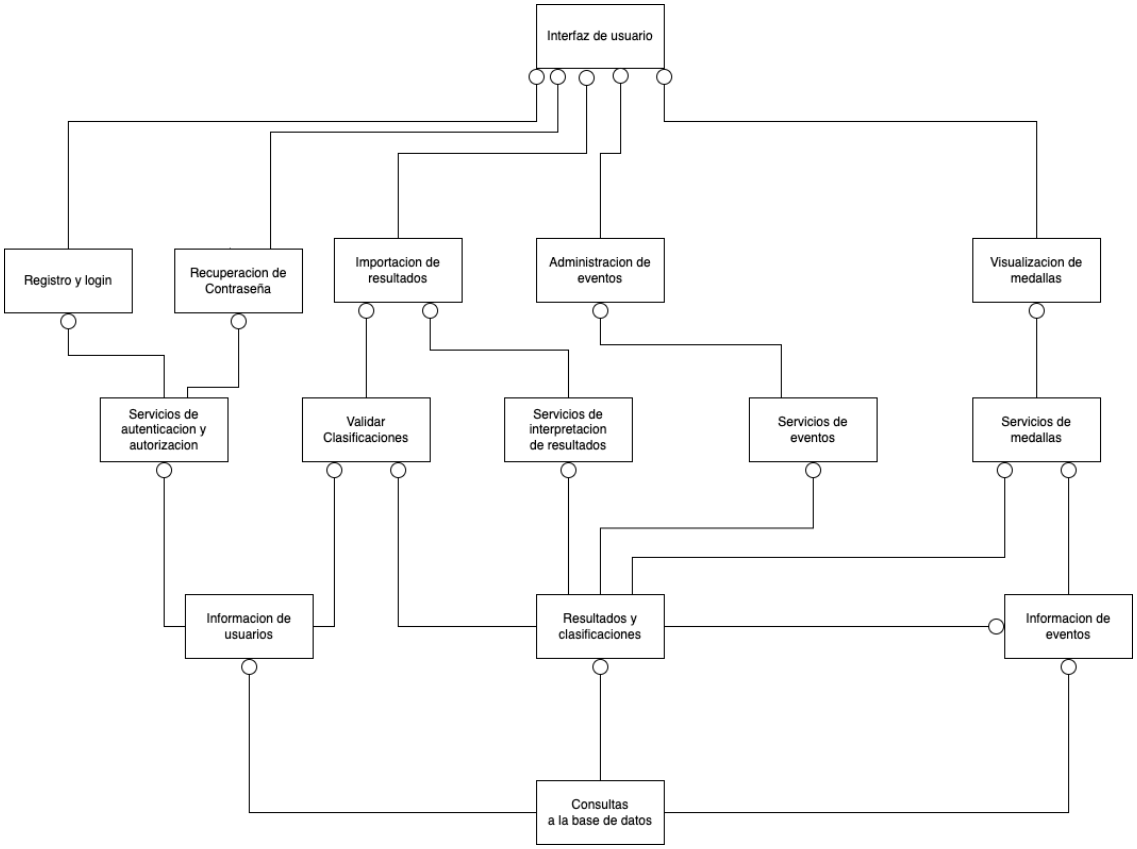
Para la realización de la plataforma consideramos los siguientes ítems

- El alta de un evento debe hacerse proporcionando su información y los resultados del mismo
- La asignación de medallas ocurre luego de que el deportista es notificado de su clasificación en un evento cargado previamente y acepta el resultado
- Se puede buscar a un deportista por nombre y apellido para revisar su perfil y las medallas que consiguió
- Por tema de tiempo, hay algunas funcionalidades que pedía el enunciado que no pudieron implementarse, las cuales decidimos dejarlas para poder enfocarnos en features que creemos son más importantes para lo que sería un mvp. Entre las funcionalidades que descartamos están:
    - La posibilidad de dejar comentarios en un evento específico
    - Poder hacer que el perfil de un usuario sea público/privado para el resto de los usuarios
    - La publicación de resultados que provengan del atleta en sí

Modelo de vistas 4+1

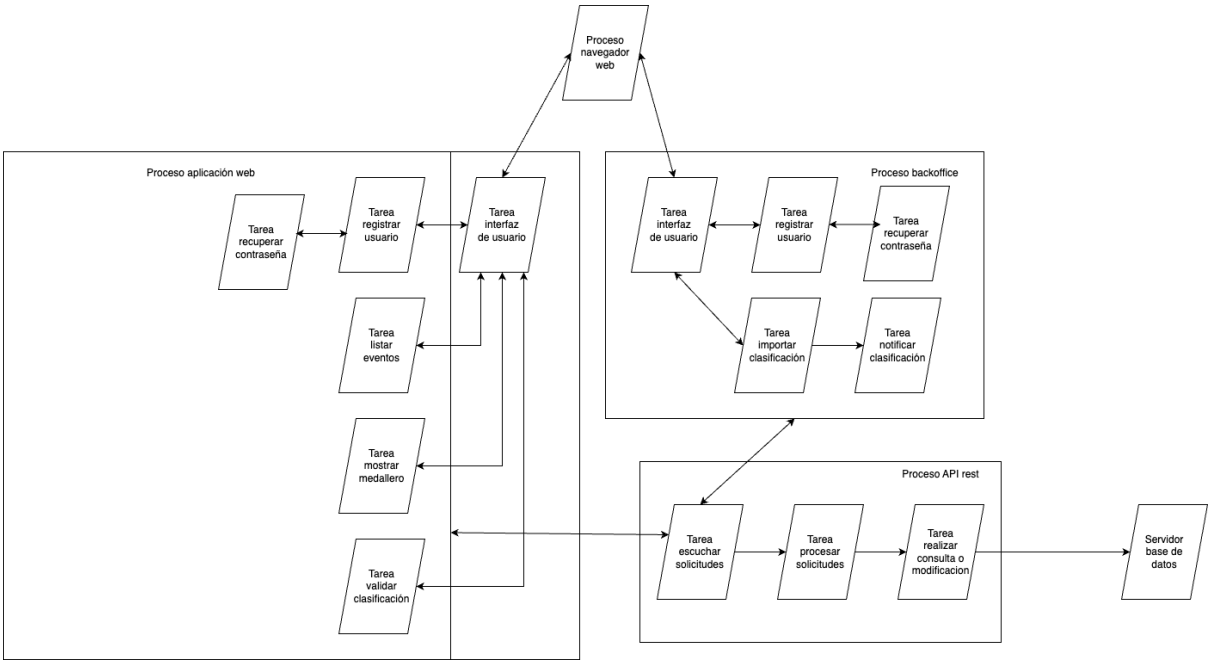
(documento inicial pre-desarrollo)

Vista lógica



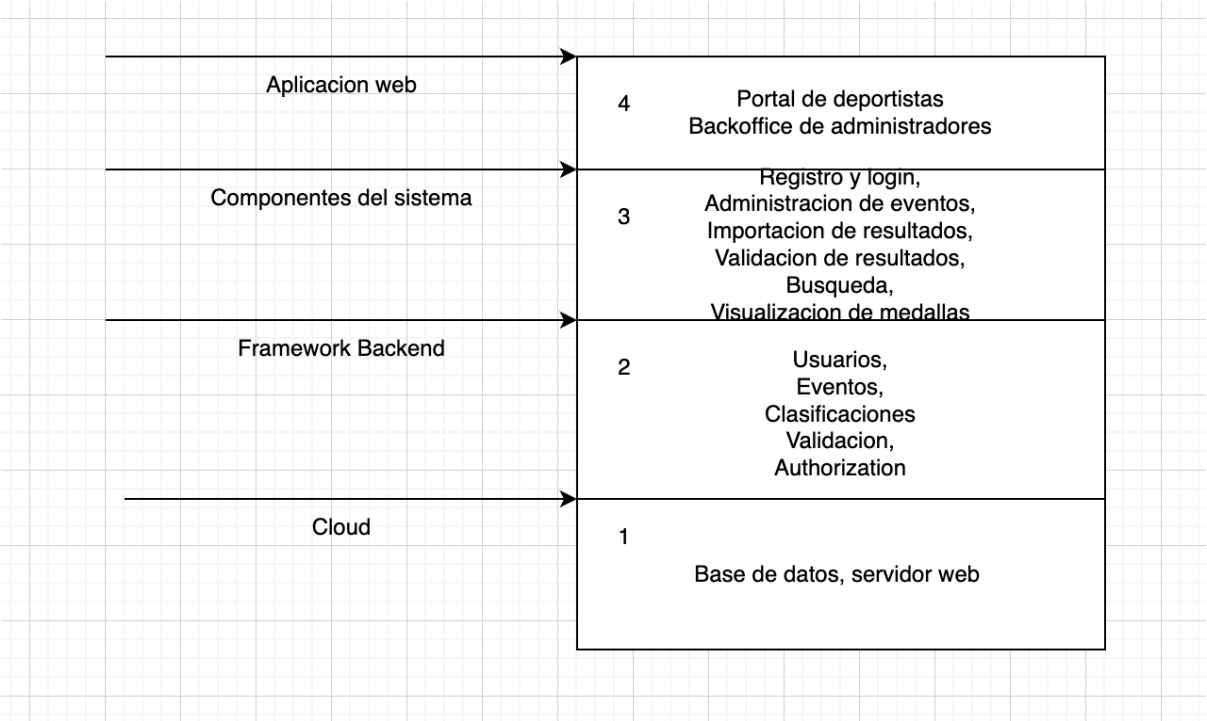
El sistema cuenta con una capa de servicios que permite implementar todas las funcionalidades e interactuar con las entidades de usuarios, eventos y clasificaciones, que se mantienen en una base de datos.

Vista de proceso



- **Proceso de base de datos de eventos**: Base de datos con la información de los eventos creados, los usuarios registrados y las clasificaciones
- **Proceso backend**: API Rest que se encarga de crear todas las entidades de las bases de datos, de notificar a los deportistas de sus clasificaciones y aprobar y rechazar clasificaciones
- **Proceso aplicación web**: Aplicación para mostrar eventos, el medallero de los deportistas y formulario de carga de resultados de deportistas
- **Proceso backoffice**: Aplicación para crear eventos y cargar resultados
- **Proceso de browser del usuario**: navegador web del cliente conectado a la aplicación a través de internet, donde puede buscar eventos, y validar sus clasificaciones.

Vista de desarrollo



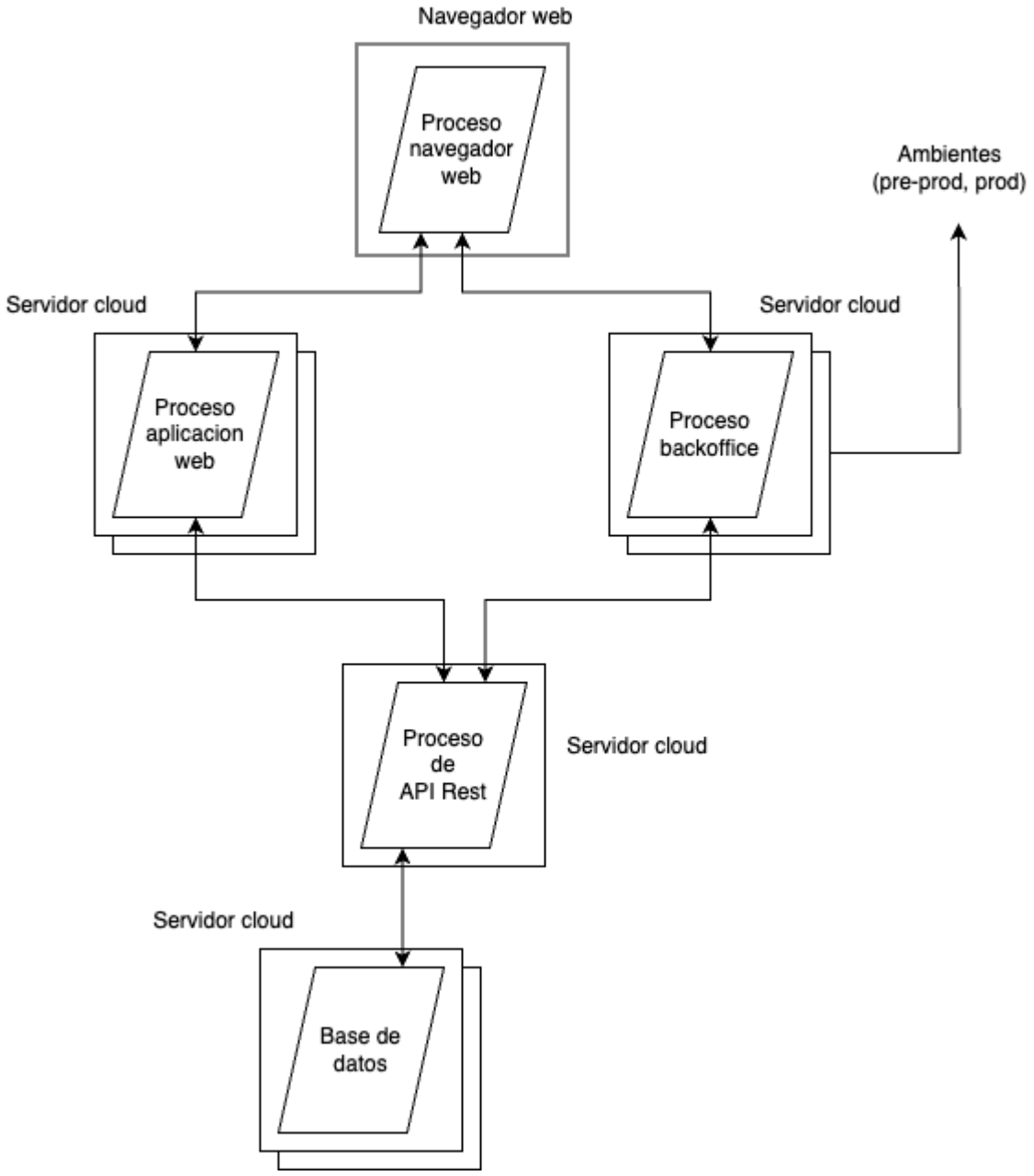
- **Capa de aplicaciones web:** Portal web de deportistas para consultar medallas, eventos y clasificaciones. Backoffice para administradores con la funcionalidad de subir clasificaciones y eventos. Usa la capa de componentes para mostrar la funcionalidad.
    - Tecnologías: NodeJS 18, NextJS 14

- **Capa de componentes:** Casos de uso implementados a través de servicios que manipulan las entidades de la capa de backend.
    - Tecnologías: React 18.2, TypeScript 5.2

- **Capa de backend:** Entidades de Java / Spring que representan al dominio de la aplicación, aplica las reglas de negocio y se comunica con la base de datos.
    - Tecnologías: Spring Boot v3.1, Java 17

- **Capa cloud:** Instancias de la base de datos y de las aplicaciones corriendo en un servidor en la nube.
    - Tecnologías: PostgresSQL 14

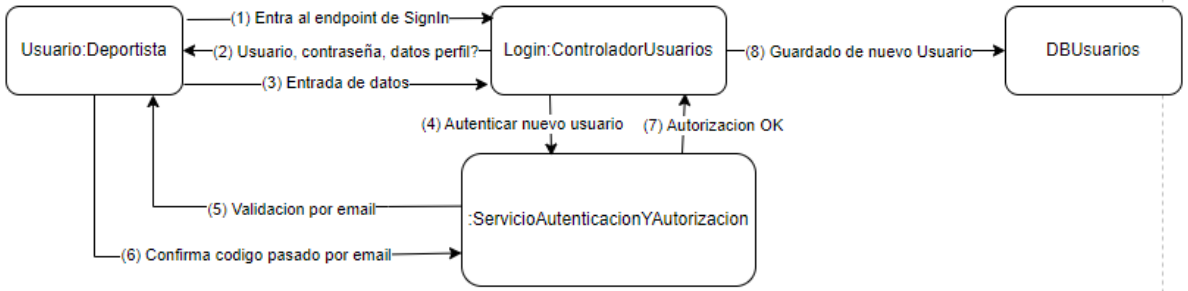
Vista física



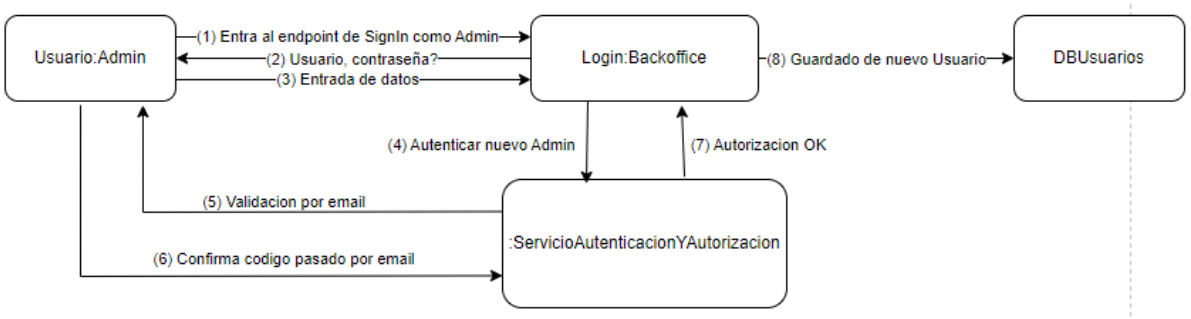
Para cada proceso de aplicación y base de datos contar con un ambiente de desarrollo (o pre productivo) que permita realizar pruebas en ambientes controlados que no expongan información sensible de los usuarios

Vista de escenarios

Alta de usuarios

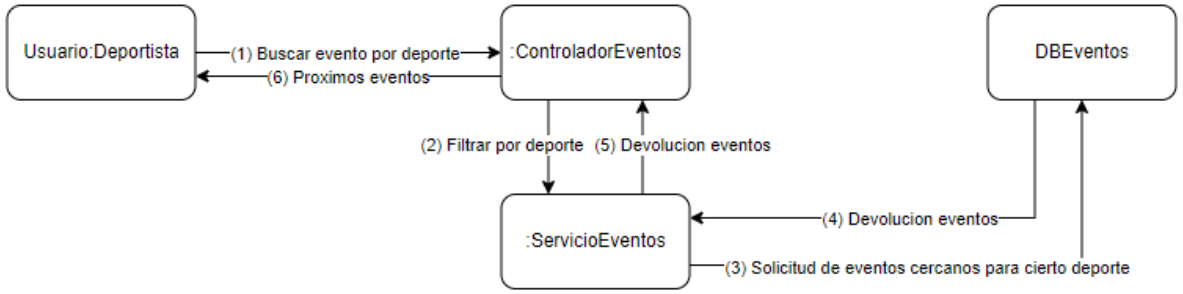


“Dado que soy deportista, cuando deseo loguearme a Medallero, el sistema me pide un usuario y contraseña que luego debo confirmar via email”



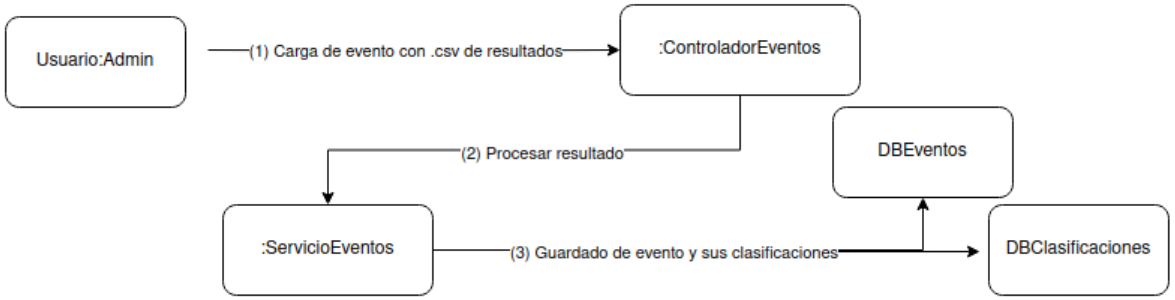
“Dado que soy administrador de la plataforma, cuando deseo loguearme al backoffice, el sistema me pide un usuario y contraseña que luego debo confirmar via email”

Busqueda de Eventos



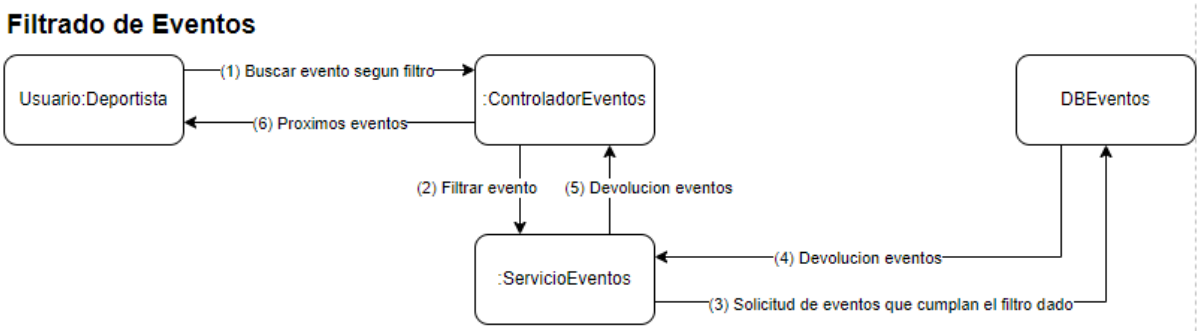
"Dado que soy deportista, cuando deseo buscar algún evento que me interese, el sistema filtra por el tipo de deporte que practico y lo muestra en el resultado de la búsqueda"

Carga de Resultados



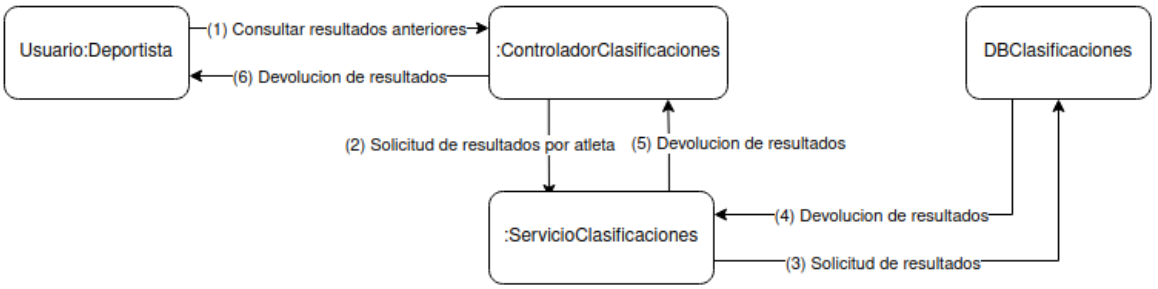
"Dado que soy administrador, cuando deseo cargar los resultados de alguna carrera, el sistema los guarda como resultados oficiales"

Filtrado de Eventos



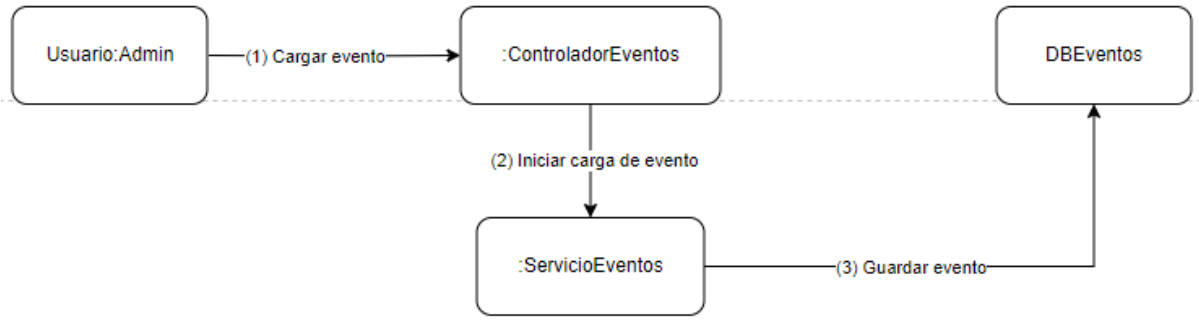
"Dado que soy deportista, cuando deseo buscar algún evento a partir de un tipo de filtro, el sistema muestra aquellos eventos que cumplan dicho filtro"

Muestra de Resultados



“Dado que soy deportista, cuando deseo ver mis resultados de eventos en los que participe, el sistema me muestra dichos resultados en mi perfil”

Alta de Eventos



“Dado que soy administrador, cuando añado un evento nuevo a Medallero, el sistema lo guarda y tiene listo para ser visto por los usuarios”

Obstáculos y evolución del proyecto

Durante la realización del trabajo práctico nos encontramos los siguientes problemas y cómo los solucionamos

- Spring Boot security, security es un tema famosamente temido por devs de springboot, es difícil de implementar, pero logramos llevarlo a cabo de una manera satisfactoria.
- Habíamos planificado tener dos bases de datos para eventos y usuarios, usando Neo4j y PostgreSQL respectivamente.
  El uso de una base en base de grafos como neo4j, está construida de una forma muy útil para analytics, y aplicaba muy bien para el caso.
  Pero al intentar configurar la base de Neo4j vimos que no funcionaban correctamente las migraciones con Docker a la hora de desplegar la aplicacion, asi que decidimos usar solo la base de Postgres que nos funcionaba bien, nuestra experiencia con neo4j, fue que el uso de querys personalizables estaba muy verde.
- Notamos que React no tiene de forma nativa inyección de dependencias ni usa arquitectura limpia para sus aplicaciones por lo que tuvimos que agregar paquetes de Node que nos permitan inyectar los servicios en los casos de uso que definimos

en la aplicación y los repositorios de cada servicio. También decidimos usar un Singleton para la comunicación con el backend que guarda las credenciales del usuario para usar la API Rest que para hacerlo funcional debimos usar variables globales que mantengan las credenciales y que no se pierdan en cada inyección.

Esta complicación nos obligó que en toda comunicación en el backend que necesitaba credenciales, no pueda usarse ‘use client’, y tenga que ser server side rendered.

Conclusiones

Las dos aplicaciones que proponemos como solución a la automatización de participación en eventos y manejar un medallero propio cumplen con lo comprometido para la entrega final del trabajo práctico, que incluye la creación de perfiles de atletas y administradores, recuperación de contraseñas, creación de eventos con resultados y visualización de medallas propias y de otros deportistas. Las tecnologías utilizadas para cumplir estos requerimientos nos sirvieron para aprender sobre el estado actual del mercado de software y la utilización de patrones de diseño y arquitectura limpia nos facilitó poder implementar soluciones apropiadas a los problemas que nos encontramos durante el desarrollo para crear un producto mantenible, reutilizable y ameno para desarrolladores que quieran agregar nuevas funcionalidades a la solución. Con Jira y Gitlab pudimos asignarnos tareas de desarrollo para luego revisarlas con code reviews e integrarlas de forma que todos participamos en la implementación y revisión de código que presentamos.