Sistema de Torneos e Competencias Deportivas

**Nome Alumno/a:**

***Jorge Sobrino Mojón***

**Curso: *2º DAM* Materia:** ***Proxecto Final Módulo Acceso a Datos***

**Docente:** ***Javier Feijóo López***

***🗐 Proxecto: https://github.com/Jsobrino98/ProyectoFinalAD***

Contido

[1. Introdución 2](#_Toc189548783)

[2. Deseño da Base de Datos – Modelo EER 2](#_Toc189548784)

[3. Tecnoloxías Empregadas e Configuración da Contorna 2](#_Toc189548785)

[4. Deseño do Backend – API REST con Spring Boot 2](#_Toc189548786)

[5. Implementación de Autenticación e Seguridade 3](#_Toc189548787)

[6. Desenvolvemento do Backend con Thymeleaf 3](#_Toc189548788)

[7. Documentación da API e Probas en Postman 3](#_Toc189548789)

[8. Conclusións e Valoración Persoal 3](#_Toc189548790)

[9. Anexos (Opcional) 3](#_Toc189548791)

# 1. Introdución

**Finalidad del sistema:** El objetivo principal de este sistema es gestionar información relacionada con equipos deportivos, jugadores, torneos y partidos. Permite registrar y consultar los datos de los equipos, los jugadores que los conforman, los torneos en los que participan, y los partidos que se disputan. Además, proporciona la capacidad de gestión de usuarios y autenticación para distintos roles dentro de la plataforma.

**Problema que resuelve:** Este sistema resuelve la necesidad de centralizar y gestionar la información sobre deportes y competiciones en una sola plataforma. Facilita el seguimiento de los equipos, jugadores y partidos, mejorando la organización y la accesibilidad de los datos. Además, permite la administración de usuarios con distintos permisos (como administradores y usuarios normales) para gestionar el contenido del sistema de manera eficiente.

**Usuarios principales:**

* **Administradores**: Encargados de gestionar equipos, jugadores, torneos, partidos y usuarios. Tienen acceso completo a todas las funcionalidades del sistema.
* **Usuarios normales**: Usuarios registrados que pueden consultar información sobre equipos, jugadores y partidos, así como interactuar con el contenido del sistema según los permisos establecidos.

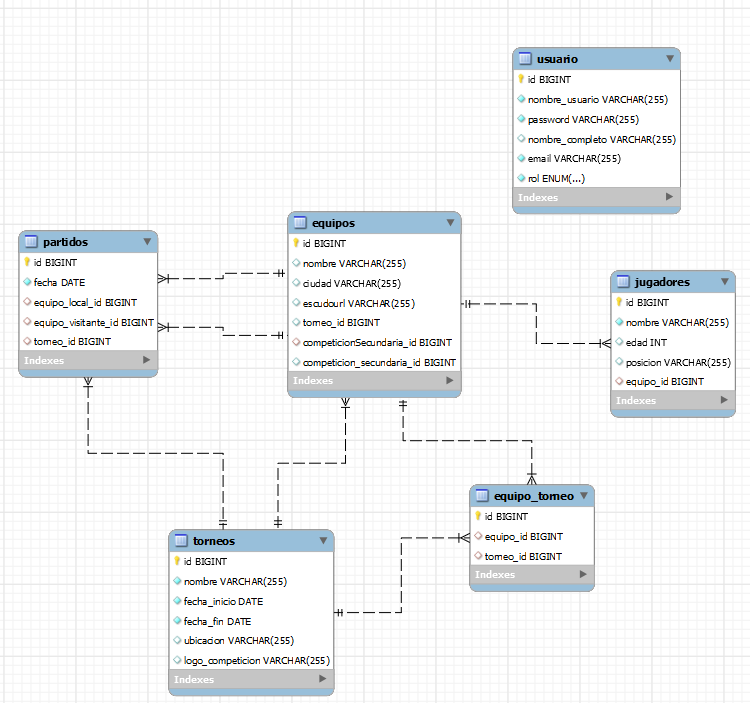
**Resumen de las tecnologías empleadas:**

* **Spring Boot (Java)**: Framework principal para el desarrollo de la aplicación backend, facilitando la creación de APIs RESTful.
* **MySQL**: Base de datos relacional utilizada para almacenar la información de equipos, jugadores, torneos, partidos y usuarios.
* **Postman**: Herramienta utilizada para la creación y prueba de los endpoints de la API.
* **Swagger**: Herramienta integrada para la documentación automática de la API, permitiendo una fácil visualización y pruebas de los endpoints.
* **Thymeleaf**: Motor de plantillas utilizado para generar las vistas dinámicas en el frontend, facilitando la interacción del usuario con el sistema.

# 2. Deseño da Base de Datos – Modelo EER

Explicación do deseño da base de datos, incluíndo:

* Modelo Entidade-Relación Estendido (EER) cunha imaxe.



**Equipos**

* **id**: Identificador único del equipo.
* **nombre**: Nombre del equipo.
* **ciudad**: Ciudad del equipo.
* **escudourl**: Imagen del escudo

Relaciones:

* Relacionado con jugadores a través de equipo\_id.
* Relacionado con partidos como equipo\_local\_id y equipo\_visitante\_id.

**Jugadores**

* **id**: Identificador único del jugador.
* **nombre**: Nombre del jugador.
* **edad**: Edad del jugador.
* **equipo\_id**: Identificador del equipo al que pertenece (clave foránea a equipos).

**Torneos**

* **id**: Identificador único del torneo.
* **nombre**: Nombre del torneo.
* **fecha\_inicio**: Fecha de inicio del torneo.
* **fecha\_fin**: Fecha de finalización del torneo.
* **ubicacion**: Pais donde se juega.
* **logo\_competicion:** Imagen de la competición**.**

Relaciones:

* Relacionado con partidos a través de torneo\_id.

**Partidos**

* **id**: Identificador único del partido.
* **torneo\_id**: Identificador del torneo (clave foránea a torneos).
* **equipo\_local\_id**: Identificador del equipo local (clave foránea a equipos).
* **equipo\_visitante\_id**: Identificador del equipo visitante (clave foránea a equipos).
* **fecha**: Fecha del partido.

Relaciones:

* Relacionado con resultados a través de partido\_id.

**Usuarios**:

* **id**: Identificador único del usuario.
* **nombre\_usuario**: Nombre único del usuario para autenticación.
* **password**: Contraseña encriptada del usuario.
* **nombre\_completo**: Nombre completo del usuario.
* **email**: Correo electrónico único del usuario.
* **rol**: Rol del usuario en el sistema (ADMIN o NORMAL).

Relaciones:

* Relacionado con publicaciones a través de usuario\_id.
* Relacionado con pedidos (en caso de un sistema de compras) a través de usuario\_id.

# 3. Tecnoloxías Empregadas e Configuración da Contorna

* Linguaxe de programación: **Java 17**
* Framework: **Spring Boot .**
* Base de datos: **MySQL.**
* ORM: **Spring Data JPA.**
* Autenticación: **Spring Security + JWT.**
* Frontend para administradores: **Thymeleaf + Bootstrap.**
* Ferramentas de documentación: **Swagger, Postman.**
* Xestión de dependencias: **Maven**.

Incluír pasos para configurar o contorno:

* IDE recomendado (**IntelliJ, VisualStudioCode**).

# 4. Deseño do Backend – API REST con Spring Boot

Explicación da arquitectura do backend:

**AD.SistemaTorneosCompeticions**

**│── config/** # Configuración general (Swagger, CORS, JPA, etc.)

**│── controllers**/ # Controladores REST (endpoints de la API)

**│── models/** # Modelos de datos (entidades JPA)

**│── models/DTO/** # Objetos de transferencia de datos (DTOs)

**│── repositories/** # Repositorios para acceso a la base de datos (JPA)

**│── security/** # Seguridad (JWT, autenticación, filtros)

**│── services/** # Servicios con la lógica de negocio

**│── Application.java** # Clase principal de Spring Boot

**Controllers**: Los controladores manejan las peticiones HTTP y se comunican con los servicios.

**Models**: Las entidades JPA representan las tablas de la base de datos.

**Repositories**: Los repositorios son interfaces que acceden a la base de datos usando **Spring Data JPA.**

**Services**: Los servicios contienen la lógica de negocio.

**Security**: Maneja autenticación y autorización con **JWT**.

**Config:** Aquí se encuentra la configuración del **Swagger**.

# 5. Implementación de Autenticación e Seguridade

Explicación da seguridade no sistema:

### **Registro e Inicio de Sesión con JWT**

* **Registro:** Se almacena el usuario en la base de datos con la contraseña encriptada usando BCrypt.
* **Inicio de sesión:** Se valida el usuario y contraseña, y si son correctos, se genera un token JWT.
* **Uso del token:** El cliente debe enviar el JWT en el encabezado Authorization en cada petición protegida.

### **Protección de Endpoints con Roles (USER, ADMIN)**

* **Definición de roles:** Los usuarios pueden tener roles como ADMIN o NORMAL.
* **Restricción de acceso:** En SecurityConfig, se define qué endpoints requieren qué roles (hasRole("ADMIN"), hasAnyRole("NORMAL", "ADMIN")).
* **Filtro JWT:** Se interceptan las peticiones y se verifica el token antes de permitir el acceso.

### **Encriptación de Contraseñas con BCrypt**

* **Uso de BCryptPasswordEncoder:** Se encripta la contraseña antes de guardarla en la base de datos.
* **Comparación segura:** Cuando el usuario inicia sesión, la contraseña ingresada se compara con la encriptada en la base de datos usando matches().

# 6. Desenvolvemento do Backend con Thymeleaf

**Configuración de seguridad**: Asegurar que solo los usuarios con el rol ADMIN puedan acceder a las rutas del panel de administración (/admin/\*\*), mediante la configuración de seguridad en la clase SecurityConfig.

**Controlador**: Crear un controlador (AdminController) que maneje las solicitudes a las rutas del panel. Este controlador debe verificar que el usuario esté autenticado y luego renderizar la vista del panel, pasando información relevante como el nombre del usuario logueado.

**Vista con Thymeleaf**: Crear una vista en Thymeleaf (dashboard.html) donde se mostrará el panel de administración. Esta vista puede incluir enlaces a funcionalidades adicionales, como la gestión de usuarios, estadísticas o configuraciones.

**Estilo y diseño**: Personalizar el diseño del panel utilizando archivos CSS para mejorar la apariencia y hacerla más funcional.

# Interfaz de usuario gráfica, Aplicación El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Aplicación El contenido generado por IA puede ser incorrecto.7. Documentación da API e Probas en Postman

Interfaz de usuario gráfica, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

* Para realizar pruebas en Postman:

1. Registrar un Usuario:

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

\**CAMPOS OBLIGATORIOS:* (***nombreUsuario***, ***password***, ***email)*** *no se pueden repetir* y ***rol (“ADMIN”, “NORMAL”)***.

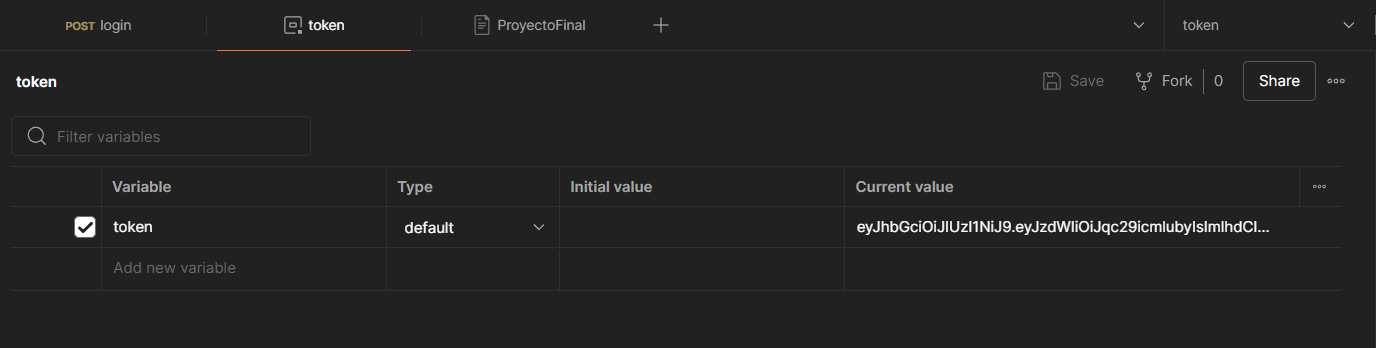
2. Hacer el login:

- Introducir nombreUsuario y contraseña (como minimo). Los datos son correctos, nos imprime el código del **token**, que tendrá una duración de 1h.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

* Código que añadiremos a la variable creada llamada **token.**



* En los Headers, añadimos la key **AUTHORIZATION,**  y como value la palabra **Bearer** seguido de la variable **token.**
* EJ:
  + Listado de los Usuarios registrados (sacando el DTO sin la contraseña visible)

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

# 8. Conclusións e Valoración Persoal

Reflexión sobre o desenvolvemento do proxecto:

### **Dificultades y Soluciones**

Lo más complicado fue implementar la seguridad con JWT y Spring Security, ya que hubo problemas con la configuración y dependencias cíclicas. Se resolvió organizando mejor los componentes de seguridad y asegurando que cada uno tuviera una función clara. También se usó BCrypt para encriptar contraseñas y se definieron bien los permisos de acceso según el rol del usuario.

### **Mejoras para Futuras Versiones**

* Ampliar la base de datos con más información sobre torneos, equipos y jugadores.
* Mejorar la gestión de usuarios, permitiendo recuperar contraseñas o cambiar roles.
* Optimizar la seguridad y la autenticación para mayor eficiencia.

# 9. Anexos (Opcional)

* Script de Base la Base de datos en la carpeta de Documentación.
* Recomendaciones:
  + Ejecutar tabla por tabla a la hora de crear la base de datos, ya que puede dar errores con las claves foraneas si se ejecutan todas de golpe.