**** **SportTech**

***GITHUB:*** [***https://github.com/Jsobrino98/ProyectoFinal***](https://github.com/Jsobrino98/ProyectoFinal)

**Nome Alumno/a: *Jorge Sobrino Mojón***

**Curso: *2º DAM***

**Módulo:** ***Proxecto Final Ciclo***

## Contido

[**1. Introducción 3**](#_Toc195008895)

[**2. Obxectivos 3**](#_Toc195008898)

[**3. Situación previa 4**](#_Toc195008901)

[**4. Tecnoloxías empregadas 4**](#_Toc195008911)

[**5. Solución proposta 5**](#_Toc195008921)

[**6. Planificación do proxecto 6**](#_Toc195008929)

[**7. Desenvolvemento e execución 13**](#_Toc195008930)

[**8. Validación, accesibilidade e testing 14**](#_Toc195008945)

[**9. Webgrafía 15**](#_Toc195008954)

[**10. Anexo I – Manual técnico de instalación 16**](#_Toc195008955)

[**11. Anexo II – Documentación de uso (manuais usuario) 18**](#_Toc195008956)

[**12. Anexo IV – Documentación técnica da API 20**](#_Toc195008957)

**13. Dificultades atopadas e solucións aplicadas …………………………………………………………..….24**

## 1. Introducción

## O proxecto seleccionado consiste no desenvolvemento dun sistema de xestión de información relacionada coas ligas de fútbol. A aplicación permite consultar que equipos están en cada liga, visualizar os seus escudos, coñecer os 10 xogadores máis caros de cada liga e acceder á lista de gañadores do Balón de Ouro dende o primeiro ata a actualidade.

## A principal aplicación deste sistema é proporcionar unha ferramenta accesible para os afeccionados ao fútbol, xornalistas deportivos e analistas, permitindo un acceso rápido e estruturado a información relevante sobre as principais ligas, equipos que as conforman e xogadores.

## 2. Obxectivos

## Os obxectivos iniciais do proxecto:

* Desenvolver unha plataforma web que permita consultar información sobre equipos, xogadores e premios individuais de forma sinxela e estruturada.
* Garantir unha experiencia de usuario intuitiva e accesible, cun deseño visual atractivo.
* Proporcionar datos actualizados e organizados sobre os xogadores máis valiosos e os gañadores do Balón de Ouro.

## Crear unha API RESTful para facilitar a integración e o acceso á información.

## 3. Situación previa

## Antes do desenvolvemento deste proxecto, a información sobre os equipos deportivos, xogadores e torneos era xestionada de forma manual ou a través de sistemas pouco integrados. Isto dificultaba o acceso rápido aos datos, a organización da información e a seguridade dos mesmos.

## Problemas identificados:

## Falta dun sistema centralizado que compile a información relevante de forma clara e accesible.

## Dificultade para atopar datos específicos sobre os xogadores máis caros de cada liga e o historial dos Balóns de Ouro.

## Dependencia de múltiples plataformas para obter esta información, o que pode resultar ineficiente para os usuarios interesados.

## Alternativas existentes:

## Uso de follas de cálculo e documentos compartidos, que presentan limitacións en seguridade e escalabilidade.

## Plataformas de xestión deportiva de terceiros, pero con custos elevados ou falta de personalización.

## O noso sistema propón unha solución eficiente e adaptada ás necesidades dos usuarios, ofrecendo unha plataforma flexible, segura e escalable.

## 4. Tecnoloxías empregadas

## Para o desenvolvemento do proxecto utilizáronse as seguintes tecnoloxías:

## Linguaxes de programación: Java para o backend e HTML, CSS e JavaScript para o frontend.

## Frameworks: Spring Boot para a implementación do backend e Thymeleaf para a xeración de vistas dinámicas.

## Base de datos: MySQL, utilizada para almacenar a información de equipos, xogadores, torneos, partidos e usuarios.

## Ferramentas de desenvolvemento:

## Postman para probar os endpoints da API.

## Swagger para documentar a API e facilitar a súa utilización.

## Spring Security para a autenticación e xestión de permisos.

## Estas tecnoloxías permitiron crear un sistema robusto, eficiente e de fácil mantemento, garantindo unha boa experiencia de usuario e seguridade dos datos.

## 5. Solución proposta

## A solución proposta consiste nunha plataforma web centralizada para a consulta de información sobre equipos e xogadores nas diferentes ligas. A diferenza doutras alternativas existentes, este sistema destaca pola súa sinxeleza, rapidez e acceso á información clave para os afeccionados ao fútbol e profesionais do ámbito deportivo.

## Diferencias clave con outras alternativas:

## Centralización dos datos: A información de equipos, xogadores, torneos gárdase nunha base de datos unificada, evitando a fragmentación e a perda de datos.

## Accesibilidade e usabilidade: A plataforma proporciona unha interface intuitiva con visualizacións claras e filtros avanzados para a consulta da información.

## Seguridade: A integración de Spring Security permite unha autenticación robusta e unha xestión de permisos eficaz, protexendo os datos sensibles.

## Automatización e eficiencia: Grazas á API RESTful, o sistema pode integrarse con outras ferramentas ou servizos externos, permitindo a automatización de tarefas e a interoperabilidade con outras plataformas.

## En resumo, a solución proposta non só resolve os problemas identificados na situación previa, senón que ademais ofrece unha plataforma moderna, segura e escalable, adaptada ás necesidades dos usuarios e das organizacións deportivas.

## 6. Planificación do proxecto

### 6.1 Modelo de desarrollo

Para levar a cabo este proxecto empregouse un **modelo de desenvolvemento en cascada**, o que facilita unha organización estruturada e eficiente. Este modelo divídese en varias fases secuenciais, onde cada etapa debe completarse antes de pasar á seguinte. Deste xeito, garántese unha progresión lóxica e controlada no desenvolvemento do software, asegurando que cada fase se execute de forma ordenada e sen solapamentos..

### 6.2 Fases do proxecto

**Semana 1-2: Planificación**

* Definir requisitos, obxectivos e alcance do proxecto.
* Crear diagramas de fluxo e esquema da arquitectura.

**Semana 3-4: Deseño**

* Crear prototipos UI/UX.
* Definir estrutura de base de datos e API REST.

**Semana 5: Configuración Inicial**

* Configurar Spring Boot (Backend) e Angular (Frontend).
* Conectar ambos con base de datos MySQL.

**Semana 6-10: Desenvolvemento Backend**

* Crear modelos, controladores e servizos en Spring Boot.
* Implementar seguridade con Spring Security.
* Desenvolver endpoints da API REST.

**Semana 8-12: Desenvolvemento Frontend**

* Desenvolver compoñentes e vistas en Angular.
* Conectar o frontend co backend mediante API REST.
* Aplicar estilos e optimizar UX/UI.

**Semana 13-14: Probas**

* Realizar probas unitarias e de integración.
* Corrección de erros e melloras.

**Semana 15-16: Despregue e Mantemento**

* Configurar servidor para despregue.
* Crear documentación técnica e de usuario.
* Monitorización e axustes finais.

### Gráfico6.3 Diagrama de Gantt:

### Diagrama, Esquemático El contenido generado por IA puede ser incorrecto.6.4 Diagrama de Casos de Uso

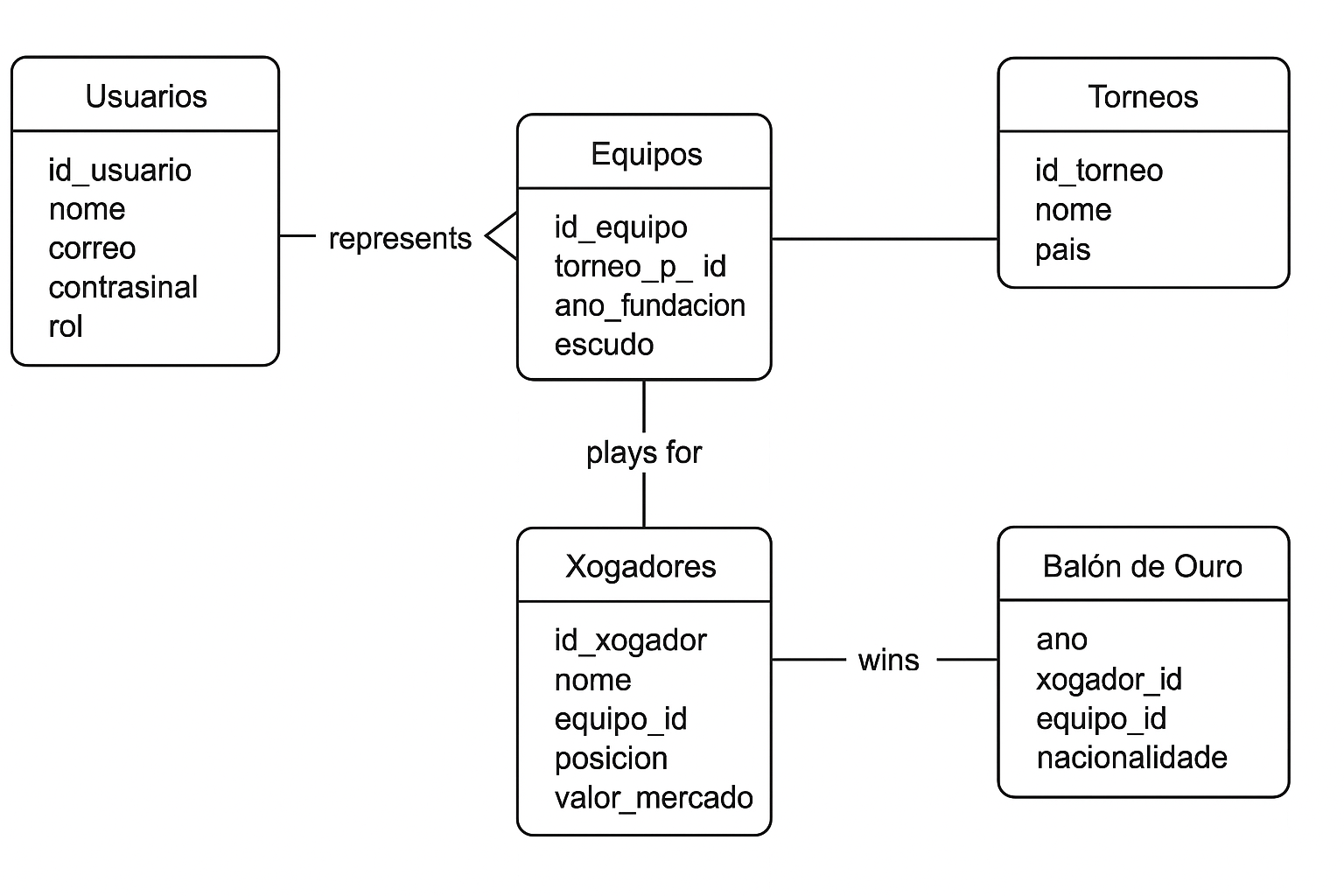
### 6.5 Táboas da Base de Datos

* **Entidades Principais:**
* **Usuarios**: Contén información dos usuarios rexistrados (id, nome, correo, contrasinal, rol).
* **Equipos**: Datos dos equipos deportivos (id, nome, liga, ano de fundación, escudo).
* **Xogadores**: Información sobre os xogadores (id, nome, equipo, posición, valor de mercado).
* **Torneos**: Representa as distintas ligas dispoñibles na plataforma (id, nome, país, categoría).
* **Balón de Ouro**: Historial dos gañadores do premio ao mellor xogador (ano, xogador, nacionalidade, equipo).
* **Relacións:**
* Un **equipo** pertence a un único **torneo**, pero un **torneo** pode ter varios **equipos**.
* Un **xogador** pertence a un único **equipo**, pero un **equipo** pode ter varios **xogadores**.
* Un **usuario** pode consultar información de varias **torneos**, **equipos** e **xogadores**.
* Un **xogador** pode ter gañado o **Balón de Ouro** en varios anos.

Esta estrutura proporciona un modelo de datos optimizado para consultas rápidas e acceso eficiente á información relevante sobre as ligas de fútbol e os xogadores destacados.

### Diagrama6.6 Diagrama de Entidad-Relación

### 6.7 Diagrama de Modelo Relacional



### 6.8 Presuposto Completo

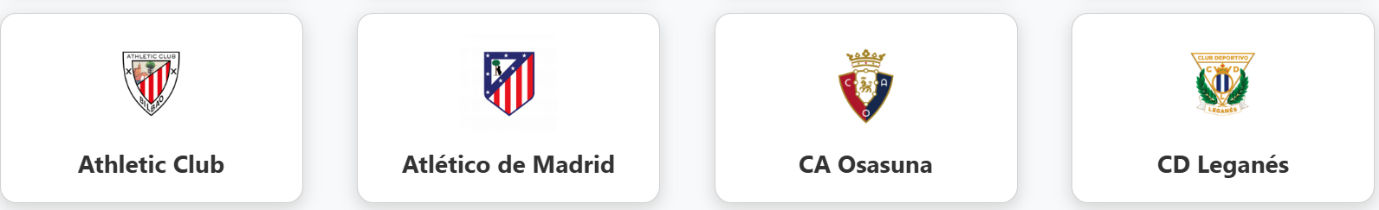
|  |  |
| --- | --- |
| Hardware: | |
| Ordenador completo (Intel i7, 16GB RAM, 1TB SSD) | 1200 € |
| Pantalla Full HD 24" | 200 € |
| Periféricos (teclado e rato) | 50 € |
| Subtotal: | **1450 €** |

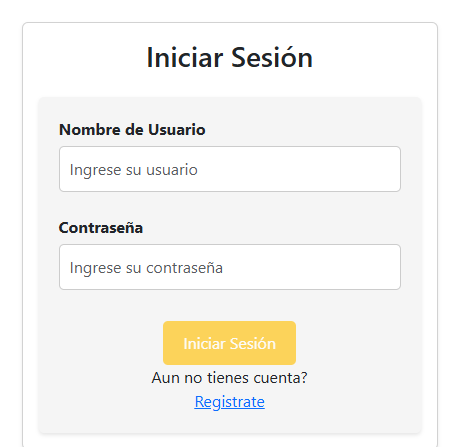
|  |  |
| --- | --- |
| Software e ferramentas: | |
| Sistema Operativo (Windows 11 ou MacOS) | 150 € |
| |  | | --- | |  |  |  | | --- | | IntelliJ IDEA Community (IDE para Java) | | Gratuito |
| |  | | --- | |  |  |  | | --- | | MySQL Workbench | | Gratuito |
| |  | | --- | |  |  |  | | --- | | Postman (Probas API) | | Gratuito |
| |  | | --- | |  |  |  | | --- | | Visual Studio Code (Editor de código) | | Gratuito |
| |  | | --- | |  |  |  | | --- | | Figma (Deseño UI/UX) | | Gratuito |
| |  | | --- | |  |  |  | | --- | | Google Chrome | | Gratuito |
| Draw.io (Diagramas) | Gratuito |
| Subtotal: | **150 €** |

|  |  |
| --- | --- |
| Total Xeral: | |
| Subtotal Hardware | **1450 €** |
| Subtotal Software | 150 € |
| Subtotal: | **1600 €** |

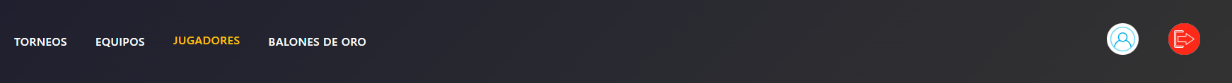
### 6.9 Mockups

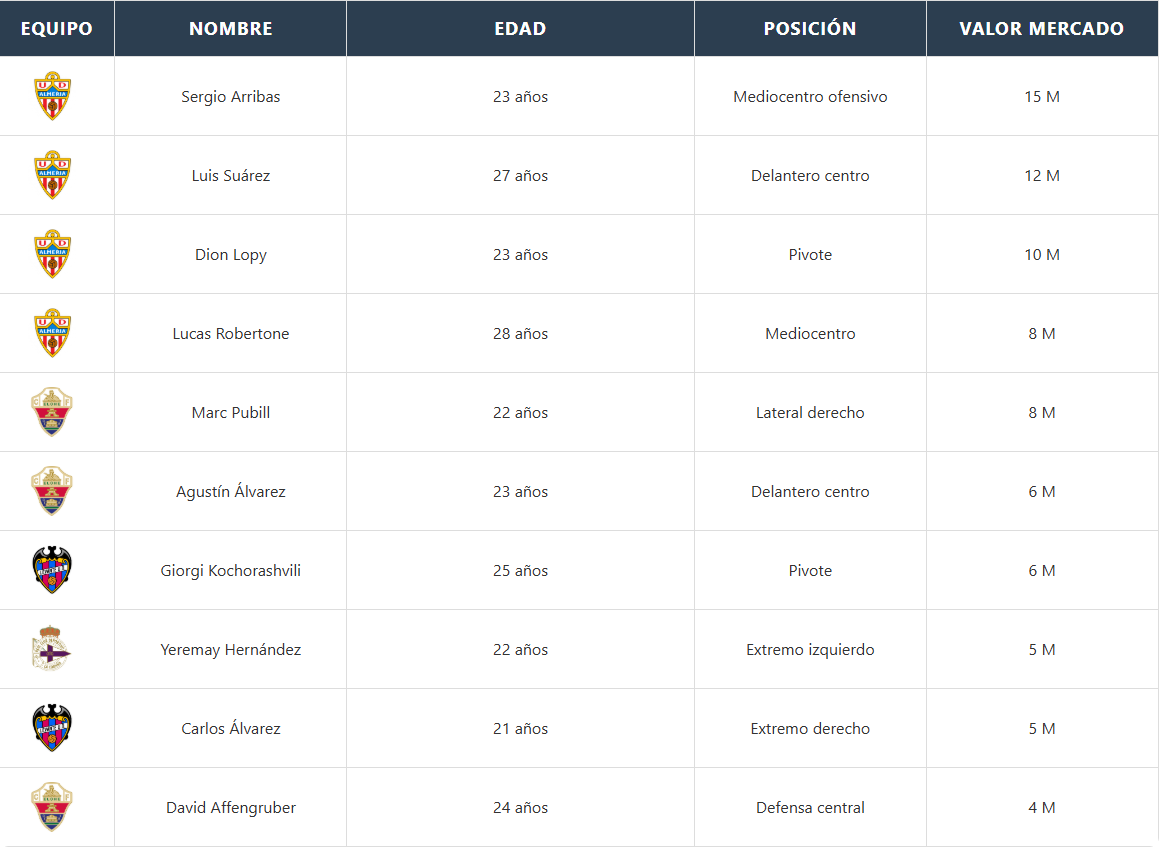
* + **Interfaz de usuario gráfica, Aplicación

    El contenido generado por IA puede ser incorrecto.Torneos:**
  + **Equipos:**
  + **Inicio Sesion/Registro**

****

****

* + **Balones de Oro**
  + **Jugadores:**

****

## 7. Desenvolvemento e execución

## Este apartado describe o proceso de despregamento da aplicación Sportech, así como o seu funcionamento técnico xeral.

### 7.1 Diagrama de despregamento

## O diagrama de despregamento mostra a infraestrutura utilizada para executar a aplicación, detallando os compoñentes do sistema, servidores, base de datos e interaccións entre eles.

### 7.2 Arquitectura da aplicación

## A aplicación está baseada nunha arquitectura cliente-servidor, cun backend desenvolvido en Spring Boot e un frontend en Angular. Os principais compoñentes son:

## Frontend: Desenvolto en Angular, encargado da interface de usuario.

## Backend: Implementado en Spring Boot, manexa a lóxica de negocio e expón unha API RESTful.

## Base de datos: MySQL almacena os datos de usuarios, torneos, equipos e xogadores.

## Autenticación e seguridade: Implementado con Spring Security e tokens JWT.

### 7.3 Proceso de despregamento

## Configuración do servidor

## O backend é despregado nun servidor con Tomcat e base de datos en MySQL.

## O frontend é servido en NGINX ou similar.

## Despregue en contedores (opcional)

## Uso de Docker para xestionar as imaxes do backend e frontend.

## Monitorización e mantemento

## Uso de ferramentas como Prometheus e Grafana para monitorizar o estado da aplicación.

## 8. Validación, accesibilidade e testing

### 8.1 Proceso de validación

## A validación do sistema realizouse mediante as seguintes técnicas:

## Revisión de requisitos: Comprobación de que se cumpren os obxectivos definidos inicialmente.

## Probas de usabilidade: Testeo con usuarios para verificar que a interface é intuitiva e sinxela.

## Probas de seguridade: Verificación da autenticación e permisos dos usuarios.

### 8.2 Accesibilidade

## Para garantir a accesibilidade, aplicáronse os seguintes principios:

## Uso de etiquetas ARIA en compoñentes da interface.

## Contrastes adecuados e fontes lexibles segundo as normas WCAG 2.1.

## Compatibilidade con lectores de pantalla para persoas con discapacidade visual.

### 8.3 Testing

A validación da aplicación realizouse de maneira manual, asegurándome de que cada funcionalidade da aplicación funcionase correctamente en diferentes fases do desenvolvemento. O proceso de testing consistiu en:

1. **Probas funcionais**:  
   Durante o desenvolvemento da aplicación, utilicei IntelliJ para executar o backend (Spring Boot) e ***"ng serve -o"*** para iniciar o frontend de Angular. A partir de aí, revisaba as diferentes pantallas e funcionalidades da aplicación, comprobando que os datos se visualizaban correctamente e que as interaccións co usuario eran as esperadas.
2. **Probas de interface de usuario (UI)**:  
   De maneira manual, validaba a interface de usuario, asegurándome de que as diferentes pestanas (como torneos, equipos, xogadores e Balón de Ouro) funcionaban correctamente. Aseguraba que a navegación entre as páxinas fose fluída e que non houbera erros ao cargar os datos.
3. **Probas de usabilidade**:  
   A aplicación foi probada en termos de accesibilidade e usabilidade. Validaba que a interface fose sinxela de navegar e accesible, garantindo que os usuarios puidesen atopar a información de maneira sinxela e intuitiva.

## 9. Webgrafía

* **Diario AS. (2024).** *Noticias de fútbol e estatísticas deportivas*. Recuperado o 2 de abril de 2025 de <https://as.com>
* **Federación Internacional de Fútbol Asociación (FIFA). (2024).** *Clasificacións e estatísticas oficiais do fútbol mundial*. Recuperado o 2 de abril de 2025 de <https://www.fifa.com>
* **Google. (2024).** *Google Images - Búsqueda de imágenes deportivas e escudos de equipos*. Recuperado o 2 de abril de 2025 de https://www.google.com/imghp
* **Transfermarkt. (2024).** *Base de datos sobre valores de mercado de xogadores e estatísticas de traspasos*. Recuperado o 2 de abril de 2025 de <https://www.transfermarkt.com>
* **Unión de Asociacións Europeas de Fútbol (UEFA). (2024).** *Información sobre torneos europeos e estatísticas de xogadores*. Recuperado o 2 de abril de 2025 de <https://www.uefa.com>
* **Wikipedia. (2024).** *Información sobre torneos e xogadores de fútbol*. Recuperado o 2 de abril de 2025 de <https://www.wikipedia.org>

## 10. Anexo I – Manual técnico de instalación

### 1. Requisitos mínimos

**Hardware:**

* Procesador: Intel i5 ou superior
* Memoria RAM: 8 GB (16 GB recomendado)
* Almacenamento: 2 GB libres
* Resolución mínima: 1366x768

**Software:**

* Sistema operativo: Windows 10/11, macOS 12+, ou distribución Linux actualizada.
* Conexión a Internet: para descargar dependencias e realizar probas.

### 2. Instalación do entorno

#### 2.1 Requisitos previos

* + - IntelliJ IDEA (versión Community é suficiente)
    - MySQL Server 8.x
    - Postman (para probar os endpoints da API)
    - Node.js e npm (versión recomendada: Node 18+)
    - Angular CLI (instalación vía npm install -g @angular/cli)
    - Visual Studio Code (opcional pero recomendado)

#### 2.2 Pasos de instalación

**Backend:**

1. Instalar Java JDK e configurar a variable de entorno JAVA\_HOME.
2. Instalar IntelliJ IDEA e abrir o proxecto backend.
3. Configurar application.properties con as credenciais correctas para a base de datos:

spring.datasource.url=jdbc:mysql**://localhost:3306/sportech**

spring.datasource.username=**usuario**

spring.datasource.password=**contrasinal**

1. Crear a base de datos sportech en MySQL manualmente ou habilitar a creación automática coas propiedades JPA
2. Executar a clase principal SportechApplication.java desde IntelliJ para iniciar o backend.
3. Usar Postman para probar os endpoints da API (GET, POST, etc.).

**Frontend (Angular con Visual Studio Code):**

Abrir unha terminal e navegar ata o directorio do proxecto Angular.

1. Executar: npm install para instalar as dependencias necesarias.
2. Iniciar o servidor de desenvolvemento con: ng serve -o

Isto abrirá automaticamente o navegador en **http://localhost:4200** cunha vista previa da aplicación.

### 3. Conexión entre Frontend e Backend

Para permitir a comunicación entre Angular e o backend Spring Boot:

* + Asegúrate de que o backend estea activo en **http://localhost:8080.**
  + No backend, engade a anotación **@CrossOrigin** nos controladores:

**@CrossOrigin(origins = "http://localhost:4200")**

* + No frontend, verifica que as chamadas HTTP usen a URL correcta dos endpoints.

### 4. Consideracións finais

Executar ambos servizos á vez: primeiro backend (IntelliJ), logo frontend (VS Code/terminal).

## 11. Anexo II – Documentación de uso (manuais usuario)

### 1. Acceso a aplicación

* A aplicación está dispoñible localmente en http://localhost:4200.
* Non require autenticación para o acceso básico á información.

### 2. Pantalla de inicio

* Mostra unha visión xeral da aplicación cun menú principal.
* Desde aquí pódese acceder ás diferentes seccións: **Torneos**, **Equipos**, **Xogadores**, e **Balón de Ouro**.

### 3. Seccións principais

#### 3.1 Torneos

* Lista todos os torneos dispoñibles.
* Ao facer clic nun torneo, móstranse os equipos participantes.

#### 3.2 Equipos

* Visualización dos equipos con escudo, nome e país.
* Ao seleccionar un equipo, móstrase o seu cadro de xogadores.

#### 3.3 Xogadores

* Mostra os 10 xogadores máis valiosos de cada torneo.
* Inclúe datos como nome, valor de mercado e equipo actual.

#### 3.4 Balón de Ouro

* Lista dos gañadores históricos do Balón de Ouro.
* Inclúe: nome, ano, equipo e unha imaxe representativa do xogador.

### 4. Funcionalidades adicionais

* **Filtros**: Filtrar por país, ano ou valor de mercado.
* **Responsive**: A interface adáptase a móbiles e tabletas.

### 5. Recomendacións de uso

* Utiliza a aplicación en navegadores actualizados como Google Chrome ou Microsoft Edge.
* Para unha mellor experiencia, recoméndase unha resolución mínima de 1366x768.

|  |  |  |
| --- | --- | --- |
| Problema | Causa posible | Solución |
| Non carga a información | Backend apagado | Iniciar o backend en IntelliJ |
| Erro 404 nos datos | Endpoint mal escrito | Verificar rutas na consola |
| Non conecta coa base de datos | MySQL non iniciado | Arrancar o servidor MySQL |

### 6. Posibles problemas e solucións

## 12. Anexo IV – Documentación técnica da API

### Endpoints RESTful

**Base URL: /api/equipo**

**🔍 GET /api/equipo**

**Descrición:** Obtén a lista de todos os equipos rexistrados.

**🔍 GET /api/equipo/{id}**

**Descrición:** Obtén os datos dun equipo específico segundo o seu ID.

**➕ POST /api/equipo**

**Descrición:** Garda un novo equipo no sistema.

**✏️ PUT /api/equipo/{id}**

**Descrición:** Actualiza os datos dun equipo existente. O id da URL debe coincidir co do corpo da petición.

**❌ DELETE /api/equipo/{id}**

**Descrición:** Elimina un equipo segundo o seu ID.

**🔍 GET /api/equipo/torneo/{torneoId}**

**Descrición:** Lista os equipos asociados a un torneo específico.

**Base URL**: **/api/balon\_oro**

**📥 GET /api/balon\_oro**

**Descrición**: Obtén a lista de todos os xogadores que gañaron o Balón de Ouro.

**📥 GET /api/balon\_oro/{id}**

**Descrición**: Obtén a información dun xogador gañador do Balón de Ouro segundo o seu id.

**📤 POST /api/balon\_oro**

**Descrición**: Crea un novo rexistro de xogador gañador do Balón de Ouro.

**🛠 PUT /api/balon\_oro/{id}**

**Descrición**: Actualiza a información dun xogador gañador.

**❌ DELETE /api/balon\_oro/{id}**

**Descrición**: Elimina un xogador gañador do Balón de Ouro.

**🎯 GET /api/balon\_oro/equipo/{idEquipo}**

**Descrición**: Obtén todos os xogadores do Balón de Ouro pertencentes a un equipo específico.

**Base URL: /api/torneo**

**🔍 GET /api/torneo**

**Descrición**: Obtén a lista de todos os torneos rexistrados.

**🔍 GET /api/torneo/{id}**

**Descrición**: Obtén os datos dun torneo específico segundo o seu ID.

**➕ POST /api/torneo**

**Descrición**: Crea un novo torneo no sistema.

**✏️ PUT /api/torneo/{id}**

**Descrición**: Actualiza un torneo existente. O id na URL debe coincidir co do corpo da petición.

**❌ DELETE /api/torneo/{id}**

**Descrición**: Elimina un torneo segundo o seu ID.

**Base URL: /api/jugador**

**🔍 GET /api/jugador**

**Descrición**: Obtén a lista de todos os xogadores rexistrados.

**🔍 GET /api/jugador/{id}**

**Descrición**: Obtén os datos dun xogador específico segundo o seu ID.

**➕ POST /api/jugador**

**Descrición**: Crea un novo xogador no sistema.

**✏️ PUT /api/jugador/{id}**

**Descrición**: Actualiza un xogador existente. O id na URL debe coincidir co do corpo da petición.

**❌ DELETE /api/jugador/{id}**

**Descrición**: Elimina un xogador segundo o seu ID.

**🔍 GET /api/jugador/equipo/{idEquipo}**

**Descrición**: Obtén a lista de xogadores dun equipo específico segundo o seu ID.

**Base URL: /api/usuario**

🔍 **GET /api/usuario?usernameOrEmail={usernameOrEmail}**  
Descrición: Permite buscar un usuario proporcionando o seu nome de usuario ou correo electrónico.

🔍 **GET /api/usuario**  
Descrición: Obtén a lista de todos os usuarios rexistrados.

🔍 **GET /api/usuario/{id}**  
Descrición: Obtén os datos dun usuario específico segundo o seu ID.

➕ **POST /api/usuario**  
Descrición: Crea un novo usuario no sistema.

✏️ **PUT /api/usuario/{id}**  
Descrición: Actualiza un usuario existente. O ID na URL debe coincidir co do corpo da petición.

❌ **DELETE /api/usuario/{id}**  
Descrición: Elimina un usuario segundo o seu ID.

🔍 **GET /api/usuario**  
Descrición: Obtén a lista de todos os usuarios do sistema (alternativo a listarUsuarios).

## 13. Dificultades atopadas e solucións aplicadas

Durante o desenvolvemento do proxecto SportTech, xurdiron diversos desafíos tanto técnicos como organizativos. A continuación, recóllense as principais dificultades atopadas e as solucións aplicadas en cada caso:

**1. Integración entre frontend e backend**

* *Dificultade:* Ao principio, existían erros CORS e problemas na comunicación entre Angular e Spring Boot.
* *Solución:* Engadiuse a anotación @CrossOrigin(origins = "http://localhost:4200") nos controladores de Spring Boot, e asegurouse que ambas aplicacións corrían nos portos adecuados.

**2. Xestión de seguridade con Spring Security**

* *Dificultade:* A implementación de autenticación e permisos causaba conflitos na exposición de certos endpoints públicos.
* *Solución:* Configurouse correctamente a clase de seguridade, establecendo rutas públicas e privadas de forma explícita, e probouse o sistema mediante Postman.

**3. Deseño responsivo e compatibilidade con dispositivos móbiles**

* *Dificultade:* Algunhas vistas non se adaptaban ben en pantallas pequenas.
* *Solución:* Usouse CSS flexbox e media queries para mellorar a adaptabilidade, probando en distintos tamaños de pantalla e navegadores.

**4. Complexidade na estrutura da base de datos**

* *Dificultade:* As relacións entre xogadores, equipos e torneos provocaban problemas á hora de facer consultas anidadas.
* *Solución:* Redeseñáronse algunhas entidades e usáronse joins e consultas personalizadas en JPA para optimizar o rendemento.

**5. Documentación da API e proba de endpoints**

* *Dificultade:* Inicialmente, a API non tiña unha documentación clara e completa.
* *Solución:* Implementouse Swagger para xerar documentación interactiva dos endpoints REST, e usouse Postman para validar as operacións CRUD