



## **TRABAJO FIN DE CICLO**

**[InaScoot]**

Autor: [Ignacio Juan Moreno Martín]

Tutor: [Ramón Salado Lucena]

Fecha: [17-06-2024]

**Técnico Superior en Desarrollo de Aplicaciones Multiplataforma**



## Índice

<b>1. Abstract</b>	<b>3</b>
<b>2. Introducción</b>	<b>3</b>
a. Estructura del Proyecto	3
b. Objetivo General	3
c. Método y Metodología	3
<b>3. Matriz DAFO</b>	<b>3</b>
a. Fortalezas	3
b. Oportunidades	3
c. Debilidades	4
d. Amenazas	4
<b>4. Análisis del Mercado</b>	<b>4</b>
<b>5. Análisis y Diseño del Software</b>	<b>4</b>
a. Casos de Uso	4
b. Diagrama UML	5
c. Diagrama de Flujo	5
<b>6. Cronograma del proyecto</b>	<b>6</b>
<b>7. Problemas encontrados y Soluciones Propuestas</b>	<b>6</b>
<b>8. Codificación</b>	<b>6</b>
<b>9. Interfaces</b>	<b>6</b>
<b>10. Pruebas</b>	<b>6</b>
<b>11. Futuras Mejoras</b>	<b>6</b>
<b>12. Bibliografía y Referencias</b>	<b>7</b>
<b>13. Anexos</b>	<b>7</b>
a. Manual del Programa	7
b. Proceso de Instalación	7
c. Otros	7

# 1. Abstract

Este proyecto será denominado InaScout. Este es un proyecto diseñado para visualizar y gestionar información de jugadores del juego RPG Inazuma Eleven. La función principal es ayudar a los usuarios a tener un acceso más concreto a personajes secundarios los cuales puedes unir a tu plantilla y así facilitar el desarrollo de su partida y darle a conocer jugadores, equipos y mucho más. Para contextualizar Inazuma Eleven es un juego de fútbol el cual se basa en superar obstáculos ganando torneos y partidos de fútbol 11, pero contiene elementos que no son reales como supertécnicas las cuales son desarrolladas por los personajes tanto fichables como los que el juego te cede para iniciar la historia. Este juego marcó mi infancia y por desgracia no tuve mucho acceso a información con lo cual quiero ayudar a aquellos usuarios nuevos a que tengan ciertas facilidades creando este proyecto el cual hice por niños como el que niño que yo fui a los 7 años feliz creando equipos. El proyecto seguirá en desarrollo y ya ha sido visto por muchos fans de la saga, lo cual me anima a seguir mejorándolo.

# 2. Introducción

InaScoot es una aplicación web desarrollada en Python-Web. Es una aplicación con un diseño atractivo, simple, pero a la vez intuitiva para facilitar al usuario su uso invitando al usuario a conocer en profundidad aspectos muy a tener en cuenta en los videojuegos de Inazuma Eleven.

## a. Estructura del Proyecto

Existen dos componentes principales en este proyecto:

- Backend: El backend se compone de dos partes:
  1. Servidor: Tengo un servidor con una estructura Docker donde monté un servidor para MySQL. La creación del contenedor sería con ese comando que escribes por consola:

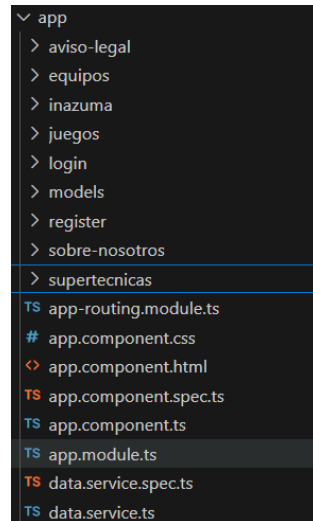
`$ docker run --name some-mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql`

siendo mysql pass la contraseña del servidor. Luego realicé la creación tanto de tablas como la inserción de datos.

2. API: Espera las peticiones del front-end proporcionando la información para ser gestionada y disfrutada por los usuarios.

- Frontend:

1. Angular: Es la parte visual de nuestra aplicación que hace que el usuario pueda visualizar e interactuar con los datos de nuestra aplicación. Dividido en componentes los cuales son:



## b. Objetivo General

Mi objetivo es crear un proyecto sin ánimo de lucro para aquellas personas que también aman esta saga a mejorar su experiencia, facilitar y mejorar el encuentro de jugadores con gran potencial para hacer que sus equipos tengan elementos distintos a los comunes, así como tener acceso a una página explícita de reclutamiento de jugadores en español castellano.

## 3. Matriz DAFO

### a. Fortalezas

DAFO matriz de InaScoot Fortalezas:

Posibilidad de implementación de datos sencilla: Es fácil añadir datos ya que las bases de datos están creadas y solo hay que añadir datos mediante la inserción.

Interfaz sencilla y orientada a la temática principal lo cual hace amena su navegación para los fans habituales y conocedores de la saga.

Contenido bien distinguido y dosificado ya que no hemos creado una página y metido todos los datos sino que cada tipo de datos está separado por componentes en Angular para una visualización más concreta.

Filtros sencillos que facilitan al usuario interactuar y buscar el contenido que desea rápidamente.

## b. Oportunidades

Expansión de la base de datos aumentando la lista de jugadores, supertécnicas y equipos teniendo en cuenta que los juegos es algo que no aumenta con tanta frecuencia porque no es una saga que desarrolle juegos en pocos años.

Colaboraciones con expertos de la saga para explorar nuevas oportunidades de mercado y expandir la página a más personas e incluso países.

## c. Debilidades

La principales debilidades de dicho proyecto son:

Proyecto individual: Al ser un proyecto individual supone que todas las ideas, conceptos e información deba ser investigada y desarrollada por un único sujeto y estos proyectos suelen ser colectivos y de larga duración. Con esto no trato de criticar como se nos exige entregar el trabajo, solo hablo de que el resultado de la aplicación influye en el número de personas que se involucren dentro de él.

En conclusión, este proyecto seguirá en desarrollo, pero será un desarrollo que tendrá actualizaciones en función de la disponibilidad del creador.

## d. Amenazas

Preferencias del usuario: Para un desarrollador es importante poder abarcar un gran público para su producto y el hecho de que este producto no pueda encajar en un porcentaje de las personas puede que haga que pierda popularidad.

## 4. Análisis del Mercado

Existen grandes aplicaciones creadas por fans de la comunidad y aunque algunas tienen información similar a la nuestra ninguna trata de forma tan explícita como yo lo he realizado. Un ejemplo de página que podría complementar de forma excelente la mía sería Inazuma Team Builder la cual ha sido diseñada para crear alineaciones con los jugadores, pero no incluye las supertécnicas que tenemos documentadas.

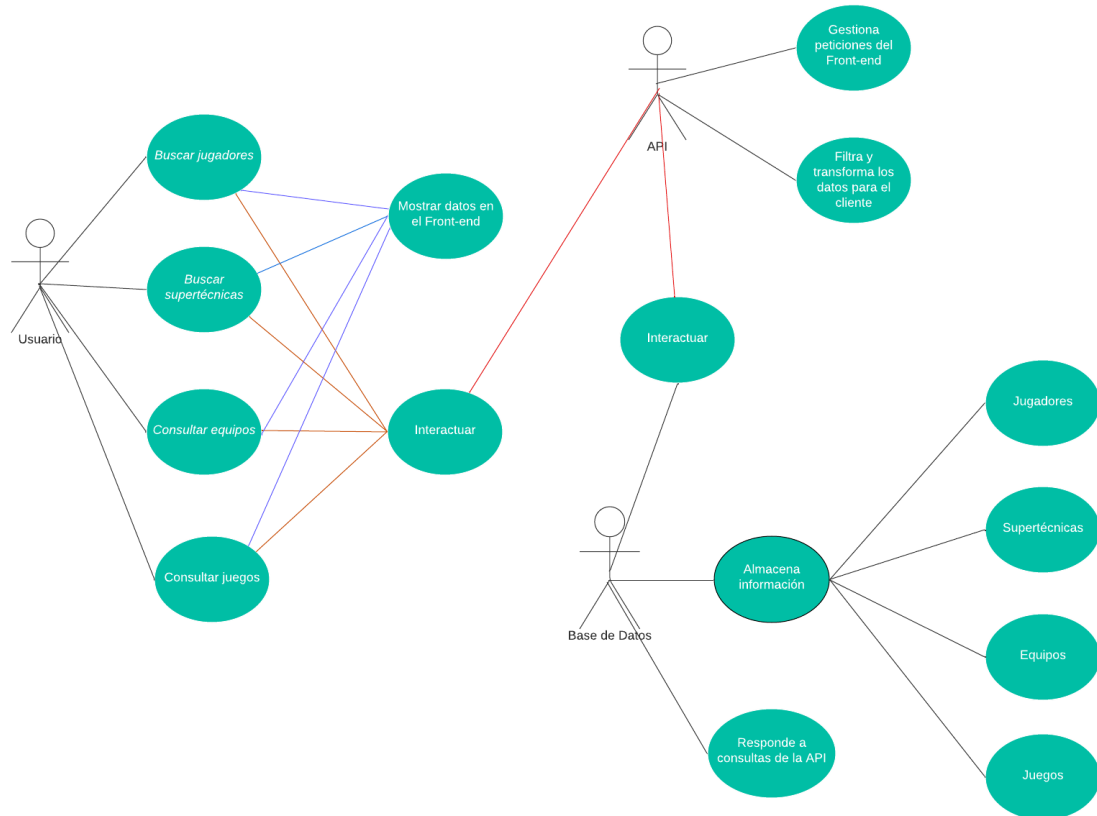
En mi caso en estos momentos la aplicación no tiene ningún ánimo de lucro, ya que esta sigue en desarrollo y tengo intención de mejorarlo y añadirle muchas más cosas.

En un futuro podría incluir un enlace dentro de la página de Paypal o Patreon y aceptar donaciones de usuarios que tengan interés en hacer pequeñas inversiones para así apoyarme a seguir mejorando.

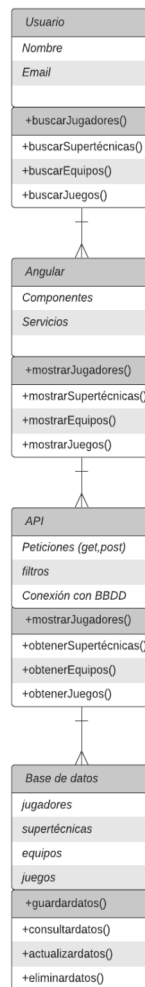
Además de añadir en un futuro próximo una caja de opiniones anónimas para aceptar sugerencias y así darle voz y voto a todos los consumidores. Incluso un servidor de Discord para crear comunidad y así ofrecer al usuario confianza para poder expresarse ya sea de la aplicación como del juego en sí o conocer más personas fanáticas.

## 5. Análisis y Diseño del Software

### a. Casos de Uso

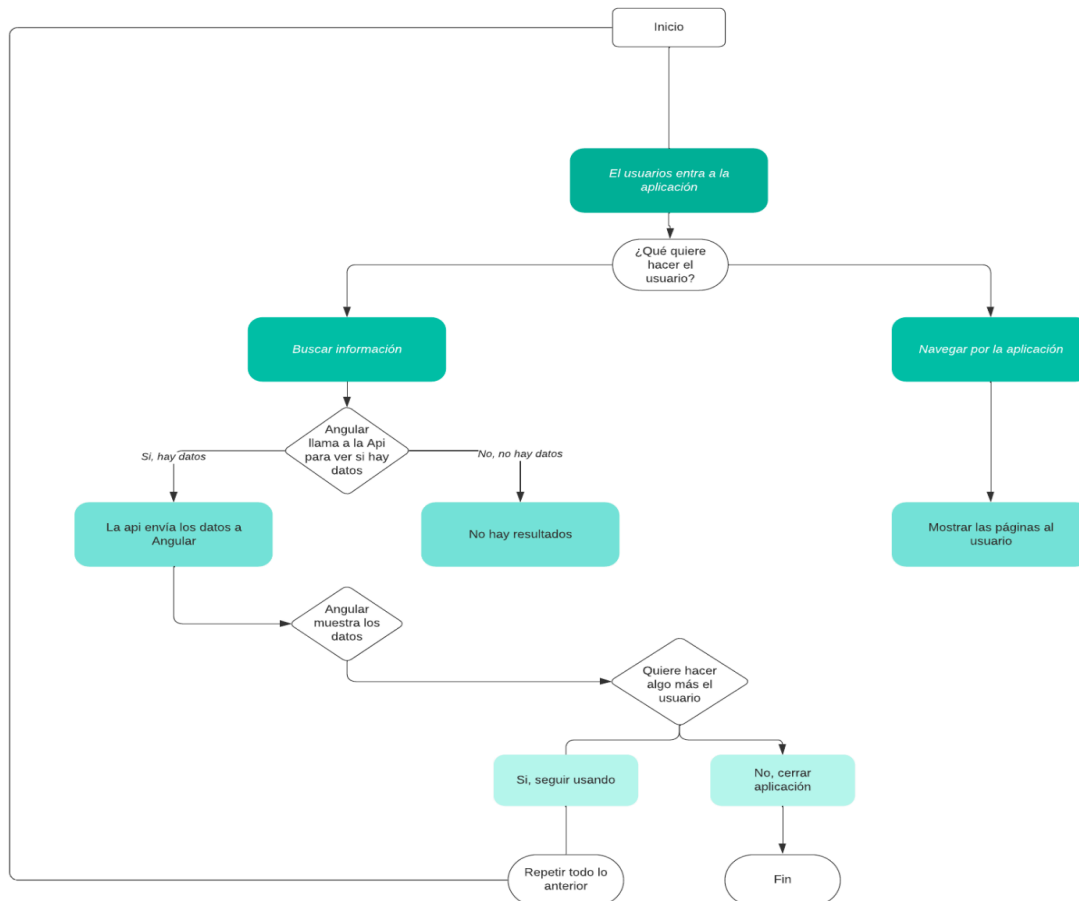


### b. Diagrama UML





## c. Diagrama de Flujo



## 6. Cronograma del proyecto

[Desarrollar una planificación creando diferentes hitos del proyecto y asignando tiempos de ejecución a cada uno de ellos.]

Fase	Semana	Tareas
Planificación y Requisitos	1-2	Inicio de proyecto, investigación y aprendizaje de python y BBDD
Base de Datos	3-6	Realización de la Base de Datos, inserciones a falta de las correcciones que realizaré durante el proyecto. Al ser datos extensos e investigados y escritos a mano me extendí para intentar garantizar una información correcta y fiable
Desarrollo Back	6 - 8	Desarrollo en python realizando tanto app, login y register para poder unir la base de datos con el back y así implementarlo en Angular
Desarrollo Front-end	9 - 12	Desarrollo front-end sumando a la intención de unirlo con el back y que sea visual. Hubo dificultades ya que aunque los códigos están correctos aparentemente el front no recibía la información de la API y era imposible poder hacer el diseño lo cual me hizo demorar en exceso. Este paso dificulta el proyecto.
Diseño Front-end	12	Una vez los datos se podían visualizar el desarrollo era posible y empecé a hacer el diseño ameno y canónico de acuerdo a la saga a la cual dedicamos esta página
Mantenimiento y mejoras	Tarea continua	A partir de que el trabajo a finalizado sumando los consejos de mejora que reciba seguiré trabajando y mejorando la aplicación

## 7. Problemas encontrados y Soluciones Propuestas

1. Problemas en la integración Back-Front: Implementamos CORS tanto en el back como en el navegador como plugin y se comprobó la api para ver si los datos se transfieren correctamente.

2. Conexión API-Angular: A pesar de que el back estaba correctamente y el front no parecía tener problemas de implementación los datos seguían sin aparecer con lo cual decidí añadir un data service para las urls de la api y un función cargar datos en la ts de cada página que fuera necesario para poder conectarla. A pesar de ellos la api empezó a funcionar en el momento que me di cuenta el orden por cual debía ejecutar los .py empezando siempre por app.py para que no hubiera conflicto en las peticiones del front.

## 8. Codificación

- **Front-end**

- Angular: Framework encargado de visualizar los datos.
- CSS: Estilos de los componentes Angular.

- **Backend**

- Flask: Framework encargado de la construcción de la API para poder implementarla
- Sqlite es la librería encargada de unir la base de datos al Back.

- **Método**

- Buscar(): Este método lo he implementado en cada componente de forma separada para crear en el TS un filtro de búsqueda de acuerdo a las características de lo que queremos buscar. Por ejemplo en el caso del componente Inazuma qué es la página principal donde salen los jugadores. La función "buscar()" quedaría así:

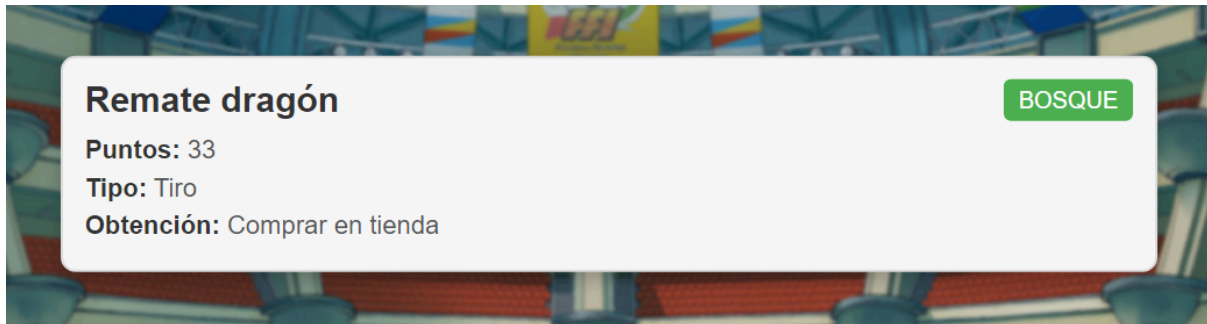
```
buscar(): void {
  const termino = this.terminoBusqueda.toLowerCase();

  this.jugadoresFiltrados = this.jugadores.filter((jugador) => {
    const nombreIncluye = jugador.nombre.toLowerCase().includes(termino);
    const afinidadIncluye = jugador.afinidad.toLowerCase().includes(termino);
    const posicionIncluye = jugador.posicion?.toLowerCase().includes(termino) || false;
    const equipoIncluye = jugador.equipo?.toLowerCase().includes(termino) || false;
    const generoIncluye = jugador.genero.toLowerCase().includes(termino);
    const supertIncluye = jugador.supert?.some((tecnica: string) =>
      tecnica.toLowerCase().includes(termino)
    ) || false;

    return (
      nombreIncluye ||
      afinidadIncluye ||
      posicionIncluye ||
      equipoIncluye ||
      generoIncluye ||
      supertIncluye
    );
  });
}
```

## 9. Interfaces

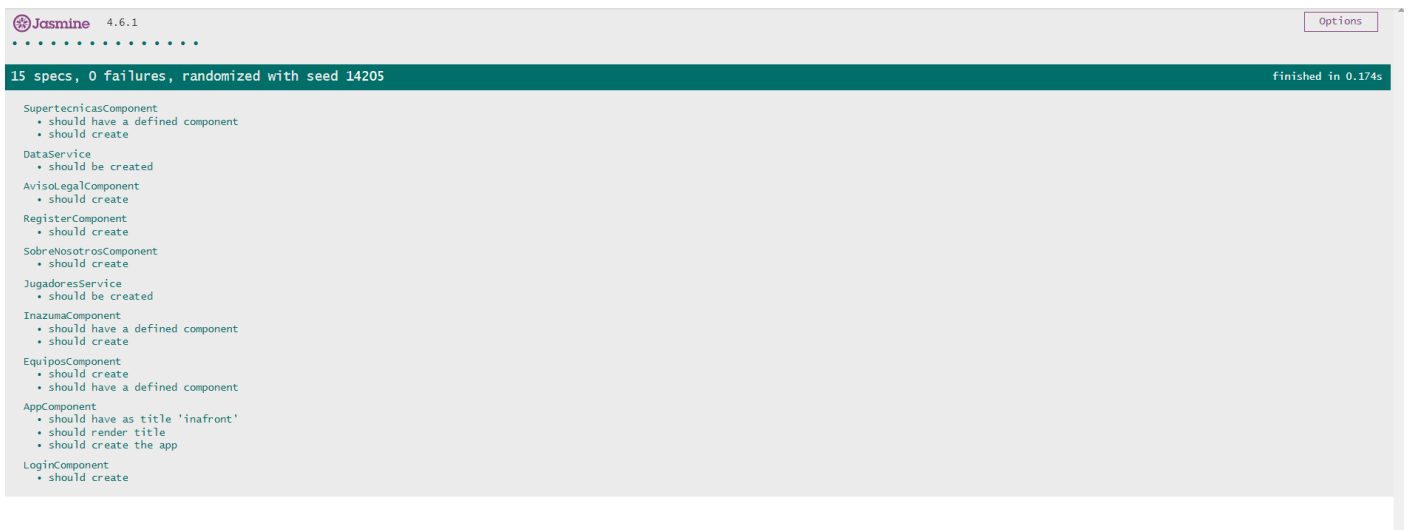
- Barra de navegación superior
  - Permanece a la vista con un buscador y un botón de cerrar sesión.
  - Colores contrastados para facilitar la lectura y navegación.
- Barra lateral de navegación
  - Incluye botones direccionales para cambiar de página y poder navegar fácilmente.
- Panel de visualización de datos
  - Los datos están organizados en unas tarjetas diseñadas de forma de que los datos por separado tengan sus características propias y explicadas de forma explícita. Pongamos un ejemplo en este caso hablemos de supertécnicas:



Como podemos ver en esta imagen tienen la siguiente estructura:

- Puntos
- Tipo
- Obtención
- Afinidad

## 10. Pruebas



He realizado la prueba jasmine. Para realizarla debes hacer ng test donde normalmente realizas ng serve y te saldrán múltiples errores. En mi caso fueron 8 errores en los spect.ts de mis componentes debido a la falta de la importación:

```
import { RouterTestingModule } from '@angular/router/testing';
```

Al añadir la importación he podido solucionarlos correctamente.

## 11. Futuras Mejoras

- Server de Discord para recibir noticias de actualizaciones y añadir canal de sugerencias al igual que la caja de comentarios
- Aumento de información en la base de datos
- Autenticación de la página con Firebase

Esta página sigue en desarrollo con lo cual esto no es más que el principio de una página con mucho potencial y sin apenas páginas similares en español-castellano

## 12. Bibliografía y Referencias

- Creación de la API: <https://youtu.be/b0ZrmhyCY4?si=QCgfT1owG52gRk8g>
- Información: [https://inazuma-eleven.fandom.com/fr/wiki/Wiki\\_Inazuma\\_Eleven](https://inazuma-eleven.fandom.com/fr/wiki/Wiki_Inazuma_Eleven)
- Colaborador de la comunidad Inazuma Eleven: [@AlexiisMach](#)

## 13. Anexos

### a. Manual del Programa

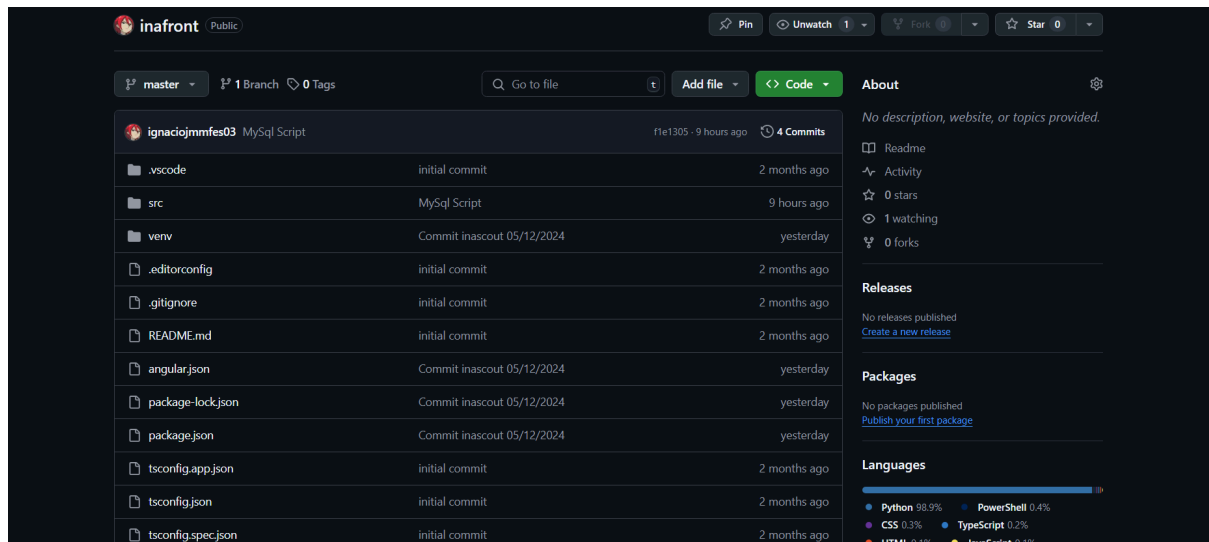
Requisitos:

- MySql Workbench 8.0 / Dbeaver (Entorno de Base de Datos)
- Visual Studio Code
- Navegador Web
- Github

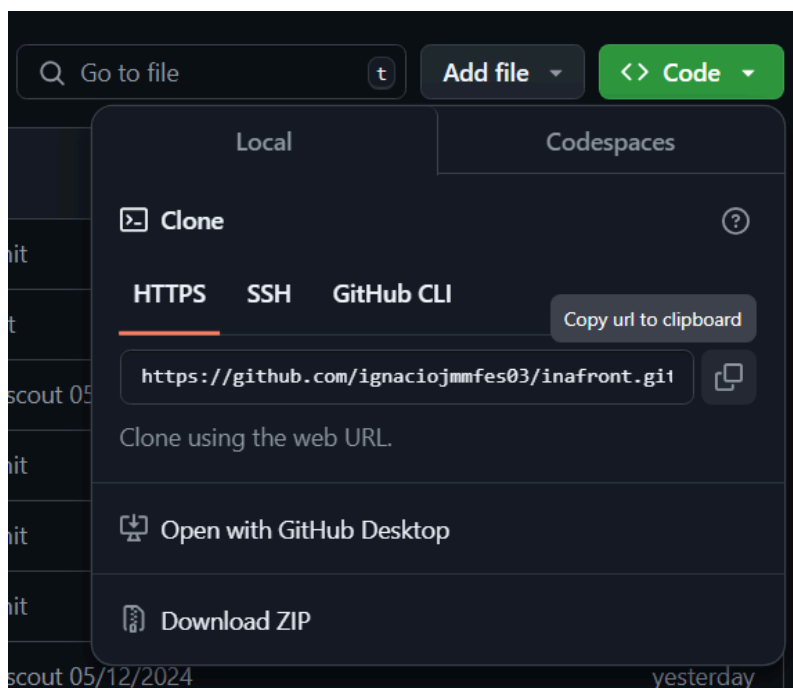
### b. Proceso de Instalación

Accedes a mi repositorio de Github

<https://github.com/ignaciojmmfes03/inafont>



Clic en la pestaña verde code



Copias ese http

- Clonamos el repositorio en Git CMD (instalar Git antes adjunto link)  
<https://www.git-scm.com>
- Una vez lo instalamos volvemos al paso vamos a Git CMD hacéis lo siguiente

```
Git CMD
C:\Users\ignac>git clone https://github.com/ignaciojmmfes03/InaScoot.git
```

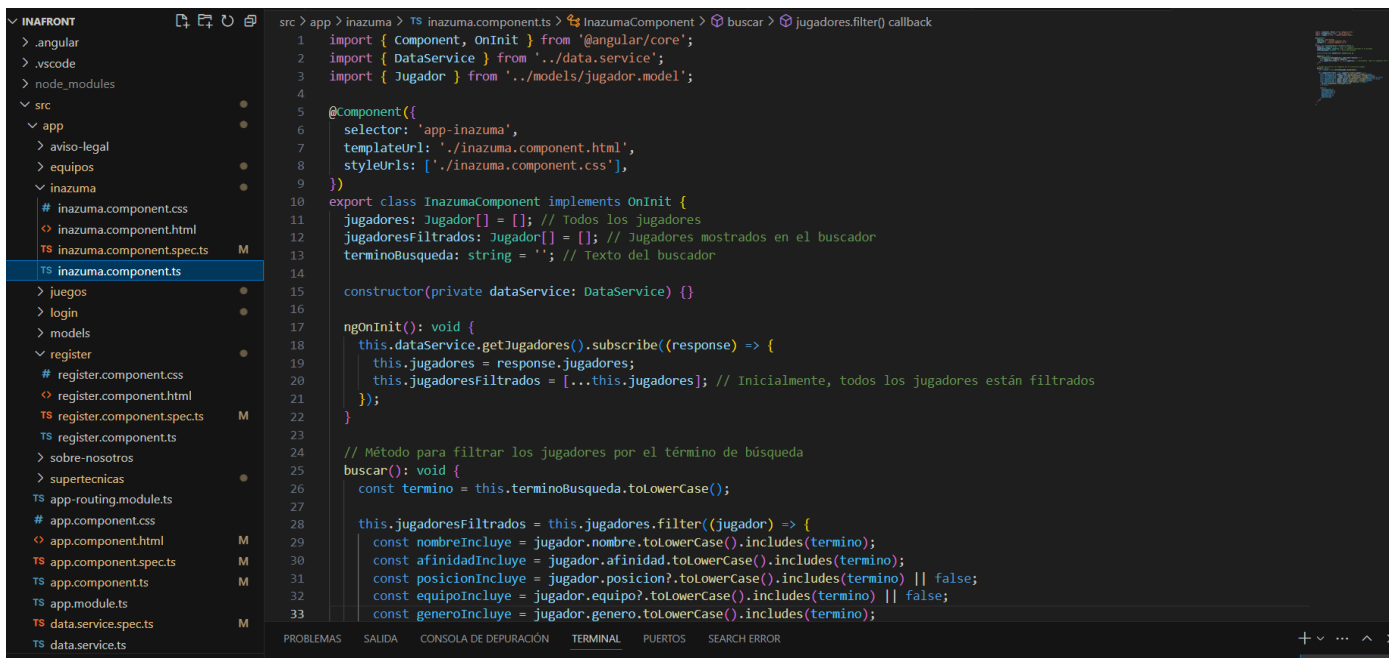
Ahora tendrás un archivo en la raíz que indica el CMD. En caso de que quieras otras la cambias con el comando `cd`

Una vez tienes el archivo lo abres con Visual Studio Code (recomendado)

Para descargar e instalar Visual Studio Code entra aquí

<https://code.visualstudio.com>

Una vez tienes visual abres el archivo en el mismo



```
src > app > inazuma > TS inazuma.component.ts > inazumaComponent > buscar > jugadores.filter() callback
1  import { Component, OnInit } from '@angular/core';
2  import { DataService } from '../data.service';
3  import { Jugador } from '../models/jugador.model';
4
5  @Component({
6    selector: 'app-inazuma',
7    templateUrl: './inazuma.component.html',
8    styleUrls: ['./inazuma.component.css'],
9  })
10 export class InazumaComponent implements OnInit {
11   jugadores: Jugador[] = []; // Todos los jugadores
12   jugadoresFiltrados: Jugador[] = []; // Jugadores mostrados en el buscador
13   terminoBusqueda: string = ''; // Texto del buscador
14
15   constructor(private dataService: DataService) {}
16
17   ngOnInit(): void {
18     this.dataService.getJugadores().subscribe((response) => {
19       this.jugadores = response.jugadores;
20       this.jugadoresFiltrados = [...this.jugadores]; // Inicialmente, todos los jugadores están filtrados
21     });
22   }
23
24   // Método para filtrar los jugadores por el término de búsqueda
25   buscar(): void {
26     const termino = this.terminoBusqueda.toLowerCase();
27
28     this.jugadoresFiltrados = this.jugadores.filter((jugador) => {
29       const nombreIncluye = jugador.nombre.toLowerCase().includes(termino);
30       const afinidadIncluye = jugador.afinidad.toLowerCase().includes(termino);
31       const posicionIncluye = jugador.posicion?.toLowerCase().includes(termino) || false;
32       const equipoIncluye = jugador.equipo?.toLowerCase().includes(termino) || false;
33       const generoIncluye = jugador.genero.toLowerCase().includes(termino);
```

Abres 4 terminales y colocas esto en cada una (en orden):

1º Terminal:

- `cd /src/app`
- `ng serve`



2º Terminal:

- cd src/app/scripts
- python app.py

3º Terminal:

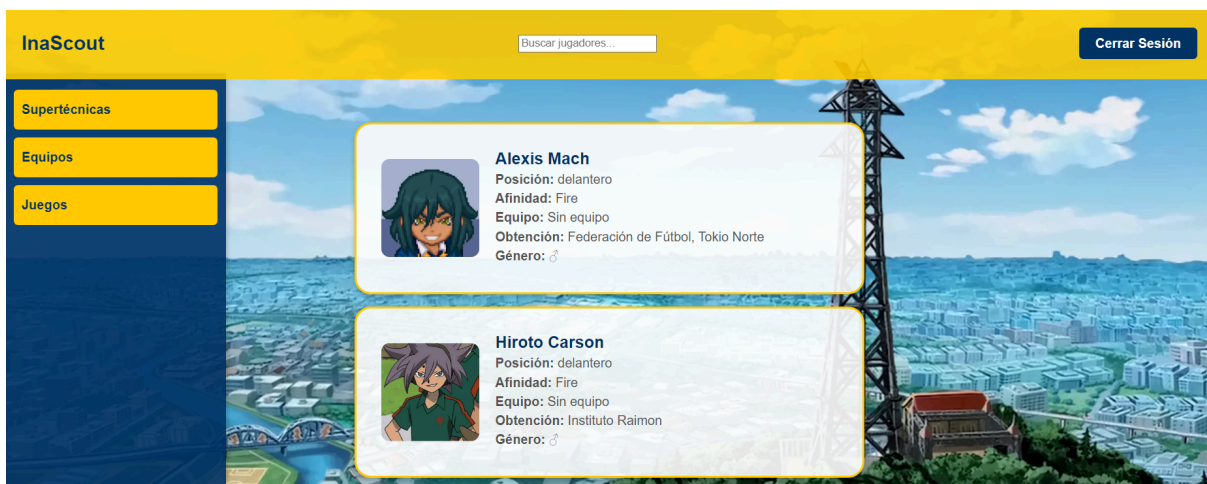
- cd src/app/scripts
- python loginapp.py

4º Terminal:

- cd src/app/scripts
- python register.py

Busca en el navegador: <http://localhost:4200/>

Si todo fue levantado correctamente verás esto:



### c. Otros

Documento con los datos de los jugadores que actualmente han sido implementados (aumentarán conforme se actualice la aplicación):

[https://drive.google.com/drive/u/0/folders/1cnwky9suAMmH1thjSnU\\_IZNUZfyvgN6V](https://drive.google.com/drive/u/0/folders/1cnwky9suAMmH1thjSnU_IZNUZfyvgN6V)