PLATFORM FOR THE ANALYSIS OF SPORTS STATISTICS

FINAL DEGREE PROJECT

PABLO MORENO CANDUELA AND

IVÁN GALLEGO CUERVO

SUPERVISORS

JOSÉ IGNACIO REQUENO JARABO

MARÍA ELENA GÓMEZ MARTÍNEZ

DEGREE IN COMPUTER ENGINEERING

FACULTY OF INFORMATICS

UNIVERSIDAD COMPLUTENSE DE MADRID

JUNE 2024

PLATAFORMA PARA ANÁLISIS DE ESTADÍSTICAS DEPORTIVAS

TRABAJO DE FIN DE GRADO

CURSO 2023-2024

PABLO MORENO CANDUELA E

IVÁN GALLEGO CUERVO

DIRECTORES

JOSÉ IGNACIO REQUENO JARABO

MARÍA ELENA GÓMEZ MARTÍNEZ

GRADO EN INGENIERÍA INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

Junio 2024

DEDICATION

To every passion. To every aspiration.

ACKNOWLEDGEMENTS

It is hard to put into words how grateful we are for our supervisors María Elena Gómez Martínez and José Ignacio Requeno Jarabo and their commitment to guide us through this adventure, without their support this project would have been unachievable.

Equally important has been the support of our closest friends and family members, vital pillars of our academic and personal journeys. Thank you for picking us up when we fell.

ABSTRACT

This project aims to develop a football analysis application, that offers professional football staffing and sports enthusiasts a comprehensive tool that covers all aspects of building a football club.

The system will provide in-depth statistics and information about thousands of football players, as well as rich data visualization elements and images that transform complex information into clear-cut insights. Users will have access to tools to filter players according to numerous advanced statistics and attributes, squad enhancement and analysis functionalities, complex player evaluation algorithms that cater to user-specific needs and other functions that ease the process of player recruitment and creating winning teams.

The system is formed by a web application based on the Django framework and an integrated database that contains information of upwards of eleven thousand players.

Keywords

Football, statistics analysis, sports statistics, player performance, web application, Django framework, data visualization, sports analytics.

RESUMEN

Este trabajo tiene como objetivo el desarrollo de una aplicación de análisis de estadísticas futbolísticas, que ofrece tanto a cuerpos técnicos profesionales como a entusiastas del fútbol una herramienta integral que envuelve todos los aspectos relacionados con la construcción de un equipo de fútbol.

El sistema proporcionará información y estadísticas detalladas sobre miles de jugadores de fútbol, así como elementos de visualización de datos e imágenes que transforman información compleja en conocimientos fáciles de comprender. Los usuarios tendrán acceso a herramientas para filtrar jugadores según diversos parámetros complejos, funcionalidades relacionadas con la construcción y el análisis de equipos, algoritmos complejos de evaluación de jugadores que se adaptan a necesidades específicas del usuario y otras funcionalidades que facilitan las tareas relacionadas con el reclutamiento de jugadores y la creación de equipos ganadores.

El sistema está conformado por una aplicación web que utiliza el marco de Django, y una amplia base de datos que cuenta con información sobre más de once mil jugadores de fútbol.

Palabras clave

Fútbol, análisis de estadísticas, estadísticas deportivas, rendimiento de deportistas, aplicación web, Django, visualización de datos, análisis deportivo.

CONTENT INDEX

FIGURE INDEX

Appendixes:

TABLE INDEX

Chapter 1 - Introduction

1.1 Motivation

When we think about football, we do not only think about the sport full of passion and excitement for a badge, a shirt or a color; there is much more to the game that moves masses in every corner of the world. Football is also a cutthroat business, with its global market size being in the billions, where every strategic decision determines the success or failure of a club.

One could blame the publishing of the 2011 film "*Moneyball*" starring Brad Pitt and based on the 2003 nonfiction book "*Moneyball: The Art of Winning an Unfair Game*", which follows the struggles of the Oakland Athletics to build a competitive baseball team under a tight budget, for the mainstream realization of the potential of data analysis in the world of sports.

The digital revolution has transformed the instruments these analysts work with, from pen and paper to a plethora of powerful software systems that incorporate the latest algorithms, data visualization tools and quantum statistical values, but there is a strong resistance to the use of such platforms, as the deeply rooted culture of football is resistant to sudden changes.

However, in the past few years there have been clear signs of change, such as several top clubs' scouting departments factoring the insights provided by these technologies into the decision-making processes; and the industry is reacting, with a blossoming of competing companies that quantify and classify hundreds of events occurring in the 90 minutes of the game.

While this data can be very valuable to any sporting organization, the formula for achieving excellence may be hidden in the visualization, analysis and interpretation of the obtained quantitative and qualitative metrics, and the market

for the ultimate football analysis platform is highly competitive, with companies' technologies and algorithms are secrets kept under wraps, as this industry has the potential to become a multi-million dollar business in the coming years.

Our project aims to enhance the decision-making process that is intricate to the world of sports management, providing well-informed insights and recommendations that can turn any club, at the amateur or professional level, into a dynasty.

It is no mystery that technology has affected every single aspect of today's world, and we think its potential in sports has not yet been completely explored and exploited. From analyzing player performance, to scouting talented players or building a promising squad, our platform provides the out-and-out toolset for any sporting director, scout, manager, or football enthusiast eager to take a leap forward and embrace the power of today's technology.

Using data-driven analytics and powerful visualization tools the boundaries of the administration of a football club are uncharted, we aspire to deliver a comprehensive application that balances the complexity of data analytics and the simplicity of a user-friendly interface.

In essence, our project is fueled by a deep-seated passion for the sport of football and the motivation to take the game to the next level, utilizing cutting-edge technology to discover the next Cristiano Ronaldo or constructing the following Leicester City-style success story.

1.2 Objectives

In order to address the challenges posed by the competitive nature of the sport, we have highlighted the following keystones to navigate the development of our project:

1. **Comprehensive Football Management Platform:** we propose to create a user-friendly platform that covers all aspects of football management, with robust tools that enwrap player scouting, squad analysis and strategic planning.
2. **Data-driven Decision Making:** by using advanced data analytics we plan to arm coaches, scouts, sporting directors and enthusiasts with insightful information that can be translated into success on the pitch.
3. **Player Scouting and Recruitment:** our focus on finding the best players that fit the club's philosophy is key in the development of the application.
4. **Squad Building:** transforming groups of players into winning teams, taking into consideration finances, managerial preferences and other external factors.
5. **Long-term Strategic Planning:** all the features of our platform stem from the core belief that the long-term sustainability of the club is fundamental. Users are able to create strategies that incorporate board expectations and monetary restrictions, such as youth development programs, player improvement pathways and season objectives, with the aim of turning any club into a winning dynasty.
6. **Customization and Flexibility:** designing the platform to be adaptable to cater to the needs and preferences of a multitude of users.
7. **Scalability and Reliability:** building a solid and scalable infrastructure that accommodates a growing user base and handles large volumes of updated data without compromising the experience.

8. **Continuous Improvement:** we are committed to a constant refinement of the platform, listening to feedback and adapting to evolving trends in football management.

1.3 Work Plan

In order to develop this software project, we have followed the "Waterfall or Cascade model". This involves various phases where the previous one must be completed before going onto the next. We have opted for this model due to the following reasons:

1. Sequential and linear nature of the model, with well defined requisites from the outset. The clear and stable nature of this model has helped us fulfill each phase methodically.
2. The predictability of this type of methodology has allowed us to organize the project into a predictable timeline, with clear and differentiated phases.
3. Time constraints have had a clear impact on the decision of choosing this model, as it allows us to have a limited set of iterations with clear objectives for each sprint.
4. The structure of rigid iterations allows our team to get constant feedback, and sets deadlines for the implementation of the functionalities, with their requirements already defined and stable throughout the entirety of the project.

The entire process has been subdivided into two-week blocks of tasks, which has allowed the project to grow with consistency and ensuring a constant input of feedback from our advisors.

We started with the planning of the project, where we created a set of plans to help guide the development. Continuing with the requirement analysis where we studied, analyzed and documented all the needs for our application. With all requirements established, we proceeded to the design phase and defined the technical specifications. Then we transformed all the requirements into actual code, following all the documentation of the previous stages. Finally rigorous testing was

conducted in order to ensure the software quality and meet all the requisites before the project deployment.

In the next chapters of this document we will explore the state of the art of the field our application lays in, explain the development process of this software application and its subphases: analysis, design, implementation and testing, and provide the necessary information to understand the management of the project, as well as the mechanisms related to project control and monitorization. Finally, a concluding chapter including the takeaways of the project and future work will be present in this document.

Chapter 2 - State of the Art

Data analysis and visualization tools are not a recent addition to the world of sports. Enthusiasts and experts have always been fascinated by the collection of data of their favorite sport, from pioneer coaching staffs to dedicated radio station sports talk shows and specialized magazines.

2.1 User Profiling

Mainly, the application is catered towards managers and other members of the staff of a football club, such as scouts, analysts and sporting directors, but we also take into consideration amateur football enthusiasts that also seek a comprehensive platform for football management and analysis.

2.2 Comparative Analysis of other Projects

Prior to designing the specifications and requirements, we drew inspiration from the existing platforms that are dedicated to the use of the aforementioned technologies within the scope of sports, and particularly football.

This has helped us realize which are the shortcomings and strengths of similar pieces of software, and which are the available opportunities to gain an edge over them.

	ADVANTAGES	DISADVANTAGES	IDEAS FOR OUR APP
Wyscout [1]	More than 600 hundred competitions including youth leagues and cups. API access. Advanced search to create player shortlists. Stats paired with video. Backed by established	Basic plan 300$ a year. No insight on possible transfer fees.	The possibility of advanced searching for players and the creation of shortlists to keep track of certain players.

	ADVANTAGES	DISADVANTAGES	IDEAS FOR OUR APP
	teams.		
Olocip [2]	AI scouting and predictive performance. Personalized dashboard. Used in different sports. Prediction and prevention of injuries. Player assessment and performance prediction. Scalability and able to incorporate future data sources and handle large volumes of data.	No possible access to specific info or data of the solutions provided by Olocip.	Using AI to predict performance of players. Scalability of the app.
StatsPerform [3]	In-depth and historical data that allows trends analysis and historical comparison. Live match statistics that allow in-game analysis and tactical adjustments. Customization for different types of users.	Lacks tools from direct competitors. Catered towards media outlets and sport bettings companies.	Allow trend analysis and comparison between players.
InStat [4]	Comprehensive video coverage. Detailed match and player analysis. Customizable reports and data visualization. Tools for recruitment and player trading.	Mainly focused on American high-school and college football teams. Lack of data for other countries.	Detailed player data and tools for player recruitment and trading.
Scisports [5]	Women football included. Stats, advanced player search and shortlisting. Calculated metrics such as transfer value, potential and development. Contemporary interface. Intelligent search engine. Own team analysis. Backed by established teams.	Limited database. No video available. Essential pack 900$ a month.	Possibility of own team analysis. Having calculated metrics such as transfer value.

Table 2.1- Competitive Landscape

In general, platforms can gather and display player and competitions data and statistics effectively, most of them also offering player reports that include graphs such as player progression, heat maps and many more.

Wyscout is the leader in terms of the sheer amount of data and the variety of competitions and clubs covered in their plans but lack some very integral functions that their main competitor, Scisports, feature in their plans, such as transfer value and player potential analysis or outstanding players.

In terms of pricing, there are no free alternatives that offer similar services, but some are more expensive than others. Scisport's approach to an upfront pricing helps us understand what value you are getting for what you are paying, whereas Wyscout's approach is offering a very limited number of features for a fixed price, and then getting a private, personalized, price for the features you request.

While some of the features required for an effective team-planner tool are present in all these platforms, such as the data and statistical analysis of players and competitions, most lack the functionality that is key for our scope, a real-time team planner and builder. The shortlist and team analysis are present in the two main players in this sector, Wyscout and Scisports, but only the latter offers market-value indicators.

Our main focus is to expand on the team-builder aspects of these platforms, adding smart-transfers, which give a comprehensive insight into the club's market opportunities, taking into account a given budget and a specific long-term sporting goal, such as building a young squad that aims for future results and market profit or immediate results.

2.3 Technologies

In order to ensure the best possible performance and fit for our vision, we analyzed the main options in terms of the development of an application, which are summarized in the table below.

	WEB APP DEVELOPMENT	MOBILE APP DEVELOPMENT
Definition	Process of designing, building, and maintaining a website app.	Creating an application for use on mobile devices.
Technologies used	HTML, CSS, JavaScript mainly	Platform-specific technologies
Compatibility	Web apps can work on multiple devices or platforms and on older operating systems.	You need to consider the diverse types of mobile devices and operating systems
Implementation	Require less platform-specific programming knowledge in order to implement.	Need specific tech stacks and separate apps for different OS.
Updates	Direct update or changes in the application.	Usually, you need the commercial app store approval for any kind of update.

Table 2.2 - Technologies

Once we settled for web app development, as it allows creating an easily accessible application across a multitude of different devices and operating systems without requiring downloads, we explored the different possibilities in terms of web application framework technologies (WAF) [6], foundations aimed to help developers build web applications, and the main takeaways can be seen in the table below.

2.3.1 Web Framework Technologies

	ADVANTAGES	**DISADVANTAGES**
React [7]	JavaScript framework. Simplified User Interface development with a declarative approach. Large community and ecosystem. Ensures scalability thanks to reusable components. Developer tools like React DevTools [8] for debugging. Uses JavaScript XML syntax, this facilitates the UI development.	High learning curve for beginners. Requires additional tooling and libraries for complex features like routing or state management.
Vue.js [9]	Efficient updates of the user interface with automatic tracking of data changes. Component based architecture facilitates code reusing and maintainability. Options API and Composition API to define component logic. Single file components that encapsulate logic, template and styles.	Smaller community and ecosystem compared to the others. Limited scalability for applications.
Django [10]	Based in python. Follows MVC architecture [11]. Large community and documentation. Simplified database interaction with Object Relational Mapping system. Automatic generation of admin interface for efficient management of database. Comprehensive set of tools out of the box. Built in security measures like the CSFR Token [12] if forms to protect apps against common web vulnerabilities like cross site request forgery. Testing framework for unity testing.	Its rigid structure may limit flexibility.
Angular [13]	Built-in features for complex single-page applications. Code maintainability with a component-based architecture. Typescript integration, which enhances	Need to follow certain conventions and patterns for development, limits flexibility. Larger bundle size compared to other frameworks, which makes a slower app.

	code quality. Animation support for a visually appealing UI. Routing support. Large collection of libraries and tools, with an extensive community and documentation.	

Table 2.3 – Web Framework Technologies

It is important to mention the many tools and technologies we chose and used to manage the relevant technical aspects of our project. Below we describe in detail these strategies and the role they play in the development of our project.

2.3.2 Web Development

Regarding web and mobile app development, we studied both approaches and decided to focus on creating a web application due to the ability to reach a wider audience and flexibility in programming. We prefer the versatility of web applications along with our web development experience obtained from the courses we have at the Complutense University of Madrid.

Another aspect we took into consideration is the appearance of our product. We are committed to providing a user-friendly interface, and being able to display powerful visualization tools while keeping a simple look and feel that does not overwhelm the user is very challenging in a mobile environment. A web application allows us to unleash the full potential of our product, and in the future, a port to a mobile environment could be considered if there was enough user-demand.

To implement our web application, we chose the Django framework based on Python, which follows the Model-View-Template (MVT) design pattern, like the Model-View-Controller (MVC), Django uses a complex, modular framework provided for our application, where model handles data management, view controls, and templates. This process facilitated our development and allowed us to maintain a clean and structured code.

For the front end of our application, we have mixed CSS and JavaScript to give the user well structured, styled, and adaptive pages. We have ensured a consistent and interactive approach to all our tools.

2.3.3 Development Environment

When coding the project, we used Visual Studio Code [14] as the Integrated Development Environment (IDE) [15] of choice. Its flexibility, wide range of extensions, and dynamic user base make this code editor a powerful tool for development.

Additionally, Visual Studio Code's native integration with Git [16] allowed us to effectively maintain version control of our code by using GitHub [17] as a remote repository. As a team of two developers, being able to make code contributions remotely and synchronously has been vital and has significantly sped up the development of the application.

2.3.4 Collaboration and Tracking Tools

We used JIRA [18] and Slack [19] for project management and project tracking. These platforms have allowed us to organize our activities, assign tasks periodically, and track project progress in detail, as well as instant messaging. The whole project since October has been broken down into 2-week blocks with each of the different tasks classified into different modules, which were flagged according to their completion status.

Together with the use of the instant messaging feature integrated in Slack, and other messaging platforms such as WhatsApp [20], we used Google Meet [21] for team meetings, which allowed us to maintain effective communication throughout the development of the project, especially in a remote work environment. We also used Google Docs [22] and Google Drive [23] to collaborate on writing all documentation and sharing other files.

2.3.5 Dataset

Finding a dataset with rich and up to date information about a large number of football players posed a challenge from day one, since the ones meeting our expectations were behind four-figure price tag paywalls.

We finally agreed on using the latest release of the Football Manager franchise, Football Manager 2024 [24] , a management simulation video game developed by Sports Interactive [25] known for its high-fidelity and realistic approach to a sports video game, even recognized by real-life clubs and managers as a source for scouting players, as the deal between Sports Interactive and Everton FC in 2008 [26] illustrates, where they allowed the club to use the game's database to scout players and opposition.

In order to obtain all the statistics that our application works with, we simulated the 2023-2024 season in the video game. After the simulation of the season, we created a custom view for the "Search" tab in the video game and we selected the fields that were relevant for our project, exporting the data of over 16.000 players into a html file, which we then processed and adapted to our necessities, generating a final csv database with over 12.000 players with more than 40 different metrics and information for each one.

One of the functionalities implemented in our application is the use of estimated transfer values for the players, for which we based ourselves in the estimates provided by Transfermarkt [27].

Transfermarkt is a German website that, along with storing footballing information such as match information, transfer news and player statistics, they estimate a player's market value according to expert evaluations and footballing market fluctuations. While their estimates are not entirely accurate, as the evaluation

of a player is subjected to many different subjective factors, they are considered to be largely on target.

To obtain the aforementioned estimates and integrate them into our dataset, we used a Transfermarkt API [28] hosted by Vercel [29], a cloud platform as a service company.

Chapter 3 - Analysis

The main aim of this stage is to completely understand the project. To do this, we studied the competitive landscape, examining the market and understanding existing solutions and their strengths and shortcomings.

Another important aspect of this stage was evaluating the existing technologies and tools, which also involved assessing the feasibility of the project. This helped us select the most suitable technological stack for development and write the State of the Art, in which we collected the outcomes from this phase.

Finally, this stage involves gathering and constructing a sound set of requirements, constraints and restrictions involved using the takeaways from the first phase of the project and translating them into the documentative guidelines for the development.

3.1 Product Perspective

Our application is designed to be a comprehensive and user-friendly football platform for enthusiasts and managers to analyze different players statistics, find the right players based on specific traits, get recommendations of new incorporations to a team, and enhance club management through long-term scope settings.

3.2 Division in Subsystems

In order to facilitate the design and implementation of the application, its functionality has been divided into three clear subsystems. In this section, a detailed explanation of those subsystems and their fundamental tasks will be provided. More detailed and technical information about the application's functionalities can be found in Section 3.3, Functional Requirements.

3.2.1 Player Management Subsystem

The Player Management Subsystem (PMS) encapsulates all the functionality related to the search and display of the players present in the database of our application. Particularly, these four main features are included in this subsystem:

1. **Player Profile:** a player's profile will display its basic information, such as their full name (or nickname), current club, shirt number, age, preferred foot, etc.
2. **Basic Search:** by entering text on the basic search box, all players whose name, current club or league match the search query will be displayed.
3. **Advanced Statistics:** a player's profile will have in-depth statistics that allow the user to get a real idea of what type of player they are, which are their strengths and weaknesses, and what play style they are going to be proficient in. When displaying a player's profile, basic information such as their age, current club or their Transfermarkt value will be shown.

   Furthermore, the user will be able to select one of the player's positions, and when this is done, a radar graph with the most relevant statistics of that position will illustrate a comparison between the player and the average player in that position. Additionally, the SmartScore of that player for that position will be calculated and presented.
4. **Advanced Search:** one can filter out the thousands of players available in our database to find the ones that have their desired characteristics. There are six categories of filters: position, general, goalkeeping, defending, creative and attacking. The metrics that we store for each of the players have been categorized into one of these and can be applied to the trait-based search to find the appropriate player.

3.2.2 Analysis Subsystem

The Analysis Subsystem (AS) contains all functionality related to the evaluation of players, squads, and signings. Particularly, these three categories of tasks can be found in the functionality related to this subsystem:

1. **Squad Builder Tool:** Users will be able to import a squad, select a starting eleven among the players in the squad, and this tool will analyze the players, giving feedback on where there is room for improvement, particularly, which players to let go and which players will take the team to the next level.
2. **SmartScore Metric:** our tool will be able to weigh-in several player's characteristics to provide a numeric grade evaluation for the player. Using an algorithm that considers the player's matchday statistics, compared to other players in the same position, the player's age, the number of international games played, the league they play at, etc., we can accurately grade their quality, giving them a numeric value from 1 to 99.
3. **Recommended Signings:** there will be recommendations for potential signings that will consider our previously mentioned SmartScore as well as several filters to find the ideal match for the club's philosophy. Filters to find a player that fits a specific playstyle will be available, and from the players that meet the criteria, the ones with the highest SmartScore (up to 30) will be presented to the user, and if the user is interested in any of them, there will be an option to save them to a shortlist.

3.2.3 User Management Subsystem

The User Management Subsystem (UMS) encompasses all the functionalities required to create and access an account in the application, as well as tweaking the application to cater to the user's preferences and requirements. These five functionalities are the ones that make up the main part of the subsystem:

1. **Sign Up:** Users are able to register in the application using an existing email address and a secure password.
2. **Log In:** With the credentials provided when creating an account, users can log into their account. Some of the functionalities such as the FutureScope settings, the Squad Builder tool and saving players to shortlists require the user to be registered in the application and be logged in to their account.
3. **Shortlisting Players:** A logged user can add the players obtained from the recommended signings functionality to a shortlist, which can be later viewed.

4. **Adding Players to Squads:** After logging in, a user can create squads from the "My Squads" page and then they are able to go to any player's profile page and add them to an existing squad. These squads can then be used in the Squad Builder tool.

5. **FutureScope Settings:** To further enrich the user experience, we came up with a system that will allow the user to achieve more realistic results. According to information about the club provided by the user (the team's budget, league, and expectations), the application will tailor the SmartScore calculation algorithm by also weighting in the aforementioned factors, which will then be reflected on player's profiles and the recommended signings.

3.3 Functional Requirements

Functional requirements will be broken down into the subsystems defined accordingly in Section 3.2. In order to generate the following graphic elements, we have used Modelio [30], an open-source modeling environment tailored for software development projects such as this one.

3.3.1 Use Case Diagram

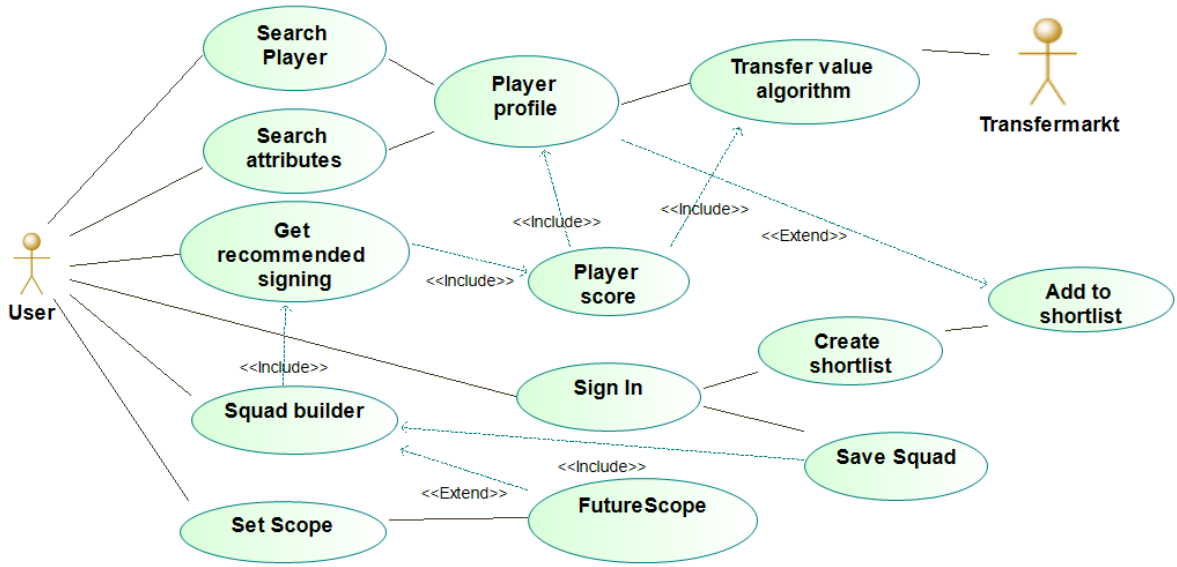


Figure 3.1 – Use Case Diagram

3.3.2 *Activity Diagrams and Structured Natural Language*

1. **Player Subsystem**
    - **Basic Search**

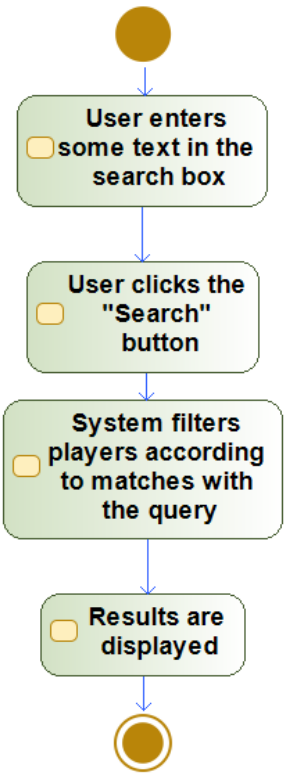


Figure 3.2 – Basic Search AD

Identifier	UC1
Use case name	Basic Search
Actors	User, System
Description	User searches players and the ones whose name, club or league matches the query are displayed
Precondition	Player database up to date.
Normal Sequence	1. User enters some text on the box and clicks the "Search" button. 2. The system looks for matches on the player's names, club or league. 3. Results are displayed.

Postcondition	Player database up to date.
Exceptions	Step 3: If there are no player matches, the user will be informed.

Table 3.1 – Basic Search SNL

- **Player Profile**

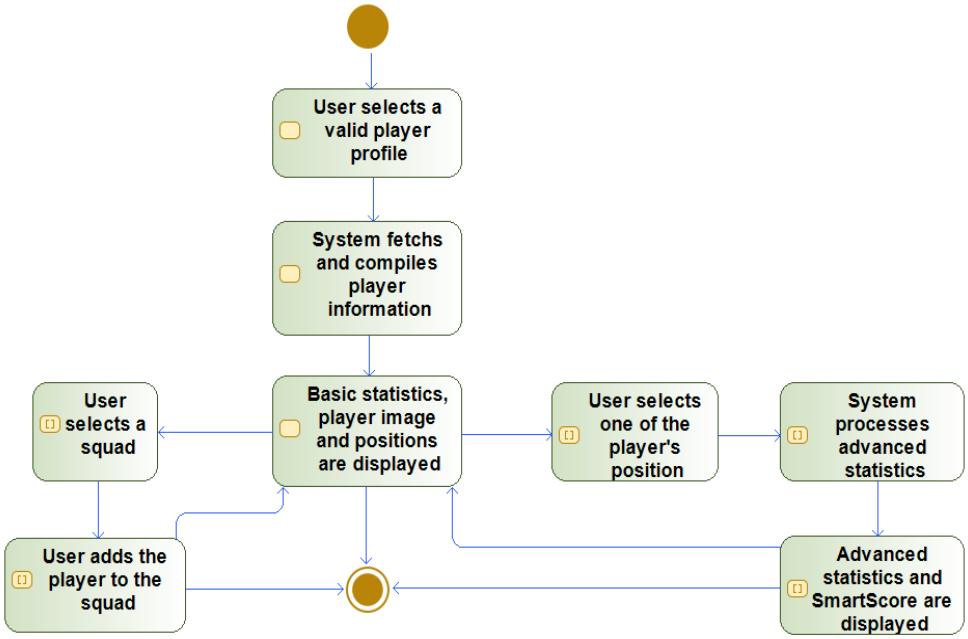


Figure 3.3 – Player Profile AD

Identifier	UC2
Use case name	Player Profile
Actors	User
Description	Retrieve and display in-depth attributes and statistics for a player's profile, including strengths, weaknesses.
Precondition	Player profile exists in the database.
Normal Sequence	1. User accesses through any valid way to a player's profile. 2. The system fetches and compiles basic player information such as their age, club, positions as well as their image. 3. System displays the information.

Postcondition	Users are informed about the selected player's information.
Exceptions	3. Once the information is displayed, the user has two optional actions: 3a: The user can click on one of the player's positions, which will make the system create advanced data visualization graphs with their statistics, as well as calculating their SmartScore for that position. 3b: The user can select one of their existing Squads to add the player to them.

Table 3.2 – Player Profile SNL

- **Advanced Search**

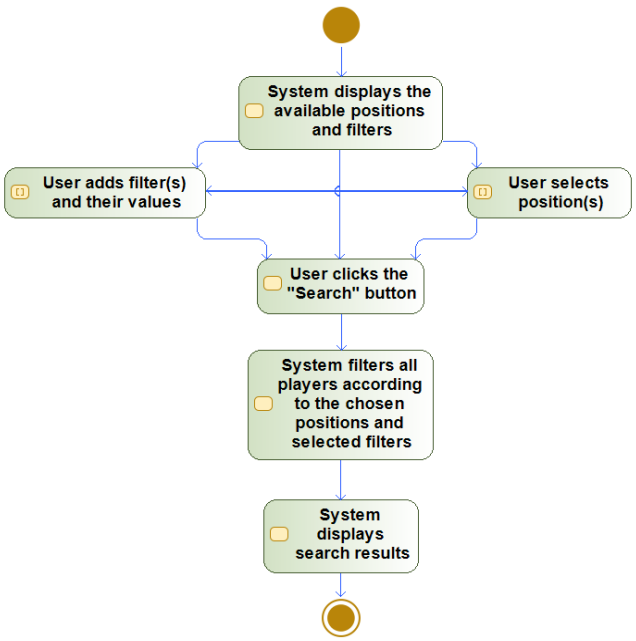


Figure 3.4 – Advanced Search AD

Identifier	UC3
Use case name	Advanced Search
Actors	User, System
Description	Filter players based on desired characteristics such as player information, physical attributes, and in-pitch statistics.
Precondition	Player database is available.

Normal Sequence	1. The system displays a set of filters and positions. 2. User can select those filters and positions, then clicks the "Search" button. 3. System filters all the players according to the selected metrics and displays them.
Postcondition	User sees a list of players that match their desired characteristics.
Exceptions	Step 3: If no results are found, the user will be informed.

Table 3.3 – Advanced Search SNL

- **Transfer Value Helper**

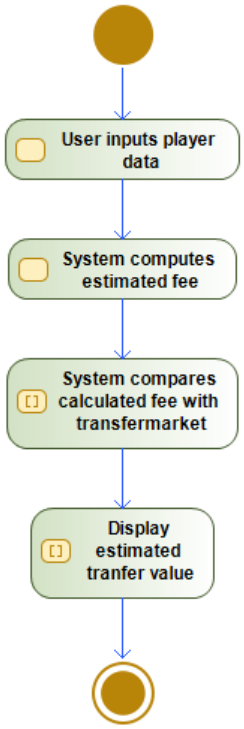


Figure 3.5 - Transfer Value Helper AD

Identifier	UC4
Use case name	Transfer value helper
Actors	User, System
Description	Estimate a player's transfer fee based on various characteristics.
Precondition	Player information and contract details are available.

Normal Sequence	1. User provides player characteristics, league, club, and contract details. 2. System calculates an estimated transfer fee. 3. System compares the estimate with Transfermarkt's market value for validation. 4. System provides the user with the estimated transfer fee.
Postcondition	User receives an estimated transfer fee for the specified player.
Exceptions	

Table 3.4 – Transfer Value Helper SNL

2. Analysis Subsystem

- **Squad Builder**

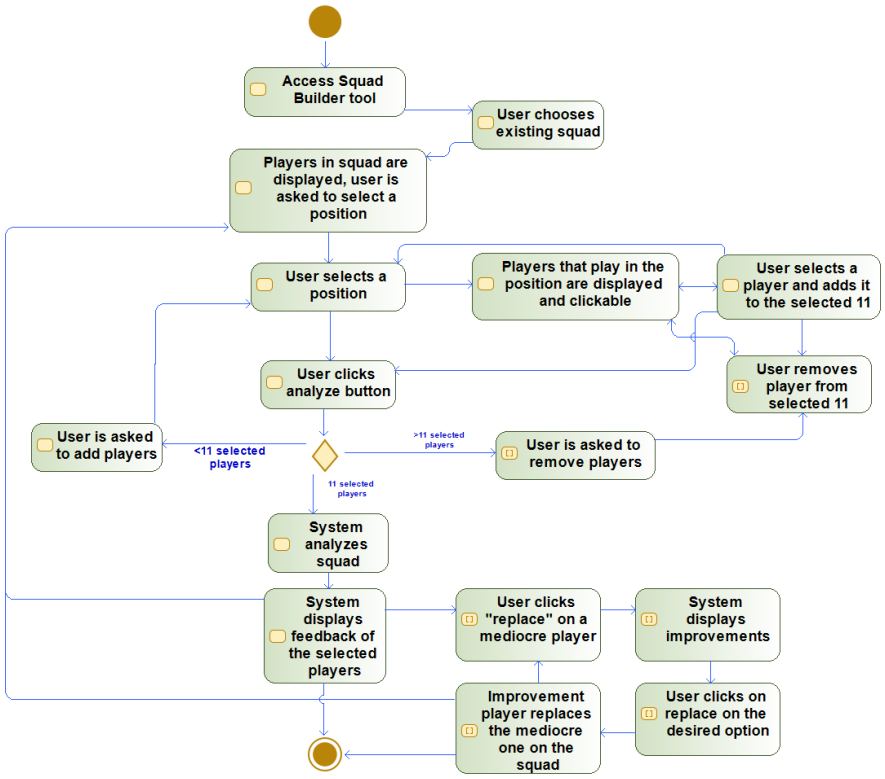


Figure 3.6 – Squad Builder AD

Identifier	UC5
Use case name	Squad Builder

Actors	User, System
Description	Analyze an existing team squad, providing feedback on improvements and player recommendations.
Precondition	Team squad information is available.
Normal Sequence	1. User imports an existing team squad. 2. User clicks on the desired positions on the field, players that play there are highlighted, and the user can choose them and add them to the selected eleven. 3. When eleven players have been selected, the user presses the "Analyze" button. 4. System provides feedback on each of the selected players, indicating their strengths and weaknesses. 5. Below-average players can be replaced if the user clicks the "Replace" button, which will prompt the system to look for players that improve the mediocre ones. The user can swap them out and replace the weak links of the squad
Postcondition	User receives feedback on their squad and can improve it.
Exceptions	Step 2: if the user desires to, they can remove the selected players by clicking on the "x" icon next to their name on the field. Step 3: if there are not exactly eleven players, the user will be informed and asked to resize their selected eleven.

Table 3.5 - Squad Builder SNL

- **SmartScore**

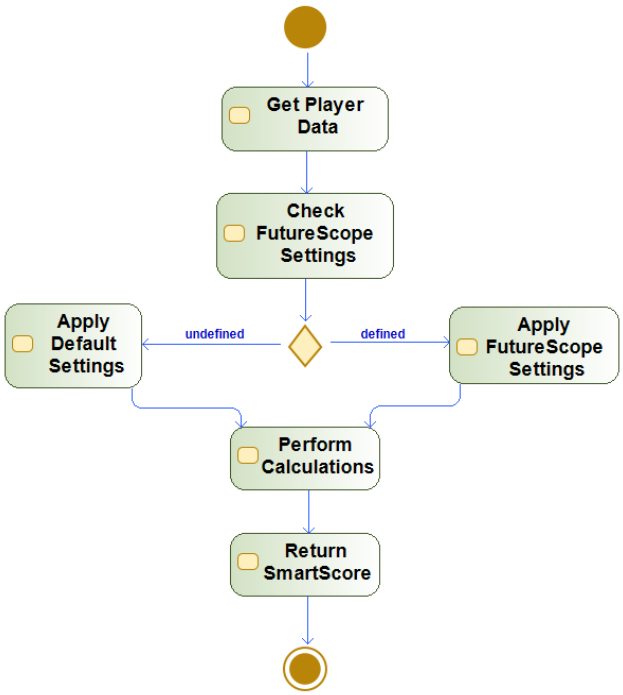


Figure 3.7 – SmartScore AD

Identifier	UC6
Use case name	SmartScore
Actors	System
Description	Analyze a player and provide a numeric evaluation from 1 to 99.
Precondition	Player information available.
Normal Sequence	1. Player information is loaded. 2. The system checks if there is any FutureScope information provided and retrieves the parameters. 3. The system performs the evaluation algorithm. 4. The system returns a numeric evaluation.
Postcondition	The system provides a numeric grade for the player.
Exceptions	Step 2: if there is no defined information for the FutureScope, the default parameters will be applied.

Table 3.6 - SmartScore SNL

- **Recommended Signings**

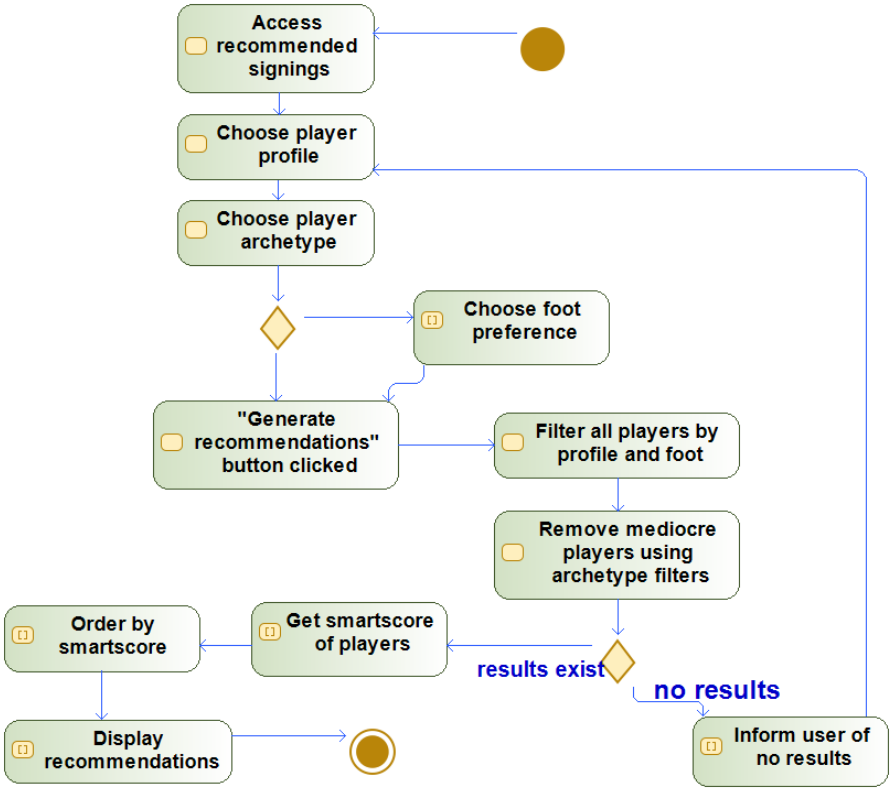


Figure 3.8 – Recommended Signings AD

Identifier	UC7
Use case name	Recommended Signings
Actors	User, System
Description	Provide users with recommended player signings based on player preferences and statistical values.
Precondition	Player database and transfer value estimates are available.
Normal Sequence	1. Users access the recommended signings section. 2. Users choose the type of player they are looking for. 3. The system filters out the players that do not meet the desired characteristics. 4. System computes the SmartScore of the remaining players.

	5. System displays the best options as recommended signings for the user.
Postcondition	User receives a list of up to 30 recommended signings with their SmartScore.
Exceptions	Step 2- It is not mandatory to add all the filters for the players. Step 3 - It can be the case that no players meet the desired conditions. In that case, the user will be informed, and they can modify the filters.

Table 3.7 – Recommended Signings SNL

3. User Management Subsystem

- **Sign Up**

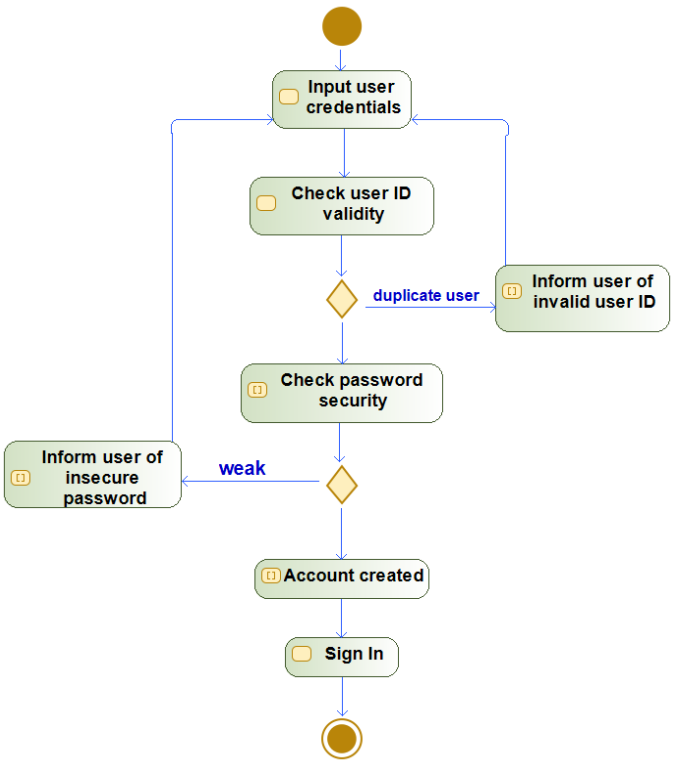


Figure 3.9 – Sign Up AD

Identifier	UC8
Use case name	Sign Up
Actors	User

Description	Create an account with valid credentials provided by the user
Precondition	No duplicate user IDs
Normal Sequence	1. User provides credentials. 2. User ID uniqueness is verified. 3. Password strength is verified. 4. Account is created. 5. Log in
Postcondition	No duplicate user IDs
Exceptions	Step 2: if the user ID is already present in the database, the user will be informed. Step 3: is the password is not strong enough, the user will be informed.

Table 3.8 – Sign Up SNL

- **Log In**

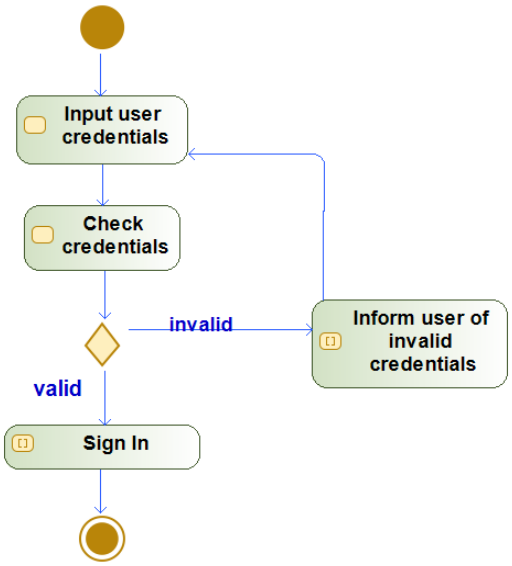


Figure 3.10 – Log In AD

Identifier	UC9
Use case name	Log In
Actors	User
Description	Check that the credentials entered by the user are valid

Precondition	No duplicate user IDs
Normal Sequence	1. User provides the credentials (input) 2. Check coincidence of the credentials with the database 3. Log In
Postcondition	No duplicate user IDs
Exceptions	Step 2: if the credentials are invalid, the user will be informed

Table 3.9 – Log In SNL

- **Add to Shortlist**

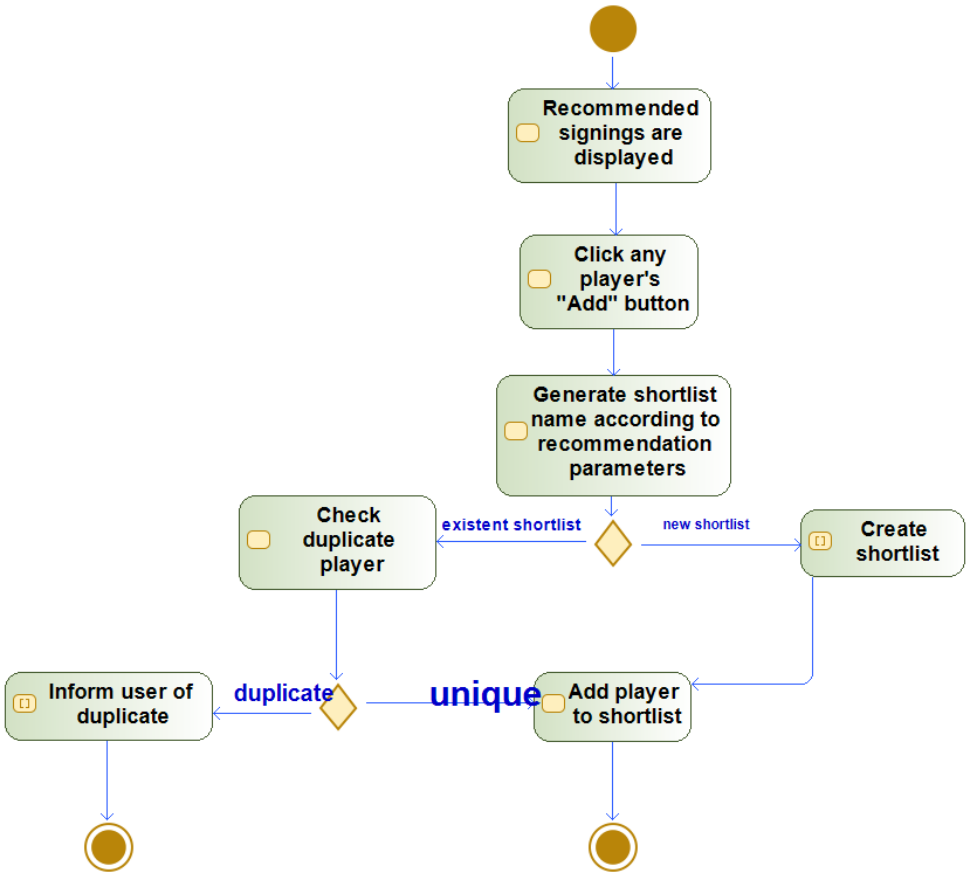


Figure 3.11 – Add to Shortlist AD

Identifier	UC10
Use case name	Add to shortlist
Actors	User, System

Description	Allow the user to save players to shortlists, that is, lists of players of interest.
Precondition	Player database up to date, UC6
Normal Sequence	1. Recommended signings are displayed. 2. A player is selected to be added to a shortlist. 3. According to the recommended signing's parameters, the name of the shortlist is generated. 4. The player is added to the shortlist.
Postcondition	Player database up to date.
Exceptions	Step 3: The system checks if the shortlist already exists. If it does and the player is already on that shortlist, the user will be informed. Execution of this use case ends.

Table 3.10 – Add to Shortlist SNL

- **FutureScope**

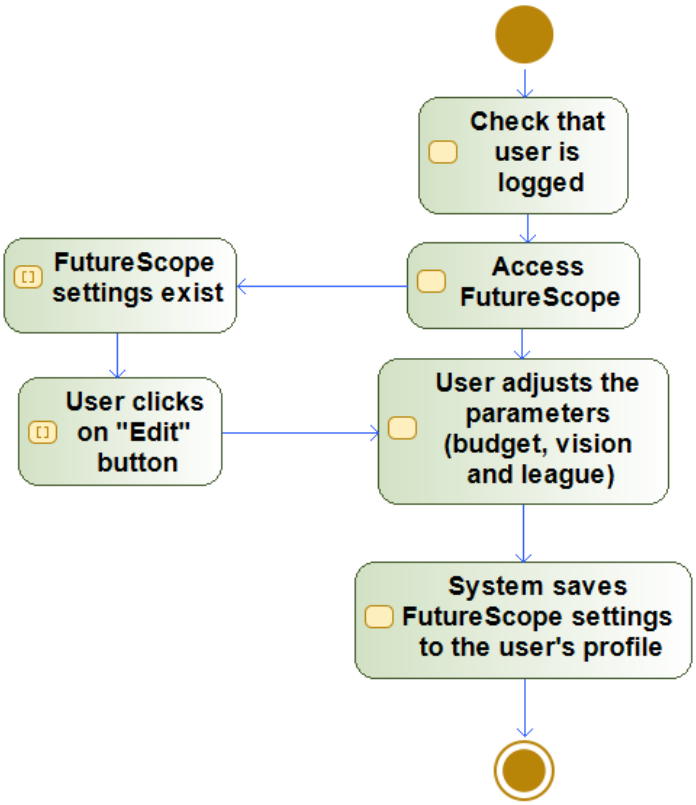


Figure 3.12 – FutureScope AD

Identifier	UC11

Use case name	FutureScope
Actors	User, System
Description	Allow the user to input transfer budget, league, and short/long-term expectations to adjust SmartScore calculation.
Precondition	Algorithm for SmartScore calculation, logged user.
Normal Sequence	1. System checks that the user is logged in. 2. User accesses the FutureScope setting section. 3. User inputs transfer budget, league, and expectations. 4. System saves settings to the user's profile.
Postcondition	Algorithm for SmartScore calculation, logged user.
Exceptions	1. If the user is not logged-in, it will be taken to the log-in screen.

Table 3.11 – FutureScope SNL

3.4 Non-Functional Requirements

3.4.1 External Interface Requirements

- The web application shall be supported in the following browsers: Google Chrome [31], Microsoft Edge [32], Mozilla Firefox [33] and Brave [34]. It is expected that the user has enabled JavaScript in their preferred browser, as the application will need these elements to function properly.
- The optimal experience is obtained on a desktop or laptop computer, as the minimum screen size needed to properly display the elements of the application is 11 inches.

3.4.2 Performance Requirements

- With filters applied, the recommendation of signings should not take over 30 seconds.
- Advanced and basic search should be performed in under 30 seconds.
- Squad analysis should not take more than 10 seconds.

- Finding replacements for a player in the squad should be done in under 15 seconds.
- Other navigation and actions should not take more than 5 seconds.

3.4.3 Data Persistence Requirements

- The application should correctly diagnose the cause of its errors and inform the developer or user accordingly.
- The application should allow applying changes and improvements to its code, database, design, and documentation. This is vital to ensure data persistence as well as allow error correction, improvements, and adaptations to different environments.
- The application should minimize the impact of involuntary or erroneous modifications by notifying the user through console outputs.

3.4.4 Security Requirements

- Integrity and confidentiality of user data is ensured by the implementation of a user-authentication process to perform certain operations, like squad building, getting player recommendations or adjusting the FutureScope settings.

3.4.5 Design Requirements

- Ensure the visual consistency of the application, having all graphical elements and layout follow a defined style and color scheme.

3.4.6 Document Requirements

- The application and its documentation will exclusively be present in the English language.

- User's manual intended to help users understand and use the functionalities of our application and help them navigate through the different pages of the platform.
- Programmer's manual is intended to help developers and other interested users comprehend the intricacies of our application.

3.4.7 Resource Requirements

- The application will employ a relational database, allowing the user to save information like squads, shorts with recommendations and the personalized FutureScope setting.
- The database will consist of upwards of 11.000 players and will be processed from a CSV document.

3.4.8 Availability Requirements

- The application is run locally, so it will be always available. However, the user must hold an active Internet connection while accessing the web application to engage with its elements.

Chapter 4 - Design

Once we had the requirements defined, and we had analyzed the available technologies and competitive landscape, the design phase began, focusing on architectural design, database schema, user interface design and other technical aspects of the application.

The main objective from this phase was the elaboration of diagrams that describe the architecture of the application, and three main documents were created: class diagrams for each of the subsystems, a detailed model describing the dataset structure and a mockup of the user interface.

4.1 Class Diagrams

We used Modelio OpenSource 4.1 to draw up the class diagrams for each of the described subsystems.

The Player Management Subsystem is represented in Figure 4.1 in which the static elements are represented in a brownish color, the logical elements in a mint green tone and the application's interface in a cyan blue color. Information regarding the class diagrams of the other subsystems has been presented in an abstract manner, as will be explained later.

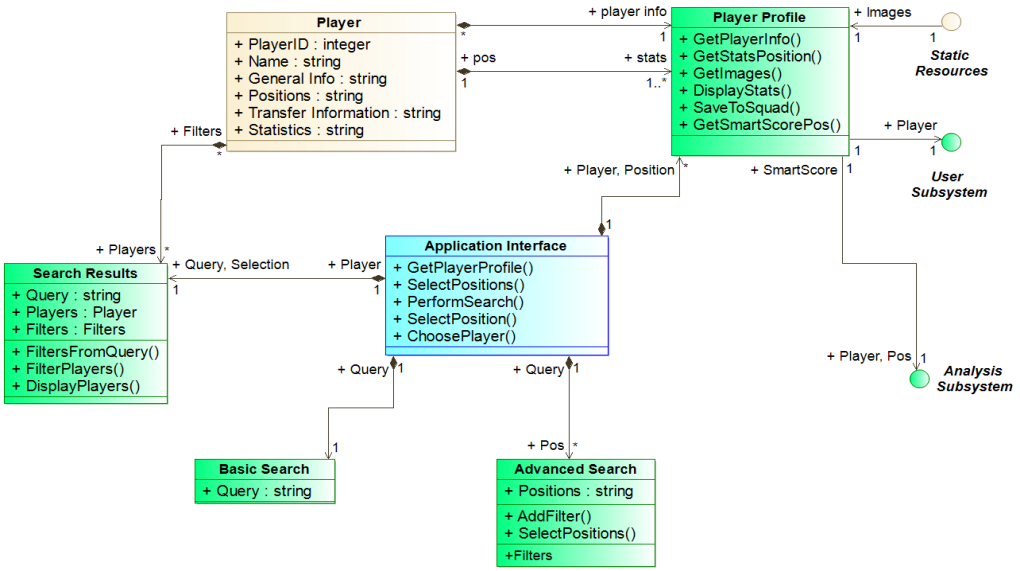


Figure 4.1 – PMS Class Diagram

For the Analysis Subsystem, as shown in Figure 4.2, we have used the color palette previously mentioned, and again choosing an abstract representation of the other two subsystems.

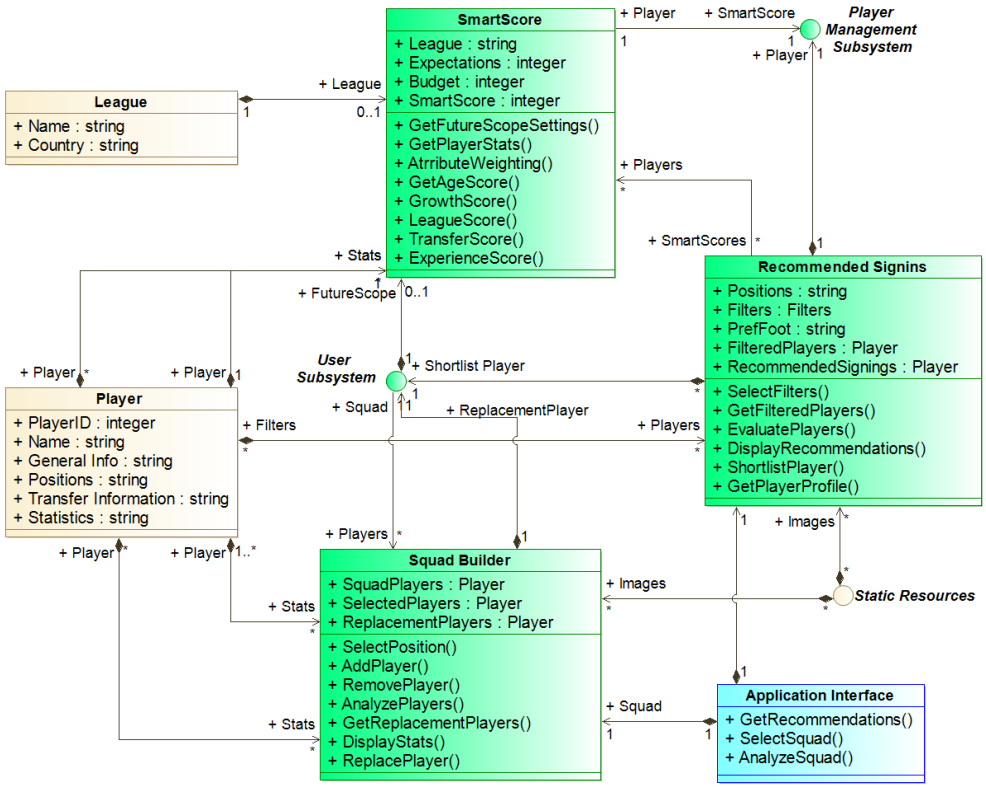


Figure 4.2 – AS Class Diagram

For the final subsystem, the User Management Subsystem, we maintain the stylistic decisions for the representation of its class diagram, which is displayed in Figure 4.3.

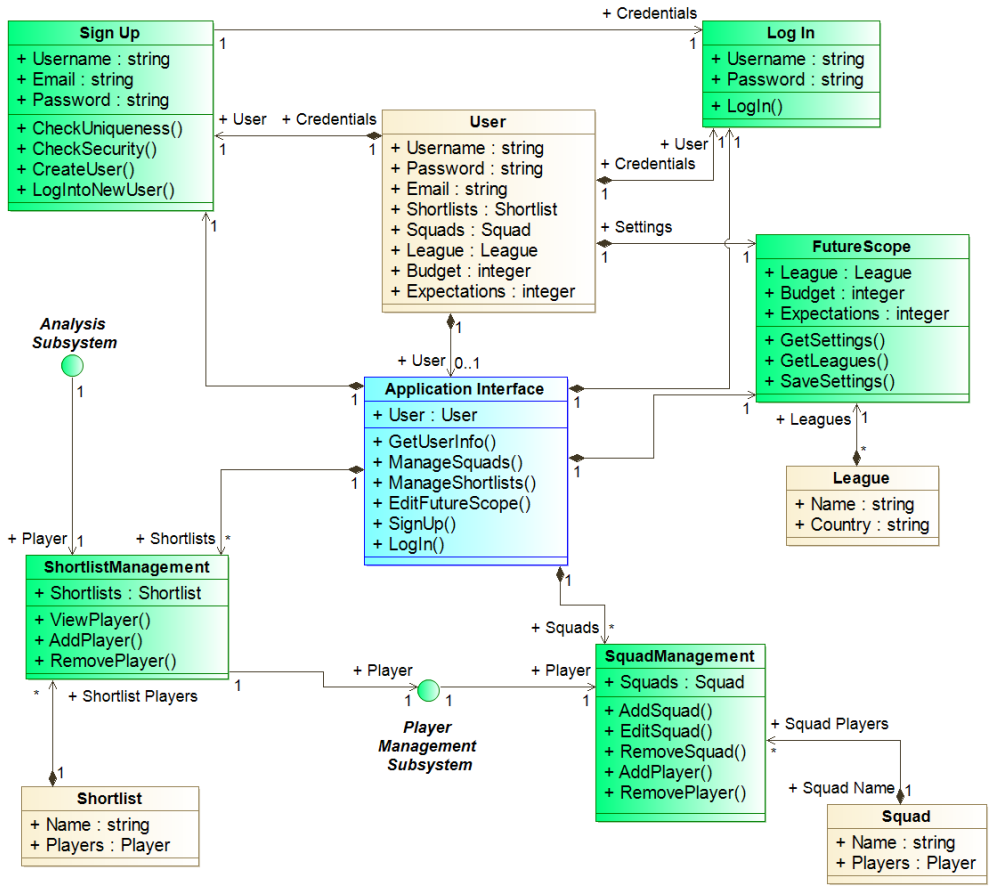


Figure 4.3 – UMS Class Diagram

4.2 Dataset Structure

Designing the structure of the dataset for this application is a crucial step in the development of the project, as it ensures data consistency and a manageable set of information. The material for the application can be classified into 7 interconnected data models. Below, we detail them and their role in the system:

- **Player**: in this model find information and statistics about each of the more than 12.000 players present in our application. Attributes include player name, age, nationality, and various performance metrics.

- **Position:** as players can play in more than one position, the Position model is going to be a field associated with the Player model. This allows more flexibility and the implementation of robust player analysis.
- **Squad:** this model is associated with the Squad Builder functionality, and each of these data types is going to contain a list of players and a squad name.
- **Shortlist:** the same way Squad is associated with the Squad Builder functionality; Shortlist is related to the Recommended Signings feature of the Analysis Subsystem.
- **League:** this is an auxiliary data model that contains the names of the leagues in our dataset and the country associated with them. This model is used by the User and Analysis subsystem, as it is involved in the SmartScore calculation and the FutureScope settings.
- **User Profile:** very common in web applications, the User Profile model handles the user database and their settings and saved elements.

4.3 Interface Mockup

In this phase we made choices to ensure that our application follows a user-centric approach, providing an intuitive user experience. The key design principles that our application follows are:

1. **Sidebar navigation:** our interface should feature a sidebar that houses the primary tools, dividing them into categories that ease browsing our application.
2. **Top bar:** the main purpose of having a top bar in our application is to complement the sidebar, providing a secondary navigation element. Here, users can find the features related to user authentication and other account-related functionality. Furthermore, a search box is provided, where the Basic Search described in the Player Management Subsystem in Chapter 3 can be performed.
3. **Responsive Design:** as described previously, a non-negotiable requirement was that our application must adapt to a wide range of devices, web browsers and screen sizes. This principle ensures a consistent and optimized user experience.

4. **Consistent Branding and Styling:** throughout the interface, we maintain a consistent color palette, typography, and iconography. By adhering to the same stylistic choices, we boost user familiarization with our application.

The deliverable from this phase was a mockup or prototype of the web application's interface, created using the Canva [35] software tool, an online visual suite and template editor. The design choices and principles previously mentioned are embodied in Figure 4.4 and will serve as the blueprint for the stylistic decisions in the implementation phase of the project.

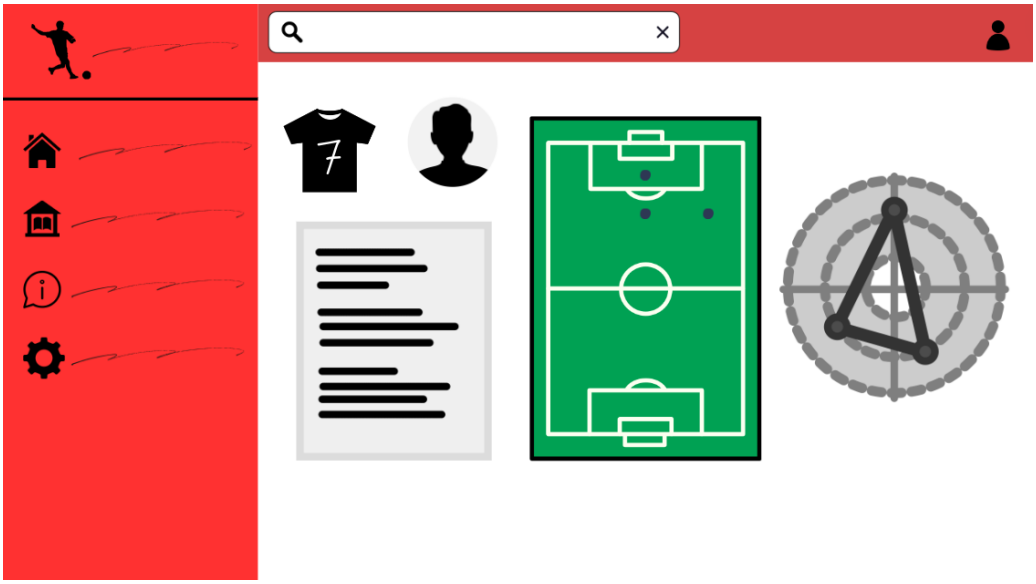


Figure 4.4 – Interface Mockup

4.4 Technological Scope

As an introduction to the chapter, we will describe the architecture of the application at a high-level. In Figure 4.5 we have a descriptive graphic representation of the architecture.

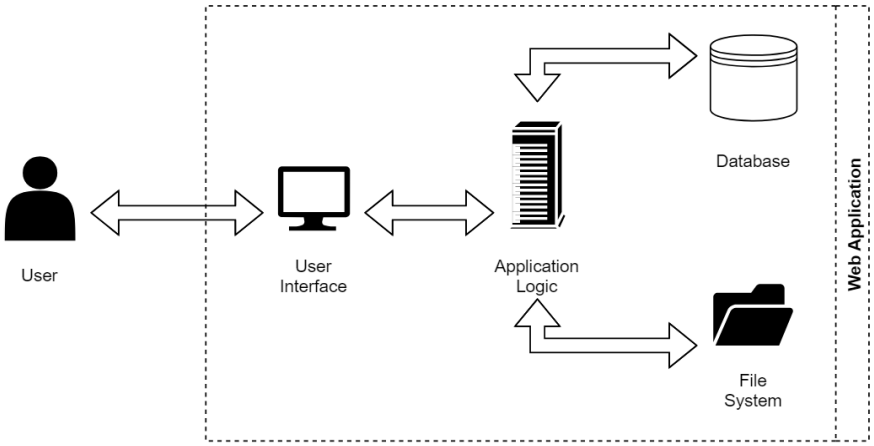


Figure 4.5 – Architecture of the Application

As described in the figure, the user will interact with the application through its interface, coded in high level languages such as HTML, JavaScript and CSS. By interacting with the interface, the user will be able to communicate with the application's logic, programmed in Python and JavaScript, which will work alongside the database, file system and algorithmic backbone of the application.

Chapter 5 - Implementation

This phase involves coding the application, which involved translating the functional requirements into programming languages. In our case, we used the Django Framework, which follows the model-template-views (MTV) architectural pattern and involves the use of programming languages such as HTML, CSS and Python.

The chosen methodology was especially important for this stage, as subdividing the development into two-week blocks, with meetings in between, allowed us to keep a constant workflow, clearly defined objectives for each sprint and a constant influx of feedback and discussion about the decisions made.

The first week of implementation was dedicated to the creation of the web application's basic prototype: a simple page where the functionality can be deployed as it is developed. This involved creating a simple interface according to the decisions taken in the design stage and introducing a reduced version of the player dataset, in order to test the first pieces of functionality with a limited amount of information.

The next weeks involved the completion of two of the main features of the Player Management Subsystem, advanced search, and the player profile. For this stage, the interface was greatly upgraded, as the first advanced data visualization elements were introduced into the application. The implementation of the advanced search feature was challenging, but it helped us understand the intricacies of working with the dataset while mixing different programming languages such as Python, JavaScript, and HTML.

The next month of development, from mid-January to the end of February, was dedicated to two of the main features of our web application, the Squad Builder tool and the SmartScore calculation algorithms. These two components of the Analysis

Subsystem posed a great challenge as they involve complex operations and complicated calculations, but once we mastered them, other obstacles were not as knotty as before.

In Figure 5.1, an example of one of the many algorithms involved in SmartScore calculation is displayed, this one is used to obtain an adjustment factor of the score when weighing players estimated transfer value and the transfer budget provided by the user.

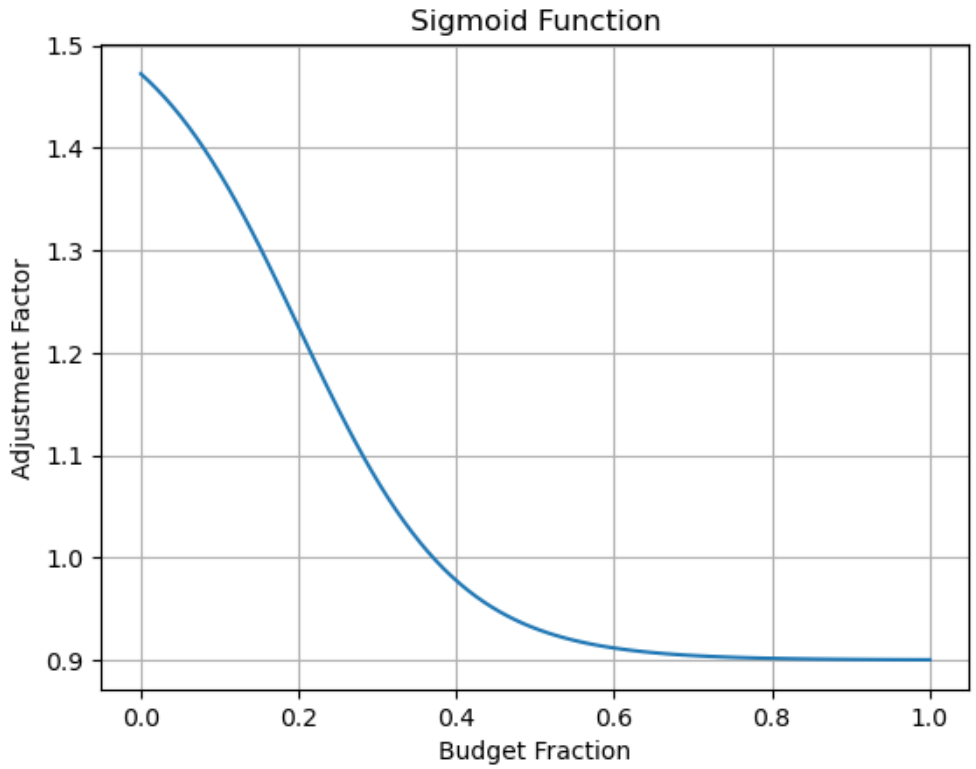


Figure 5.1 – Budget Adjustment Factor

In Figure 5.2, another example of the calculations made in the process of evaluating players through the SmartScore value is shown. To calculate the factor related to a player's age, we are going to be using the FutureScope settings chosen by the user. In this case, when selecting long-term success (Growth Factor = 2), the

algorithm places more value in the player's age, giving more extreme age score factors to very young or very old players.

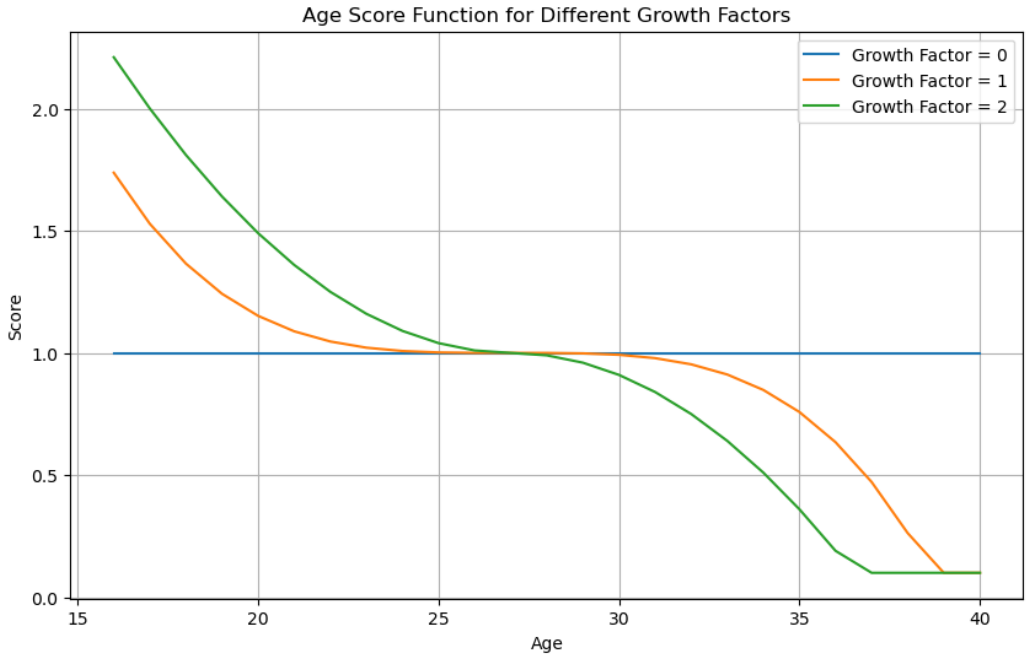


Figure 5.2 – Age Adjustment Factor

The culminating efforts of the implementation stage were dedicated to the development of the recommended signings tool of the Analysis Subsystem, as well as the FutureScope settings related to the User Management Subsystem and final tweaks to the CSS and HTML code related to the visual aspects of the web application's interface.

The final deliverables of this stage were a definitive web application, a User's Manual, and a Programmer's Manual, which are included in the appendixes of this document.

Chapter 6 - Testing

After the development of the code and the manuals, rigorous testing and quality assurance procedures were conducted to ensure that the software met the specified requirements and constraints. We have conducted both unit tests and integration tests to cover all code of our application.

We have followed the testing guidelines defined in *Software Testing and Quality Assurance* [36], to ensure that this process is done efficiently and meets the quality requirements posed by these testing standards.

Unit testing consists of testing individual modules and components of the system, validating the behavior of the functions that are included in the code. On the other hand, integration testing is the process of verifying the interactions between different parts of the system as a unified group.

6.1 Unit test

To cover as many test cases as possible, we have used a combination of two testing techniques:

- Black-Box testing: implies testing the functionality of the program without prior knowledge of the internal workings of the system, just the inbound and outbound parameters of a function.
- White-Box testing: requires complete knowledge of the system that is being tested.

For the testing of our web application, we have mainly used equivalence and assertion to check the behavior or the program in different scenarios. As defined by the aforementioned testing and quality assurance guidelines, equivalence class partitioning is used in cases where the input domain is too large for all its inputs to be used as test inputs. For example, for testing the User subsystem, we have used this

technique to create equivalence classes of the login process of a user, whether the credentials are valid, or they are not.

An example of the use of white-box testing, also known as structural testing, can be found in the validation of the system's connection to the Transfermarkt API. The focus in this particular case is on the data flow and control flow, that is, the logic behind the data retrieval and processing and error handling mechanisms.

6.2 Integration test

Once the modules of the application have been individually tested, in the process known as unit testing, the stability of the system as an ensemble must be verified. The main objective of integration testing is to ensure the correctness of the communication between the different modules that make up the application and have been unit-tested.

There exist various methods to conduct integration tests: Big Bang, Top-bottom, Bottom-top and Sandwich. Due to the size of the system, we have discarded Bing Bang, because it involves integrating all system components at the same time. In our case, we have used the Top-bottom model, which begins with high level modules and integrates low level modules progressively, opposite to Bottom top that begins with low level modules. Sandwich integration represents a midpoint between top-bottom and bottom top.

For instance, in this quality assurance process we have tested the recommended signings functionality corresponding to the Analysis subsystem. This tool involves communication between different modules: SmartScore calculation, player searching with constraints in the dataset, the use of the FutureScope settings, etc.

6.3 Coverage

Following quality assurance and testing guidelines is crucial to ensure the functionality and reliability of any software application and having a wide test coverage guarantees that all critical features of the software meet the quality standards. This is why every single line of the logic behind our application has been tested, as shown below.

Name	Statements	Miss	Coverage
Smartscore/admin.py	8	0	100%
Smartscore/apps.py	4	0	100%
Smartscore/dictionary.py	3	0	100%
Smartscore/forms.py	31	0	100%
Smartscore/models.py	87	0	100%
Smartscore/smartscore.py	79	0	100%
Smartscore/urls.py	3	0	100%
Smartscore/utils.py	287	0	100%
Smartscore/views.py	387	0	100%
tfg\settings.py	19	0	100%
tfg\settings.py	3	0	100%
TOTAL	**912**	**0**	**100%**

Table 6.1 - Test Coverage

In total, we have created 111 different definitions of test functions that have helped us to reach the 100% coverage of the application milestone. Utilizing the pytest-cov [37] plugin during the development of these tests, we were able to obtain real time insights into the percentage of code covered by them (Table 6.1 - Test Coverage demonstrates the complete coverage of the modules of the application), and additionally generating detailed HTML documents that indicate which parts of the code remain to be verified.

Chapter 7 - Project Management

To correctly develop this software application, the application's development process has followed a waterfall or cascade approach, as previously explained in the Introduction. The implementation of the waterfall model in this project has involved dividing the efforts into two-week blocks to ensure constant improvement and feedback.

7.1 Estimating: Function Point Analysis

In order to accurately estimate the size of our software project, we have used the measurement technique known as Function Point Analysis, described by Allan J. Albrecht [38]. This standardized estimation technique is commonly used because it brings a clear picture of measures such as the scope, duration, and quality of the project.

For all the following subsections, we have used Microsoft Excel [39] to create the tables and perform the mathematical calculations related to the Function Point Analysis estimation technique.

7.1.1 Unadjusted Function Point

Function Point Analysis involves categorizing software requirements and using mathematical formulas to weight their size and obtain an Unadjusted Function Point Count (UFP), which is an indicator of the countable functionality provided by the application to the user.

Firstly, we split the software requirements into the different functions the requirement is composed of, and we classified them into one of five categories of components specified by this type of analysis:

- **Internal Logical Files (ILFs):** set of data present in the system. It includes the logical files used to store and manage data within the system.
- **External Interface Files (EIFs):** logical files accessed by the system but maintained by a third party (external) entity.
- **External Inputs (EIs):** user interactions that are captured and are passed into the system to be processed or stored.
- **External Outputs (EOs):** the system presents information to external users or systems.
- **External Queries (EQs):** includes input and output components of the interaction between the system and external entities.

Secondly, after the requirements have been classified into component categories, their complexity level is categorized into Low, Average or High. This classification is made according to the number of elements referenced by each component, which can be one of the following:

- **Data Element Type (DETs):** single, unique, non-repetitive data field.
- **File Type Referenced (FTRs):** file types referenced by a transaction. They can either be ILFs or EIFs.

EXTERNAL OUTPUTS (EOs)						
	# of	# of	Complexity			
List of Outputs	DETs	FTRs	Low	Average	High	Notes
Player Profile Display	6	3	0	1	0	
Advanced Statistics Graphs	6	3	0	1	0	
Recommended Signings Display	5	4	0	1	0	
Search Result Display	3	1	1	0	0	
Squad Analyzing Display	6	2	0	1	0	
Saved Squads Display	3	2	1	0	0	
Saved Shortlist Display	3	1	1	0	0	
Summary			**3**	**4**	**0**	

Figure 7.1 – External Outputs

After tables for the five components are created in similar fashion to Figure 7.1, we proceed to the main step of the process, obtaining the UFP. Function Points (FPs) are calculated by weighing the tasks according to their complexity level.

The Unadjusted Function Point Count is obtained by adding the Function Points of each of the five categories: *UPF = SUM(FPs)*

UNADJUSTED FUNCTION POINT COUNT (FP)					
Function Type	Functional Complexity	Count	Weight	Function Points (FPs)	FP %
Internal Logical Files (ILFs)	Low	3	7	21	
	Average	3	10	30	
	High	2	15	30	50%
External Interface Files (EIFs)	Low	0	5	0	
	Average	0	7	0	
	High	1	10	10	6%
External Inputs (EIs)	Low	5	3	15	
	Average	2	4	8	
	High	2	6	12	22%
External Outputs (EOs)	Low	3	4	12	
	Average	4	5	20	
	High	0	7	0	20%
External Queries (EQs)	Low	0	3	0	
	Average	1	4	4	
	High	0	6	0	2%
Total Unadjusted Function Point Count				**162**	**100%**

Figure 7.2 – Unadjusted Function Point Count

7.1.2 Value Adjustment Factor

The Value Adjustment Factor (VAF) is used in Function Point Analysis to adjust the UFP according to several factors that influence the quality and complexity of the software application.

In this methodology, there are 14 factors, the General System Characteristics [40] (GSCs), that are graded on a scale from 0 to 5 and described briefly below.

1. **Data Communications:** communication facilities to aid in the exchange of information with the system.
2. **Distributed Processing:** distributed data and processing functions handling.
3. **Performance:** requirements related to response time and throughput.
4. **Heavily Used Configuration:** importance of specific equipment requirements in the design process.
5. **Transaction Rates:** frequency of transaction processing and its influence on the development of the project.
6. **Online Data Entry:** percentage of the information that requires online data exchange.
7. **Design for End User Efficiency:** importance of user efficiency in the design of the application.
8. **Online Update:** the application's ILFs are updated online.

9. **Complex Processing:** involvement of elaborated logical or mathematical processing in the application.
10. **Usable in Other Applications:** the project has been developed to ensure adaptability to meet an array of users and platforms.
11. **Installation Ease:** complexity of conversion and installation of the application.
12. **Operational Ease:** information related to the usage of the application is provided and its complexity.
13. **Multiple Sites:** consideration of multiple installations during the design and development of the project.
14. **Facilitate Change:** the application is conceived taking into account facilitating change.

The sum of all the grades for the GSCs is calculated to give out a value called Total Degree Influence (TDI), which is used in the calculation of VAF. In the table below, the grade for each GSC and a brief explanation is provided, as well as the formulas and values for the TDI and VAF.

VALUE ADJUSTMENT FACTOR (VAF)		
General Systems Characteristics	Degree of Influence (0-5)	Description
1. Data Communications	3	Fetches information from external sources
2. Distributed Processing	0	Processing is done locally
3. Performance	4	Requires real time data updates and calculations
4. Heavily Used Configuration	2	Basic browser compatibility requirements
5. Transaction Rates	3	Involves frequent transactions
6. Online Data Entry	2	Users interact mostly with local data
7. Design for End User Efficiency	3	Interface designed for responsiveness
8. Online Update	5	Updates through git repository are instantaneous
9. Complex Processing	3	Elaborated algorithms that involve multiple operations
10. Usable in Other Applications	5	Open Source, documentation provided
11. Installation Ease	2	Documentation explains the installation process
12. Operational Ease	5	Clear documentation
13. Multiple Sites	0	Only local installation is considered
14. Facilitate Change	5	Thanks to Git repository and documentation
Total Degree of Influence (TDI)	42	*Calculated (sum of the above)*
Value Adjustment Factor (VAF)	**1,07**	*Calculated ((TDI*0.01)+0.65)*

Figure 7.3 – Value Adjustment Factor

7.1.3 Adjusted Function Point

Adjusted Function Point Count (AFP) is the metric of Function Point Analysis that considers the size of the project and its complexity. It is calculated using UFP and VAF, calculated in the previous sections.

Summary Estimates		
Unadjusted Function Point Count	162	*From FP worksheet*
Processing Complexity Adjustment Factor	1,07	*From PCA worksheet*
Adjusted Function Point Count (AFP)	**173**	*Calculated: (FP*PCA)*

Figure 7.4 – Adjusted Function Point Count

7.1.4 Size of the Software Project

The next step of this stage is to estimate the project size of the project. In order to do this, we are going to determine the size of the project according to the IFPUG Function Point Analysis Method [41].

Relative Size	Size Code	Function Point Size
Extra-extra-small	XXS	< 10
Extra-small	XS	10 – 30
Small	S	30 – 100
Medium1	M1	100 – 300
Medium2	M2	300 – 1000
Large	L	1000 – 3000
Extra-large	XL	3000 – 9000
Extra-extra-large	XXL	9000 – 18000
Extra-extra-extra-large	XXXL	> 18000

Table 7.1 - Size Categories

According to the table above, our project falls into the Medium1 (M1) category, as it consists of 173 Function Points.

7.1.5 Effort Estimation

The final step of this process consists of the calculation of an effort estimate, measured in person-days. The formula used to get this estimate is the following:

*Effort Estimate = (Adjusted Function Point Count / Delivery Rate) * Days per person-month*

- The Adjusted Function Point Count (AFP) was calculated previously (AFP = 173)
- The Delivery Rate (DR) is the estimated Function Points that can be delivered by a person in a month and are measured in Function Points per person-month. In this project, we have established that DR = 18 FPs/person-month.
- Days per person-month (DPM) refers to the conversion rate of person-months of effort to person-days of effort, that is, the number of working days of a person in a month. For the estimation of our application, DPM = 17,5 days per month.

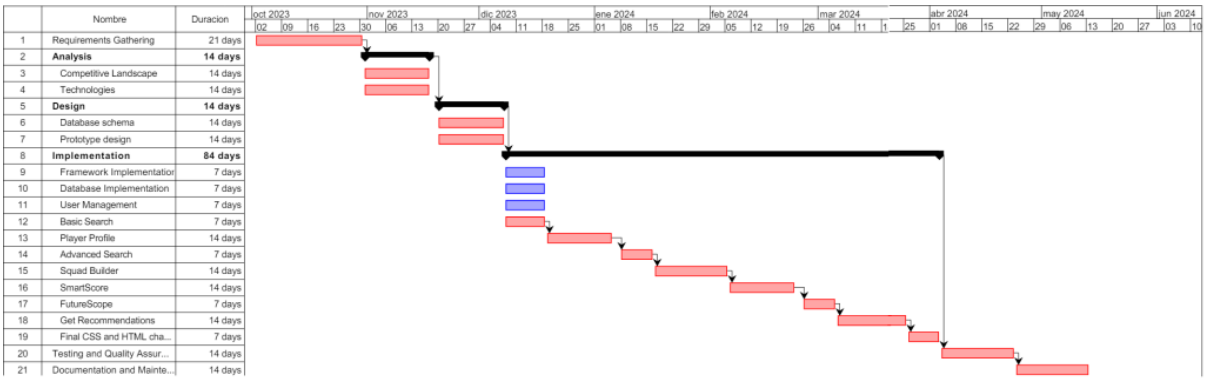
Summary Estimates		
Unadjusted Function Point Count	162	*From FP worksheet*
Processing Complexity Adjustment Factor	1,07	*From PCA worksheet*
Adjusted Function Point Count (AFP)	173	*Calculated: (FP*PCA)*
Delivery Rate (DR)	18	*In FPs/person-months*
Days per Person-Month (DPM)	17,5	*Days per month*
Effort Estimate (EE)	**168**	*Calculated: (AFP/DR) * DPM*

Figure 7.5 – Effort Estimate

The Effort Estimate for the implementation of the project is 168 person-days.

7.2 Planning

The estimation for the timeline of the development of this project have been obtained by using the project management software tool ProjectLibre [42]. This open-source software has been used to generate a Gantt chart for our project, which is included below in Figure 7.6.



	Nombre	Duracion
1	Requirements Gathering	21 days
2	**Analysis**	**14 days**
3	Competitive Landscape	14 days
4	Technologies	14 days
5	**Design**	**14 days**
6	Database schema	14 days
7	Prototype design	14 days
8	**Implementation**	**84 days**
9	Framework Implementation	7 days
10	Database Implementation	7 days
11	User Management	7 days
12	Basic Search	7 days
13	Player Profile	14 days
14	Advanced Search	7 days
15	Squad Builder	14 days
16	SmartScore	14 days
17	FutureScope	7 days
18	Get Recommendations	14 days
19	Final CSS and HTML cha...	7 days
20	Testing and Quality Assur...	14 days
21	Documentation and Mainte...	14 days

Figure 7.6 – Planning Gantt Chart

The main takeaways drawn from the chart are that most of the efforts have been dedicated to the Implementation stage, which expanded from the months of December to the beginning of April.

Figure 7.7 – Dependency Chart

7.3 Resources

The development team for this project consists of four members, on one hand, Gallego and Moreno, in charge of the development of the application as well as writing every piece of documentation, and two advisors, professors Elena Gómez and José Ignacio Requeno.

To illustrate the resource/task assignment for this project, we are going to use the following table, which considers the main activities that make up the project.

Activity	Contributor
Project Management	Iván Gallego Pablo Moreno
Counseling, Consulting	Elena Gómez José Ignacio Requeno
Analysis	Iván Gallego Pablo Moreno
Design	Iván Gallego Pablo Moreno
Coding	Iván Gallego Pablo Moreno
Testing	Iván Gallego Pablo Moreno
Documenting	Iván Gallego Pablo Moreno

Table 7.2 – Resource Allocation

In terms of financial resources, the only capital invested in this project was destined to the purchase of a copy of Football Manager 2024, which was used for the objective of obtaining the database of the application.

7.4 Risk Management

In software development, assessing the possible circumstances or events that may hinder the project's success is a fundamental part of minimizing their potential negative consequences. In this chapter, we will identify which are these circumstances and assess their importance and provide strategies to minimize their impact.

7.4.1 Risk Identification and Categorization

Technical risks

- Dependency on external APIs: reliance on an external API to get the player's Transfermarkt valuation, this could pose a risk if the API experiences downtime or becomes unavailable.
- Compatibility issues: some browsers or devices may cause compatibility issues due to variation in settings, rendering or specifications.
- Data security: risks associated with data security include unauthorized access, vulnerabilities in the system or data breaches.

Resource risks

- Limited development resources: constraint the project timeline and quality of deliverables.
- Infrastructure limitations: the chosen technologies and infrastructure may lead to performance issues or suboptimal user experience.

Functional risks

- Incomplete functionality: the application may face risks related to incomplete or inconsistent functionality, compared to their description in the project's requirements.
- User interface shortcomings: poor usability, lack of responsiveness or confusing navigation may affect user's satisfaction and adoption of the product.

Operational risks

- System maintenance: changes to the database, image folder or fragments of the code may require downtime or disrupt normal operations.

Management risks

- Communication challenges: poor communication among team members could lead to misunderstandings, conflicts, or delays.
- Scope creep: in the case that new features or requirements are introduced during the project development phase, scope creep may occur, leading to delays or changes in constraints.
- Poorly defined scope: issues may arise if the scope has not been well defined or if the development process is not clear.

Quality risks

- Testing limitations: inadequate testing coverage or ineffective strategies may result in undetected defects, bugs, or usability issues in the final product.
- Insufficient documentation: a lack of documentation or low-quality one may create issues that hinder a robust quality assurance process.

External risks

- Market changes: changes in consumer needs, preferences or the competitive landscape may impact on the project's success.

7.4.2 Risk Analysis and Mitigation

Only when we comprehend and assess the importance of the previously mentioned risks, we are able to tackle them and mitigate them.

To do this, the risks must be analyzed, taking into consideration the likelihood of them occurring and the impact they would have on the project if they were to appear, performing a qualitative analysis and giving their negative impact a numeric grade ranging from 1 to 25.

Furthermore, mitigation strategies must be defined, explaining what actions are taken to minimize or completely eliminate the negative impacts of these circumstances or events, taking into account the priority defined previously.

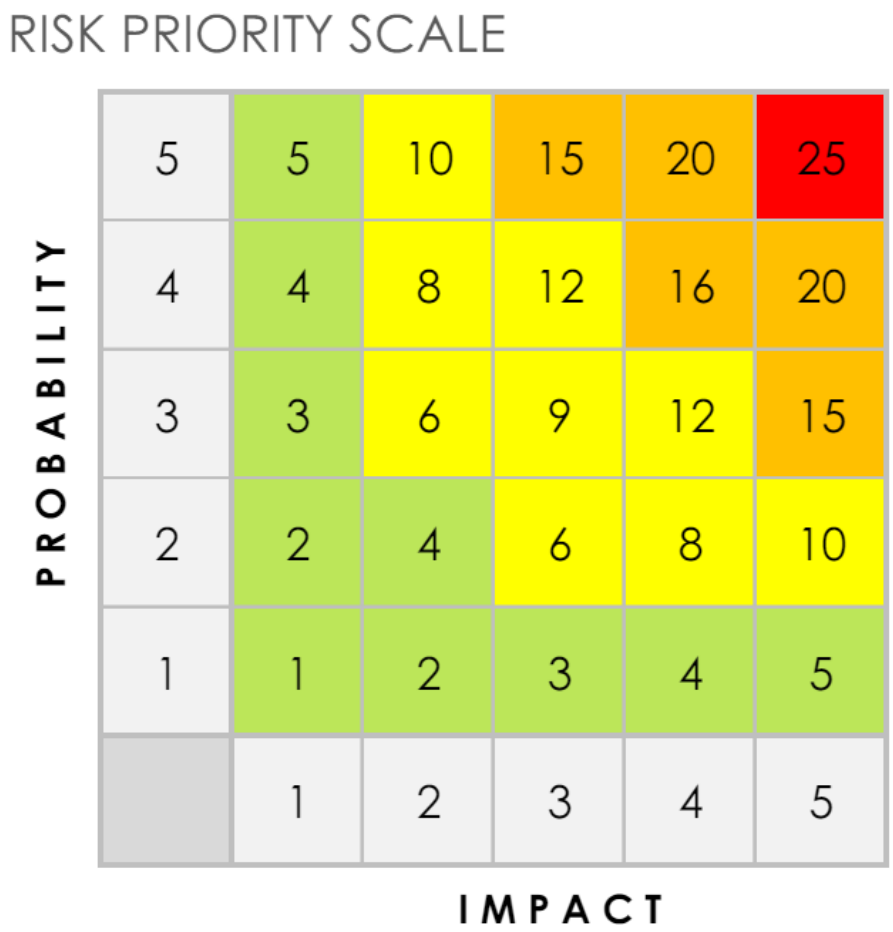


Figure 7.8 – Risk Priority Scale

Using Microsoft Excel, we have created two tables. The first one is depicted in Figure 7.8 and describes the scale used for identifying the risks' priority according to the probability of occurrence and impact, and the second one, Figure 7.9, includes a description of the impact, priority level and mitigation strategies for each of the risks.

This is the main reason why the previous evaluation of risks has been taken into consideration in each of the stages of development of this project, together with the pertinent monitoring and control of these risks.

RISK DESCRIPTION	IMPACT DESCRIPTION	IMPACT LEVEL	PROBABILITY LEVEL	PRIORITY LEVEL	MITIGATION STRATEGY	RISK TYPE
Give a brief summary of the risk.	What will happen if the risk is not mitigated or eliminated?	Rate 1 (LOW) to 5 (HIGH)	Rate 1 (LOW) to 5 (HIGH)	(IMPACT X PROBABILITY) Address the highest first.	What can be done to lower or eliminate the impact or probability?	Risk categorization
Dependency on external APIs	Data may become unavailable	4	2	8	Pre-load API data when first loading the dataset, monitor API availability	Technical Risk
Compatibility Issues	User experience could be suboptimal, some features may not work	2	3	6	Ensure that the application works with most common systems	Technical Risk
Data Securiry	Data leaks, security breaches	5	2	10	Handle non-sensitive information, provide robust user authentication	Technical Risk
Limited Development Resources	Constraint project timeline, affect quality of derivables	3	4	12	Robust project plannification, control and monitorization	Resource Risk
Infrastructure Limitation	Performance issues, suboptimal user experience	2	1	2	In-depth comparative analysis of the chosen technologies.	Resource Risk
Incomplete Functionality	Inconsistent or incomplete functionality compared to project objectives	3	3	9	Realistic objectives and strong requirement analysis	Functional Risk
User Interface Shortcomings	Suboptimal user experience	2	2	4	Simple and optimized interface design	Functional Risk
System Maintenance	Disruption of normal operations	4	1	4	Local handling of data and logic	Operational Risk
Communication Challenges	Misunderstandings, conflicts or delays in the project's development	3	3	9	Transparent communication, clear objectives and requirements	Management Risks
Scope Creep	Delays or changes in project constraints	3	1	3	Clear objectives and requirments that are stricly followed throughout development	Management Risks
Poorly Defined Scope	Delays or changes in project constraints	4	2	8	Definition of objectives and requirements before the implementation stage	Management Risks
Testing Limitations	Undetected bugs, issues or usability issues	4	3	12	Exhaustive testing of the entire application	Quality Risks
Insuficient Documentation	Hinder quality of the final product, user experience	3	3	9	Complete documentation of every aspect of the project	Quality Risks
Market Changes	Impact the project's success	5	3	15	Strong state of the art and competitive landscape to anticipate consumer need trends	External Risks

Figure 7.9 – Risk Evaluation

Chapter 8 - Project Control and Monitoring

8.1 Change Control, Monitoring and Verification

The project has been developed using GitHub as a version control tool, having a joint repository in which each of us coded in our own branch, and the consistency of the code was ensured through pushing, pulling, and merging branches from the GitHub repository.

To monitor the progress of the project and the status of the tasks corresponding to the biweekly block, we have used a Jira Board. In the Jira Software tool, a project can be created, and by using its built-in chronogram, backlog and board features, we were able to effectively follow project progress. For each two-week effort we had a board with the tasks that were set as objectives for the period, which were classified under different modules and given a status tag according to their completion situation. Using this software allowed us to always account for the tasks that needed to be done in that period, and the ability to assign tasks to a user permitted seamless collaboration between the team members.

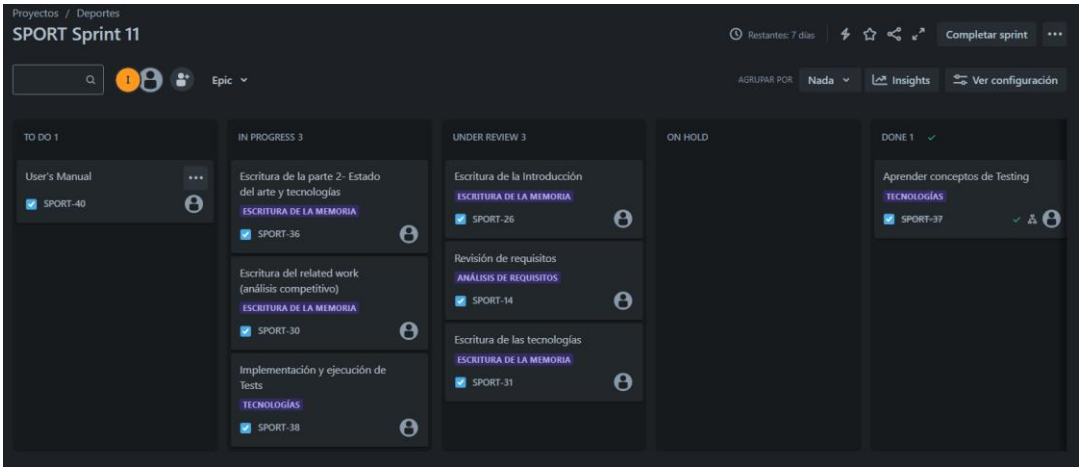


Figure 8.1 – Jira Board

Finally, in our biweekly meetings, the verification of the tasks was done together with our tutors. This is a way of ensuring that the project is developing in the right path, and if mistakes or possible enhancements are found or questions are posed, they are quickly and effectively tackled. These meetings were held either in person, in the faculty, remotely, through the Google Meet platform, or as a mixture of the two, depending on the availability of the members of the project.

8.2 Difficulties Found

Holding the development of this project to high standards has posed a challenge in each of the stages of the affair.

In the first stage, analysis, the main obstacles that we found were that the existing platforms in the market were extremely secretive and hermetic about their modus operandi, and we had little information about the technologies they used or the way they were implemented. Furthermore, when pondering what technologies were suitable for the vision that we had of our project, we were required to investigate and adopt numerous concepts and ideas we were unfamiliar with.

Secondly, defining the requirements and gathering the constraints to hold our project to the highest standard was tedious, especially considering that the knowledge regarding these procedures was deeply buried in our memories.

The implementation stage was clearly the bumpiest one. Being the one that took the longest time and effort, we were finding new challenges every single day. First, we were not experts using the Django framework, so it took some getting used to all its intricacies and peculiarities. The early days of this stage were dedicated to having a working web application, which we then could use as a backbone to then implement the features at a time.

Having a working database was a challenge. Standardizing formats for each of the fields took some discussion among the team, and deciding what to do when some of the fields were unavailable also posed a challenge. The biggest debate at those moments was the size of the database: we wanted to have as many players and as much information about them as possible, but without compromising the performance of the application, and finding a balance between those two took some time.

Another issue we found was displaying some visualization tools and having them be dynamically resized according to the user's screen size. The CSS and JavaScript code for these types of functionalities is not hard, however, finding the right settings was a matter of a lot of trial and error.

New challenges came when we had to implement the "Get Recommended Signings" feature. The algorithms to assess the quality of each player took way too long, as the application had to perform a lot of logical operations for each individual player on the database (as previously mentioned, this number is upwards of 12.000). The average turnaround time for these operations was around five minutes, which is not what was specified in the Performance Requirements. To tackle this problem, we added some preliminary filters to this functionality and modified the original performance requirements. By filtering preemptively, the players by positional profile (for example, including players that play in any of the midfielder positions) and behavioral archetype (their playstyle, for example, creative midfielders), as well as their preferred foot, we were able to preselect a set of players that have outstanding metrics, to then apply the grading and selection algorithms to only those players. While it is still possible to get recommendations without selecting these filters, it is not recommended, and the performance requirements only specify the expected turnaround time when applying them.

Chapter 9 - Conclusions and Future Work

To conclude, it is important to review the objectives that were determined at the beginning of the planning process and evaluate if the project meets the expectations that were set:

1. **Comprehensive Football Management Platform:** the application covers the aspects mentioned, providing the user with robust tools that enwrap scouting, squad building and strategy scheming.
2. **Data-driven Decision Making:** the algorithms described in the Analysis Subsystem use advanced statistics to evaluate the quality of players and squads, and the data visualization tools illustrate the conclusions.
3. **Player Scouting and Recruitment:** the application focuses on providing the user with robust tools to determine which players fit their project best.
4. **Squad Building:** the Squad Builder tool maximizes the potential of football teams by providing extensive insight and allowing them to improve their weaknesses.
5. **Long-term Strategic Planning:** the influence of the FutureScope settings, which are fully customizable by the user, on every aspect of the application allows it to provide insight that caters to the user's circumstances and preferences.
6. **Customization and Flexibility:** every design choice has been made to cater to the needs and preferences of a variety of users. That includes both stylistic and functional decisions.
7. **Scalability and Reliability:** the application has passed rigorous quality assurance and testing procedures and meets the requirements specified in the analysis stage.
8. **Continuous Improvement:** by providing clear and comprehensive documentation, the project and its code is accessible and customizable, which allows tweaks and improvements to be easily introduced.

In the future, our maximum priority will be hosting the application in a suitable web hosting environment. Services such as Amazon Web Services [43] will be considered as options to allow the users to access our product without having a copy in their local system. Code, database and elements such as player images and flags will be accessed remotely.

While our database provides the user with a wide range of statistics and attributes for a multitude of players, we aim to substitute it with one that does not use simulated football matches to obtain the data. If the budget would allow us to, we would implement one of the databases that uses real-world information, as we think that the users would greatly benefit from being able to analyze the actual performances of the players.

Another big step for our application would be storing information about football matches in our database. Working with parameters such as the number of shots on target, the number of dribbles or the accuracy of lateral crosses during games and implementing technologies such as deep learning and neural networks, we would be able to analyze and extract the factors that make teams win fixtures.

With the aforementioned data and technologies, our Squad Builder functionality could be greatly enhanced, by analyzing the squad as a unit and finding which areas of the team need to be improved to take it to the next level.

Chapter 10 - Personal Contributions

While we have had different tasks throughout the development of the project, we have been working alongside each other in every stage of it. The workload has been balanced ensuring equal implication all through the entire process, as it can be seen in Figure 10.1, obtained from our GitHub Repository.

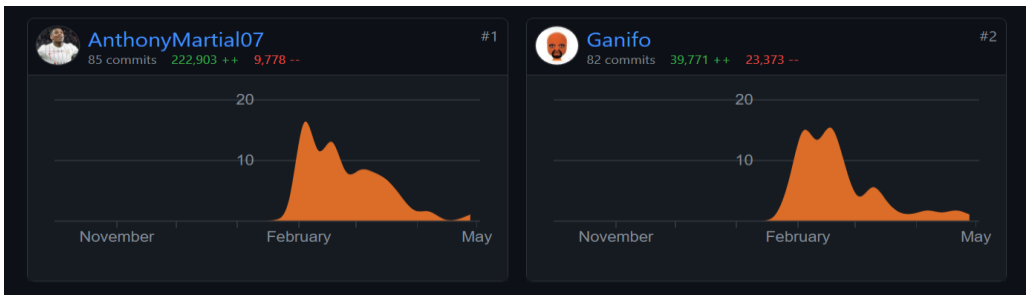


Figure 10.1 – GitHub Contributions

Iván Gallego

Starting from the analysis stage, I researched the competitive landscape, evaluating the strengths and weaknesses of similar platforms, as well as trying to get in touch with their developers for insight on the technologies they used and the approach they followed. While this was unsuccessful, it helped us realize the competitive nature of the field we were getting into and gave us ideas to gain an edge over the competition.

Belonging to the same stage, I also oversaw designing the division in subsystems of the requirements we had agreed upon. This involved ensuring that the objectives we had defined for the project were met by the subsystems, and that the functionalities associated with them were clearly defined.

An important aspect of the analysis stage is defining the functional requirements for the project. As explained in Chapter 3, we used the Modelio software for creating the activity diagrams, and I was responsible for half of them (and their

corresponding structured natural language tables), while my teammate oversaw the remaining half.

The next step of the project was the Design stage, which involved the creation of class diagrams, the dataset schema, and a prototype of the interface. The principal objective of my work was to generate exhaustive class diagrams that cover and explain the different interactions and functionalities throughout the set of components of our system. They were also created using the Modelio tool, which has been a key tool in this project.

As explained previously, for the dataset of our application, we were going to extract the information from the Football Manager 2024 videogame. To do this, I simulated the 2023-2024 football season, and once the processing ended, I created a custom view inside the "Search" tab in the video game and exported the information of the players, generating raw HTML file which after being treated would serve as the dataset for our web application.

The implementation stage of this software project was the time to put the gathered ideas to work, and while we both were simultaneously working on the same features, lending a helping hand when the other one became stuck with a problem, we had differentiated priorities when it came to our development efforts.

My first efforts in the implementation stage were focused on having a structured interface, a baseline for the development of the functionality, and that is why I made the creation of a rudimentary interface that followed our design principles (sidebar navigation, an auxiliary top bar and consistent styling and a responsive framework) my number one priority.

When that was done, I directed my focus onto my first important piece of functionality, the advanced search tool, corresponding to the Player Management

Subsystem. This was particularly challenging as at that point of development, we were not familiarized yet with working with the elements of the dataset while combining multiple programming languages.

An important piece of work was also obtained from the development of this functionality, which is the football field with the interactive player positions. Having the positions be dynamically placed on the field (the field's size is dynamically adjusted according to several parameters such as screen size) posed an important challenge, as it involved a lot of work transforming elements from one programming language to another, while maintaining the responsiveness. However, the efforts were worthwhile, as the element of the football field with the positions is present in several functionalities of our application.

The remaining part of my efforts were then dedicated to the Analysis Subsystem. We divided the work related to the squad builder functionality, for which I was responsible for the analysis part of the functionality, that is, the evaluation of the players and the algorithms to find the replacement players that improve the weak links of the football squad,

Regarding the SmartScore algorithms, generating algorithms that generate accurate evaluations of the players occupied my focus for a couple of weeks, and then tinkering with their parameters to fine tune the algorithms was a constant process during the remaining part of this stage.

Having a deep understanding of the SmartScore algorithms and being familiarized with the intricacies of player evaluation, it made the most sense for us that I was the one responsible for the recommended signings feature of our application. This was the last major addition I made during the implementation stage, with my remaining efforts being dedicated to smaller changes and tweaks or helping Pablo Moreno with any difficulties he found.

For the testing stage, Pablo Moreno was the main contributor, and my focus was dedicated to aiding him with the creation of tests of the pieces of code I was most familiar with and helping him with any issues he encountered.

The last weeks of the development my focus was on writing the documentation for the project, as I am confident of my skills and have more affinity towards tasks of this nature.

As a conclusion, I think it is important to note that, while having a clear division of work, every single aspect of the project was a joint effort. As we had clear and effective communication, constant contributions were made by the both of us when they were needed, and a continuous exchange of ideas is one of the main characteristics I would associate with the development of this project.

Pablo Moreno

Having a clearly defined plan of action and structure for the development of this project has personally helped me guide my efforts and follow the strict deadlines we had set for ourselves.

During the first stage of the project, analysis, my focus was directed towards studying the available technologies and analyzing and evaluating which ones had the characteristics that were best suited for the project we had on our hands.

Once we settled on the technology we were going to use and the scope we were going to cover, I was responsible for the definition of the non-functional requirements and the creation of half of the activity diagrams and structured natural language tables for the functional requirements, which was not particularly complex, but it involved copious amounts of work.

For the design stage, I was responsible for transforming the document corresponding to the dataset of our application that Iván Gallego generated from the Football Manager 2024 video game, into a CSV file that our platform was going to use, as well as reduced versions that were going to be used in earlier versions of the application during the development process for functionality testing purposes.

In this stage I was also tasked with the creation of a mockup of the interface of the application. For the creation of this prototype, I captured the ideas we had agreed on during the analysis stage as well as the principles defined during the weeks allocated to the design of the project and portrayed them in a digital render.

With the documentation and other byproducts of the analysis and design stages, we were ready to enter the most tedious phase of the software development process, the implementation, with clear and concise guidelines.

Firstly, I was responsible for the creation of the framework, that means generating the necessary files and documents for our Django application to have a foundation onto which we were able to deploy the functionalities related to the application.

My subsequent efforts were dedicated to implementing several elements of the Player Management Subsystem, namely, the basic search and the profile for the players. Using the most reduced version of the datasets we had created in the design phase, I was able to have a working basic search bar that displayed players that matched with the search query and then I created a basic player profile, extracting the attributes from each of the players in the database and displaying the basic ones.

The challenging part of the Player Management Subsystem was generating the graphs and other visualization tools related to the advanced statistics functionality.

This involved working with different programming languages and creating functions and algorithms for attribute comparison, display, and categorization.

From the Analysis Subsystem, I worked closely with Iván Gallego to create the squad builder tool, which contains different elements such as analysis algorithms, selection tools, data visualization and user management. Personally, I found this to be one of the most demanding tasks of the project, especially considering that we were beginning to become familiar with the structure of our application and discovering the intricacies of the technology we had chosen for it.

My focus shifted towards the functionalities related to the User Management Subsystem as soon as the previously mentioned tool was implemented. While I closely collaborated with Iván Gallego for the FutureScope settings, I was mainly responsible for these functionalities due to my previous experience with similar affairs.

I created the code related to user credentials management, sign up, login and other aspects of user account management, which was not especially complicated, but required a lot of work to ensure that the security requirements were met. Dealing with user's squads and shortlists was also a key aspect of the efforts of these weeks, as was their integration into other parts of the application (for example implementing the shortlisting of suggested players in the recommended signings tool).

For the last stage of the project, I was at the wheel of the creation and evaluation of the testing applications. With some help from my teammate and a lot of constant work, we were able to guarantee that 100% of the code was tested after the finalization of this phase.

Finally, I helped Iván with his commitment to finishing all of the pertinent documentation related to the project, including manuals and other elements that needed to be done.

Bibliography

[1] Wyscout. [Online]. Available: https://wyscout.hudl.com/app/?.

[2] Olocip. [Online]. Available: https://olocip.com/.

[3] StatsPerform. [Online]. Available: https://www.statsperform.com/.

[4] InStat. [Online]. Available: https://instatscout.com/.

[5] Scisports. [Online]. Available: https://www.scisports.com/.

[6] DocForge, "Web Application Framework," [Online]. Available: https://web.archive.org/web/20150723163302/http:/docforge.com/wiki/Web_application_framework.

[7] React. [Online]. Available: https://react.dev/.

[8] React, "React DevTools," [Online]. Available: https://react.dev/learn/react-developer-tools.

[9] Vue.js. [Online]. Available: https://vuejs.org/.

[10] Django. [Online]. Available: https://www.djangoproject.com/.

[11] University of California, Berkeley, "Model-View-Controller," [Online]. Available: https://patterns.eecs.berkeley.edu/?page_id=42.

[12] CGISecurity, "The Cross-Site Request Forgery," [Online]. Available: https://www.cgisecurity.com/csrf-faq.html.

[13] Angular. [Online]. Available: https://angular.io/.

[14] Visual Studio Code, [Online]. Available: https://code.visualstudio.com/.

[15] Amazon Web Services, "What is an Integrated Development Environment (IDE)," [Online]. Available: https://aws.amazon.com/what-is/ide/#:~:text=An%20integrated%20development%20environment%20(IDE,easy-to-use%20application..

[16] Git. [Online]. Available: https://git-scm.com/.

[17] GitHub. [Online]. Available: https://github.com/.

[18] JIRA. [Online]. Available: https://www.atlassian.com/.

[19] Slack. [Online]. Available: https://slack.com/.

[20] WhatsApp. [Online]. Available: https://www.whatsapp.com/.

[21] Google Meet, [Online]. Available: https://meet.google.com/.

[22] Google Docs, [Online]. Available: https://docs.google.com/.

[23] Google Drive, [Online]. Available: https://www.google.com/drive/.

[24] Football Manager 2024, [Online]. Available: https://www.footballmanager.com/games/football-manager-2024.

[25] Sports Interactive, [Online]. Available: https://www.sigames.com/.

[26] Eurogamer, "Everton signs Football Manager database," 17 November 2017. [Online]. Available: https://www.eurogamer.net/everton-signs-football-manager-database.

[27] Transfermarkt, [Online]. Available: https://www.transfermarkt.com/.

[28] "Transfermarkt API," [Online]. Available: https://transfermarkt-api.vercel.app/. [Accessed January 2024].

[29] Vercel. [Online]. Available: https://vercel.com/.

[30] Modelio. [Online]. Available: https://www.modelio.org/index.htm.

[31] Google Chrome, [Online]. Available: https://www.google.com/chrome/.

[32] Microsoft Edge, [Online]. Available: https://www.microsoft.com/edge.

[33] Mozilla Firefox, [Online]. Available: https://www.mozilla.org/firefox/.

[34] Brave, [Online]. Available: https://brave.com/.

[35] Canva, [Online]. Available: https://www.canva.com/.

[36] K. Naik and P. Tripathy, Software Testing and Quality Assurance, Wiley, 2008.

[37] Pytest-cov, [Online]. Available: https://pypi.org/project/pytest-cov/.

[38] A. J. Albrecht, in *Proceedings of IBM Application Development Symposium*, 1979, p. 83.

[39] Microsoft Excel, [Online]. Available: https://www.microsoft.com/en/microsoft-365/excel.

[40] Function Point Modeler, "General System Characteristics," [Online]. Available: http://www.functionpointmodeler.com/fpm-infocenter/index.jsp?topic=%2Fcom.functionpointmodeler.fpm.help%2Fditafiles%2Fconcepts%2Fcon-99.html.

[41] International Fuction Point Analysis Users Group , "Function Point Analysis Method," [Online]. Available: https://ifpug.org/ifpug-standards/fpa.

[42] ProjectLibre, [Online]. Available: https://www.projectlibre.com/.

[43] Amazon Web Services, [Online]. Available: https://aws.amazon.com/.

[44] Django, "Download," [Online]. Available: https://www.djangoproject.com/download/.

[45] Python, "Downloads," [Online]. Available: https://www.python.org/downloads/release/python-3120/.

[46] Git, "Downloads," [Online]. Available: https://git-scm.com/downloads.

[47] Django, "Instructions," [Online]. Available: https://django-extensions.readthedocs.io/en/latest/installation_instructions.html.

[48] Pytest, "Get Started," [Online]. Available: https://docs.pytest.org/en/7.1.x/getting-started.html.

APPENDIXES

Appendix A - Programmer's Manual

The objective of this manual is to provide the basics about our program so that any developer can adapt, modify, or improve the code for our application. The main objectives of this document are that any interested readers can:

1. Understand what programming languages, libraries and add-ons are needed to modify and execute the code.
2. Install any of the aforementioned requirements into their development environment.
3. Comprehend the structure of the application.
4. Understand how the code can be modified to amend the changes introduced by a third party.
5. Run the modified version of the application.

Requirements

Our web application is based on the Django web application framework, particularly, using version 5.0.1, which can be installed from their Downloads page [44].

Django is a high-level web framework based on the combination of several programming languages. In our project, the following languages have been used:

- HTML
- CSS
- JavaScript
- Python

The first three languages do not require any installation, but in order to compile Python programs, a valid version of it must be installed. As our application uses the 5.0.1 version of Django, which only supports Python versions 3.10, 3.11 and 3.12, one

of these has to be installed, however, we encourage the developers to get the latest version of Python 3.12 from their Downloads page [45], which was the one we used for the development of the application.

The application is published on a GitHub repository, so it is essential that the local machine has installed Git, which can be done from their Downloads page [46].

Python Libraries

While the Django Framework takes care of most of the Python libraries we have used in the development of the project, there are some that do need to be installed on the client's environment to be able to modify the entirety of the project.

Firstly, the Django-extensions library must be installed. Django-extensions is a collection of custom extensions for the Django Framework, which can be installed by following the guide on their Instructions page [47]. The purpose of these extensions in our project is to be able to import a different database. In /smartscore/import.py one can change the source file for the application's database, and after making the pertinent changes, the developer would need to run the following command inside the project's directory:

py manage.py runscript smartscore.import

The second module the developer would need to install is the pytest framework, which can be done by following the instructions in their Get Started page [48]. This module is used to perform unitary tests on the application, and they can be found under /smartscore/tests.

Structure of the Application

The entirety of the project is contained in the same git repository, and as previously mentioned, it is structured according to Django's Framework Model-View-Controller (MVC) architecture, which harmonizes the different components and allows a developer-friendly environment. In this Chapter we are going to expand on each of the subcomponents of the application, which we have divided into 6 categories.

Project Directory

The Project Directory is the nucleus of the application, containing the tools to run, manage and migrate it. Its main file is /manage.py, which can be located in the root directory of the application and other files are present under the /tfg folder.

Application Directory

When we mention the Application Directory, we are referring to the files that define the behavior and functionality of the application. They are present in the /smartscore directory, and they handle the logic of the web application. If a developer needs to introduce changes to the project, the most likely place for these changes is going to be one of the files present here.

Templates

Under the subcategory of Templates, we can find the user-facing layer of the application. The files present in the /smartscore/templates folder are written using the HTML language and contain the pages the user is going to see when using the web application.

Static

The Static subcategory is envisioned to contain the additional files that the web application is going to need to offer the best user experience. Under /smartscore/static we can find different types of Static components:

- JavaScript Content: present under /smartscore/static/js_content, these files define additional functionality of the HTML pages present in the Templates subcategory. The JavaScript code found in this folder often serves as a tool to provide extra functionality to the user-facing layer of the application or to communicate this layer with the Application Directory.
- Styling Content: present under /smartscore/static/style.cs, these documents specify the presentation of the interface of the application to the user.
- Multimedia Content: files such as images, fonts and other UI elements can be found under /smartscore/static, particularly, in the *img*, *images* and *fonts* folders. In these directories content such as player's images, countries' flags and other icons is located.

Database

This subcategory consists of the database files for the application. Currently, the data is obtained from a csv document and after importing it to the Django Framework, it is present as /db.sqlite3. This file contains all the relevant information related to the application, that is, the players and their statistics and the users and their saved information (squads, shortlists, settings, etc.).

Tests

The Tests category consists of the code related to the evaluation and the inspection of the functionality of the web application. The files related to testing can be found in the /smartscore/tests directory.

Execution of the Application

In this section, we will cover the information related to the execution of the application and the related content.

Downloading the Application

As previously mentioned, the application is hosted in a GitHub repository. To run the project locally, it is necessary to clone the repository to your local machine using the following command: *git clone* https://github.com/TGF-2023-24/Deportes

Running the Server

Running the server for the web application is a simple process once the requirements specified previously are met. This task consists of two simple steps:

1. Navigate to the directory where the manage.py file of the project is located.
2. Execute the following command: *py manage.py runserver*

A quick check will be performed and if no errors are found, the development server will start.

Imports

To make imports of the database, we will need the django-extensions module installed. Once this is done, navigate to the project's directory and run the following command: *py manage.py runscript smartscore.import*

Migrations

To change databases, we need to perform the migration procedures, which consist of the following steps:

1. Navigate to the project's root directory.
2. Run the following command: *py manage.py makemigrations*
3. Run the following command: *py manage.py migrate*

Accessing the Web Application

Once we have completed the steps to start the server we can either:

- Ctrl + click on the address shown in the console.
- Open a web browser and navigate to http://127.0.0.1:8000/

To access the admin side of the web application, the admin must navigate to http://127.0.0.1:8000/admin and log in as a user with admin privileges.

Appendix B - User's Manual

The objective of this manual is to provide the basics about our program so that any user can understand how to properly use our application. The main objectives of this document are that any interested readers can:

1. Run the application on their personal system.
2. Understand the different functionality of the application.
3. Comprehend the different metrics and statistics present on the application.

Requirements

Our web application is based on the Django web application framework, particularly, using version 5.0.1, which can be installed from their Downloads page.

To compile Python programs, a valid version of it must be installed. As our application uses the 5.0.1 version of Django, which only supports Python versions 3.10, 3.11 and 3.12, one of these must be installed, however, we encourage the developers to get the latest version of Python 3.12 from their Downloads page, which was the one we used for the development of the application.

The application is published on a GitHub repository, so it is essential that the local machine has installed Git, which can be done from their Downloads page.

Execution of the Application

In this section, we will cover the information related to the execution of the application and its functions.

Downloading the Application

As previously mentioned, the application is hosted in a GitHub repository. To run the project locally, it is necessary to clone the repository to your local machine using the following command: *git clone* https://github.com/TGF-2023-24/Deportes

Running the Server

Running the server for the web application is a simple process once the requirements previously specified are met. This task consists of two simple steps:

1. Navigate to the directory where the manage.py file of the project is located.
2. Execute the following command: *py manage.py runserver*

A quick check will be performed and if no errors are found, the development server will start.

User Signup and Login

The first step is to create an account to access every feature of the application, which can be done by clicking any of the "Sign Up" buttons. Once there, the user must provide valid credentials.

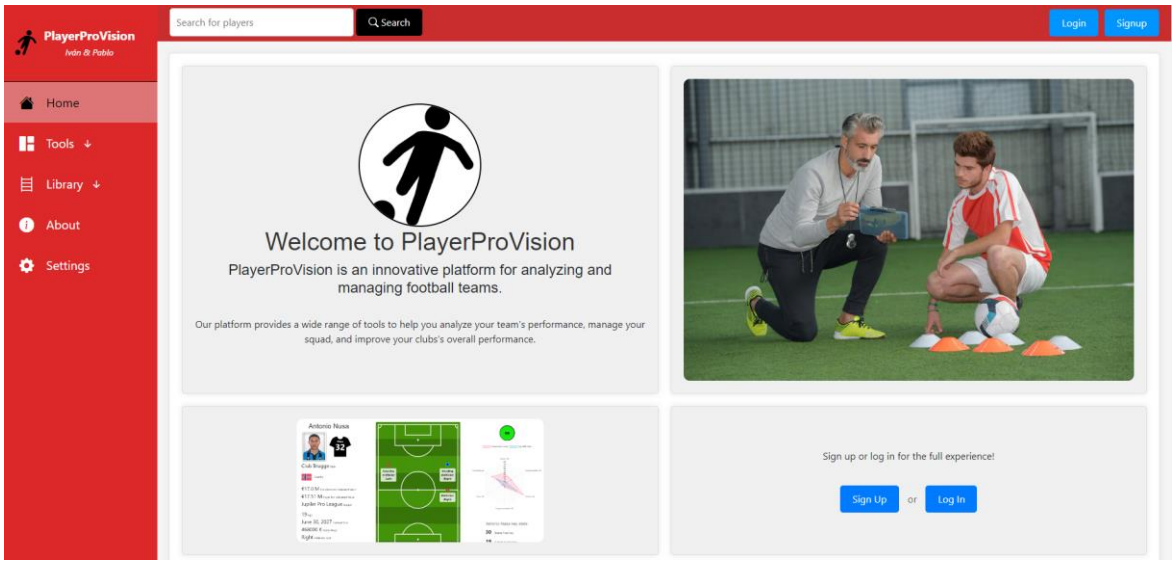


Figure B.1 – Home

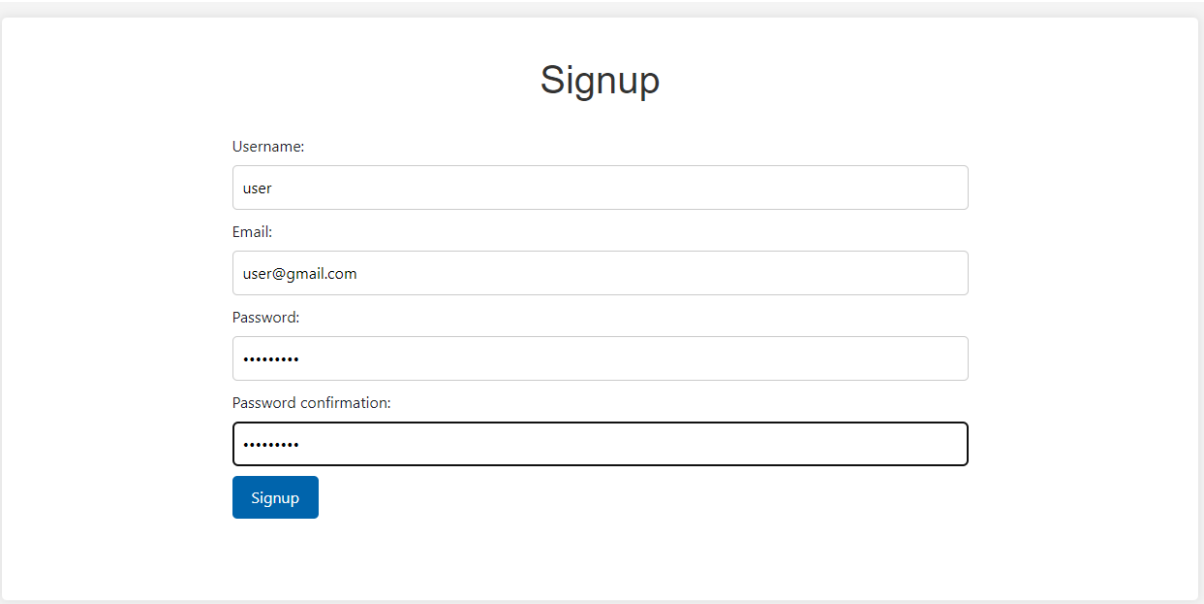


Figure B.2 – Sign Up

Once that step is completed, the user will get redirected to the home page where they can start to use the different features explained in this document. To log out, the user needs to click on the upper right button.

If a user is not logged in, it has limited access to the application's tools (most features of the application require the user to be identified to provide their complete functionality).

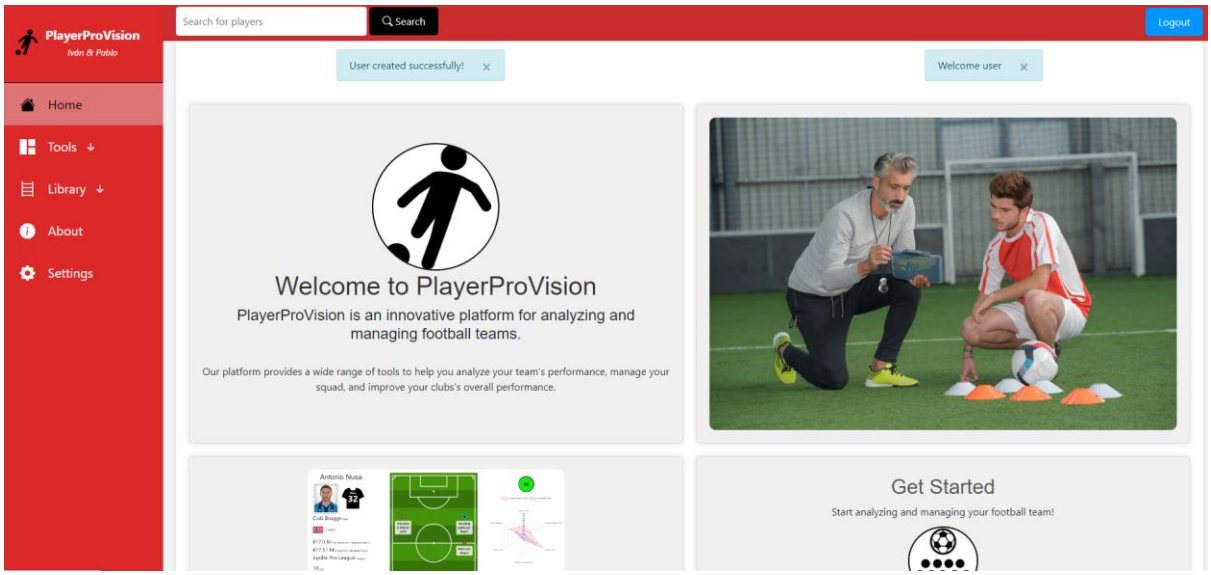


Figure B.3 – User Created

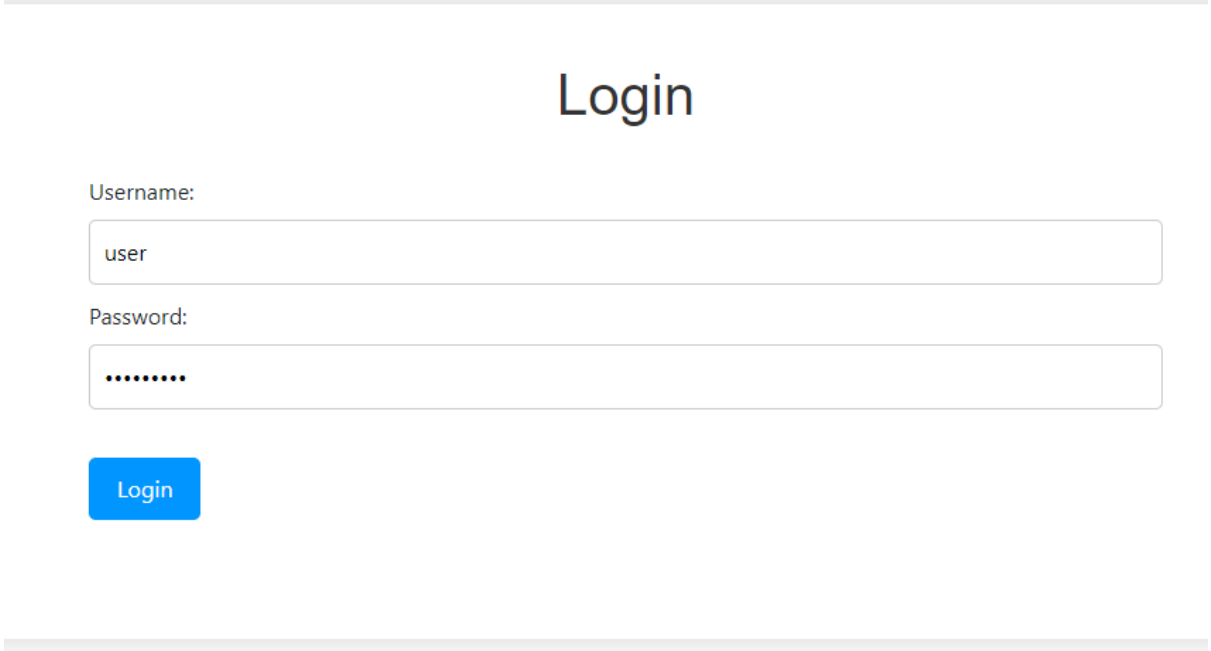


Figure B.4 – Log In

Sidebar

The sidebar contains dropdown menus which the user can interact with by hovering the mouse over. The main tools of the application are found inside one of the categories of the sidebar.

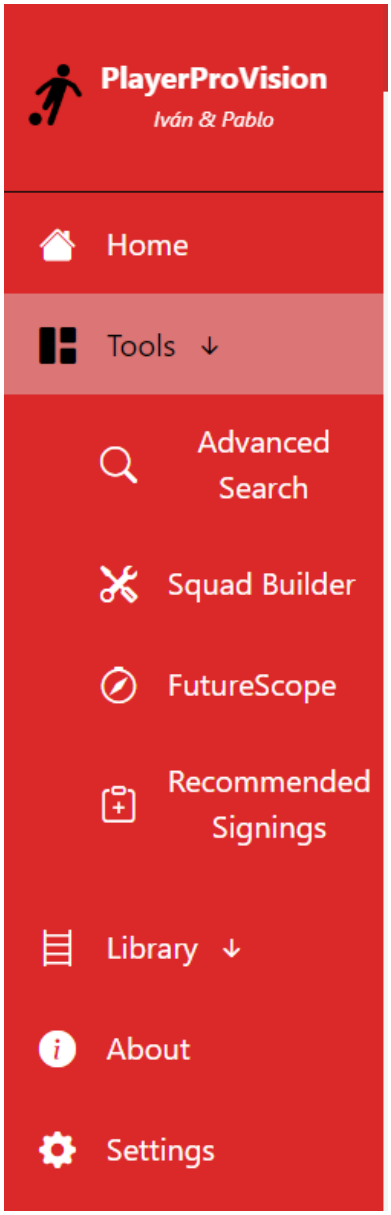


Figure B.5 – Sidebar Tools

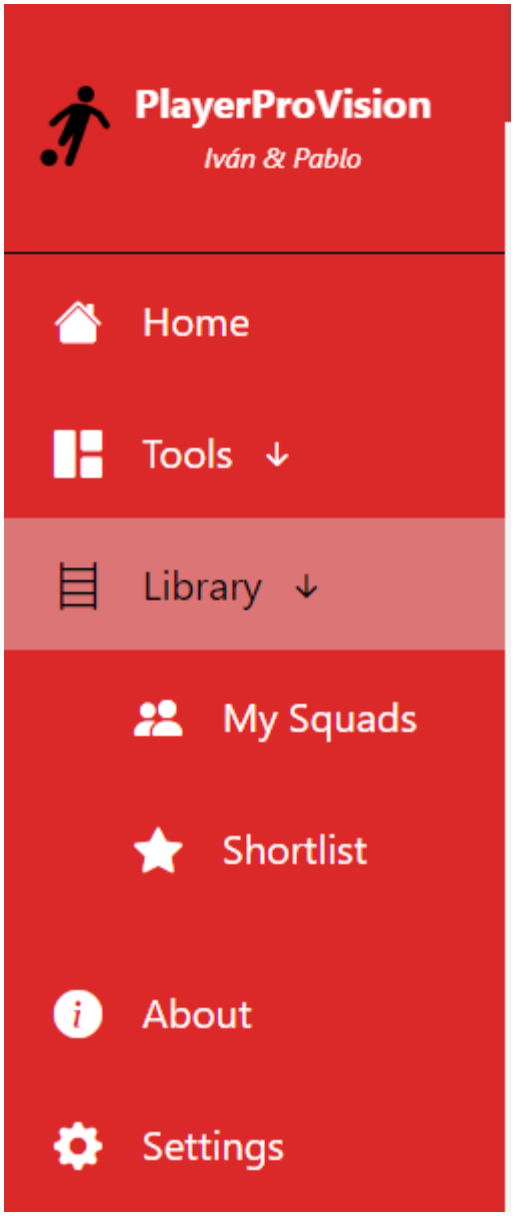


Figure B.6 – Sidebar Library

User Settings

A "Settings" button can be found on the sidebar, which will direct the user to that page, where the information related to the account can be updated.

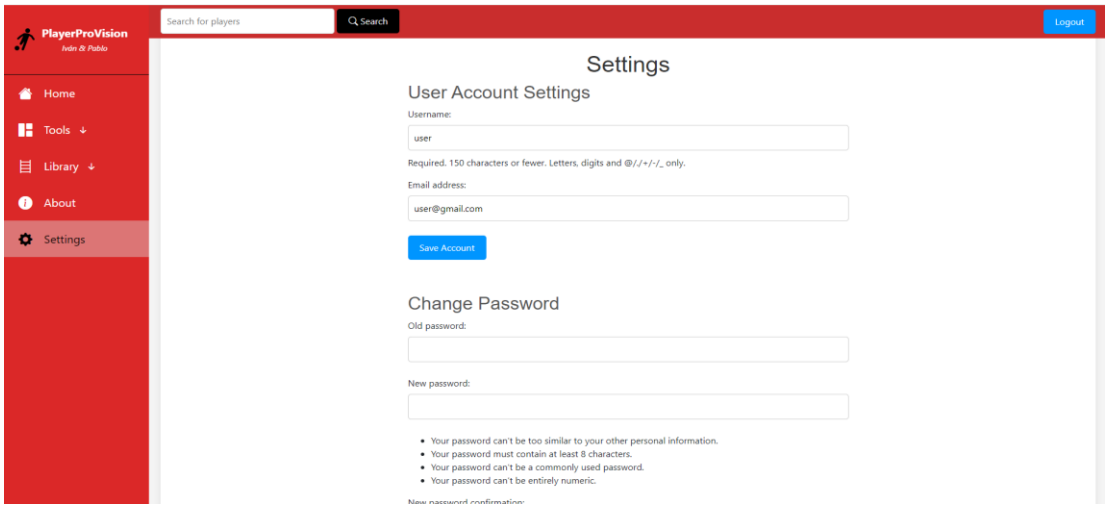


Figure B.7 – User Settings

Basic and Advanced Search

There are two different types of search queries that can be performed in our application:

- **Basic Search:** using the search bar present at the navigation bar (top of the interface). User's query will be matched with every player's name, club, and league.

Figure B.8 – Basic Search

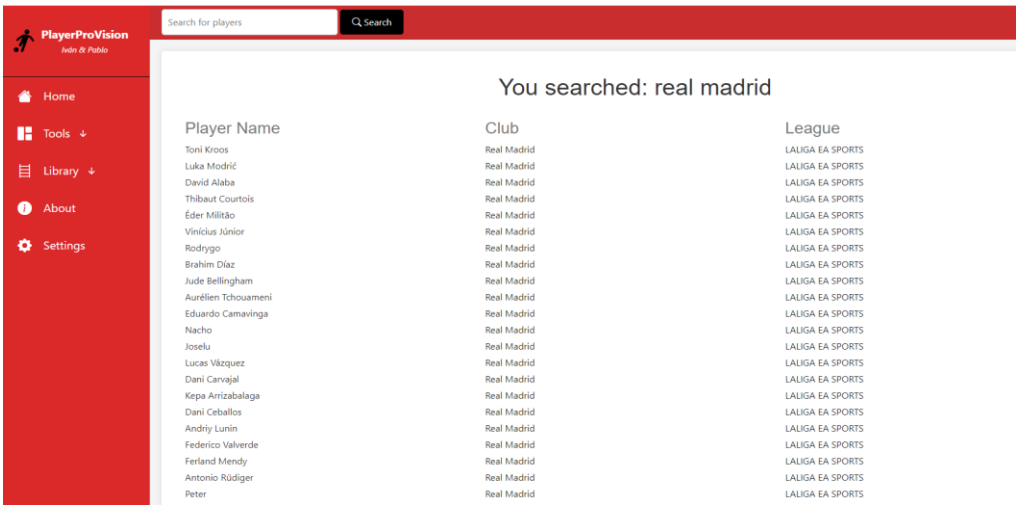


Figure B.9 – Basic Search Results

- **Advanced Search:** present under the "Tools" category on the sidebar. There is a football field image where the user can choose the positions they want the filtered players to play in. The user can also add filters related to players attributes and match statistics by clicking on the "Add Filter" button and selecting between the options presented on the dropdown menus.

Figure B.10 – Advanced Search

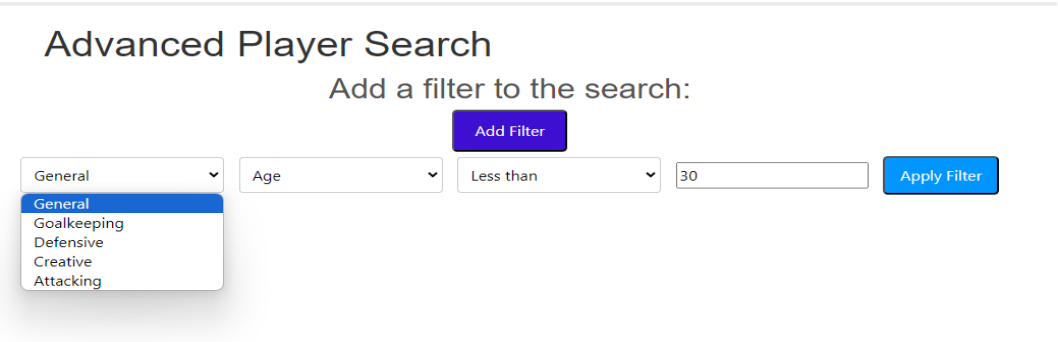


Figure B.11 - Advanced Search Add Filter

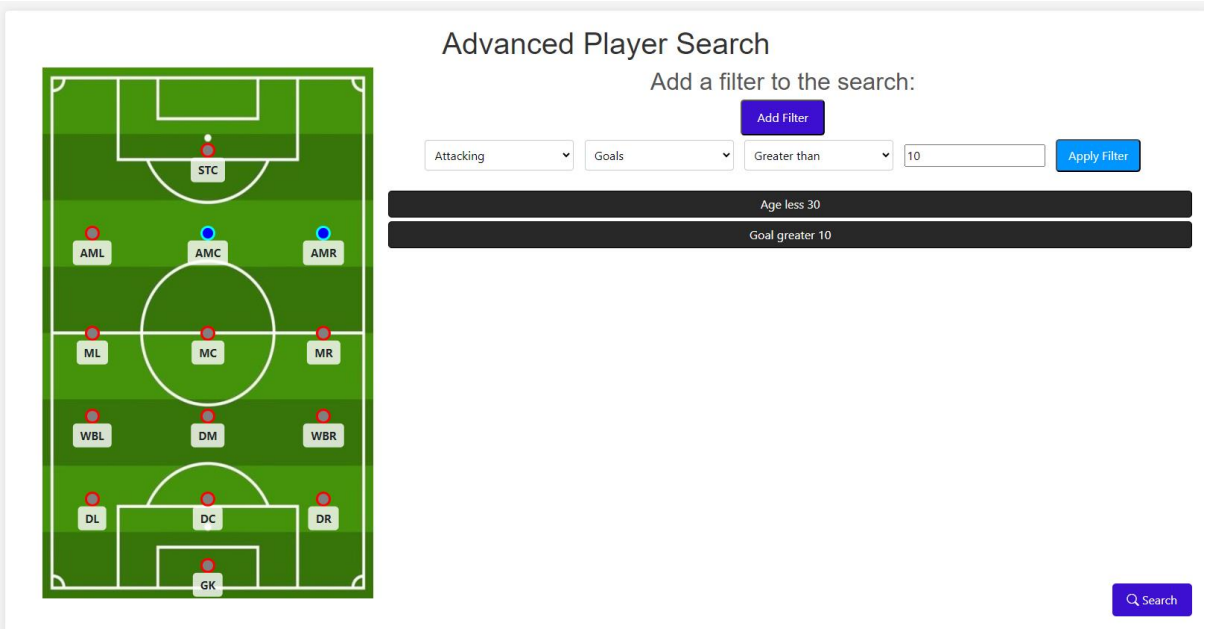


Figure B.12 - Advanced Search Query

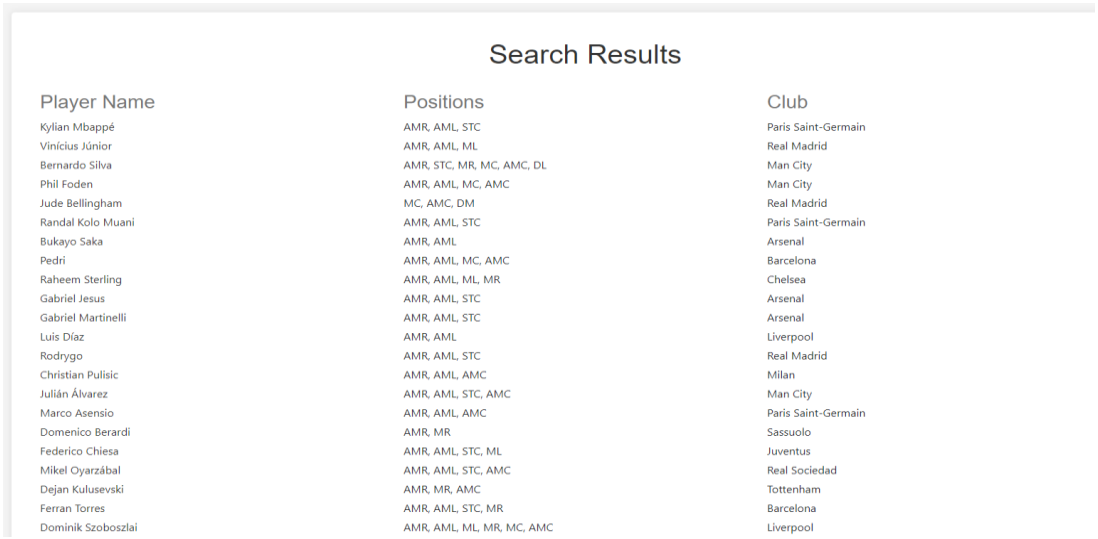


Figure B.13 - Advanced Search Results

Player Profile

There are multiple ways of reaching a player's profile, one of them being by searching (as explained previously) and clicking on the player's name in the "Search Results" page.

In a player's profile we can find basic information about the player, together with the positions where they are able to play. When selecting one of these positions, a graph comparing the player's matchday statistics to the average of other players in that position will show up, as well as a summary of those statistics and the player's SmartScore for the selected position.

Furthermore, if the user is logged in and has previously created a Squad, they can add the player to the squad.

Figure B.14 – Player Profile

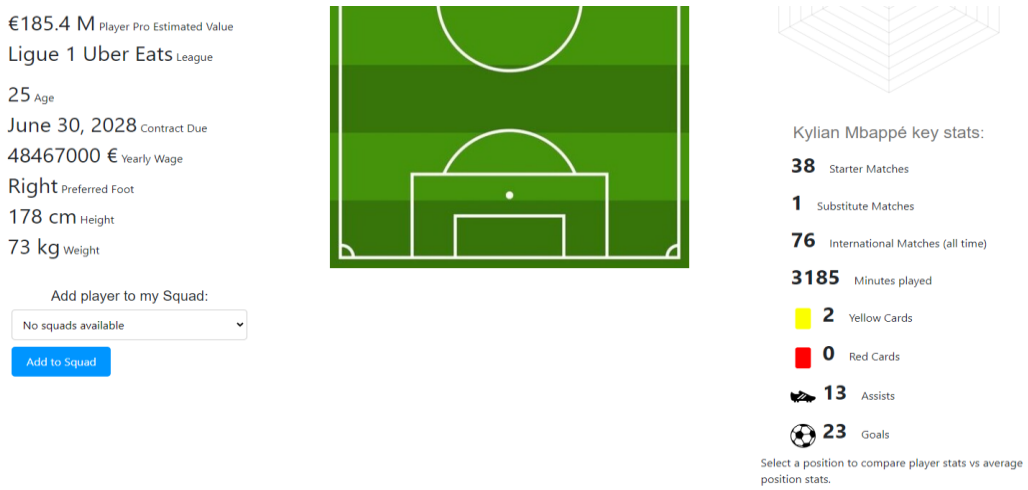


Figure B.15 – Player Profile Attributes

Figure B.16 – Player Profile Statistics

Creating a Squad

The first time the user clicks on the "My Squads" section of the sidebar, under "Library", the page will show up empty, letting the user create new squads.

After adding players to the created squads, the user is able to edit the created squads (changing their name or removing players from them) or removing squads from their library.

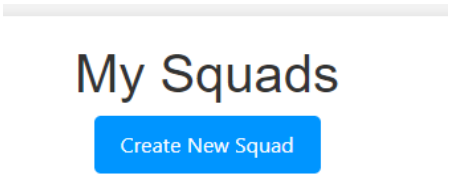


Figure B.17 – Create Squad

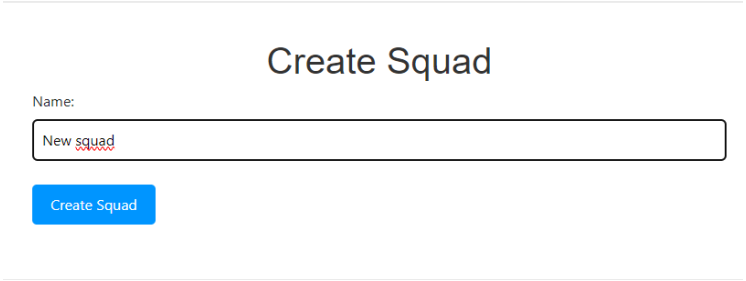


Figure B.18 – Squad Name

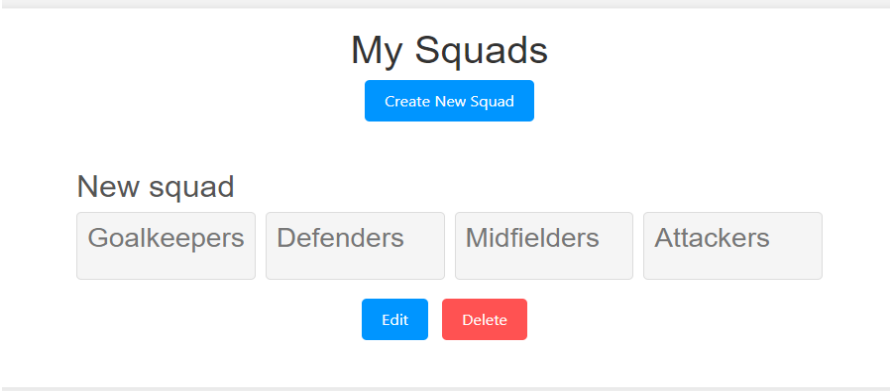


Figure B.19 – My Squads

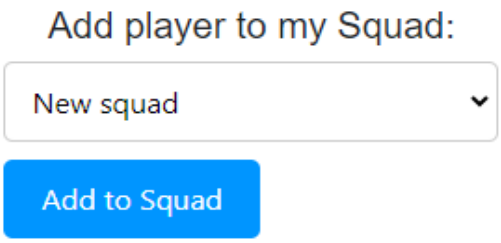


Figure B.20 – Add to Squad

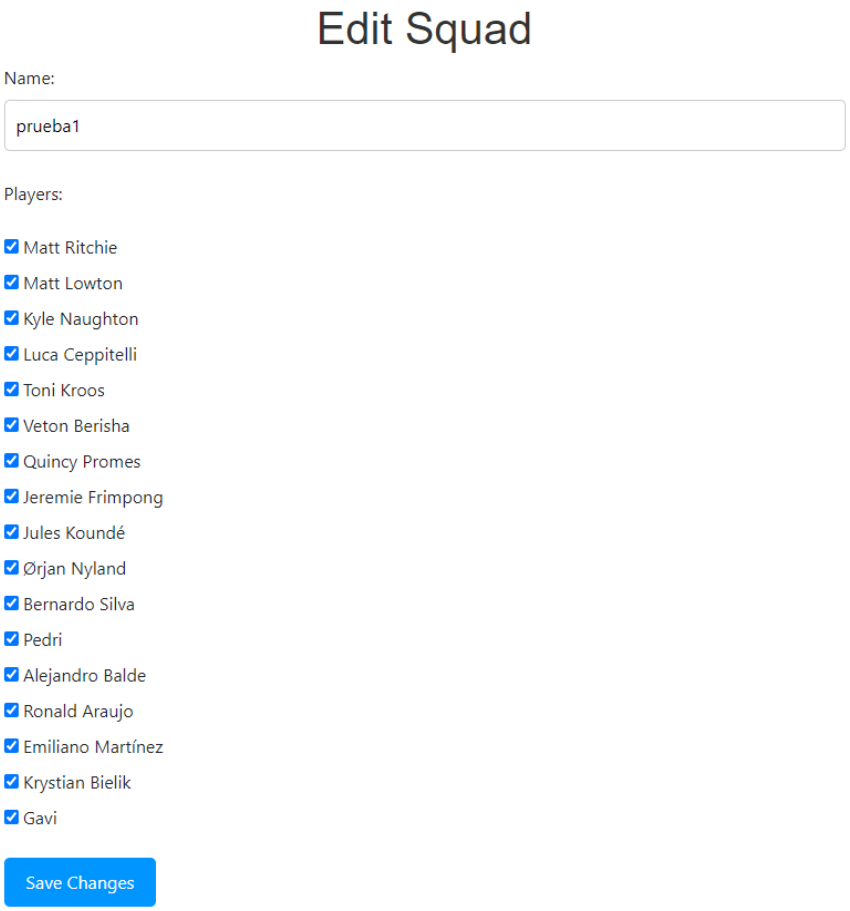


Figure B.21 – Edit Squad

Squad Builder

The Squad Builder tool, found in the sidebar, requires that the user has created a squad prior to the access to this page. Here, the user will be able to select their squad of choice.

Once you have selected which one the user wants to work with, they are able to add players by selecting a position and choosing any of the players that are able to play there. They can be removed by clicking on the "x" on the top right corner of their image.

Once 11 players are selected on the field, the squad can be analyzed by clicking the corresponding button. This will prompt the system to compare each of the player's stats on the selected position against the average values, providing an insight on the performance of every selected player.

If a player has below-average statistics, it will let the user find a replacement player, by displaying a list of players that will improve the squad, as well as detailed visualization tools that help illustrate the comparison.

Whenever the user feels that they have found the perfect improvement, they can click the "Replace" button, confirm the action and automatically the improved player will replace the previous option.

Figure B.22 – Squad Builder Select Squad

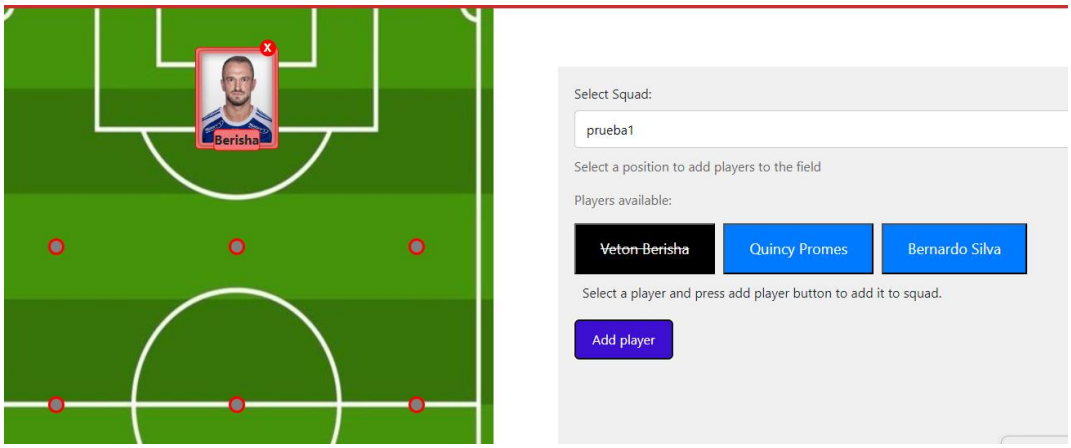


Figure B.23 - Squad Builder Add Players

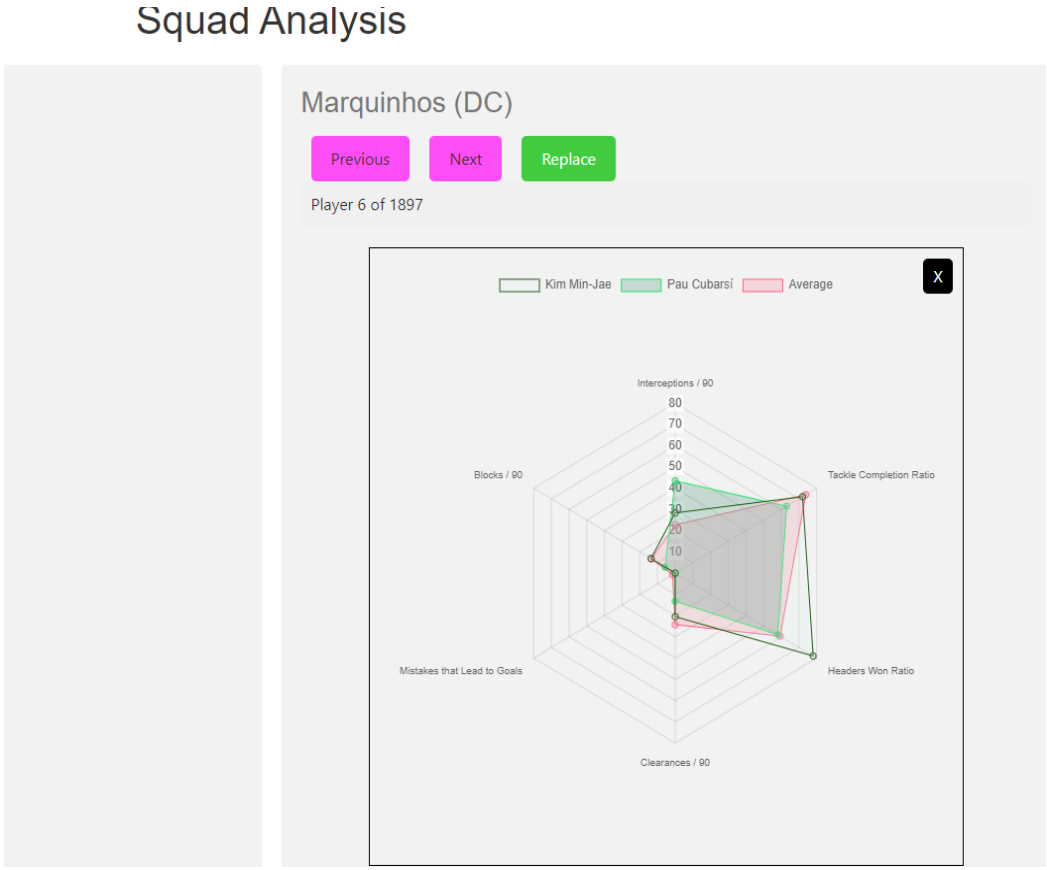


Figure B.24 - Squad Builder Replace Player

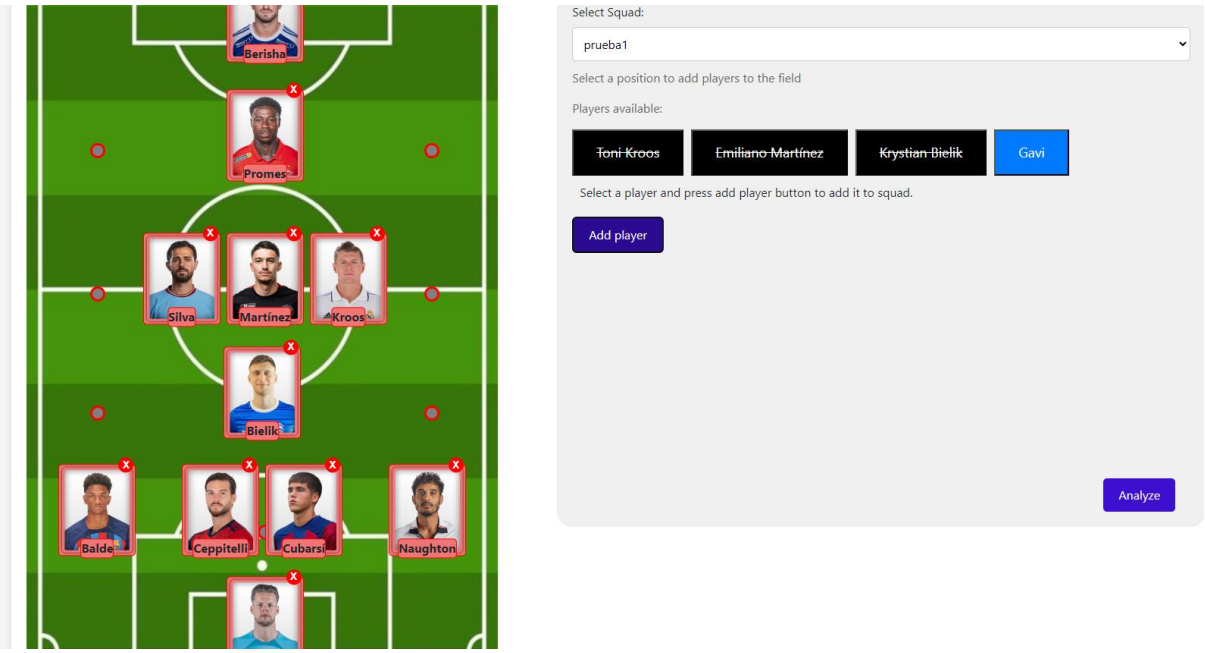


Figure B.25 - Squad Builder Players

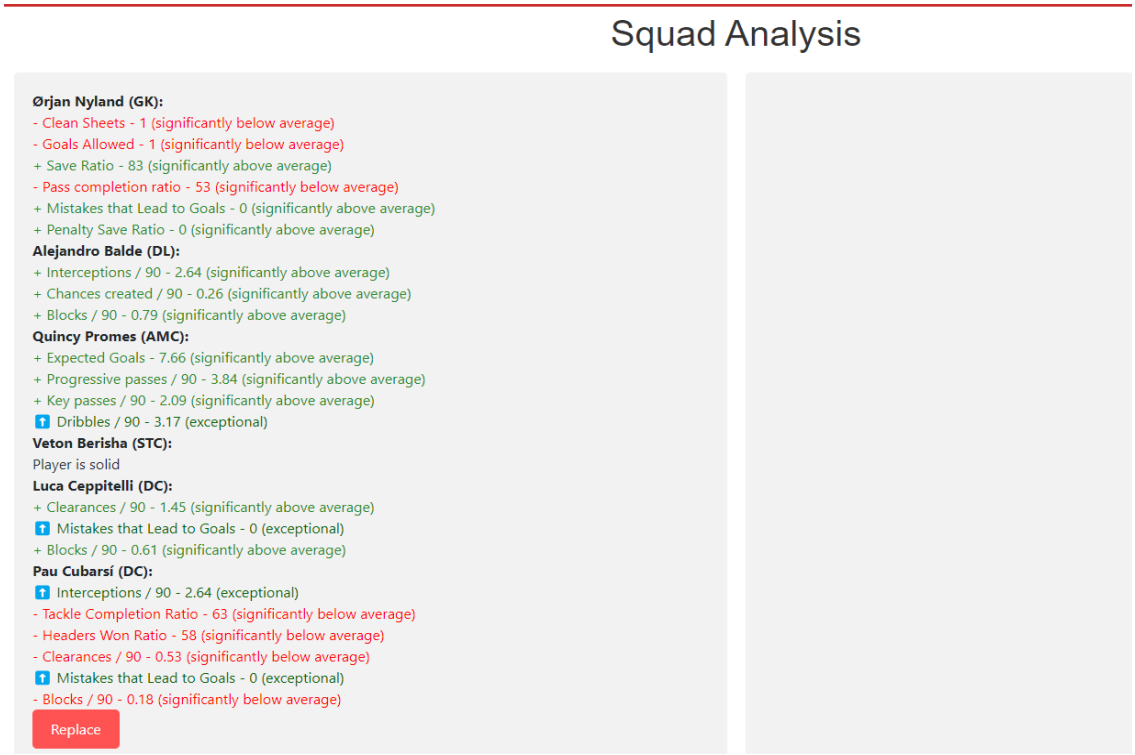


Figure B.26 - Squad Builder Analysis

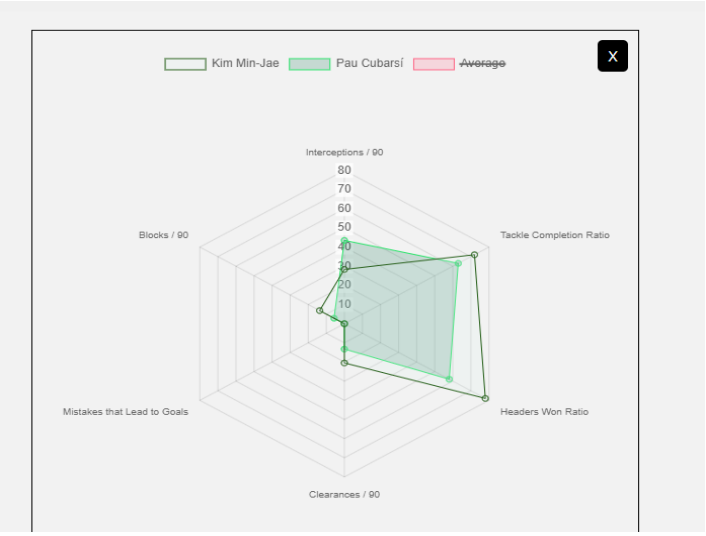


Figure B.27 – Squad Builder Comparison

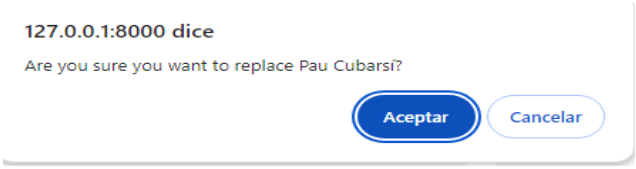


Figure B.28 – Squad Builder Replace Player

Figure B.29 - Squad Builder Replaced Player

FutureScope

The FutureScope settings' objective is to tweak the SmartScore associated to each player according to several adjustable parameters that aim to cover a wide range of users' circumstances.

This tool allows the user to set a transfer budget, a football league, and their competitive expectations, which will affect how the SmartScore algorithm evaluates the different characteristics of players, to cater to users' needs. Once set, they can be modified by clicking the "Edit" button.

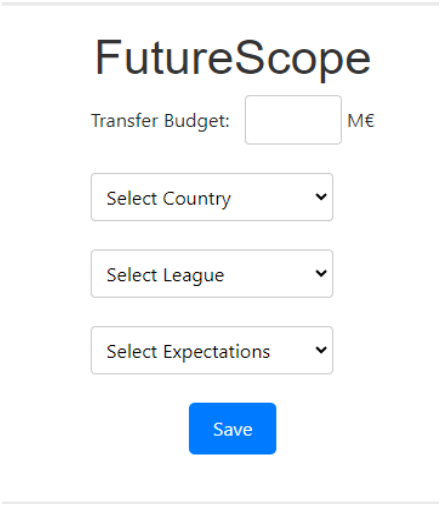


Figure B.30 – FutureScope Settings

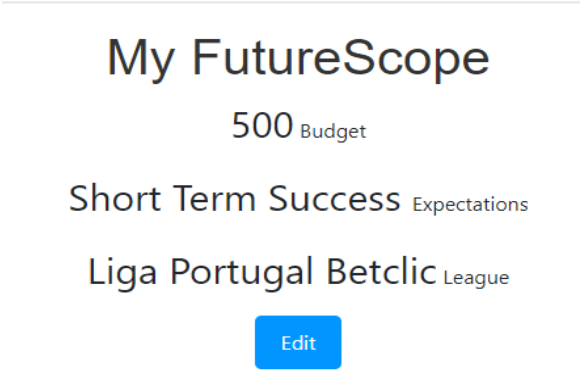


Figure B.31 – My FutureScope Settings

Recommended Signings

To obtain player selections recommendations, the user can select what type of player they are looking for (profile and archetype), as well as their preferred foot. Once selected, by clicking the "Generate Recommendations" button and briefly waiting, up to 30 suggested players will be displayed.

The user can navigate the list and click on any player's image to navigate to their profile or save them, which will create a shortlist with the player or players the user has saved.

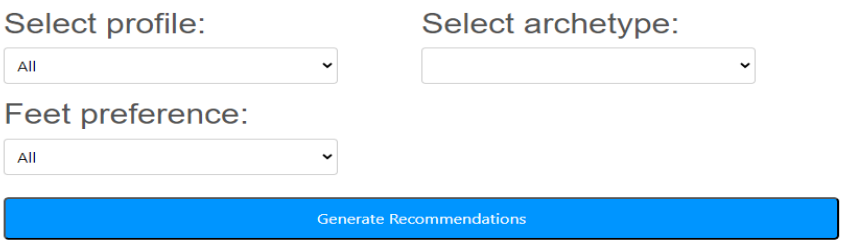


Figure B.32 – Recommended Signings Filters

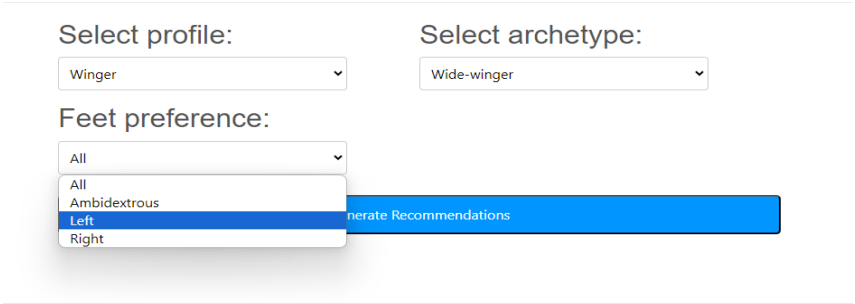


Figure B.33 – Recommended Signings Select Filters

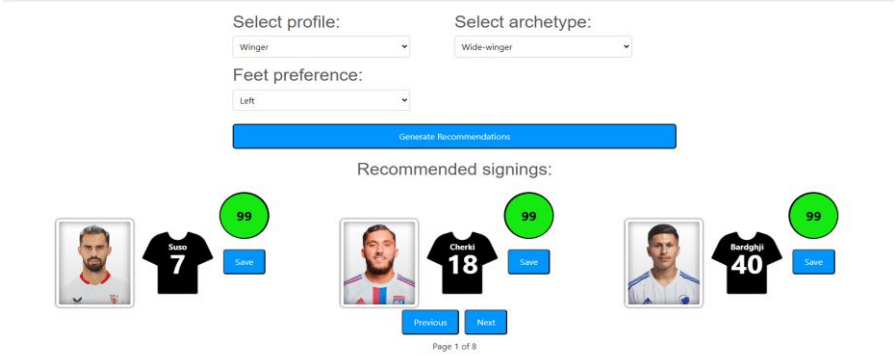


Figure B.34 - Recommended Signings Players

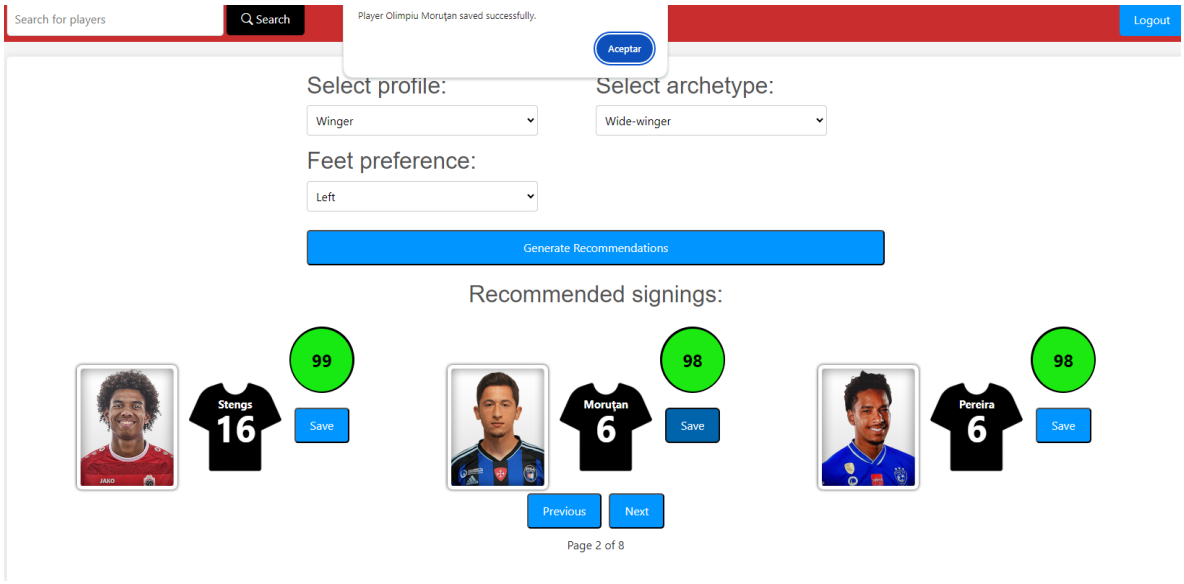


Figure B.35 - Recommended Signings Save

Shortlists

Users can keep track of players they were recommended and have saved in the shortlist page (found under "Library" in the sidebar). For each of the shortlists, the user can see player's wages, market values and the expiration date of their contracts, as well as navigating to their profile (by clicking view) or removing them from the shortlist.

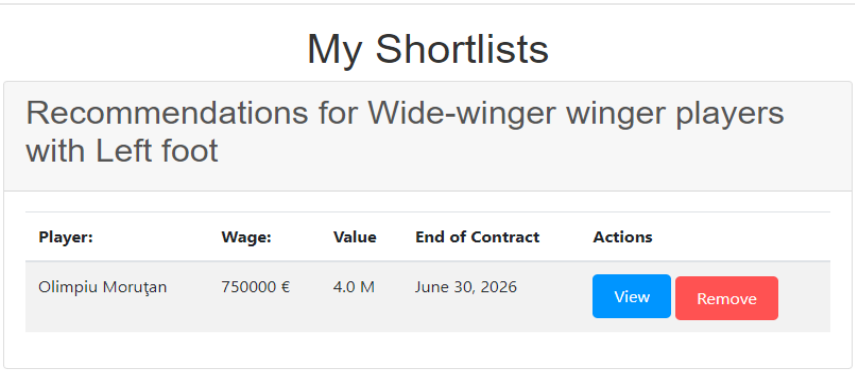


Figure B.36 – My Shortlists

Glossary

The purpose of this section is to help the user understand each of the statistics and attributes of the players.

- **Name:** player's name.
- **Nationality:** player's represented country.
- **International_match:** number of official international matches played.
- **Club:** current players' club.
- **League:** the league the player plays in.
- **Pos:** list of positions the player can play at.
- **Pref_foot:** preferred foot.
- **Age:** player's age.
- **Height:** player's height in cm.
- **Weigh:** player's weight in kg.
- **Salary:** yearly salary in Euros.
- **End_contract:** date their current contract expires.
- **Strater match:** number of games played as a starter.
- **Res_match:** number of games player as a bench player.
- **Min:** number of minutes played.
- **Goal:** number of goals.
- **Asis:** number of assists.
- **xG:** expected goals.
- **Gol_90:** goals per 90 minutes.
- **Asis_90:** assists per 90 minutes.
- **Goal_allowed:** goals allowed as a goalkeeper.
- **Clean_sheet:** number of clean sheets.
- **Sv_rat:** save rate over 100.
- **xSv_rat:** expected save rate over 100.
- **Pen_saved_rat:** penalty saved rate over 100.
- **Faga:** fouls against.
- **Fcomm:** fouls made.
- **Yel:** yellow cards received.
- **Red:** red cards received.
- **Dist_90:** distance covered per 90 minutes.
- **Key_tck_90:** key tackles per 90 minutes.
- **Key_hdr_90:** key headers per 90 minutes.
- **Blocks_90:** blocks per 90 minutes.
- **Clr_90:** clearances per 90 minutes.
- **Int_90:** interceptions per 90 minutes.
- **Hdr_rat:** headers won rate over 100.

- **Tackles_rat:** tackles won rate over 100.
- **GI_mistake:** mistakes leading to goals.
- **Pass_rat:** passes completed rate over 100.
- **Pr_pass_90:** progressive passes rate over 100.
- **Key_pass_90:** key passes per 90 minutes.
- **Cr_c_90:** crosses created per 90 minutes.
- **Cr_c_acc:** crosses accuracy over 100.
- **Ch_c_90:** chances created per 90 minutes.
- **Drb_90:** dribbles per 90 minutes.
- **Poss_lost_90:** possession lost per 90 minutes.
- **Shot_rat:** shot rate over 100.
- **Conv_rat:** conversion rate over 100.
- **Dorsal:** player's shirt number.
- **Country_league:** country of the player's league.
- **market_value:** player's estimated market value.