Introduction

Our project aims to create a system for accessing and analyzing football data (datasets from Transfermarkt), aimed at both non-expert fans and industry experts.
As required we divided data into two sections:

1. **Dynamic data** with a reasonably fast change rate that are stored in MongoDB collections and are: appearances, games, clubgames, gameevents, gamelineups and users.
2. **Static data** which change few times that are stored in PostgreSQL tables and are: clubs, players, competitions and player valuations

Our project works with a **constellation of servers**, there is a **main server** implemented in Javascript (Express.js) which simply interacts with the other two servers that are the **Express server**, implemented in Javascript (Express.js), responsible for retrieving data from MongoDB and the **Spring Server**, implemented in Java with Spring Boot framework, responsible for retrieving data from PostgreSQL

Task: realize user access

Solution:

The login page is a single page that allows users to log in or register. We decided to store user data in a MongoDB database because it handles dynamic data well. The main server extracts the data and sends it to the Express server, which is responsible for registering users or verifying credentials. The credentials are saved in the database with hashed passwords using bcrypt library.

Issue:

One problem we encountered was with password encryption. We resolved this by using bcrypt library, a library that provides functions to encrypt and decrypt strings.

Requirements:

Our solution uses bcrypt library for password encryption, fulfilling the requirement of securely storing user passwords in the database. Additionally, we have implemented session storage to manage user sessions, ensuring that user authentication and session management are handled effectively. This implementation covers the specified requirements, but didn’t manage the security of sending credentials between the client and server.

Limitations:

Our current implementation has a few limitations. Firstly, there is no mechanism for users to recover forgotten passwords, which requires administrative intervention to reset them. Secondly, the system uses HTTP for communication between the client and server, meaning that credentials are transmitted in plain text and are vulnerable to interception and man-in-the-middle attacks.

Task: realize chat page

Solution:

The chat page design is structured around the concept of group conversations based on competitions. Each competition has a dedicated chat room, allowing users to choose and

participate in discussions relevant to the specific competition they are interested in. This design is implemented using Socket.IO, which facilitates real-time, bidirectional communication between clients and servers. To be able to chat you need to be logged.

Issue:
One of the major challenges was ensuring message persistence within the current session when a user switches between different chat rooms. Specifically, when a user changes chat rooms and then returns to a previous one, all the messages from that session need to remain intact and accessible.

Requirements:
Real-time Communication: Achieved using Socket.IO.
Organized Conversations: Met by segregating chat rooms by competition.
User Flexibility: Users can choose their preferred competition to chat about.
Accessibility: User-friendly design with an initial regulation page ensures easy navigation and usage.

Limitations:
Currently, the chat only supports message persistence within the active session. Extending the chat functionality to save messages in a database would allow for persistent message history across sessions, providing users with a more robust and continuous conversation experience.

Task: realize search pages

Solution:
There are four pages dedicated to searching for information:

1. **search competition page:** shows the user the competitions grouped by country and allows searching by name
2. **search club page:** very similar to the previous one but groups the clubs by initial letter
3. **search players page**: by default it shows the most expensive players and allows you to search by name or filter players by nationality, competition and role
4. **search matches paye:** the user is able to see the matches of a competition sorted by day of the week, by default the Serie A matches are loaded

Issue:
For the search pages, the primary issue was finding a solution to load a large amount of data quickly and efficiently. The goal was to enable users to find what they are looking for with minimal wait time, despite the vast quantity of data being processed. Ensuring fast load times and accurate search results was critical to providing a seamless and satisfactory user experience.

Requirements:
We try to realize these pages as flexible as possible, for example we allow the user to filter players by competition, nationality and role but no parameters are required, they are all optional!

Limitations:

We think that our solution has some limits in the specificity of the searches and applicable filters, which could go into much more detail, especially for players, but also for the matches page for which filters could be improved and could be introduced of search system.
Another important improvement would be the introduction of pagination, especially for players

Task: realize competition_page

Solution

The page is entirely dedicated to show statistics, matches, standings of a competition. Our solution tries to be as flexible as possible, in fact the page changes dynamically based on the type of competition, that are **normal league** (for example: Serie A) where all the teams face each other twice and there is no direct elimination phase, **group cup** (for example: uefa champions league), where there is a first phase with different groups and a second direct elimination phase and the **simple cup** (for example: italy cup) that only have a direct elimination phase.

The page shows **generic information** about the competition, obtained from the PostgreSQL competitions table, **ranking of the best players** of the competition by **value**, obtained through the PostgreSQL players table and of the **best players by goals scored** since the 2012 season, obtained through the MongoDB appearances collection.

The page also shows **all matches of the competition,** divided by season and round, and divided by **group stage** and **knockout stage,** and it also shows all the clubs participating in the competition and the rankings, divided into groups if the competition requires them

Issues

Ourmain issue was trying to create the competition page in such a way as to be extremely flexible with respect to the type of competition, we must admit that from this point of view the code could be refactored and improved. Furthermore, the section on the knockout phase also gave us some problems and the logic with which the round order is chosen could be improved

Requirements

We tried to create a solution on the competition page in order to satisfy a large portion of people, from the **most passionate and experienced** who, for example, are able to immediately view the best players in the competition or view the different types of rankings, but even towards the **less passionate** who can quickly see the results of the matches, the teams competing and other basic information.

Limitations

We believe that our solution **can also be extended for other types of competitions** that were not present in the starting dataset, for example national competitions such as the World Cup or European Championships, but a limitation could be the way in which we determine whether a competition has the phase in groups.

Task: realize match page

Solution

The match page aims to clearly show the result, scorers, players on the pitch and other significant information of any match. The page shows **basic match information** like result, home and away team and date that are obtained via MongoDB's games collection, shows all **events of the match** (goals, yellows, substitutes…) divided between the home team and the away team, obtained via MongoDB's game events collection, shows the **lineups** of the home team and away team, where the starting players grouped by position and players on the bench are displayed, obtained via MongoDB's game lineups collection and shows the **"head to head"**, i.e the number of victories for the home team, draws and victories for the away team.

Issue

The biggest challenge we have faced are:

1. Make the events section **flexible to possible changes in the type of events**, for example trying to capture the differences between a goal scored normally and a penalty kick, the logic used to choose between the various types could be improved
2. **Create the section dedicated to lineups**, in fact we chose to show the lineups only for the games of the 2023 season, since the data on the lineups are taken through the MongoDB game lineups collection which unfortunately does not maintain data of all past games

Requirements

Our solution was designed to be easily understood by all people (experts and non-experts) interested in football, we thought that **experts** will be more attracted by the graphics in the "head to head" section and by the events of a match and **non expert** will focus on understand who played the match, be clear about the result, which team won and other basic information.

Limitations

We think that our current solution has some limitations, including the logic of assigning the type to an event in the match and the section dedicated to the formations that could be made even more interactive by showing information on the players in the match.

Task: realize data analysis:

Solution:

We analyzed the provided datasets by importing them into Jupyter Notebooks and using Python libraries such as pandas, matplotlib, and seaborn. We then conducted statistical analyses to identify patterns and trends, and integrated the various datasets to generate comprehensive insights. Finally, we created visualizations, including charts and graphs, to effectively communicate our findings.

Issue:

The main issue was ensuring the data was clean and consistent across all datasets to allow for accurate analysis. Additionally, integrating multiple datasets to extract meaningful insights was challenging due to potential discrepancies in data formats and missing values.

Requirements:

Cleaning and preprocessing of data.Merging and integrating datasets where necessary.Performing statistical analysis and visualization.Interpreting and reporting findings

Limitations:

Incomplete or missing data may have affected the accuracy of the analysis.The quality of insights depended on the accuracy and comprehensiveness of the provided data.

Conclusion

In conclusion we think we did a good job and that we applied what we learned in class, trying to create correct non-blocking communication between servers using JSON, apply asynchronous programming correctly, create a chat with the techniques seen in class…
We think that our solution is not perfect but can easily be improved and we think that new features can be introduced such as better user management or a better information filtering system.

Division of works

Accornero: realization of competition_page, squad_page, searchCompetition_page a bit of player_page and and some work in the search filters,MainServer
Aiello:documentation (swagger + jsDocs e javaDocs),chat,login/register,searchClub_page, player_page,ExpressServer
Bonincontro:documentation (swagger + jsDocs e javaDocs), landingPage, player_page, matchesPagesearchPlayers_page, schema database,queryexamples,modular structure for routing,Division of folders,JavaServer,Error404_page

Extra information

Adding the following indexes into your MongoDB collections is strongly recommended:

1. In **appearances** collection on player_id, game_id and competition_id fields
2. in **clubgames** collection on game_id
3. in **gameevents** collection on game_id
4. in **games** collection on game_id, competition_id and season
5. in **gamelineups** on game_id

Without indexes, site performance degrades, especially when executing queries that include multiple collections

Bibliography

https://docs.spring.io/spring-data/jpa/reference/jpa/specifications.html we have used JpaSpecificationExecutor to realize player filter query with nullable parameters (competition,nation and role)
https://chatgpt.com/ generative AI that helped us in brainstorming ideas, solve some specific programming tasks and also generates the logo (logo.png) and the manager image (managerLogo.webp)
https://tmssl.akamaized.net/images/logo/header/{competitionId}.png we have used this link to obtain competition logos basing on competition id in our datasets
https://tmssl.akamaized.net/images/wappen/head/{clubId}.png we have used this link to obtain club logos basing on club id in our datasets
https://boxicons.com/, https://icons8.it/ for most of the icons on the site
https://www.locofy.ai/ a generative AI that helped us convert from figma prototype to HTML+CSS