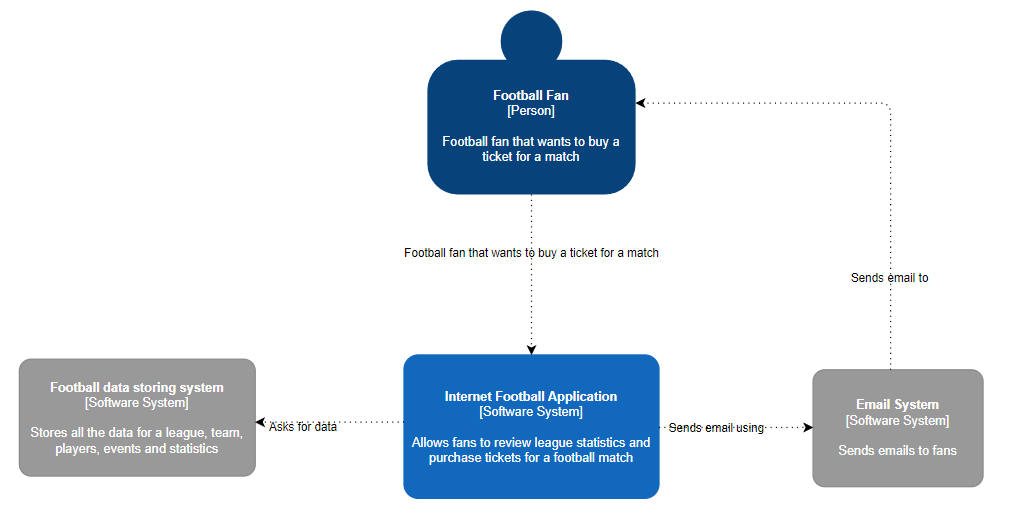
Design document

Architecture constraints and design decisions

Why did I choose the following technologies for my project?

* Java: it is an OOP and a statically typed language which makes the code more predictable and maintainable like C#. Unlike C#, Java is more independent on which platform it can run. It can run on any OS that has a JVM while C# is a Microsoft service which makes it harder to run on other OS.
* Spring Boot: it manages a lot of small configurations for me, so that I don’t waste time. Also, Spring Boot allows you to package your application and this way it simplifies deployment.
* React: It is based on component architecture which I personally like because it lets me break the code into smaller pieces and if a problem happens I will be able to identify where it is and fix it easier and faster. Another thing I like about react is that it lets you write in JSX, which is a syntax that is similar to HTML
* MySQL: this DB has a big community which will help me find a solution to a problem that I might face easier. MySQL is also one of the fastest and efficient DBs.
* Rest API: it is not dependent on a single programming language which makes it more flexible. I am going to use Java and JavaScript so I don’t have to worry about compatibility later on. The architecture of REST APIs allows stateless data transfer which helps save memory.

I am always focusing on improving my code by making it scalable and consume as less memory as possible.



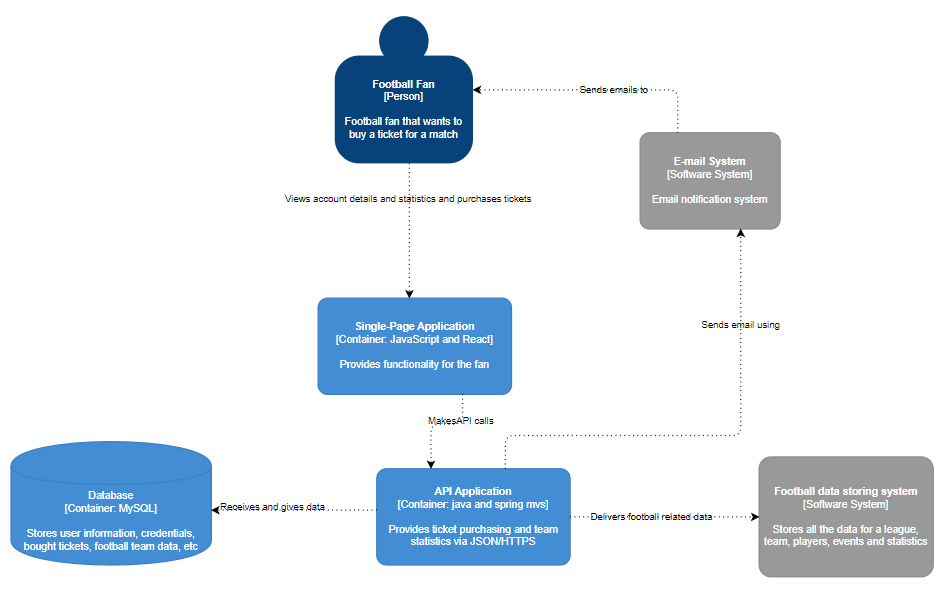
Context diagram - C1

This diagram represents a system that football fans can use to purchase tickets to attend football matches.

The light blue box in the middle represents the application which the fan is going to use.

The grey box on the left represents an external entity which will provide data for the matches.

The grey box on the right represents an external entity that is responsible for checking and sending email notifications to the fan

Container diagram – C2

This diagram represents a zoomed in version of the application shown on the diagram above.

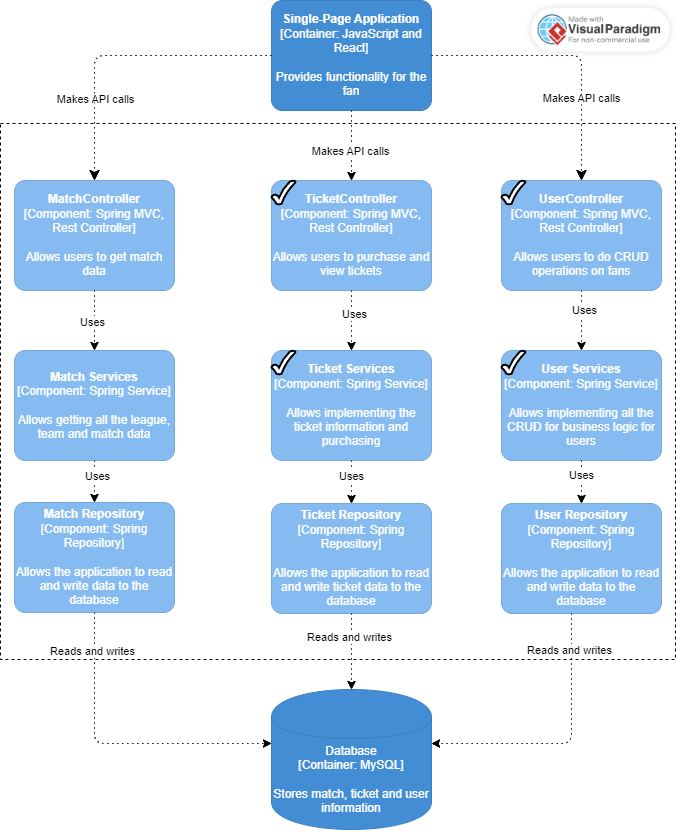
Single-Page Application represents the website that the user can see. The purpose of it is to represent data in a pleasant way to the user.

API Application represents the server side of the application that is responsible for manipulating data, getting a request and giving a response to the Single-Page Application and also saving data for future use in a safe way.

Database represents a storage unit that contains all kinds of information needed from the application in order for it to function.

Football data storing system represents a third-party service that stores information that is common. This way the application and the people responsible for it save time and energy to implement that same system manually.

E-mail system represents another third-party service that is responsible for checking if the user email address is valid and sends email notifications to them.



Component diagram – C3

This diagram zooms more into what the Single-Page Application contains for it to achieve results.

It contains 3 components for each entity. For now, all of the entities have the same type of components that are responsible for a task.

1. Controller component – handles requests from the front-end. It is the middle-man between the front-end and the business logic. They are also responsible for delivering the response to the front-end.
2. Service component – contains the business logic of the application. They are responsible for data validation and manipulation and following business rules.
3. Repository component - it is responsible for retrieving, deleting and re/writing data to a database.

By following this design we assure that every component has a Single Responsibility and the code is kept simple and maintainable.

The ticks in the image show which components were finished based on the current requirements.