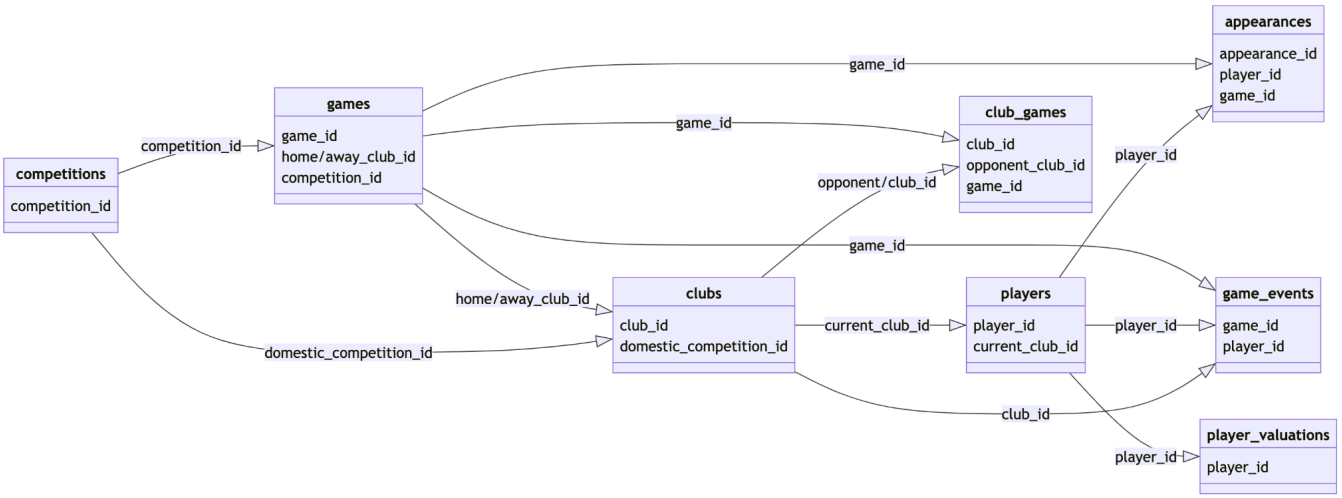
# Assignment IUM-TWEB for 2023-2024

## 1. Introduction

This assignment is primarily concerned with applying the ideas that are being presented in the module on methods for accessing the Web and making sense of its content. In providing a solution, you are required to use the methods and techniques taught in the module.
You are given a set of Football Data from Transfermarkt. This is the data schema:




The data is provided as a set of CSV files.
You are requested to create a system able to support access and analysis by fans (non experts) and pundits (experts).

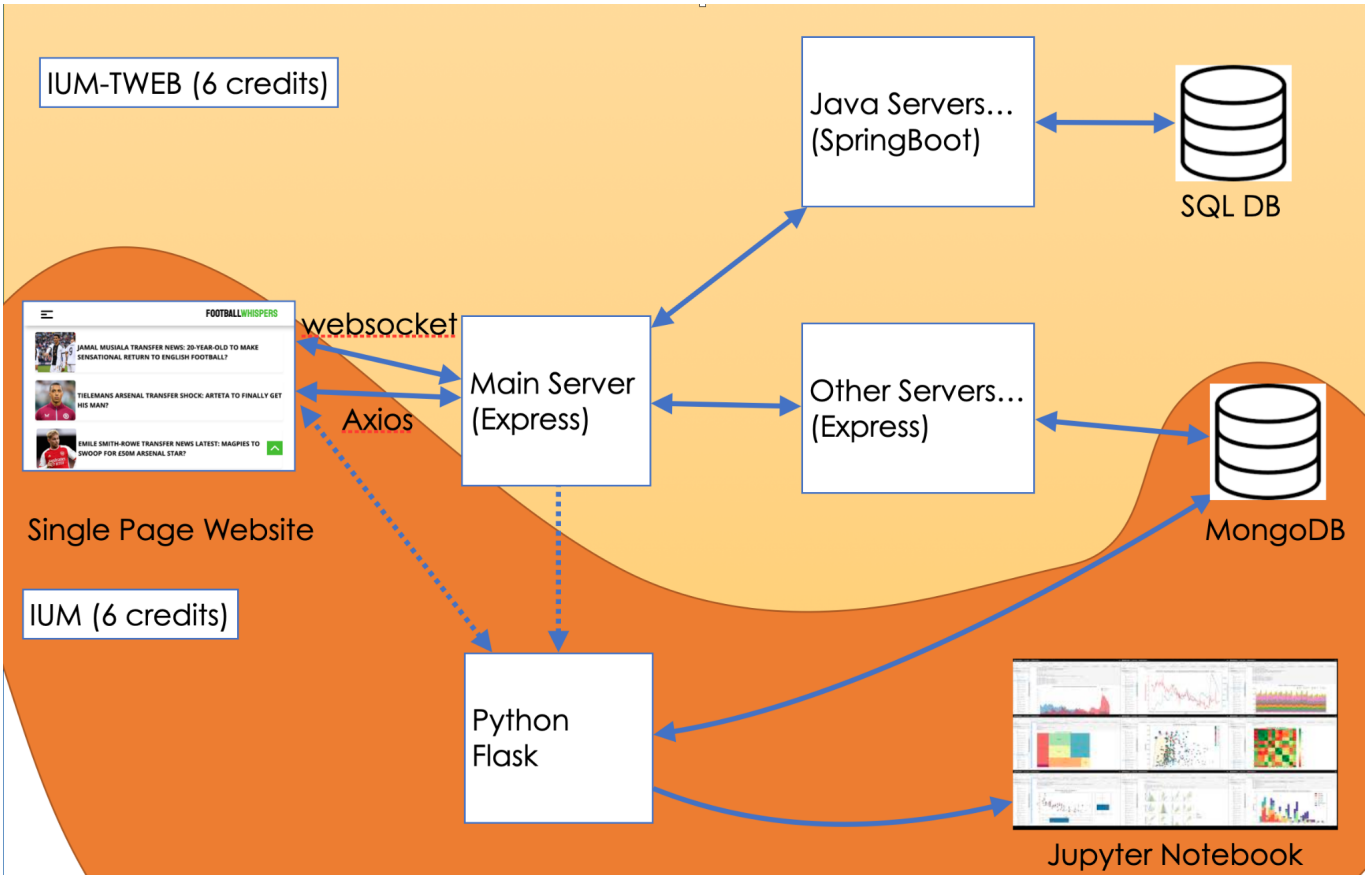
## 2. Requirements

Here is the outline of the required architecture. Note that according to the module you are taking (IUM-TWEB 12 credits, IUM-TWEB 6 credits or IUM 6 credits) a different architecture is required.
The architecture is composed of a number of servers, some of them will be written in Javascript (Express), some of them in Java Spring Boot and some of them in Python (Flask). They interact with two types of databases: MongoDB and Postgres where the Transfermarkt data is to be stored.

**Functionalities**:
The goal of the work is to provide the possibility for fans and pundits to access data about football. You must design  service with the following features:

- An HTML+Javascript+CSS page enables querying and exploring the data.
- The servers:
    - IUM-TWEB (6 and 12 credits):
        - Queries are sent to a central server implemented in Express which will communicate to other servers for database access, at least one written in Express and one in Java Spring Boot. The central server must be fast and be able to serve thousands of users, so it must not perform any heavy duties.

    - IUM (6 credits) and IUM-TWEB (12 credits):
        - Queries are sent to a Python Flask server
- The data will have to be divided into two subsets:
    - the dynamic data with a reasonably fast change rate e.g. the games, scores, goals, etc. This should be stored in a MongoDB.
    - the more static data such as the data about the players, their valuation, the names of the competitions, etc. This should be stored
        - in a PostGres database (IUM-TWEB - 6 and 12 credits)
        - a MongoDB (IUM 6 credits)
- For IUM-TWEB (6 and 12 credits), the solution should also implement a chat system among fans and pundits to discuss topics in specific topic based rooms (e.g. Juventus room, English Premiership room, etc.). This should be implemented using socket.io.




# 3. Large Scale Data Analytics using Jupyter Notebooks

If you are taking IUM (6 credits) or IUM-TWEB (12 credits), you are requested also to develop one or more Jupyter Notebooks that enables analysis of data.
The Web interface should enable the user to query the database in order to identify a subset of the data (e.g. stats about the Italian players, stats about German teams etc.). The specs of the data subset are then sent via a Flask server to a (set of) Jupyter Notebook(s) for analysis. That is, the Jupyter Notebooks are pre-set, i.e. they are designed to perform a specific set of statistics and analysis. But they are parametrised so that, when run, they perform this analysis to the specified subset of data.

# 4. Groups

The assignment is to be done in groups. Experience has demonstrated that it is not really possible for one person to do all the work for the assignment, unless they are an exceptionally competent

programmer with previous knowledge of Web programming. Therefore, the assignment is organised on the basis that people should work in groups. Groups must be composed of

- a maximum of 3 members if you are taking IUM-TWEB (6 credits)
- a maximum of 4 members if you are taking IUM-TWEB (12 credits) or IUM (6 credits)

Students are also allowed to do the assignment in pairs or on their own.

# 5. Material Allowed - Plagiarism

You can freely use the lecture notes, lab classes examples and all materials used during the module. No third party code can be used in the assignment, except what has been explicitly provided in the lectures or lab classes. For example you are allowed to use some code given in the lecture slides but you are not allowed to download any code from the Web or to use any other software that will perform a considerable part of the assignment. Unauthorised re-use of third party software will be considered plagiarism. In case of doubt ask the lecturer for permission before using any third party code. Libraries allowed, despite not being mentioned in the lecture notes, are css/js/html libraries to improve the look and feel of any interface (e.g. Angular, Bootstrap, Vue, React). However note that the use of these libraries must be limited to the look and feel of the interface, i.e. as a replacement of HTML/CSS code. **You must not use any support for server implementation or communication with the Express or Spring Boot servers**. For example you cannot use any functions that would replace an Axios call. For other libraries, please ask the lecturer before using them.

# 6. Marking schema

Each part of the assignment will carry marks divided as follows:

- 25% for the documentation in the project report.
- 50% for the implementation of your solution (quality of code and data analysis). See appendix A for further details on the specific marking schema.
- 25% for the correctness of results.

# 7. Handing in

Your solution must be submitted in a self-contained directory named after the group name (<MainDirectory> in the following).
The directory must contain:

1. The code of the solution (please note that we will both inspect and run the code).
    a. The interface and the communication with the server must be developed in HTML/CSS /Javascript. The server must be developed in Javascript and Java SpringBoot (IUM-TWEB - 6 and 12 credits) and/or Python (IUM - 6 credits - and IUM-TWEB 12 credits). We must be able to run your solution without problems on a standard computer. It should not need any applications or code or libraries to be installed in order to run. Use *npm* or *pip* or *Gradle* to make sure the system knows what libraries to load automatically.
    b. All the code should be in the directory **`<MainDirectory>/solution`**. Please note that the quality of the code carries a relevant portion of marks, so be sure to write it properly.
    c. The Postgres database schemas (e.g. created with a SQL dump) must be stored in **`<MainDirectory>/databaseschemas`**
2. A report documenting your work. The report must be contained in the directory **`<MainDirectory>/report/`**. An outline and set of requirements for the report are provided in Appendix A.
3. The documentation within the code (Javascript/HTML/Java/Python) files must be of very high quality, both in terms of Javadoc (or equivalent) or in terms of Swagger documentation for the

servers. Please note that this documentation carries a relevant portion of marks, so be sure to write it properly. For more information on guidelines for this type of documentation see `http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html`

4. Some screenshots of at least 3 queries' results must be stored in **`<MainDirectory>/queryexamples`**. These are used to check when we run your solution that the results are what you would expect.
5. The filled self assessment form.

## 7.1. Private Github/Gitlab Repo

The solution must be provided via a link to a Gitlab or Github repository. The repository must be used **regularly** during the development of the project by **all members** of the group. The repository will be used during the examination phase to:

- check that the project has actually been developed by the group
- provide clear indications of the contribution of each group member

For this reason, each group member will have to upload (commit) **personally** all the changes to the repository, ideally for each day of work on the project. That is, (i) the group must not delegate the task of making commits to a single member and (ii) commits must not be made only when the code (or a part of the code) is fully completed. We should see the intermediate work, ideally in separate branches.
The name of the group members must be clearly identifiable with name and surname (e.g. Giovanna Rossi instead of xyzaa). Be very careful when you have more than one user e.g. in Github not to submit with multiple anonymous identities.
You are allowed to use several github/gitlab repositories, each for a different part of the project, e.g. one repo for each of the servers, etc.
**Please note: if the group does not provide a satisfactory recording of the commits to demonstrate the actual progress of the project, the project will be rejected and the assignment redone from scratch.**

# 8. Appendix A: Report Outline and Marking Schema

It is important that you produce a high quality report. Be sure not to leave the report at the very last minute. It is important that you do not ignore techniques and examples provided to you during the lectures and lab classes. Referring back to them during your planning/implementation stages will give you the base from where to start, as well as a point of comparison/discussion where your ideas differ from what already presented to you (i.e., do not reinvent the wheel – if it has been done before, reuse it and complement it where it lacks functionality). You may use diagrams to aid your explanation in these sections, but please note that a diagram alone is not acceptable and must be accompanied by an explanation/discussion.
Your report must be organised according to the following outline, which is designed to help you ensure that all the required information is provided.
Length: **maximum 5 pages**. Any additional pages will be ignored.

**Index Table**

[No marks – OPTIONAL] - You may or may not want to include an index table on your report. This is completely up to you. This does not count against the page limit.

**Introduction**

[no marks] – This should give the marker a guide as to what to expect from your report. Please make the marker's work easier by being honest.

**For each technical tasks:**
Create a subsection in your report for each of the technical requirements. It is very important that they are documented individually by explicitly following the organisation below.

- **Solution:**
  - Design and its motivations: explain how your solution works and explain why you chose to design your solution in this particular way. Does it have advantages/disadvantages over other design choices?
- **Issues**
  - Introduce the task this section refers to and the challenges that you were faced with.
- **Requirements**:
  - How does this design comply with the requirements specified in the original assignment sheet? Are you meeting all requirements?
- **Limitations**:
  - Have you thought about exceptional situations that may limit your solution? Is your solution extensible? Can it be easily adapted for other requirements? Remember that no design is flawless and that is ok!
- **Conclusions**
  - [no marks] - You should include here any relevant conclusions you've collectively arrived at regarding the process of designing the solution for this part of the assignment as well as any lessons learned.
- **Division of Work**
  - [No marks – MANDATORY] – Have all the group members shared the workload in a balanced way? In what way? E.g. what has each member provided – be precise!
- **Extra Information**
  - [No marks – MANDATORY] – This section should include any extra details needed to run your code. If no extra configuration is needed, please explicitly say so in this section.
- **Bibliography**
  - [no marks] - Do not forget to cite any sources and reference these within the text where appropriate (e.g., “(...) we have used the techniques as per [1].”).Make sure to use a standard format style. No need to cite the lecture notes or lab classes.