

1Hibernate Exam – 28 July 2019

Football Information

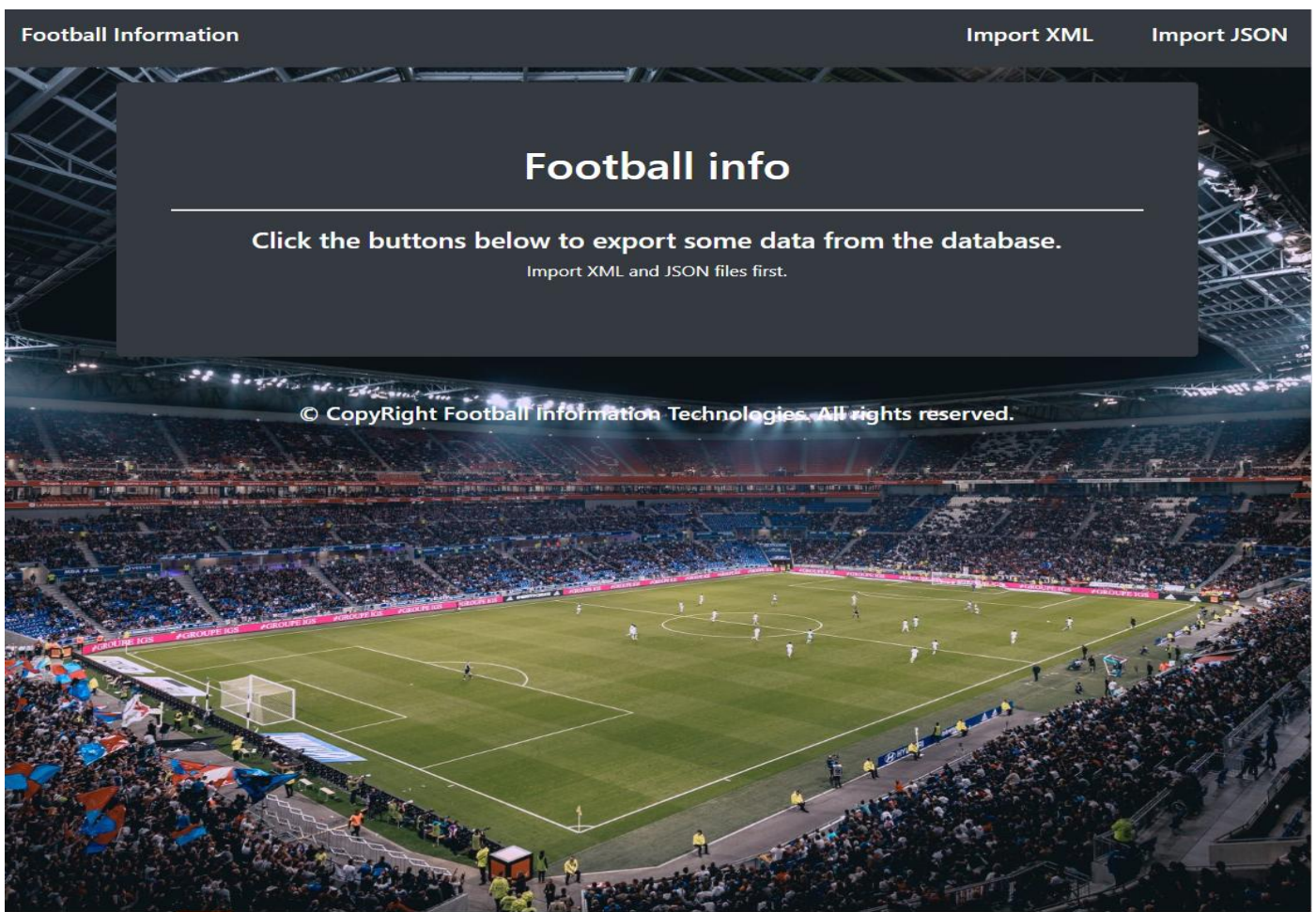
You have been employed by the Next Level Technologies Ltd. to finish the database layer, which supports basic functionality like importing data and exporting some results.

1. Functionality Overview

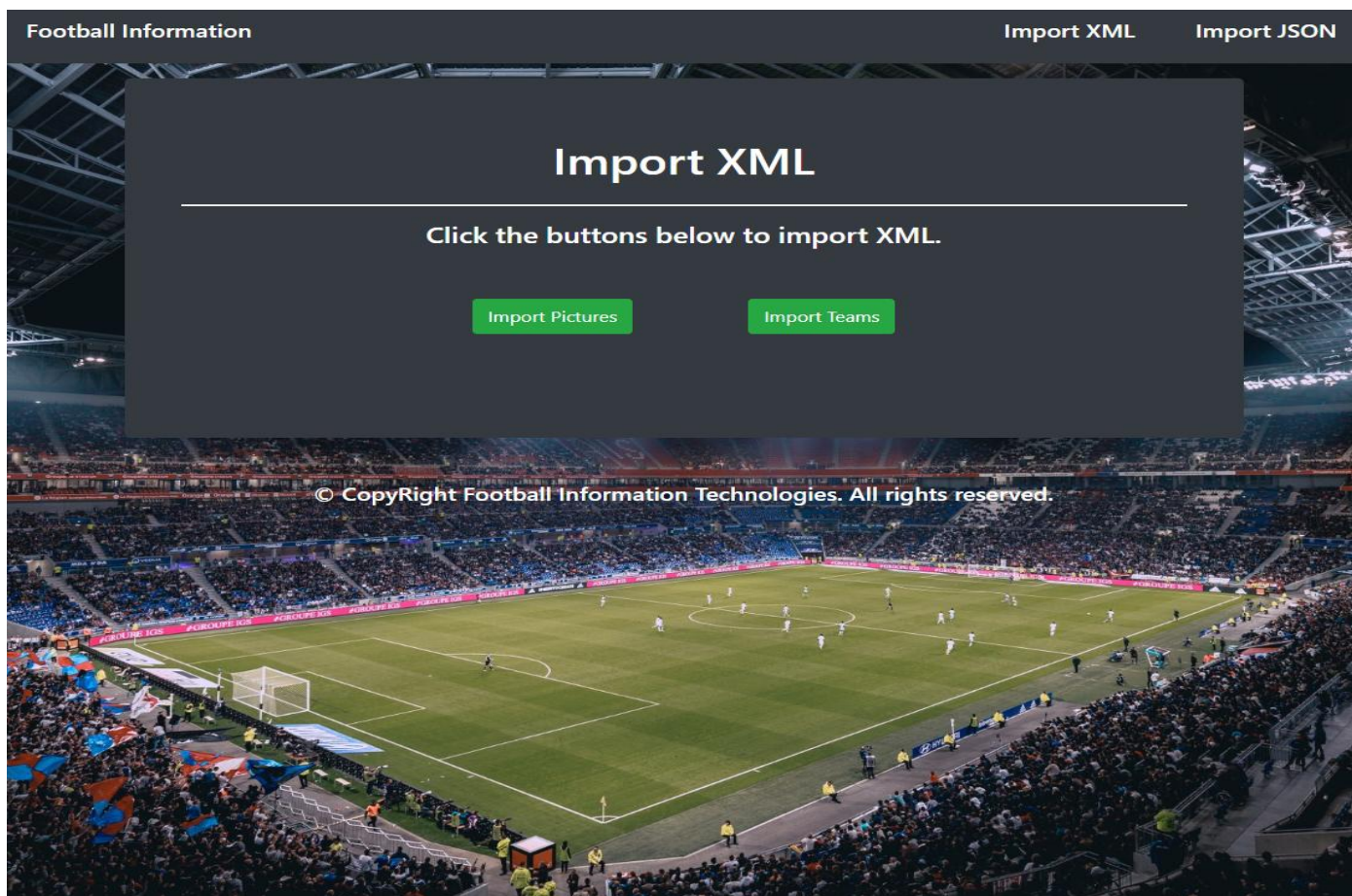
The firm has hired you as their application developer, to implement the **database layer**. The application should be able to easily **import** hard-formatted data and **support functionality** for also **exporting** the imported data. The application is called – **Football info**.

Look at the pictures below to see what must happen:

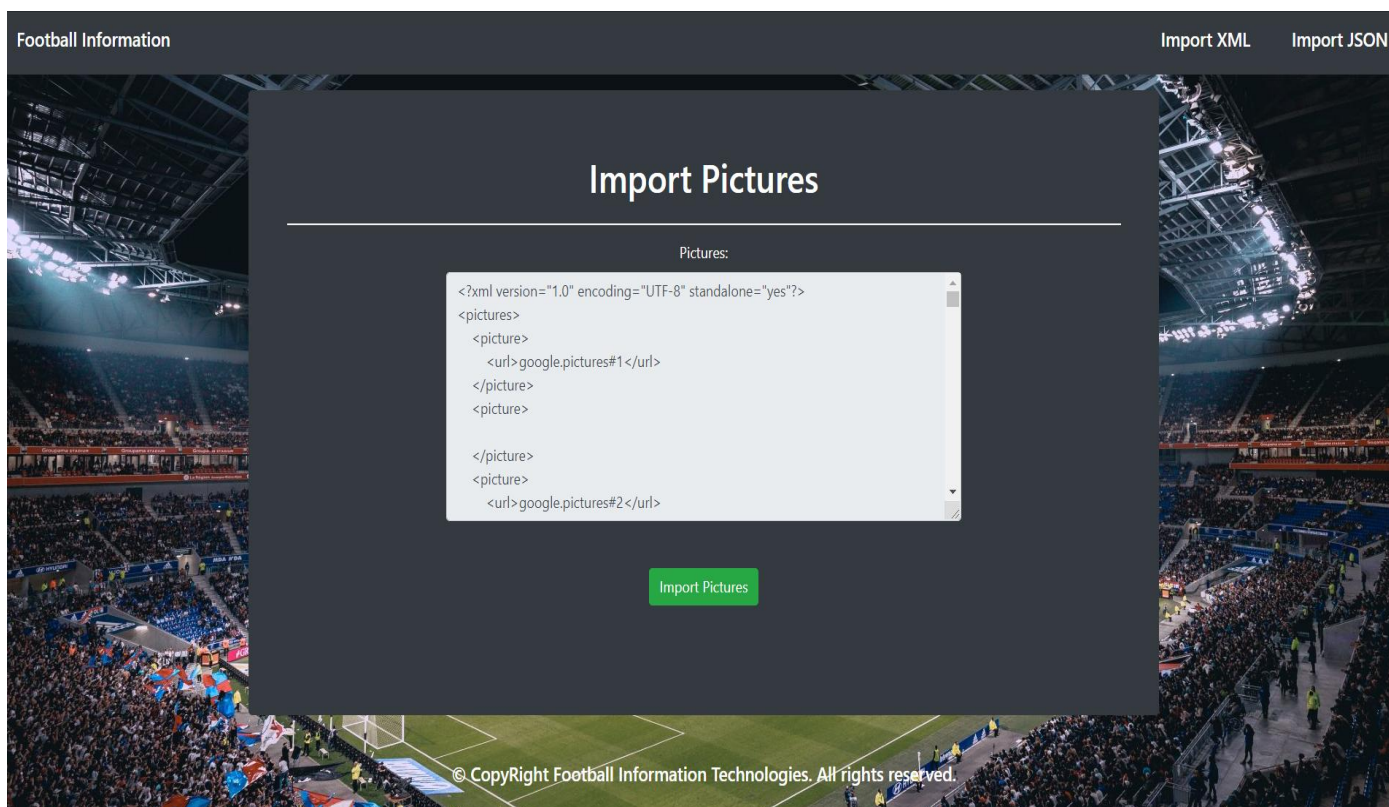
- Home page before importing anything:



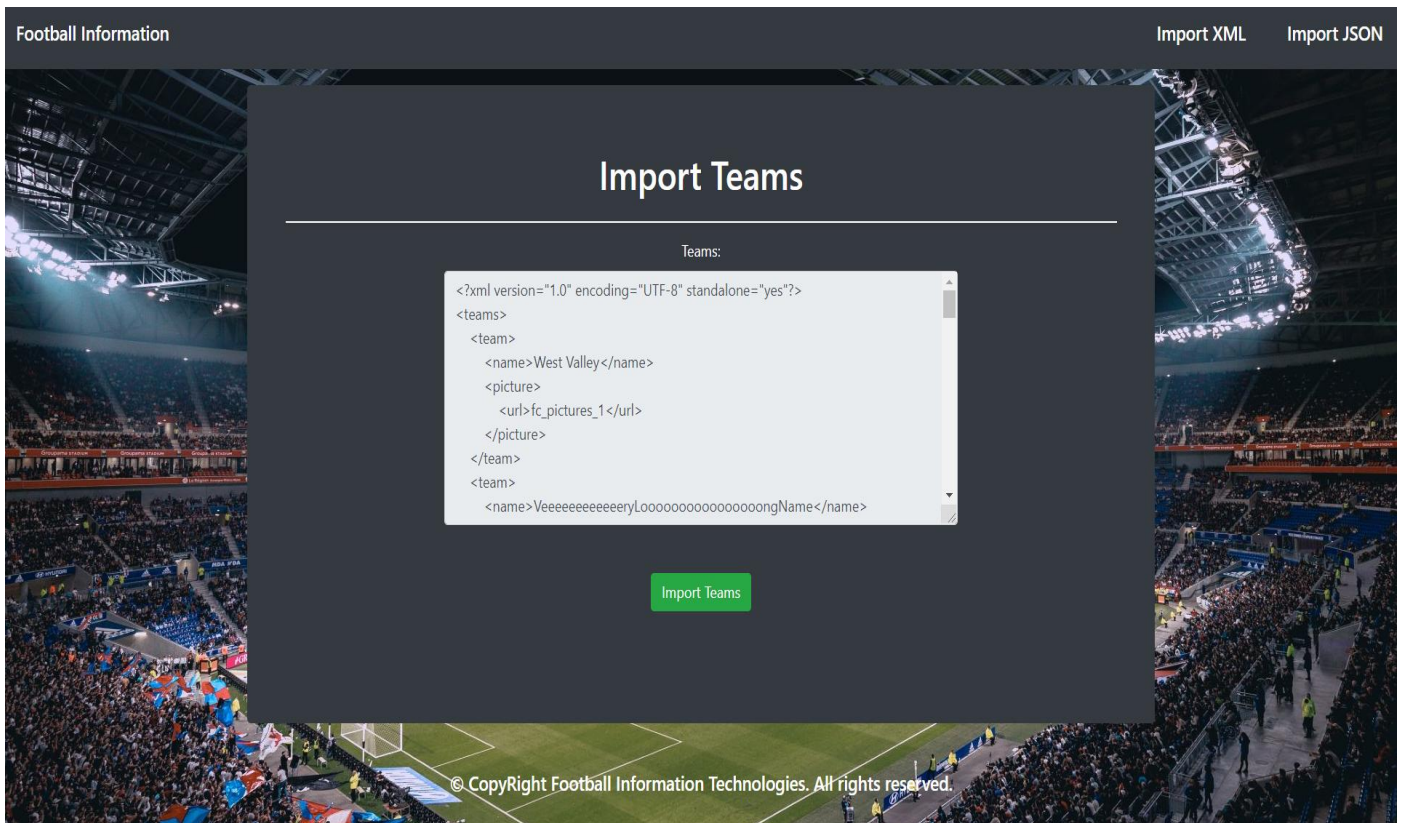
- Import XML page before importing anything:



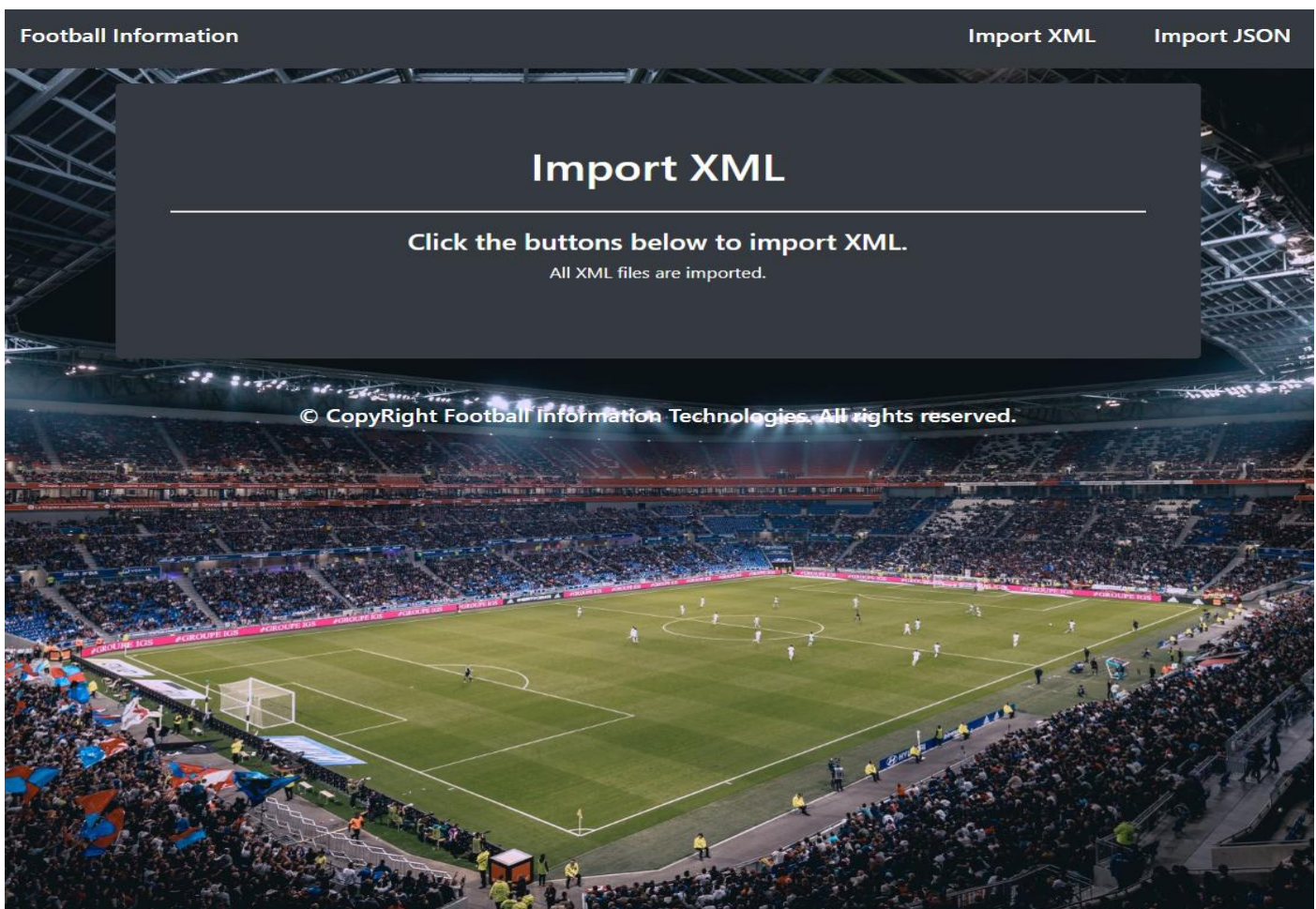
- Import Pictures first:



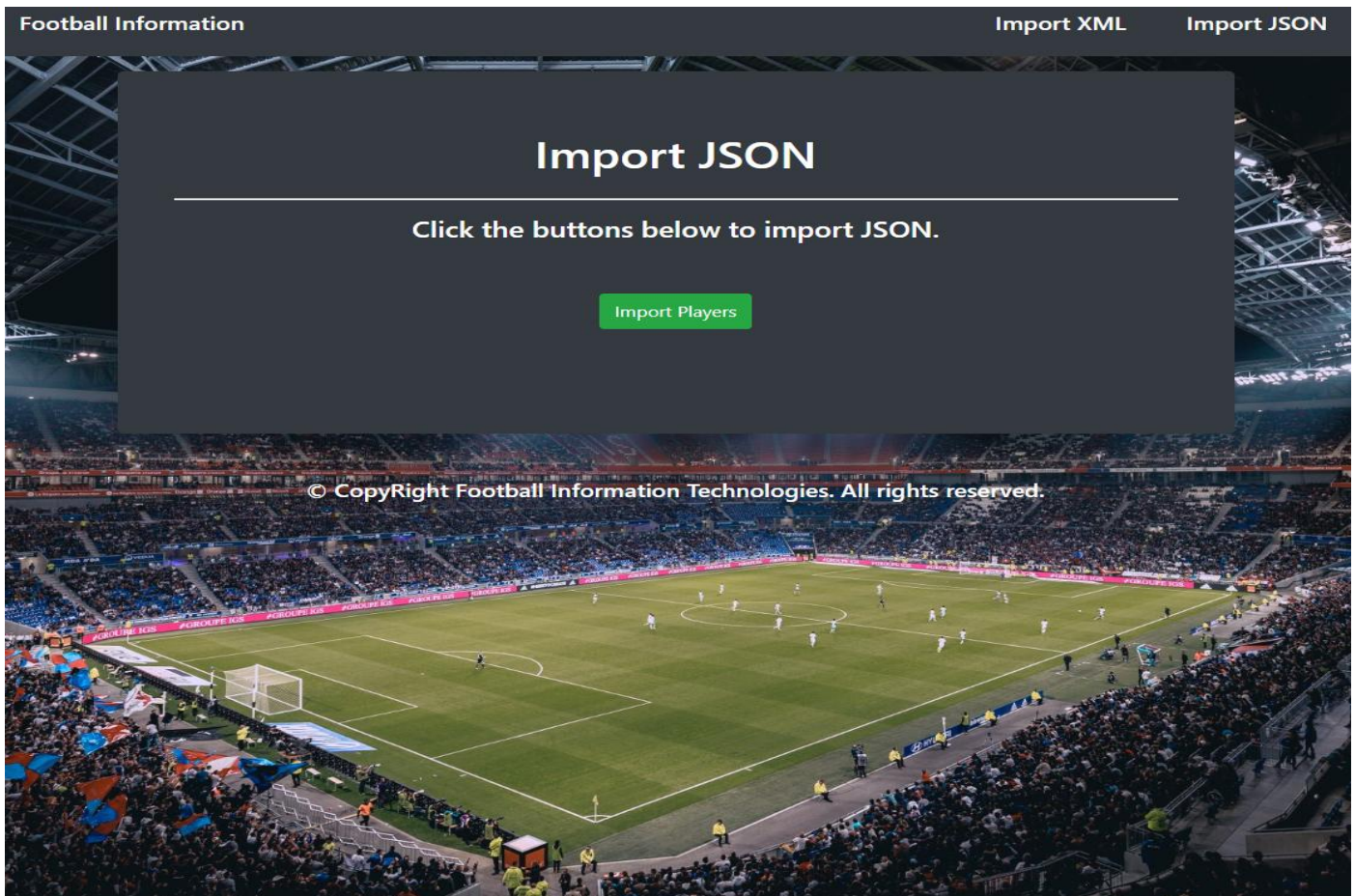
- Import Teams second:



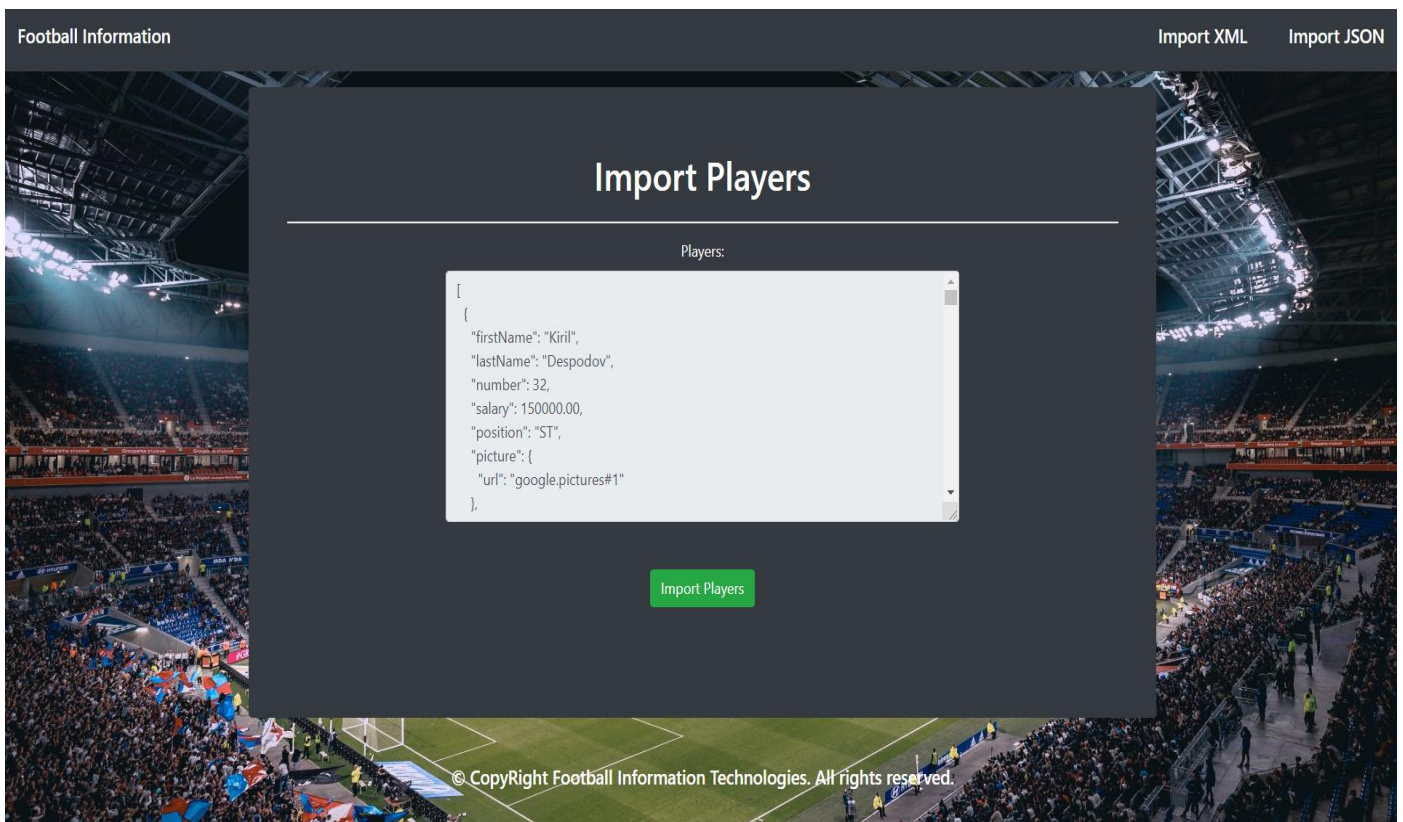
- Import XML page after importing both files:



- Import JSON page before importing the given data:



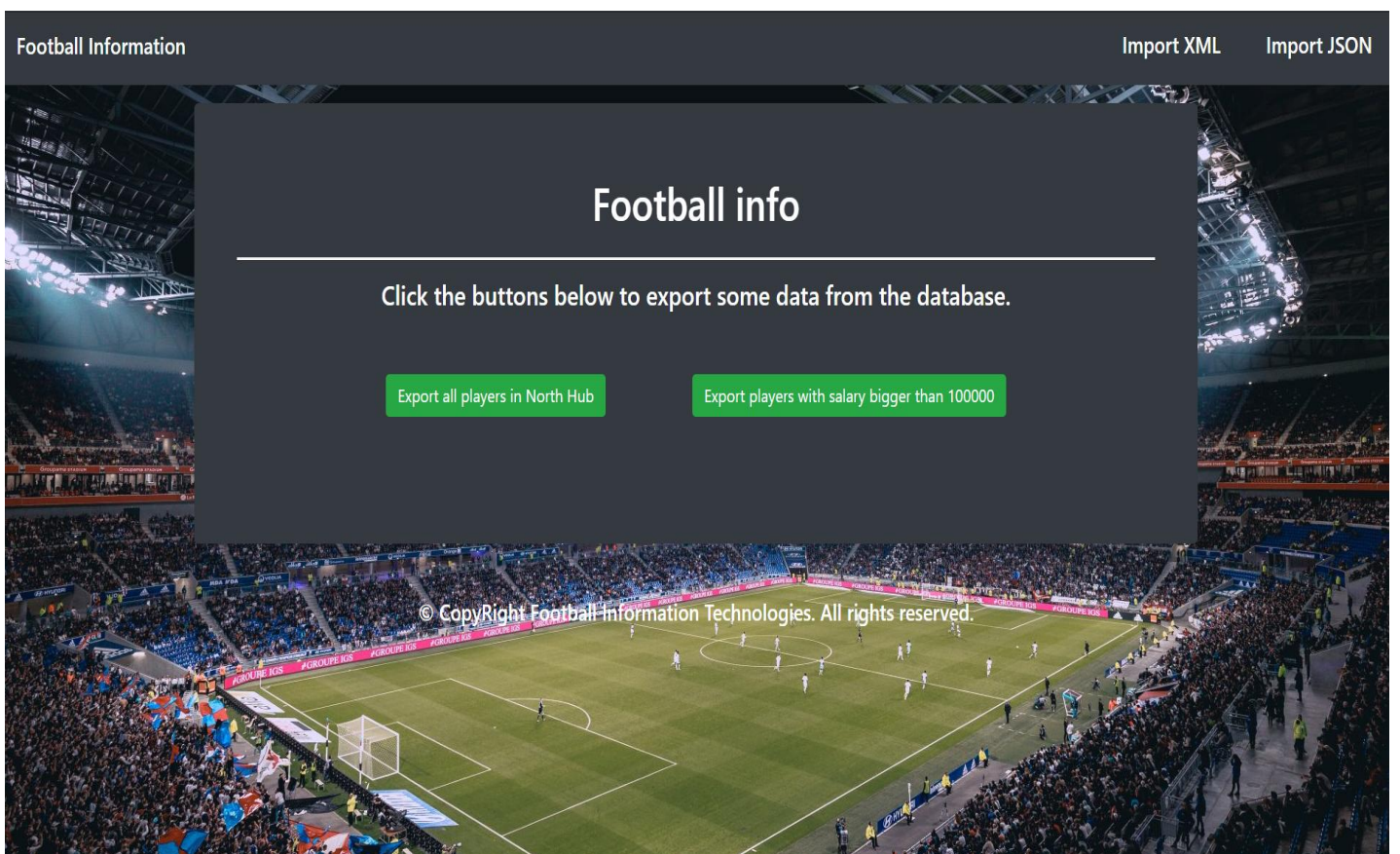
- Import JSON page after importing the data:



- Import JSON page after importing the data:



- Home page after the data is imported:



- Export all players in North Hub:

The screenshot shows a web application interface with a dark header. On the left, it says "Football Information". On the right, there are links for "Import XML" and "Import JSON". The main content area has a dark background with a large, semi-transparent white box in the center. The box has the title "Players in North Hub" and a subtitle "Players in North Hub:". Below the subtitle, there is a list of players for the "Team: North Hub". The list includes: Lionel Messi - ST (Number: 5), Harry Kane - ST (Number: 27), Edinson Cavali - RB (Number: 54), and Fernando Karanga - ST (Number: 31). The background of the application is a live-action football match in a large stadium.

Football Information Import XML Import JSON

Players in North Hub

Players in North Hub:

Team: North Hub

- Player name: Lionel Messi - ST
Number: 5
- Player name: Harry Kane - ST
Number: 27
- Player name: Edinson Cavali - RB
Number: 54
- Player name: Fernando Karanga - ST
Number: 31

© Copyright Football Information Technologies. All rights reserved.

- Export players with salary bigger than 100000:

The screenshot shows the same web application interface as the previous one. The main content area has a dark background with a large, semi-transparent white box in the center. The box has the title "Players with salary bigger than 100000" and a subtitle "Players with salary bigger than 100000:". Below the subtitle, there is a list of players with salaries greater than 100,000. The list includes: Ousmane Dembele (Number: 61, Salary: 7800000.00, Team: Raptors), Lionel Messi (Number: 5, Salary: 6000000.00, Team: North Hub), and Kylian Mbappe (Number: 4). The background of the application is a live-action football match in a large stadium.

Football Information Import XML Import JSON

Players with salary bigger than 100000

Players with salary bigger than 100000:

- Player name: Ousmane Dembele
Number: 61
Salary: 7800000.00
Team: Raptors
- Player name: Lionel Messi
Number: 5
Salary: 6000000.00
Team: North Hub
- Player name: Kylian Mbappe
Number: 4

© Copyright Football Information Technologies. All rights reserved.

2. Project Skeleton Overview

You will be given a **Skeleton**, containing a **certain architecture(MVC)** with **several classes**, some of which – completely empty. The **Skeleton** will include the **files** with which you will **seed** the **database**.

3. Model Definition

There are 3 main models that the **Football info** database application should contain in its functionality.

Design them in the **most appropriate** way, considering the following **data constraints**:

Picture

- **id** – integer number, **primary identification field**.
- **url** – a **string (required)**.

Team

- **id** – integer number, **primary identification field**.
- **name** – a **string (required)** between 3 and 20 characters.
- **picture** – a **Picture entity (required)**.

Player

- **id** – integer number, **primary identification field**.
- **first_name** – a **string (required)**.
- **last_name** – a **string (required)** between 3 and 15 characters.
- **number** – a **Integer (required)** between 1 and 99.
- **salary** – a **BigDecimal (required)** min 0.
- **position** – a **ENUM (required)**.
- **picture** – a **Picture entity (required)**.
- **team** – a **Team entity (required)**.

NOTE: Name the entities and their class members, **exactly** in the **format stated** above. Do not name them in snake case with the dashes, of course. But if a field is specified as **first_name**, you are to name it **firstName**.

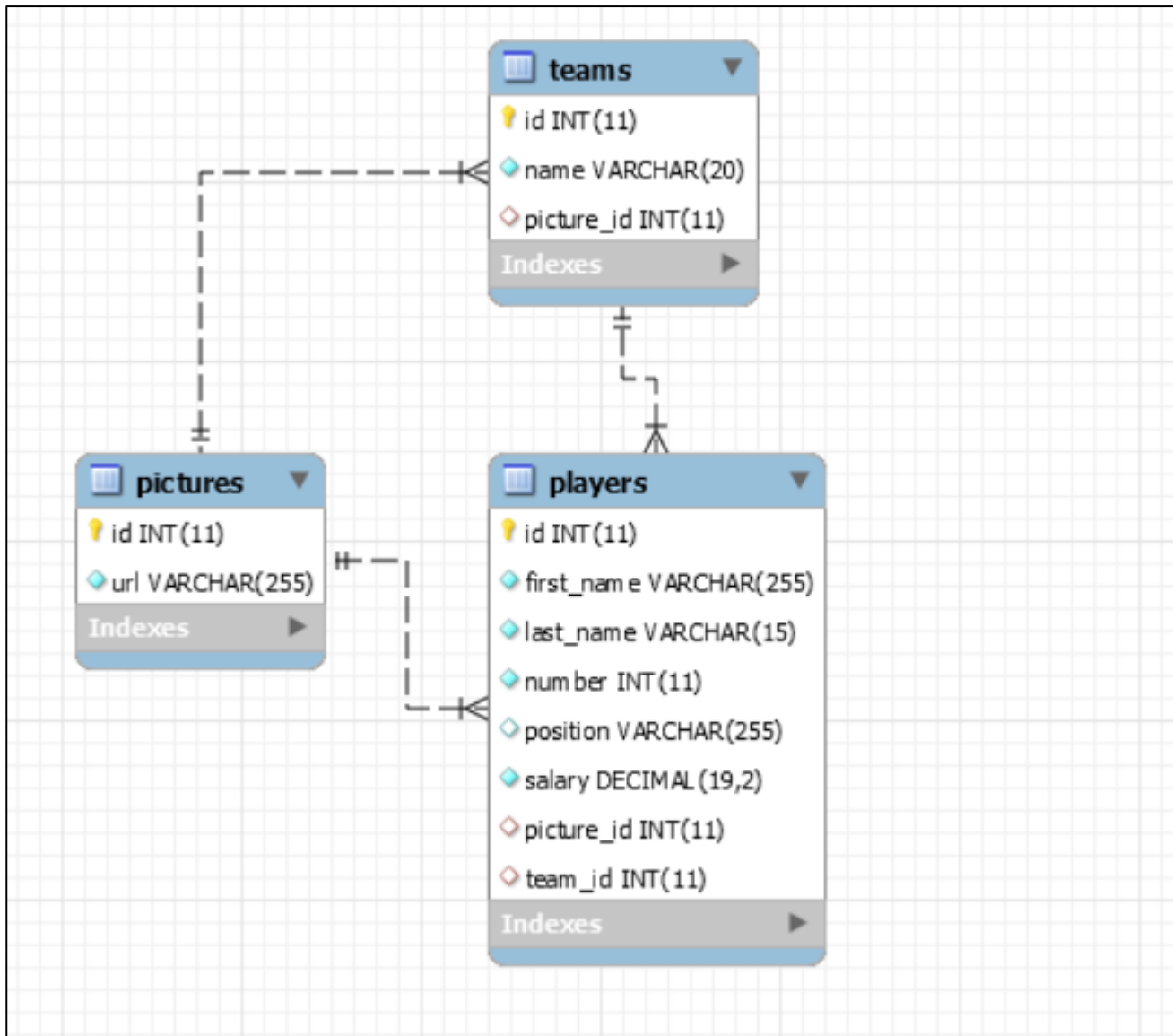
Relationships

The Football info decided to give you a little hint about the more complex relationships in the database, so that you can implement it correctly.

One **Team** may have only one **Picture**, and one **Picture** may have many **Teams**.

One **Team** may have many **Players**, and one **Player** may be appointed to only one **Team**.

One **Player** may have only one **Picture**, and one **Picture** may have many **Players**.



4. Data Import

Use the provided files to populate the database with data. Import all the information from those files into the database.

You are not allowed to modify the provided files.

ANY INCORRECT data should be **ignored** and a message **"Invalid {picture/team/player}"** should be printed.

- **NOTE:** An incorrect data input is an input which is **missing required fields**.

When the import is finished

"Successfully imported {picture/team/player}- {url/name/firstName lastName}"

XML Import

The **Football info** have prepared some XML data for you to import.

Picture (pictures.xml)

pictures.xml
<pre><?xml version="1.0" encoding="UTF-8" standalone="yes" ?> <pictures> <picture> <url>google.pictures#1</url> </picture> <picture> </picture> <picture> <url>google.pictures#2</url> </picture> . . . </pictures/></pre>
Successfully imported picture - google.pictures#1 Invalid picture Successfully imported picture - google.pictures#2

Team (teams.xml)

teams.xml
<pre><?xml version="1.0" encoding="UTF-8" standalone="yes" ?> <teams> <team> <name>West Valley</name> <picture> <url>fc_pictures_1</url> </picture> </team> <team> <name>VeeeeeeeeeeeeryLooooooooooooooooongName</name> <picture> <url>noPicture</url> </picture> </team> <team> <name>Samurai</name> <picture> <url>invalidURL</url> </picture> </team> . . . </teams></pre>
Successfully imported - West Valley Invalid team Invalid team

JSON Import

Player (players.json)

players.json

```
[
  {
    "firstName": "Kiril",
    "lastName": "Despodov",
    "number": 32,
    "salary": 150000.00,
    "position": "Invalid",
    "picture": {
      "url": "google.pictures#1"
    },
    "team": {
      "name": "West Valley",
      "picture": {
        "url": "fc_pictures_1"
      }
    }
  },
  {
    "firstName": "Christian",
    "lastName": "Rodrigues",
    "number": 121,
    "salary": 100000.00,
    "position": "RB",
    "picture": {
      "url": "google.pictures#2"
    }
  }
  . . .
]
```

Invalid player
Invalid player
Invalid player
Successfully imported player: Rubin Star
Successfully imported player: Serj Smokey

5. Data Export

Get ready to export the data you've imported in the previous task. Here you will have some pretty complex database querying. Export the data in the formats specified below.

Export all players North Hub

Export all players which are playing in North Hub:

- Extract from the database, the name of the **team** and information about the **player ordered by id**.

"Team: {Name}

Player name: {playerOne firstName} {playerOne lastName} - {playerOne position}

Number: {playerOne number}

Player name: {playerTwo firstName} {playerTwo lastName} - {playerTwo position}

Number: {playerTwo number}

. . . "

Export players with salary bigger than 100000

Export all the players with salary bigger than 100000.

- Export the **player's full name, number, salary and team name order by salary descending**.

"Player name: {firstName} {lastName}

Number: {player number}

Salary: {player salary}

Team: {team name}

. . . "