# TDW Module 1b - NBA Followers

Diogo Filipe Amaral Carvalho - NMec: 92969

October 2022

Github Pages Link: `https://diogocarvalhoo.github.io/TDW/TDW_Mod1b/mini-projeto/players.html`

## 1 Introduction

Servidores Deca Link: `https://labmm.clients.ua.pt/deca_22TDW/deca_22TDW_11/TDW_Mod1b/mini-projeto/index.html`

This report is the result of the work performed to complete the project of module 1b on "Tecnologias e Desenvolvimento Web" course at University of Aveiro. In this project, the goal was to develop a unique website that displays content on a dynamic way by exploring an API. Basic knowledge on HTML, CSS and Javascript was going to be evaluated. Considering this, NBA Followers is a website that can be used by anyone that loves NBA. It makes use of two different APIs. The main API retrieves information about players and teams and is called data.nba.net. Additionally, a second API, nba-stories, is used to get recent news about NBA world. Finally, due to the fact that the main API don't have any information about players images, the images are obtain on a static url from the nba website, by changing the player unique id.

## 2 Description of dynamic elements displayed on the website

### 2.1 News Side Column

In the right section of the homepage, a list with recent news is presented to the user. This list is built based on the response of the request performed to the secondary API. Each new is built on a dynamic way using Javascript and contains a title, a source and an external link to read more about it at the source. It is possible to filter the number of news presented on the list. The options available are 3, 5 and 10.

## 2.2 Players Card List

When the user enters "Players" page, a list with the players presented on NBA is displayed. Each player is represented by a card containing his image and name. The information about the players is obtain by a request performed at a specific endpoint on the main API. Afterwards, the images are obtain on a static url from nba website, using the id retrieved by the main API. A default image is used when the players have no image on the static url.

## 2.3 Teams Card List

When the user enters "Teams" page, a list with the teams presented on NBA is displayed. Each team is represented by a card containing the team's logo and name. Additionally, the card contains an event that when it is clicked it opens the team profile page. The information about the teams is obtain by a request performed at a specific endpoint on the main API. Regarding the images, they were previously downloaded and stored since i have not found a way to obtain them from the same url used with the players.

## 2.4 Paginator

On both "Players" and "Teams" page, the maximum number of players/teams displayed is 16. Considering this, a paginator was created in order to list the previous or next 16 results. Since the main API has not support for pagination, all the logic behind the paginator was created from scratch. The paginator supports the filters that can be applied to both "Players" and "Teams".

## 2.5 Filters

On both "Players" and "Teams" page, it is possible to filter the results by name using search bar. The players or teams that start by the input placed on the search bar are presented to the user. Additionally, in both pages a counter of the total number of players or teams (depending on the page were the user is) in the nba, in the west conference and in the east conference is presented to the user. Those numbers can also be used to filter players or teams by conference. It is also possible to combine both filters.

## 2.6 Team Profile Page

Finally, the last element of the website is the team profile page. From the list of teams presented on the "Teams" page, if the user clicks on the card of a team it is redirected to the team profile page were the information about the team is displayed on a deeper level. In this page, it is also displayed the roster of the team and the stats, meaning, the best player of the team on a specific set of stats. These information is obtained from the main API.

### 2.7 Game Page

As professor requested, I have created a simple game with the goal of using localstorage. The game section contains 2 different pages. The first one, contains a simple login that only asks for a username and presents the ratings of current players. The second page, after providing a username, have 5 player cards. Each card contain an input with auto completion. The user must select 5 players (one for each card), according to the positions of the players (user must select 1 center, 2 guards and 2 forwards). The user cannot select the same player for both guard position or forward position. In the end, the user is able to play and a random score is assign to each player performing a total score point for the team built by the user. If the same username is passed again on the login page, the previous team built by the user is displayed and changes can be made.

## 3 Implementation Strategy

Regarding the strategy used to create the website, I have separated the code in different files. Each section of the website (homepage, teams page, players page and team profile page) has a specific .html and .js file with the HTML and Javascript used to create that page. A folder named "assets" contains all the CSS styles and static images, as well as the Javascript from the initial template that I have used to create the website. Bootstrap 5 was integrated and used to apply styles on the website. The Javascript code was decoupled with different functions for each purpose. This structure as well as the comments that I have written on the code increase the ability to read, understand and maintain the whole code.

## 4 Main technical challenges and their solutions

During the completion of this project some challenges emerged. In this section I will describe the problems and how I have overcome them.

### 4.1 APIs Selection

In the initial stage I had to choose the theme of the website and the main API. A large set of APIs were available, however the best ones I found had a limit of requests per minute and per day. The API I chose is free, however it is not as completed as the other ones. Information was not totally correct. I had to remove duplicated and wrong records.

### 4.2 Player Images

Another problem that I faced was related to the players images. In my website, each player needs to have an image. The API I choose does not have images.

To solve this, I found a static url that uses the same player identification used in the API and I use that link to get the images of the players.

### 4.3 Paginator

The last problem I faced was related to the number of results I show to the user. The total number of players retrieved by the API is 826. It is impossible to display all this information to the user at the same time. Considering this, I created a paginator that allows the user to change the current displayed results.

## 5 Obstacles not overcome

As I said before, I used an initial template to create my website. Despite this approach saved me some time, the template comes with lots of extra elements that are not used. I have cleared all the extra HTML elements, however my .css file contains styles that are not used. An additional problem is related to CORS policy. In my localhost development environment I have no problem in sending requests to the API, however, after deploying the website I realised that the main API requests are not answered due to CORS policy issues. I had no time to change my API since it would change my whole website.

## 6 Conclusion

To sum up, the goal of the project was achieved. Working on it allowed me to practise my front-end development skills.