TDW Module 2a - NBA Followers

Diogo Filipe Amaral Carvalho - NMec: 92969

December 2022

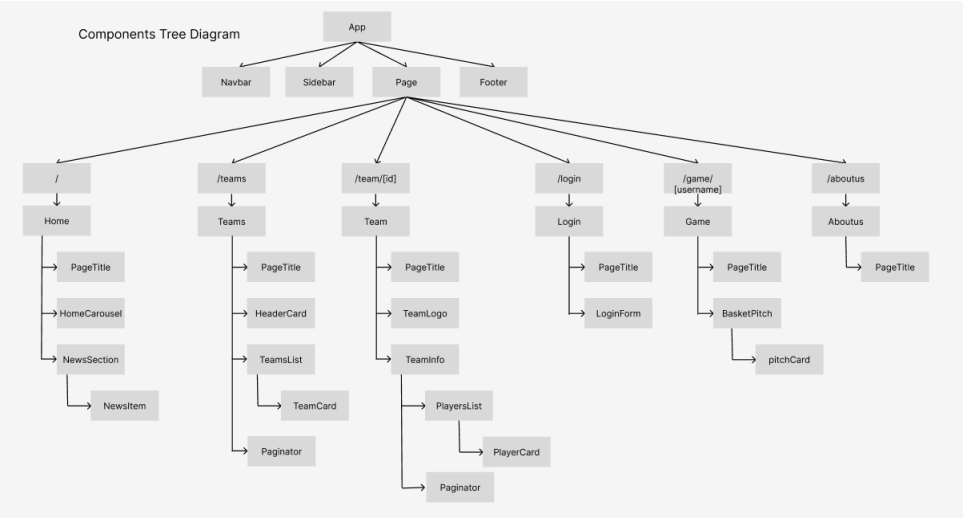
Heroku Link: `https://nba-app.herokuapp.com/`
Repository Link: `https://gitlab.com/pw-mctw/tdw-2022/tdw-mp2-diogo-carvalho`

1 Introduction

In this project, the goal was to develop a front-end React, Razzle or Next application that consumes data from a public source and represent that data. Advanced knowledge on React related content such as routing, state management with Redux and Redux toolkit, perform asynchronous actions with Redux-Thunk and RTK-query 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 three different APIs. The main API retrieves information about teams and players per team and is called api-nba. Additionally, a second API, data.nba.net, is used to get information about all NBA players (since the first API only retrieves data about players of a specific team and has a limit amount of requests per minute). Finally, a third api called newscatcher is used to retrieve recent news about NBA world.

2 Application Structure and Design

Concerning the design, a light theme with light colors was selected to provide easy to read interfaces. Functionalities were separated in different pages that can be accessed through a side menu that the user can open and close whenever he wants. The application follows a component tree structure. Components where created to separate UI elements, removing duplicated code and improving code readability and maintenance. The developed components tree is described by the figure below:



The application code is separated in several files for better comprehension. The code contains the following folders/sections:

- components: Contains folders for each component in the application. An additional "styles" component was created to store some of the styled components objects created that are used across multiple components.
- data: Contains static data structures. It only has a dictionary for team's id translation between the two different APIs used to retrieve information about players.
- pages: Contains file for each page in the application.
- public: Contains app static images.
- redux: Contains store creation file and Reducers for API calls and store management.
- styles: Contains global CSS files.
- utils: Contains a function used to interact with localStorage, which is required in app mini-game to keep records between sessions.

3 Implementation Strategy

In this section we are going to discuss the implementation strategy. The main technologies used to develop the application are listed below:

- Nextjs: React framework that serve as the base foundation.
- Typescript: Programming language that guarantees higher variable type control.

- Redux Toolkit: Manages the state of the application.
- RTK-Query: Manages the asynchronous calls to external APIs.
- Styled Components: Used to manage CSS applied to HTML elements.
- Gitlab: Code repository.
- Heroku: Platform used to deploy my application.

Some additional technologies were also used such as React-bootstrap, React-icons, and Google Fonts for styling purposes and ESLint for code analysis.

3.1 State Management

Application state was managed using Redux Toolkit and Components Props. Components Props were used to manage local state that was not necessary to be stored in the global store. Global state was managed by Redux Toolkit to share state between components in different parts of the tree. Two slices were created, one to manage filtered teams information and another for players.

3.2 Async API Calls

Asynchronous calls to external APIs is handled by RTK-Query that manages all states of the request, such as waiting for response, getting error and getting valid response. Two apis were created. One is used to make requests to the news' API. The remaining was created to be used to send requests to the main API however it is not used since the first request that gets all teams is done in server side rendering and the filters are applied manually since the external API doesn't have endpoints to support it.

3.3 Server Side Rendering

One of the main advantages of using Nextjs is server side rendering. Server side rendering was developed in all pages of the application. In static pages, Nextjs does it automatically. In the remaining pages, initial request are performed inside getServerSideProps method using Fetch and the data is passed to components props.

3.4 Routing

Routing between pages is performed using Nextjs router system.

3.5 Pagination

To achieve a better UX, a paginator component was developed that is used in teams list and also in players list inside a specific team. The results, displayed in the lists that the paginator manages, are gathered by external calls managed

by RTK-Query, meaning that, those results are in cache avoiding to repeat same external API calls when we go back in the paginator.

3.6 CI/CD Pipeline

As an extra point, I created a CI/CD pipeline to build, test and deploy my application in the Heroku platform. This pipeline contains three stages. The first builds the application, the second is the test stage and runs lint (unitary tests should be added in the future) and finally the deploy stage deploys the application to Heroku.

4 Challenges

The biggest challenged that I had was related to the APIs that I chose. The main API (api-nba) doesn't have an endpoint to retrieve all players (only players per team) and has a limit of 10 requests per minute. Since I needed all players in my mini-game, I had to use multiple APIs. Another challenge was the limit of requests per minute that lead to adapt app development to that situation. Using new technologies such as Redux Toolkit and RTK-Query was challenging aswell.

5 Obstacles not overcome

The main problem in my development was that, since I was developing the app alone, I made a big bang commit which is not a good approach. Another obstacle was related to players images. In the mini-game, players contain images. In the list of players per team, players don't have images. This happens due to the use of different APIs in each page.

6 Conclusion

To sum up, the goal of the project was achieved. Working on it allowed me to upgrade my skills with Nextjs, Typescript, Redux Toolkit and RTK-Query.