universidade de aveiro
departamento de electrónica,
telecomunicações e informática

Tecnologias e Programação Web

Project no. 2

André Ribeiro
NMec: 112974

Rúben Garrido
NMec: 107927

Violeta Ramos
NMec: 113170

December 17, 2024

Contents

1 Introduction

This report will highlight the main changes made in this second iteration of the project, as well as show the app's architecture, access information, deployment resources and local development instructions.

Remembering what is arbitrage betting

Arbitrage betting is a strategy that exploits differences in odds offered by various bet houses (bookmarkers) to guarantee a profit, regardless of the outcome of the event. This occurs when the implied probability calculated using a combination of odds (Win, Draw, Lose) from different bookmakers are less than 100%, creating an opportunity for risk-free profit. By placing bets on all possible outcomes across multiple betting platforms, a gambler ensures that one of the bets will win, covering the losses of the other bets and generating an overall profit.

Mathematically, for a n-outcome event, arbitrage exists if:

$$\text{Arbitrage Condition} = \frac{1}{\text{Odds}_1} + \frac{1}{\text{Odds}_2} + ... + \frac{1}{\text{Odds}_n} < 1.$$

If this inequality holds, placing proportional bets on both outcomes will yield a guaranteed profit. The challenge lies in identifying and executing these opportunities quickly, as bookmakers adjust their odds frequently.

Role of LaBet

LaBet is a platform designed to automate the process of scraping odds from multiple betting houses and identifying arbitrage opportunities in real-time. By leveraging technology and data analysis, LaBet aims to simplify the complex process of sports arbitrage, enabling users to maximize their potential returns efficiently.

2 App Features

The *LaBet* application offers a comprehensive set of features to facilitate sports arbitrage betting by automating the process of collecting and analyzing odds from various betting houses.

2.1 Main features from Project 1

Business Model Tiers

Different tiers offer varying levels of functionality, such as the maximum amount of money per month, minimum odd arbitrage, and more. Users can choose a subscription tier that best fits their betting strategies and needs.

Data Analytics and Betting History

By accessing their betting history, users can review past bets, monitor their profit, and identify patterns in their betting behavior.

Odd scraping & Task Scheduling with `django-q2`

The app interacts with discovered APIs from various betting houses. To keep odds data up-to-date, the app leverages `django-q2` for scheduling background tasks. This setup ensures odds are scraped and updated at regular intervals.

Authentication and Authorization

The app includes a robust authentication and authorization system built using Django's built-in features, which ensures that only registered and authorized users can access specific features, such as viewing odds, performing arbitrage calculations or managing data.

Team Name Similarity Algorithm

To reconcile team names that may differ between betting houses and the app's database, the Ratcliff/Obershelp algorithm is employed (using Python's `difflib`). This string similarity algorithm calculates the similarity between team names, ensuring an accurate match and reducing errors when comparing odds from multiple sources.

2.2 New features added in Project 2

Administration panel

Admin users can now manage users and bets. In both cases, a list of each is shown, so that admins can check everything related to LaBet usage.

Regarding users, their profile, tier and permissions can be changed from the web, without resorting to manual database queries. Users can be banned.

When it comes to bets, administrators can check the user and the amount bet, while being able to delete them one by one and restore the money to the respective users.

Game popularity

Users can now see, in each game, how many people already placed a bet for it. Not only this gives an important statistic about whether a game is good or not for a bet, it also pushes our customers into making more bets and potentially subscribe the higher cost tiers.

Team listing

The app now features a team listing, with search options. This allows users to directly check games from their favorite teams, as well as get their odds.

Reports page

There is now a dedicated reports page, which includes charts that help users get a different view on their profits and bets.

Redesigned home page and wallet

Both home and wallet pages now feature a brand-new full responsive design, with a layout that prioritizes space and improves design, when compared to the one in Project 1.

3 Architecture

LaBet is now a multilayered app, with a separate backend and frontend that communicate between each other using REST.

3.1 Backend

The backend is made in Django, and uses Django REST Framework as the RESTful framework.

Authentication is handled using `django-rest-knox`, which manages the users' access tokens by itself, with little to no configuration.

OpenAPI docs were also implemented using `drf-yasg`, which generates an OpenAPI schema and hosts a Swagger instance. The endpoints are documented using decorators in their functions, which specify metadata.

Big datasets - such as teams and games - feature pagination support, so that the backend doesn't have to send all the data to the client in one time.

3.2 Frontend

The frontend is made using Angular, with Flowbite as the chosen UI.

TanStack Query was used in this project. It eases the data querying and mutating progresses by creating a "capsule" around the traditional fetching function, which automatically deal with potential errors, loading times, infinite queries, and more. It also acts as a cache, which maintain state about recent queries. This package has adapters for each framework, which means that in Angular it acts as injectables - `injectQuery` and `injectMutation`, for queries (getting data) and mutations (changing data), respectively.

To help in some fetching abstraction, ky was used to create the API services. It is based on the Fetch API, yet features some helpful utils like hooks, common fetch options, type inference and more. It's as powerful as axios, yet with a renewed and lightweight codebase.

The infinite scrolling - in both teams and games lists - was implemented with the Angular Infinite Scroll package, which deals with scroll positions and automatically calls a callback. In this case, the callback is the `getNextPage()` method from a TanStack Query's query.

4 Access information

4.1 Deployment

The backend of this app is deployed at Python Anywhere, whereas its frontend is deployed at Heroku.

Regarding the API, a maximum of 100 CPU-seconds is allowed per day, which means that the current 10-minute external API scrapper would quickly run out of resources. Instead, a fake game database was used to mock the real behavior. This does not affect local environments (check Section 4.3)

4.2 Users

Database migrations include some default users: an admin user, and three normal ones. Although all information can be checked in the `resources/data_user.sql` file, the most important authentication data is referenced as follows:

User	Password	Plan	Admin
admin	`Test#1234`	Ultra	Yes
andreribeiro87	`Test#1234`	Pro	No
RGarrido03	`Test#1234`	Pro	No
VioletaR	`Test#1234`	Free	No

Table 1: Default user information

4.3 Local execution

4.3.1 Backend

To locally run this project's backend, some packages must first be installed. Those are explicitly listed in the `requirements.txt` file. The detailed instructions are as follows:

1. Have a recent version of Python
2. Create a virtual environment

```
python3 -m venv venv
source venv/bin/activate
```

3. Install the required packages

```
pip install -r requirements.txt
```

4. Apply migrations and start scheduling

```
python3 manage.py migrate
python3 manage.py qcluster
```

5. In another shell, run the app

```
python3 manage.py runserver
```

4.3.2 Frontend

The frontend uses Bun, a lightning-fast npm-compatible package manager. However, it can also be used with a typical npm installation.

1. Install project dependencies

```
bun install
npm install  # Legacy, should not be used
```

2. Run the server

```
bunx ng serve
npx ng serve  # Legacy, should not be used
```

5 Conclusion

The second iteration of the LaBet project introduced significant enhancements to improve functionality, user experience, and system performance. Key additions include the *Administration Panel* for managing users and bets, *Game Popularity* insights to drive engagement, and a *Team Listing* feature for easier access to favorite games.

Architecturally, the adoption of a Django backend with REST APIs and an Angular frontend ensures scalability and maintainability. Tools like Django REST Framework, TanStack Query, and Angular Infinite Scroll optimize data handling, responsiveness, and user interaction. The redesigned *Home Page* and *Wallet* further enhance usability with a modern, responsive interface.

In summary, this iteration builds on a solid foundation, delivering essential features and improvements that streamline sports arbitrage betting while positioning LaBet for future growth and innovation.