Explanation

WebScrapingAPI

Author: Zilai Denis

1. Choices

The choices for this project, in what regards the programming languages, were:

- NodeJS:
  - allows you to use JavaScript for both server side and client side development
  - works well with the Puppeteer module, a browser automation tool, making it suitable for web scraping
- Puppeteer:
  - high level API to control browsers
  - it allows interaction with web pages
- Express:
  - suitable option for building APIs
  - simple and minimal framework
- HTML:
  - standard language for building the structure of web pages
- Tailwind:
  - allows quick and simple styling of web pages, without writing custom CSS code
- Insomnia:
  - a suitable tool for manually testing the endpoints of the application
- Visual Studio Code:
  - support for many programming languages
  - integration with NodeJS and JavaScript

The choice of this languages and environments is an excellent choice for simplifying the development of a web scraping application.

2. Proposed standout features

- File save feature: the proposed standout feature for this application is a save function, which allows the user to save the fetched data from the target browser, under a .txt file. This feature would boost the API's usability and attractiveness to users, as it would be useful in further uses of the fetched data, such as storing them in a database, or sharing the data in an easier manner.

3. Learning experience

This project was a challenge, pushing me to step out of my comfort zone, by experiencing new programming languages (Tailwind) and putting my mind to work in finding different approaches to solve the issues that I've encountered during the development of the application. For example, the biggest challenge of this project was to determine the suitable tool for the browser automation (between Puppeteer and Cheerio), as initially was challenging to determine if a web page is dynamically loading or static. I had an attempt to design the application with Cheerio, but it did not work well on the target website, even though it worked on other test websites. I had to rethink the approach, and after some research, I have found that Puppeteer is the perfect tool for the browser automation, being able to dynamically interact with web pages.

During my previous work experience, I did not have the chance to continue using NodeJS. The opportunity to develop applications again, encouraged me to further expand my knowledge and abilities which I had from university past projects.