Jay Calhoun
Holberton
Cohort 14.5

Custom-API

27th February 2022

OVERVIEW

In this project I will be scraping wikipedia for data on the United States National Parks. The wikipedia html gets stripped of everything except for its paragraph data. The project will include user authentication, and after going through the login portal a random National Park will be selected from the database and its data rendered to the homepage. There will also be a socket open that loads images for each park based on pagination displayed on page.

GOALS

1. Create an ETL for wikipedia data
2. Setup an API for connection to the client
3. Communicate with the database to render data to the client
4. Setup a socket for sending image data to client
5. Manually store photos for each dataset
6. Adjust frontend for aesthetics
7. Complete design documents

SPECIFICATIONS

Begin collecting the data you need, set up your database based on your design docs, build an ETL process to populate the database. Even if you (and your users) are manually inputting data into your database instead of collecting data from other sources, you do need to set up a database for this project for your API to interact with.

Once your database is in place and populated with data, it will be time to consider how you build your API to interact with your data. In this project, your API must do the following:

- Authenticate users of your API
- Allow for pagination of data

- Allow for caching of data to reduce hits to your database when possible and improve responsiveness to your users

Your API must also implement one of the following features:

- Queuing systems (for long-running process on your server)
- Web sockets (two-way communication between client and server)

An API is only as good as the user's ability to use it. For this task, build a small, single-page application that utilizes your API and presents data to your user in a relevant way.

It does not need to be pretty, though that always improves any product - the most important thing is that it is able to use your API to authenticate users, retrieve data (including the use of pagination), and in retrieving data your user will end up using a caching system at times (when appropriate). This page should utilize the API where a queuing system or web sockets are implemented, as well.

TOOLS

axios v0.26.0
bcrypt v5.0.1
csv-parser v3.0.0
ejs v3.1.6
express v4.17.3
express-flash v0.0.2
express-session v1.17.2
method-override v3.0.0
mongoose v6.2.3
passport v0.5.2
passport-local v1.0.0
socket.io v4.4.1
dotenv v16.0.0

MILESTONES

Functioning Backend

The data is loaded. The http requests are being made successfully. The data is rendering to the frontend. The socket has open communication. Estimated 50 hours.

Friendly UI Frontend

The pictures are loading. The text is easy to read. The page is responsive. There is something pleasing to look at in the form of colors and pictures. Estimated 16 hours.

Documents Complete

The design documents and project proposal are complete. The blog post about the project and my learning experience is written and published. Estimated 6 hours.

Estimated Time

72 hours.