Documentation - The Dev Challenge

Author: Guilherme Santos Souza

Access the application: https://the-dev-challenge.herokuapp.com/

GitHub repository: https://github.com/GlermS/the-dev-challenge

This application reads purchase files (tab separated) and counts the total gross income of each file and total all-time gross income, according to the current user.

Settings

Language, libraries and frameworks

Ruby

- Ruby v2.5.0
- Node.js v10.1.0
- Yarn v1.22.10
- Ruby on Rails v5.2.6
- Gems:
    - omniauth-rails_csrf_protection
    - omniauth-google-oauth2 - authentication
    - dotenv-rails
    - jwt

Javascript

- Jquery v3.6.0

System

- Ubuntu v18.04

Database

- Sqlite3 - Development
- Postgres - Production

Project Structure

This project was made using the Ruby on Rails framework. Consequently, the MVC pattern was adopted to exploit the framework resources.

Models

The first model was created which represents the purchases (each line of the file). The data was persisted in the database, with the following fields:

- id: integer
- purchaser_name: string

- item_description: string
- item_price: float
- purchase_count: integer
- merchant_address: string
- merchant_name: string
- user_id: integer
- created_at: datetime
- updated_at: datetime

The second model represents the users. The data was persisted in the database, with the following fields:

- id: integer
- name: string
- email: string
- created_at: datetime
- updated_at: datetime

The database setting and migrations are saved in the folder "db". The model class is defined in the folder "app/models".

Views and Controllers

Views and controllers are both responsible for client interaction. Three controllers were created, one to receive the first request plus respond with the home page HTML, the second to process the request from the form and the third controls sessions. The two first controllers have a set of views to respond to the client. All of them are located in the "app" folder.

The controller **home** only receives the GET request from the route "/". The corresponding view is an *HTML* page and its JS and CSS files are saved in the folder "lib/assets".

The controller **purchases** receives a POST request from the route "/purchases/post_file" with the ".tab" file, processes the text, and determines the total gross income. In this case, the response is a ''.json' file that contains de gross income values.

The controller **sessions** receives a GET request from the route "/auth/google-oauth2/callback " with the data from the Google Authentication api mediated by the omniauth middleware and finish session. In this case there is no view, the interaction is made via cookies and the user is redirected to home.

The authentication token is encoded using JWT which contains the user email in the payload. The Rails' sessions also encrypt the cookies data, reinforcing protection of sensitive data.

The api keys and the JWT secret are saved in the ".env file".

Setup environment

To work with Rails, it is necessary to install de following prerequisites:

- Ruby- Installation guide: https://www.ruby-lang.org/en/documentation/installation/

  Check installed version

  ```
  >> ruby -v
  ```

- SQLite3 -Installation guide: https://servermania.com/kb/articles/install-sqlite/

  Check installed version

```
>> sqlite -v
```

- Node.js - Download: https://nodejs.org/en/download/

  Check installed version

  ```
  >> node -v
  ```

- Yarn - Installation guide https://classic.yarnpkg.com/en/docs/install#windows-stable

  Check installed version

  ```
  >> yarn -v
  ```

Install Rails:

```
>> gem install rails -v 5.2.6
```

Development Environment

To start the development database, run the following code on the terminal:

```
>> rails db:migrate RAILS_ENV=development
```

In order to start the development server, run the following server:

```
>> rails s -e development
```

Tests

The test files are saved in de folders "/test". In addition, the example file plus three alterations were saved in the "/test/fixtures" to be used during tests.

To run the tests, it must start the database in the test environment by the following code:

```
>> rails db:migrate RAILS_ENV=test
```

And then, run the following code:

```
>> rails t
```

Production Environment

In order to start a local production server, it must start the database in the environment by the following code:

```
>> rails db:migrate RAILS_ENV=production
```

And then, run the following:

```
>> rails assets:precompile RAILS_ENV=production
>> rails s -e production
```

Heroku

To deploy the application using Heroku, follow the instructions of the link below:

https://devcenter.heroku.com/articles/getting-started-with-rails5

Run the following command before using `git push heroku master` to change the Heroku's stack:

```
>> heroku stack:set heroku-18
```

Then start the database with the following command:

```
>> heroku run rails db:migrate
```

Configurations

General settings are saved in the "/config" folder. The server settings are in the "puma.rb" file.