VanHack

Voucher Pool Microservice

Summary

General description

The voucher pool microservice was developed using Laravel 5.5, Bootstrap 4, jQuery libraries Dynatable for dynamic tables and datepicker.

Postman was used to test the API calls.

Installation

1. Clone the repository (https://github.com/AngelGris/voucher_pool_microservice.git)
2. Create an empty database for this project in your database engine.
3. Configure the database connection in *config/database.php* file.
4. Copy **.env.example** file to **.env** and change the database connection configuration there too. Optionally, the application name can be set in the .env file too. (If the application name has more than one word it's required to enclose them in quotes, i.e. "My App")
5. In a command line window go to the Laravel project folder and update dependencies running ***composer update***.
6. Still in Laravel's project folder run ***php artisan migrate*** to create the tables in the database.
7. Optionally seed the database by running ***php artisan db:seed***. This creates 50 recipients, 5 special offers and the corresponding voucher codes. One of the recipients created is john@doe.com, which is used later for testing.
8. Now run ***php artisan key:generate*** to generate a unique encryption key for this Laravel instance.
9. Run ***php artisan serve*** in the command line to start Laravel server. Now you can enter your web browser and go to http://localhost:8000 to see it working.

Web interface

The system doesn’t have and security or authorization integrated, so accessing the sites takes the user right tot he main page without any login required.

Home

The main page shows some basic statistics of the system (total vouchers, unused vouchers and used vouchers), and below them a list of the created vouchers displayed in a dynamic table that allows sorting by clicking on the headers, changing the number of results per page and run a quick search.

There’s also a blue *Add vouchers* button to create new offers and vouchers.

Add vouchers

To add new special offer click the blue button that will take you to the *Add vouchers* form. The form only has 3 fields: 2 text boxes for the offer name and discount and a date picker for the expiration date.

All fields are mandatory and after submitting them a new special offer is created in the database and a new voucher for each recipient is generated.

API Endpoint

An API endpoint is provided to get a list of the valid vouchers for a given recipient and to redeem a voucher code. Since no authorization system is implemented, no API key or token are required.

The API endpoint will be available at http://localhost:8000/api while the Laravel server is running

GET vouchers/{email_address}

If the provided email address is valid and corresponds to a registered recipient, a list of the valid vouchers for that recipient will be retrieved. The retrieved list include the special offer name, voucher code, discount and expiration date.

Valid vouchers are those that hadn't been used yet and are not expired.

POST voucher

This POST call uses 2 parameters: recipient email address and voucher code. If the email is valid and belongs to a registered recipient and the voucher is for a valid voucher for that same recipient, then it's redeemed using the current date and the discount to be applied retrieved.

Testing

To test this application, testing code has been written both in Laravel, which uses phpunit for testing, and in Postman.

Tests run using john@doe.com recipient, so seeding the database, or manually creating this recipient and some vouchers for it, is required before testing.

Testing in Laravel

To run the test cases for Laravel open a command line prompt, go to the project folder and run ***./vendor/bin/phpunit***

Laravel tests for response codes on the different URLs and API calls, views display and form validations for the *Add vouchers* form.

Postman collection has further testing for the API calls.

Testing in Postman

The Postman collection found in the *docs* folder includes API calls and testing code for them. The **POST voucher** call need a valid voucher code in order to work, otherwise it'll return an *"invalid voucher code"* error.

You can use the **GET vouchers** call to get the list of valid voucher codes and use one of those in the **POST voucher** call, using the same recipient email.

If the POST voucher call is successful, it means the voucher was used and trying to use it again will throw an "Invalid voucher code error", so it requires a new valid voucher code each time it's successfully tested.