

Sergio Patricio Fautsch Badin– A01421602

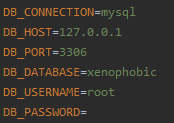
**Extracting sentiment and emotion features (Laravel with PHP)**

February - June 2021

Intelligent Systems

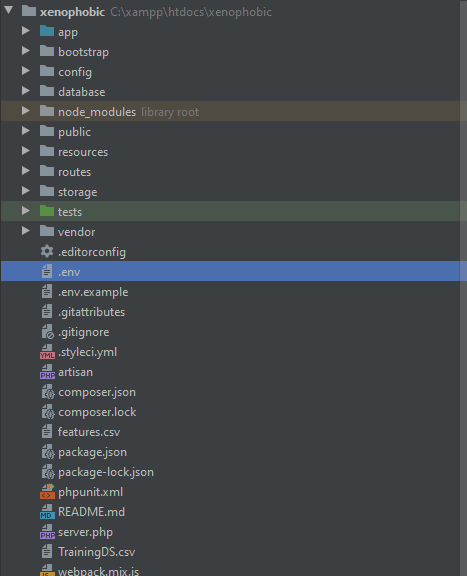
**Initial Setup**

* Clone the repository: **git clone** [**https://github.com/PatoFb/xenophobic.git**](https://github.com/PatoFb/xenophobic.git)
* Run **composer update** in terminal.
* Run **npm install**.
* Rename **.env.example** file to **.env**
* Fill your MySQL configuration, mine looks like this (I’m using xampp to run it, so my path is C:/xampp/htdocs/xenophobic):



* In terminal, run **php artisan migrate** to create the table.
* Inside your database manager, import the **TrainingDS.csv** file provided by the repository (it is the same file that’s in Kaggle, but data is filled with “?”).
* To execute the program, run **php artisan serve** in your terminal and in your web browser go to **http://127.0.0.1:8000/**.

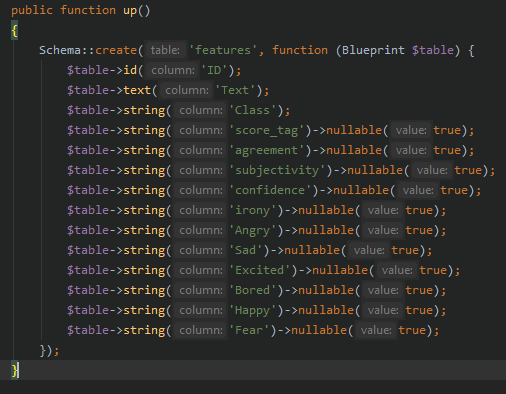
**Structure**

The project structure will look like this:

To make this code, I first imported some dependencies:

* **composer require laravel/ui**
* **composer require meaningcloud/meaningcloud-php**
* **composer require paralleldots/apis**

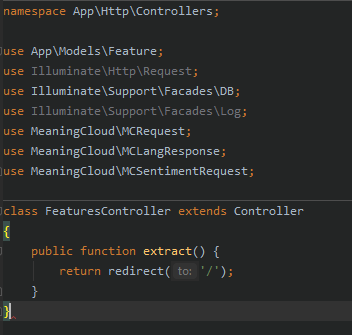
Once the project was set, I ran **php artisan make:migration create\_features\_table** and filled that new file inside **database/migrations** with the table structure I wanted (**php artisan migrate** runs these migrations).



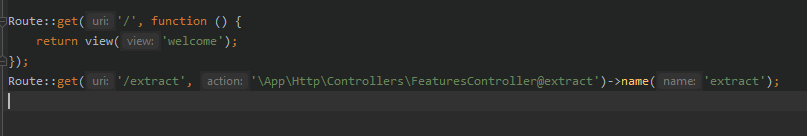
I ran **php artisan make:model Feature** to create the model for the table (Laravel links models and tables with the same name automatically) and added the **fillable** property for all the columns (models are inside **App/Models**).



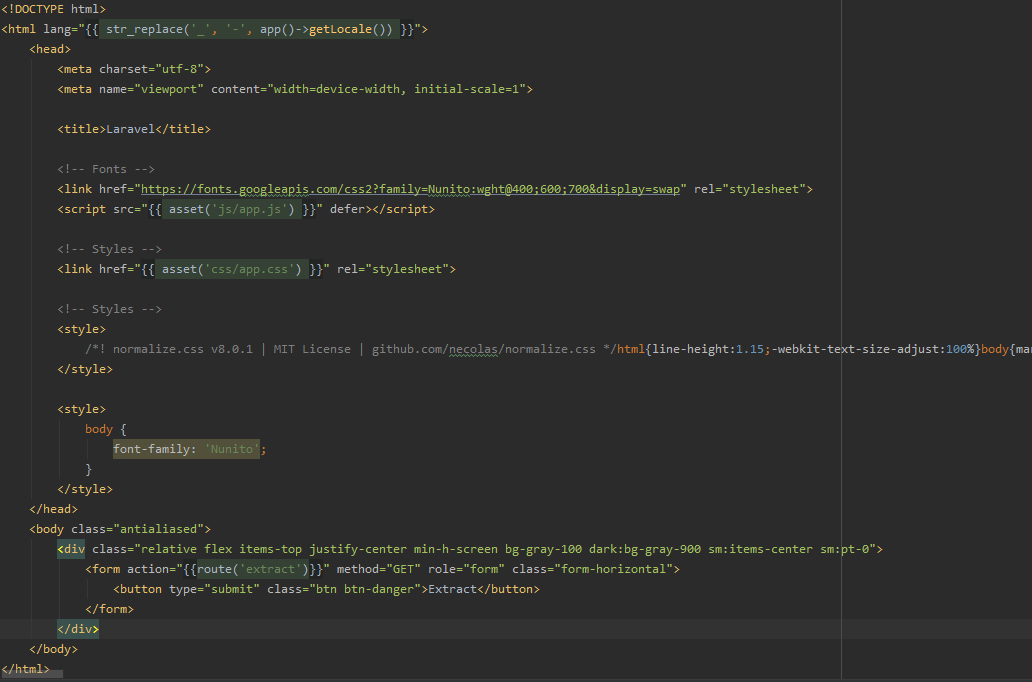
I ran **php artisan make:controller FeaturesController** to make the controller file located inside **App/Http/Controllers**. I created the function **extract** and left it empty and added all the **MeaningCloud** dependency files I needed according to the turorial, the **Feature** model file, the **DB** file inside Facades to make queries to the databes and the **Log** file to show outputs inside a log file, mostly for testing.



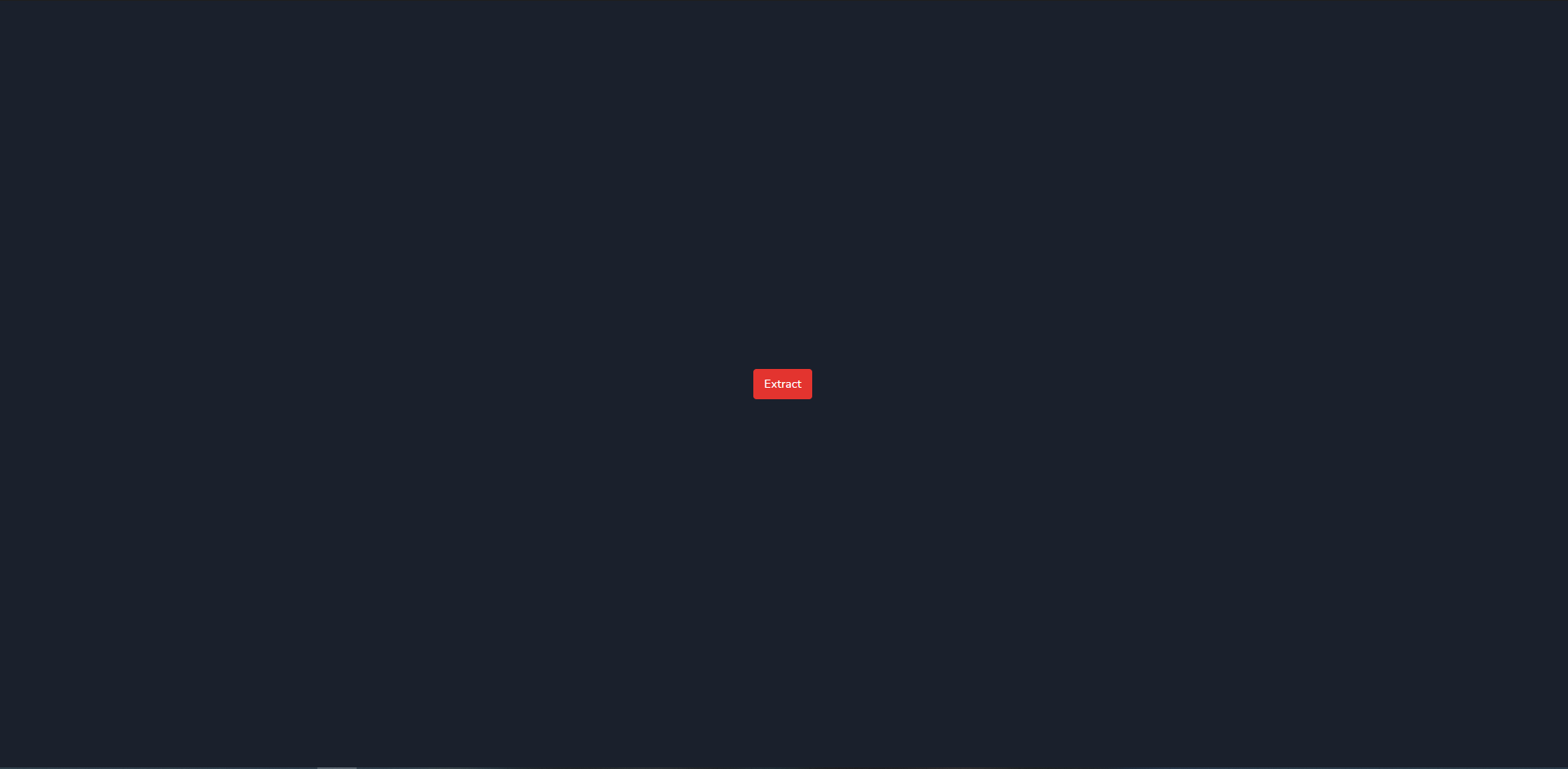
I added a route inside **routes/web.php** that points to the extract function.



I modified the main Laravel view, located inside **resources/views**, adding a form with just a button to point to the **extract** route.

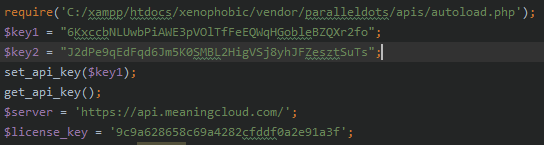


In the browser looks like this:



Once you click the **Extract** button, it will call the route that points to the **extract** function inside the **FeaturesController**.

I followed both tutorials from ParallelDotsand MeaningCloudto make requests to the API. MeaningCloud has namespaces so the **use** function works. ParallelDots doesn’t have a namespace so you have to **require** the **autoload.php** file from the dependency. It’s located inside **vendor/paralleldots/apis**. I set up two keys for this since we have the limitations, storing them in variables and set the first one with **set\_api\_key($key1)**. I also set up the server and license key from MeaningCloud as indicated on the tutorial.



Note: I had to edit the host in the **$url** variable inside the paralleldots autoload file because it was not detecting it when I made emotion requests, so I set it up manually.



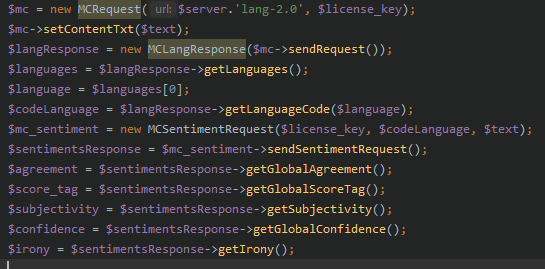
I called all tweets by calling the model using Eloquent and ordering them by ascending IDs and initialized a counter.



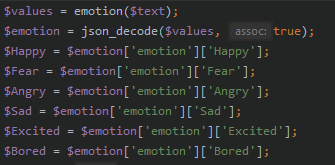
Inside a foreach statement, I ran through all my **Feature** objects and saved the **Text** and **ID** column values inside a variable.



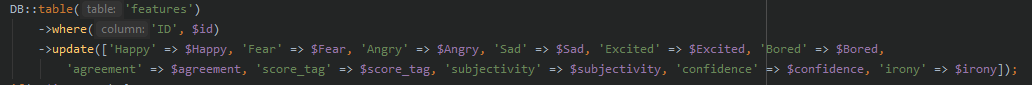
I followed the tutorial to make a request to the MeaningCloud API, grabbing the text, server, and license key to set up the request, then making a request to detect the language and finally making the sentiment request, storing it in a variable, finally I called each sentiment feature individually and stored them inside variables of the same name.



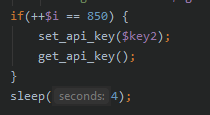
Next, I made the request to ParallelDots, following their tutorial. I stored the json response inside a variable that called the **emotion** function and then decoded it to turn it into a PHP array. I retrieved each value individually and stored it inside a variable.



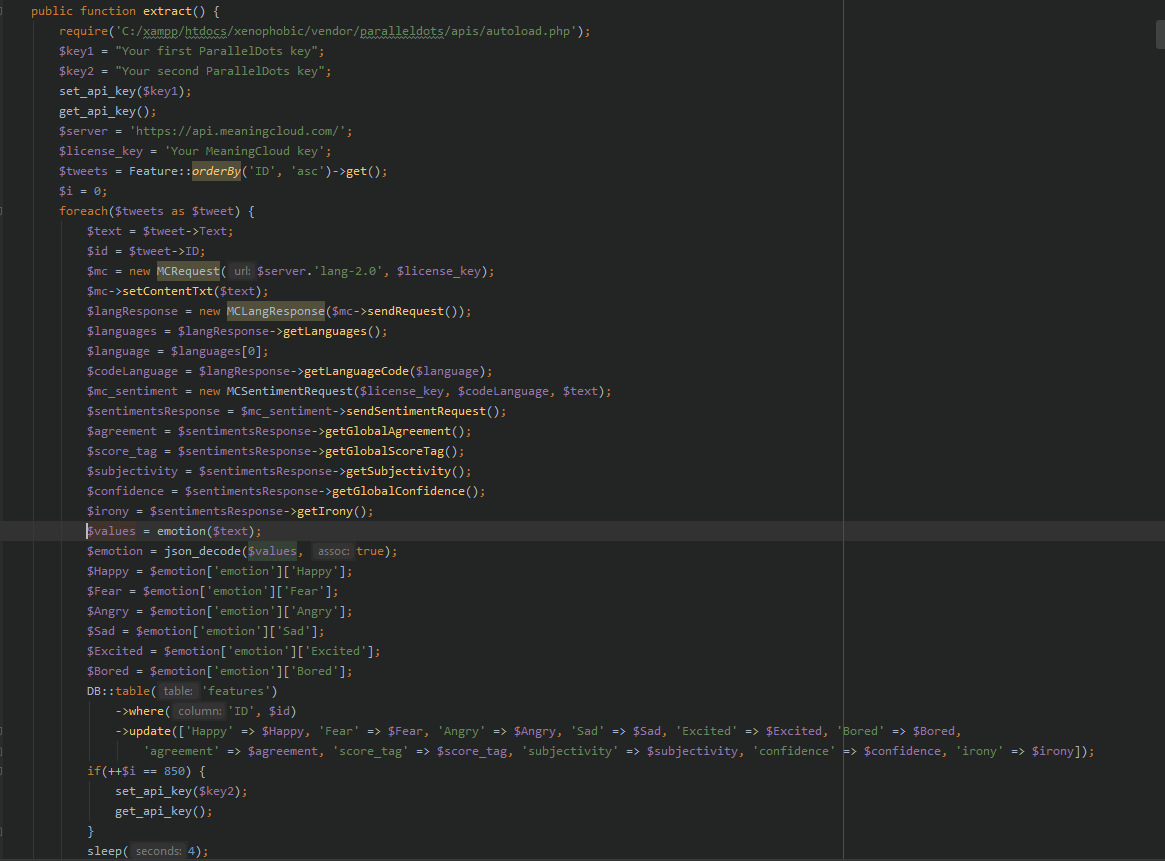
Finally, I made the Query to update the **Feature** object inside the database with the new values.



I added at the end a **sleep(4)** function since we have a 20 requests per minute limit with the ParallelDots API, and I also added an **if** statement that once the counter reaches 850, it will start using the second ParallelDots key, since we are limited to 1000 requests per day.



The complete **extract** function looks like this:



The process is like 1 hour 40 minutes, so I also had to edit my **php.ini** file, setting the **max\_execution\_time** to 10000 seconds.

The features are saved in the database and I exported the table to a file called **features.csv**, also included in the repository.