User Guide

Welcome to the user manual for the Pokémine API!

This API allows you to interact with a Pokémon card database, facilitating access, personal collection management, and deck creation. It is built in Java and uses a MariaDB database for data storage.

Prerequisites

- **Access to API Documentation**: to view a complete overview of all endpoints.
- **MariaDB database access**: to import our database.
- **JDK and IDE**: Requires JDK version 17 or later.
- **Basic Knowledge of RESTful APIs** and familiarity with API request tools, such as Postman or cURL.

General Structure

The Pokémon Card API is organized into endpoints that respond to specific resources and actions. The main categories of endpoints are:

1. **Cards**: for searching, filtering and viewing Pokémon cards.
2. **Users**: for user registration and management.
3. **Collections**: for viewing, adding, and removing cards from collections.
4. **Decks**: for creating, validating and managing Pokémon card decks.

This is a step by step guide on how to setup, install and run our API.

Get ready to play!

Index

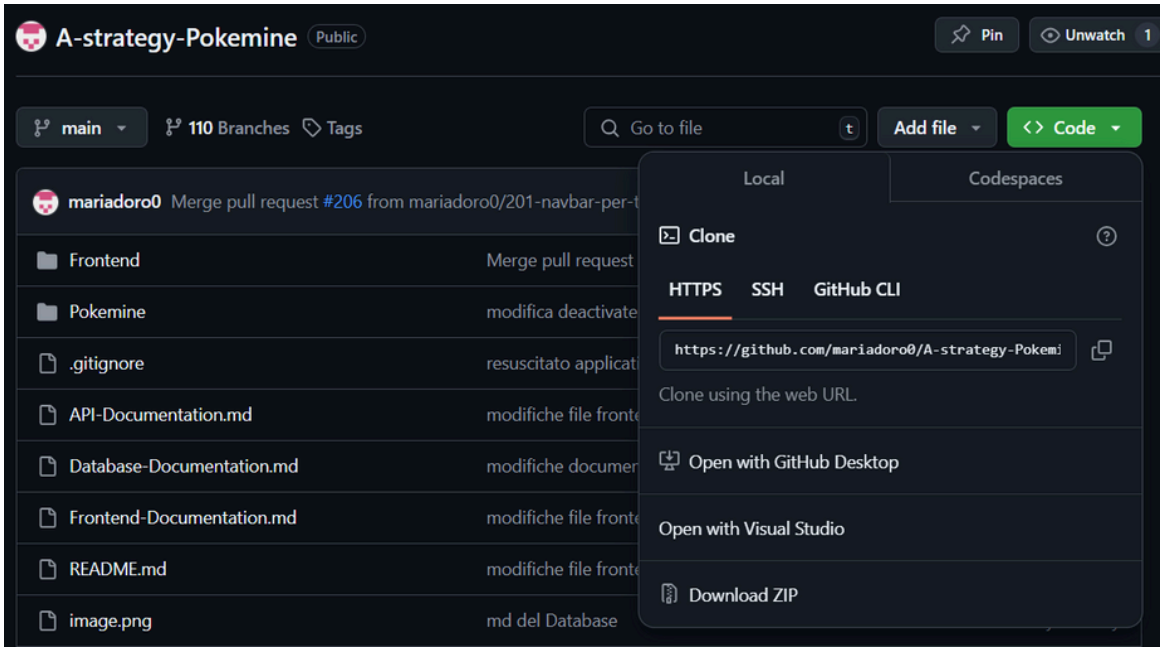
Step 1

Installing the needed technologies

- **Download the Pokémine GitHub Repository**

Go to https://github.com/mariadoro0/A-strategy-Pokemine and download the repository from the “Code” button and save it wherever you like on your computer.

If you download the ZIP version, remember to unzip it or it won’t be accessible.



- **Download the Java JDK 17 or above.**

Go to https://www.oracle.com/java/technologies/downloads/ and download the latest version of the Java Development Kit (JDK) for your architecture.

Note : the installed version must be at least version 17.

Once the download is completed, you can proceed by installing the JDK via the installer.

If you need help or encounter any trouble, here is the official installation guide by Oracle: https://docs.oracle.com/en/java/javase/23/install/overview-jdk-installation.html

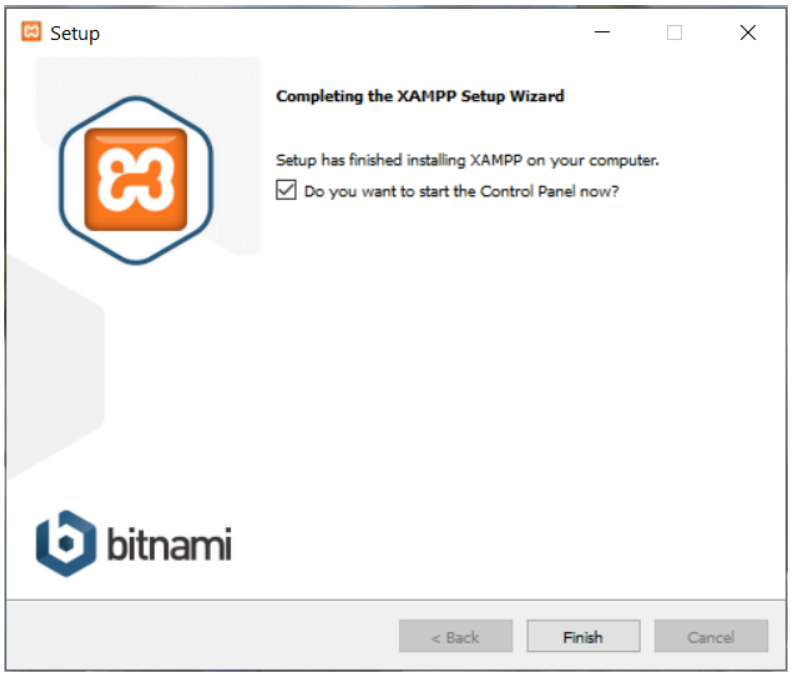
- ## Download XAMPP.

Open your web browser and go to the official XAMPP website (https://www.apachefriends.org/index.html) and choose the latest version of XAMPP compatible with your operating system.

Once the download is completed, run the installer file you downloaded and, when prompted by User Account Control, click Yes to allow the installation.

Follow the installation prompts, choosing components (Apache, MySQL, PHP, etc.) and the installation path as desired. Default components are generally sufficient.

Click Finish to complete the installation.



- **Download Postman.**

Go to the official Postman Download Page (https://www.postman.com/downloads/).

Once the download is completed, you can proceed with the installation process.

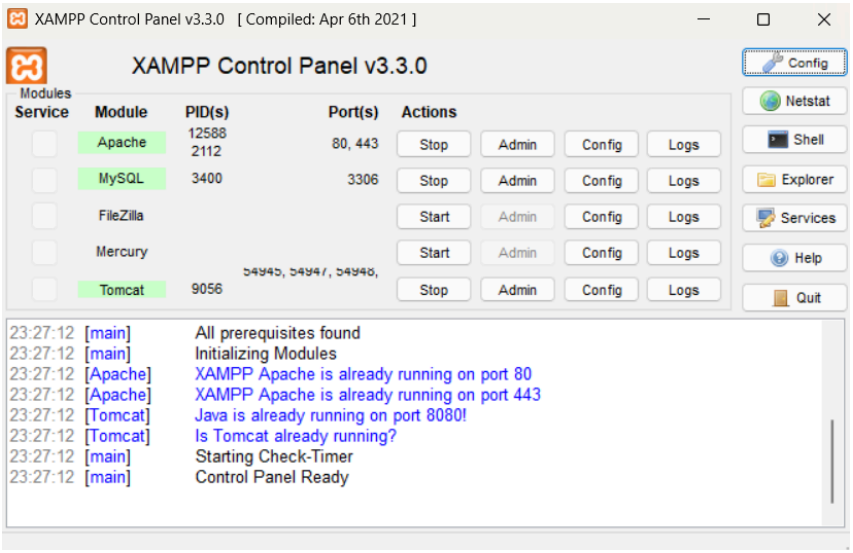
If you are having any trouble during the installation, please check the official Postman Guide (https://learning.postman.com/docs/getting-started/installation/installation-and-updates/)

If you don't want to install Postman on your machine, Postman also offers a web version that only requires a simple sign up to their platform.

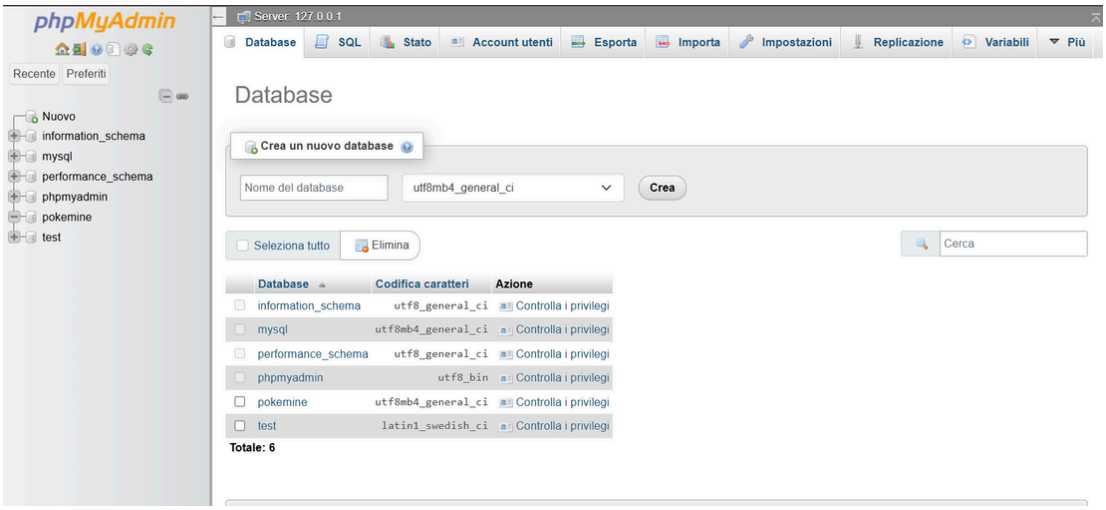
Step 2

Setup the Pokémine Database

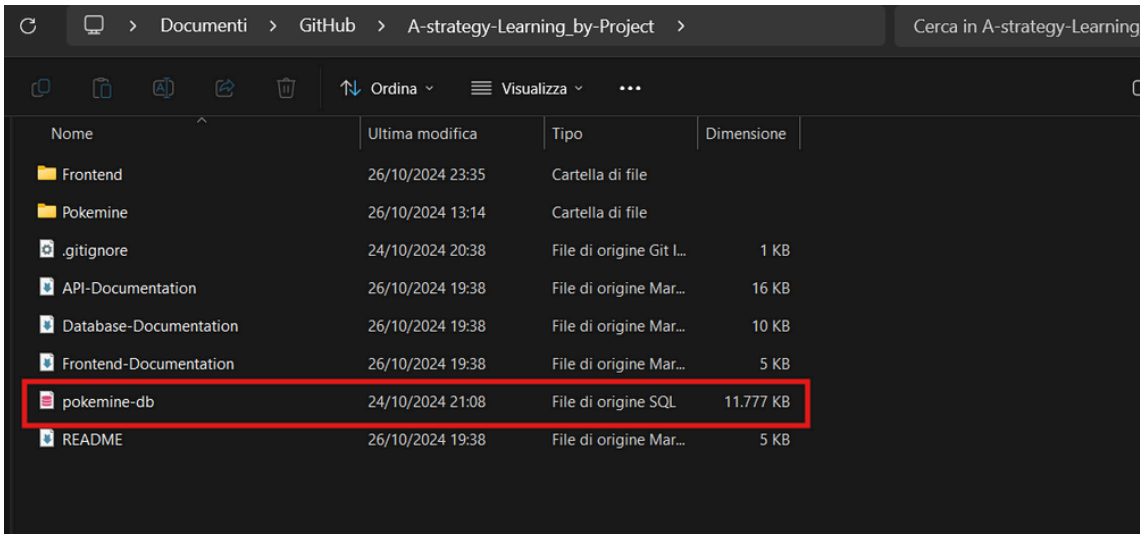
- Open your XAMPP Control Panel and Start the MySQL service. Your panel will look something like this:



- Click on the 'Admin' button near the MySQL instance: this will open the phpMyAdmin Database Management System with the default 'root' user. You will be redirected to a web page like this one:



- Open the unzipped Pokémine repository you earlier downloaded from GitHub: inside of it, you will find a file named 'pokemine-db.sql'.



- Go back to the phpMyAdmin page and click on the 'Import' tab, select the 'pokemine-db.sql' file and click 'Go' to import the database.

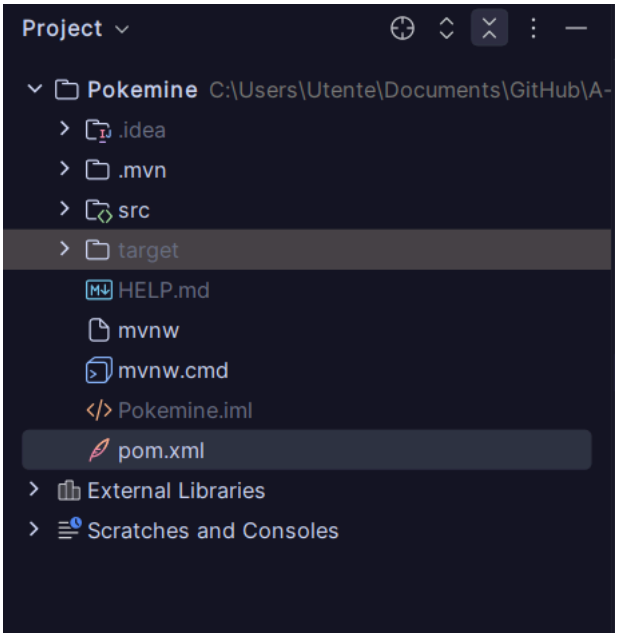
If the operation is successful, you will see the 'pokemine' database in the left menu.

For **further documentation** on our Pokemine Database, please read the documentation we provided on the repository (https://github.com/mariadoro0/A-strategy-Pokemine/blob/main/Database-Documentation.md.)

Step 3

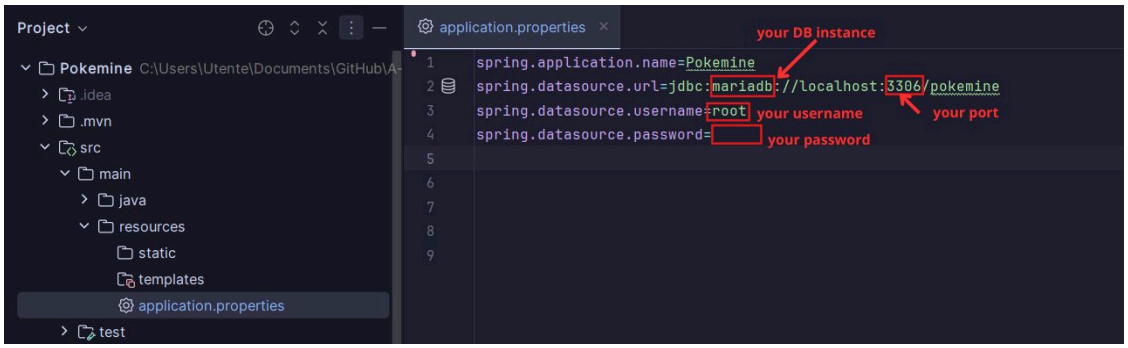
Starting the Pokémine Application

Select your favourite IDE (Eclipse, IntelliJ Idea, Visual Studio Code …) and open the Pokemine folder with it. You will see the project structure in the Project explorer.

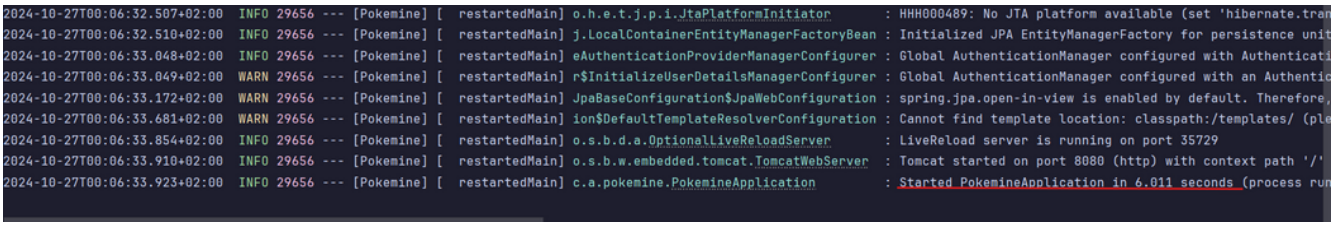


If you set up XAMPP with the default features, you probably won't need to change anything.

However, if you opted for another relational-based DBMS you may need to change the application.properties file in order to run the application correctly.



Once your setting are updated, it's time to run Pokémine: look for the 'Run' button on your interface, and click on it: this will open up a console that will display something like this:



Step 4

Try it with Postman!

Now, if everything is working correctly, we should be able to use Postman to make requests to our API e get responses. There are four main endpoints for the multiple operations we can do, and they are:

- ***/cards*** : main endpoint for the card research
- ***/users*** : main endpoint for the user management
- ***/collection*** : main endpoint for the collection management
- ***/decks*** : main endpoint for the deck management

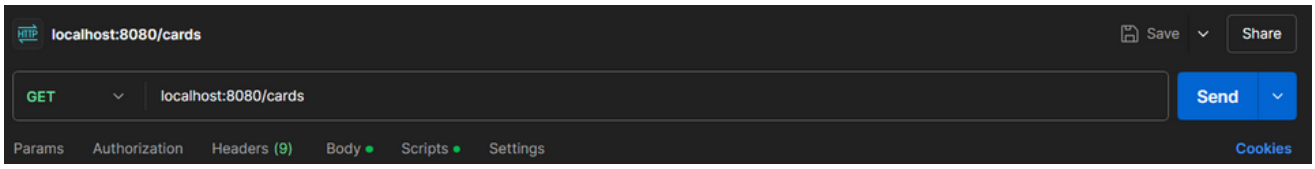
Let's try them all!

If you don't want to test the endpoints one by one, we offer a JSON file that can be Imported into Postman and will download for you our collection of calls: see at the bottom to see how to do that.

- ## Card Research
  - ### Get all cards

    ```
    localhost:8080/cards
    ```

    This endpoint asks for the complete list of cards via GET method. This is how your Postman interface should look like:



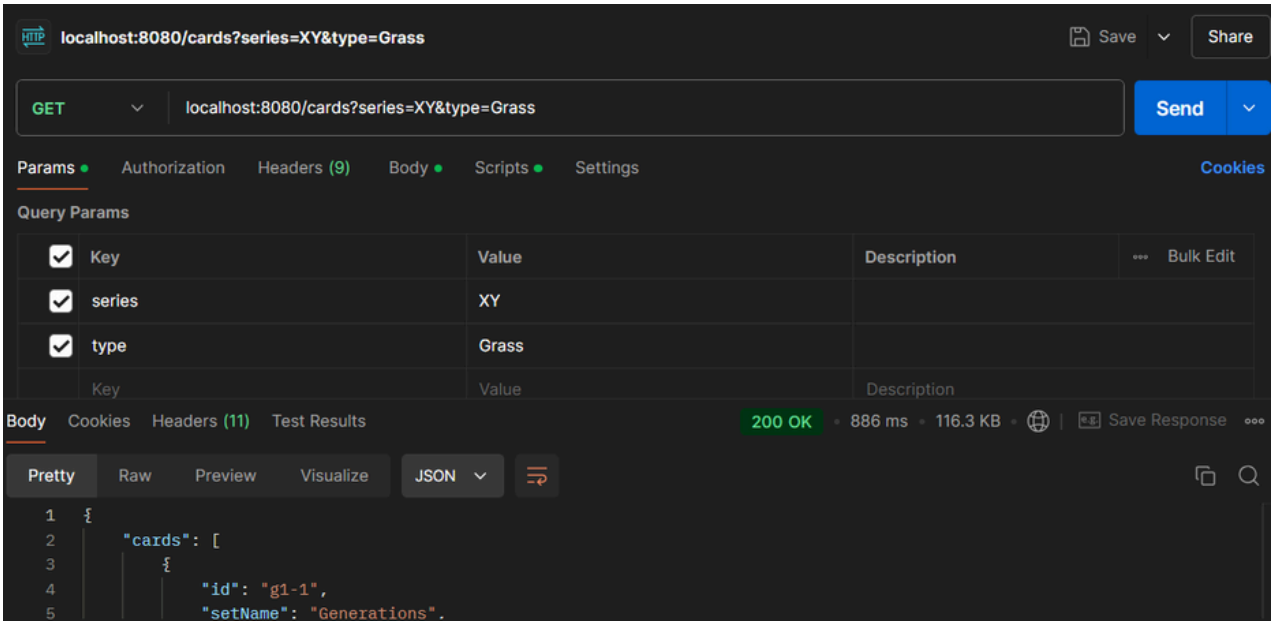
    Clicking on the 'Send' button, you will receive a list of cards in JSON format. At the end of the response body, you will also visualiza how many pages of results the call obtained, how many cards in total and the current page you are receiving: each page contains approximately 100 cards.

This endpoint can receive multiple parameters in order to filter cards based on:

- id
- name
- type
- subtype
- artist
- series
- set
- generation
- rarity

You can specify your chosen parameters in the Postman 'Params' section.

For example, if I want to look for all the cards of the series 'XY' whose Pokémon is of type Grass, my interface will look like this:



The '200 OK' is a signal for you that the server correctly interpreted your request and was able to give you a proper response.

You can now experiment with all the parameters previously listed!

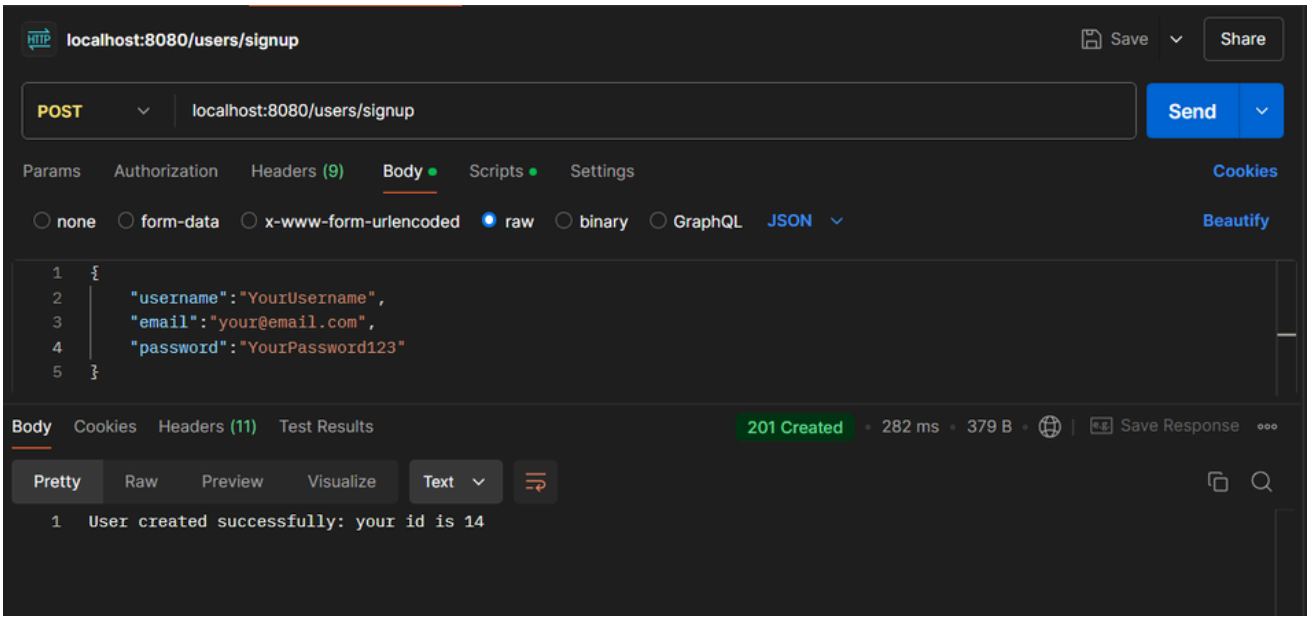
- # User Management
  - ## Registration

    `localhost:8080/users/signup`

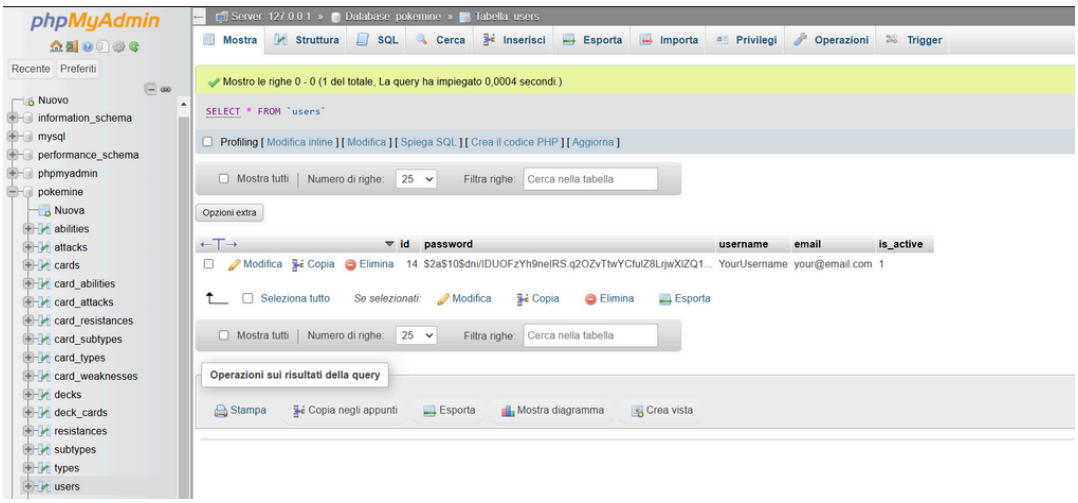
Now, it's time to create our account on Pokémine: to do so, we will just need a unique username, our email and a password. This call is made via POST method and will need a Body containing

the user information previously mentioned.

Your request and consequent response should look like this:



For further confirmation of the user being created, you can open your DBMS and check the users table, that should contain all the information we just sent:



As you can see, the password is encrypted for safety reasons, so don't forget it!

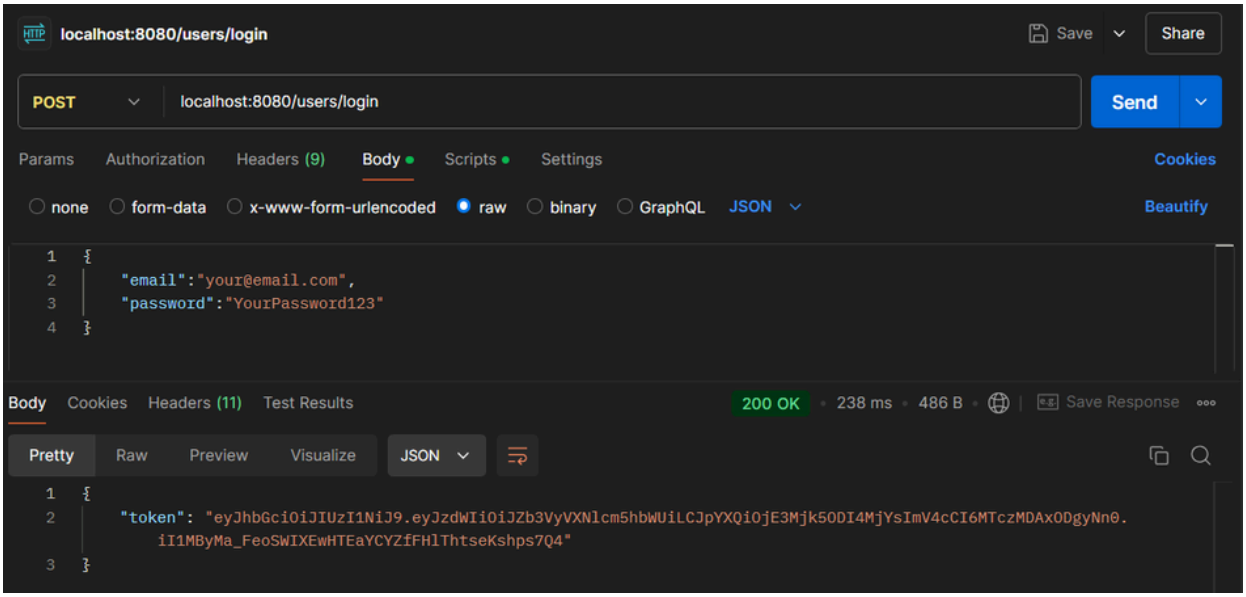
- ## Login

  `localhost:8080/users/login`

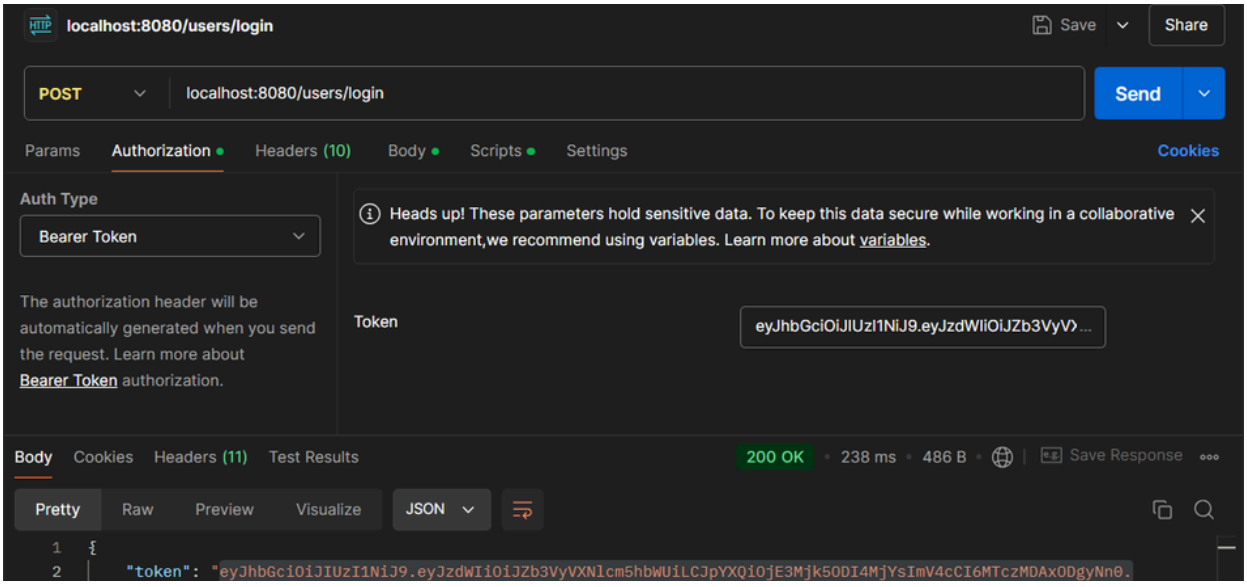
Once the registration is completed, you will be able to log into your account in order to access your collection and your decks. The login operation requires your email and password in order to authenticate you. Via POST, we will send in the request body our email and password and will receive a JWT token in response.

We will use this token from now on to access following requests that would otherwise normally be forbidden.

Let's start by logging in:



Now, we need to take the token and change our Authorization method by clicking on the tab and selecting 'Bearer token', in order to paste it in the text box.



WARNING

From now on, every call, besides the ones mentioned before, will need this token in order to be sent, so always make sure this Authorization settings always match.

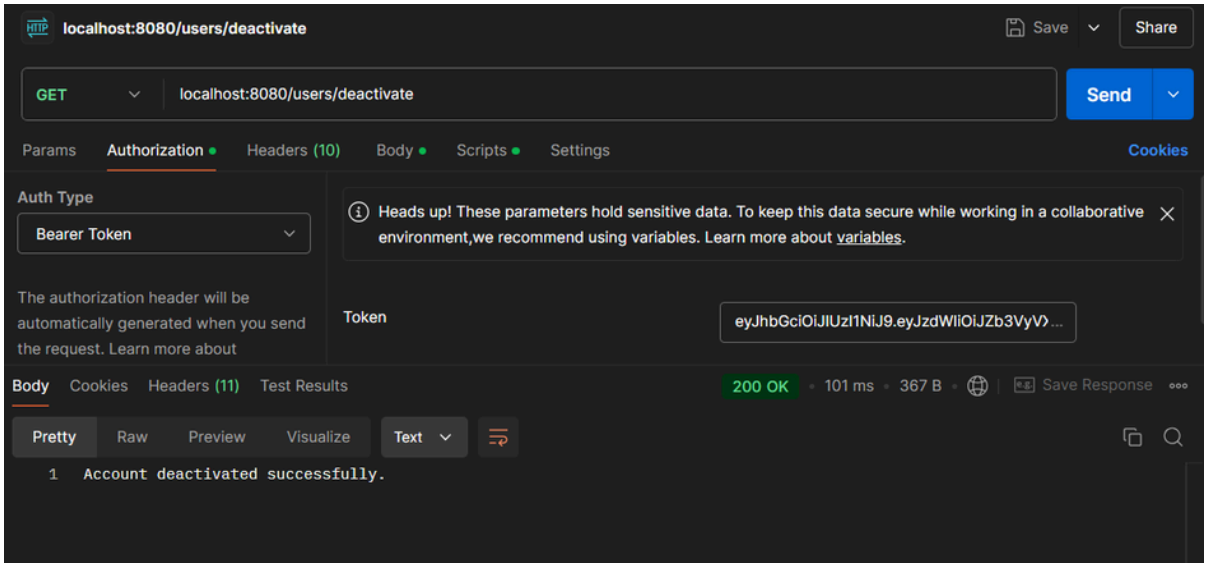
The token expires in one hour, so you will need to log in again.

- ## Deactivate account

  `localhost:8080/users/deactivate`

If you plan on deleting your account, but you don't want to lose your collection and your decks, you can deactivate it: you will still be able to visualize them but not to modify them.

This request is made via GET.

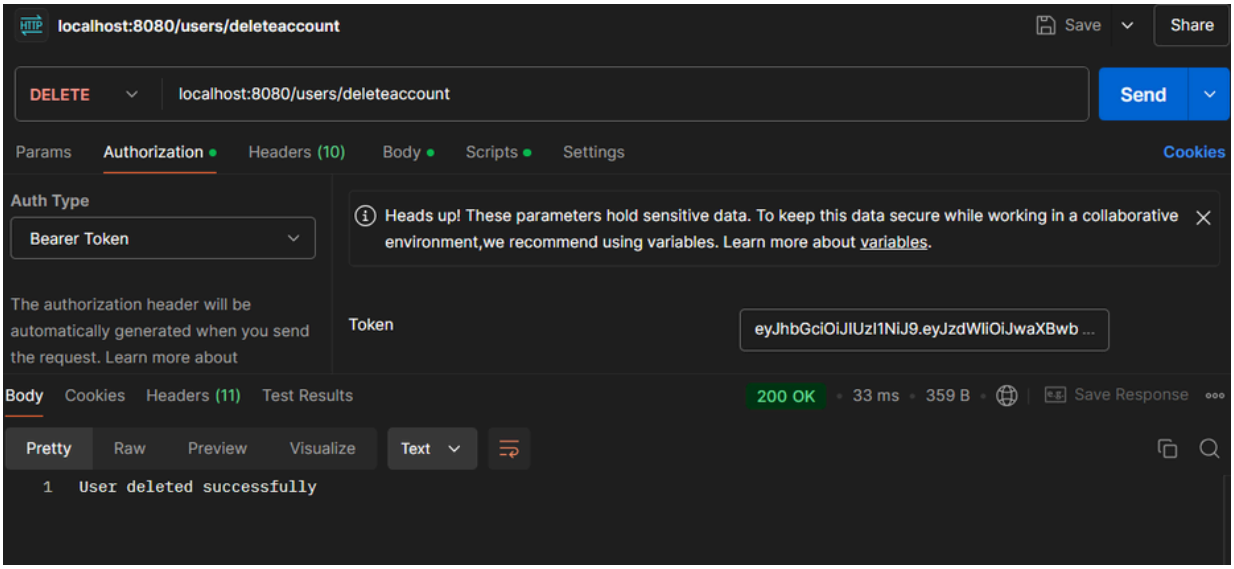


- ## Delete account

  `localhost:8080/users/deactivate`

However, if you want to permanently delete your account and all the information attached to it, you can call this endpoint via DELETE.

> **WARNING**
> **This action is irreversible!**

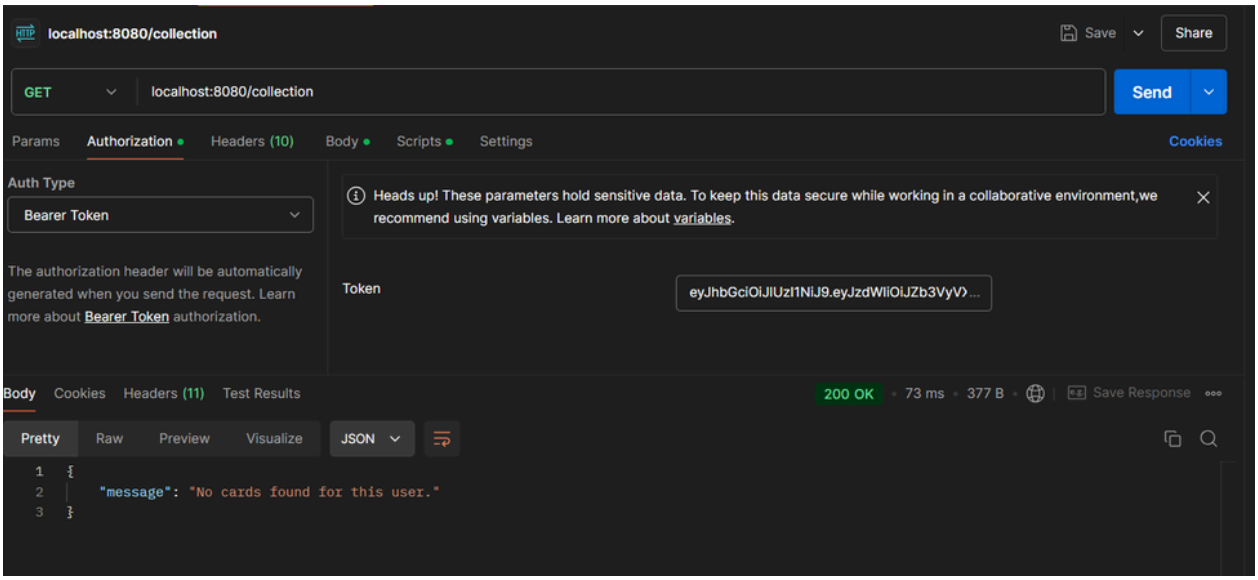


- **Collection Management**
  - **Display you collection**

```
localhost:8080/collection
```

This call enables you to visualize the content of your collection via GET.

If you don’t have any card in your collection, your output will look like this:



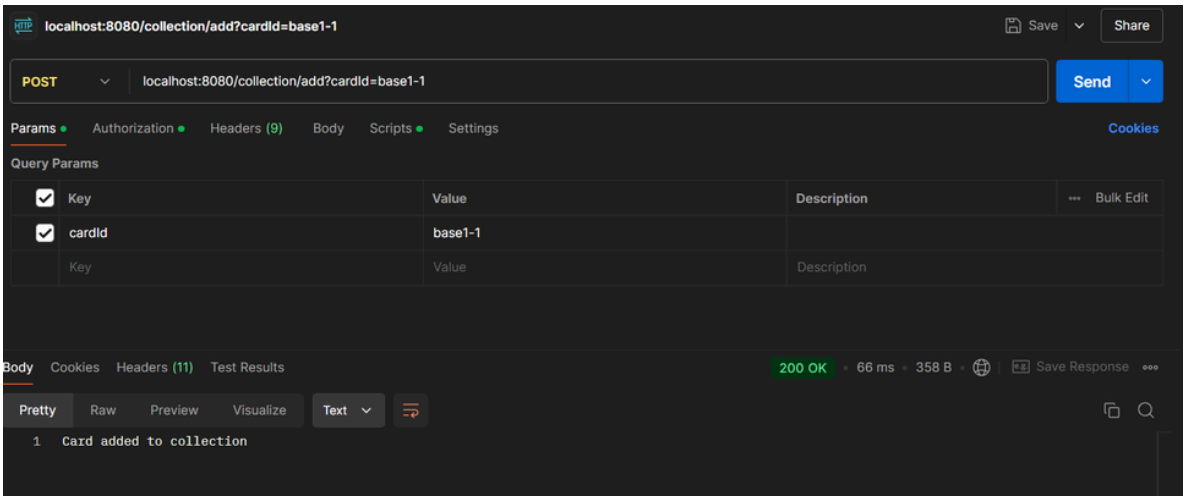
I will now explain to you how to add a card to your collection:

  - **Add a card to your collection**

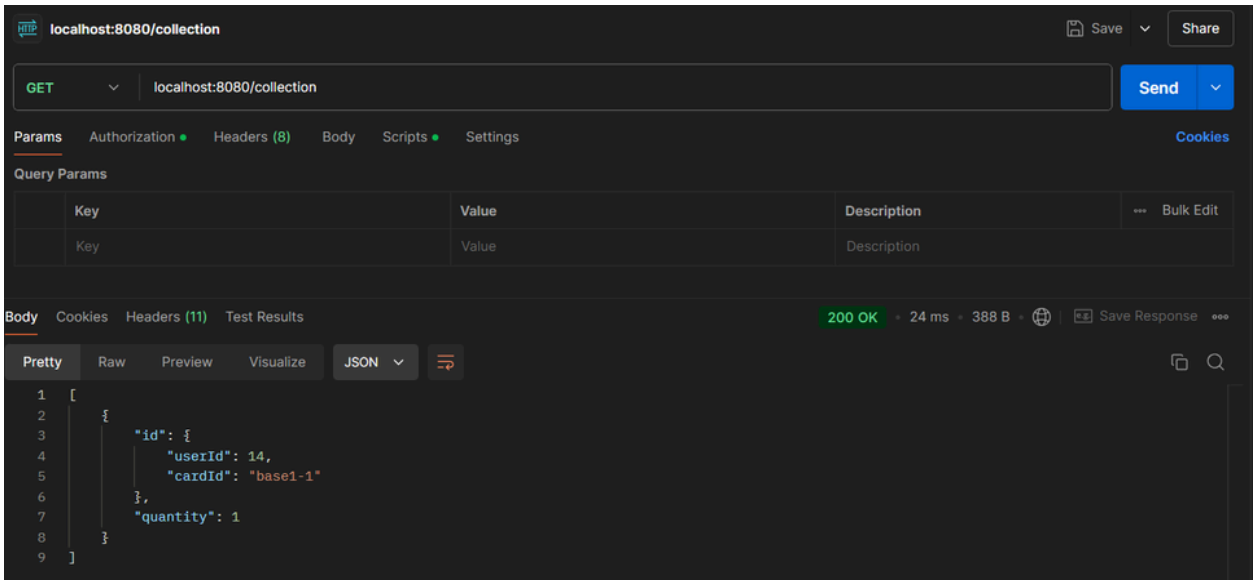
```
localhost:8080/collection/add
```

This endpoint will give you the opportunity to add a specific card by its id: every card has a unique id that you can retrieve from the ‘/cards’ research endpoint. This request is made via POST method.

Let’say we want to add the ‘base1-1’ Alakazam Base card. All we have to do is set the cardId parameter and send our request!



If we you check your collection now, with the previous endpoint, you will see the card you just added appear in your collection:



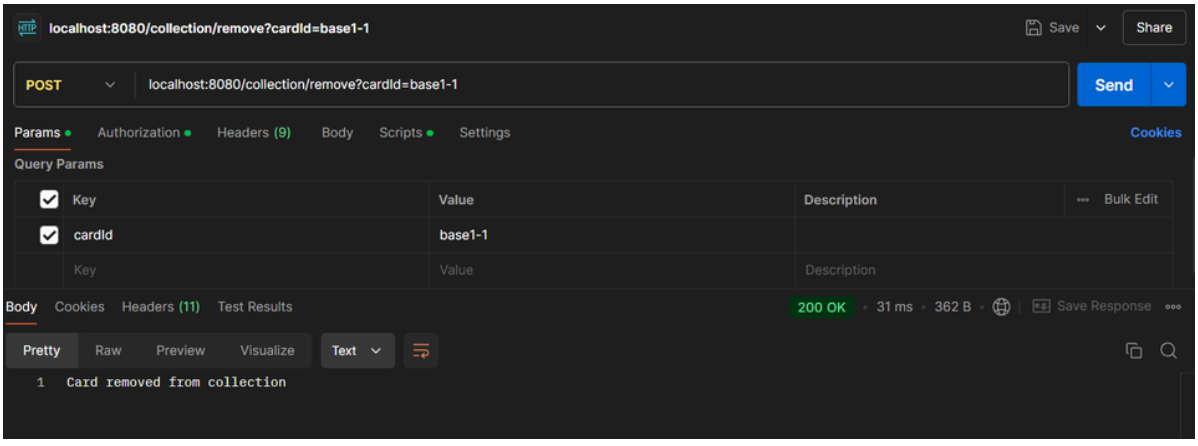
The card will be added one by one, increasing the quantity of your copies owned.

- ## Remove a card from your collection

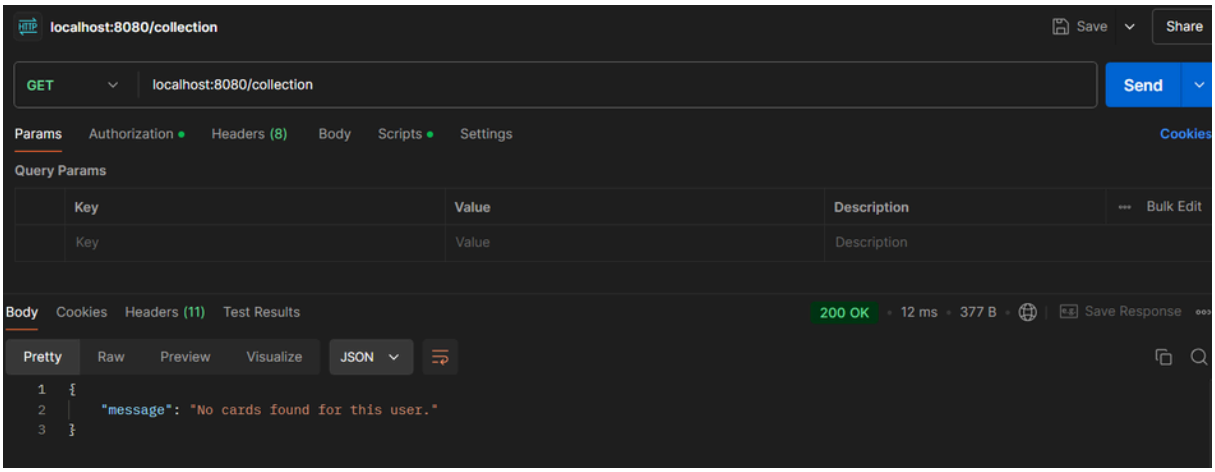
  `localhost:8080/collection/remove`

This endpoint will give you the opportunity to remove a specific card by its id: every card has a unique id that you can retrieve from the '/cards' research endpoint. This request is made via POST method.

Let's remove the card we just added by setting the cardId parameter to 'base1-1'.



Now that the card is removed, our collection will be empty again, so if we call it, we should see something like this again:



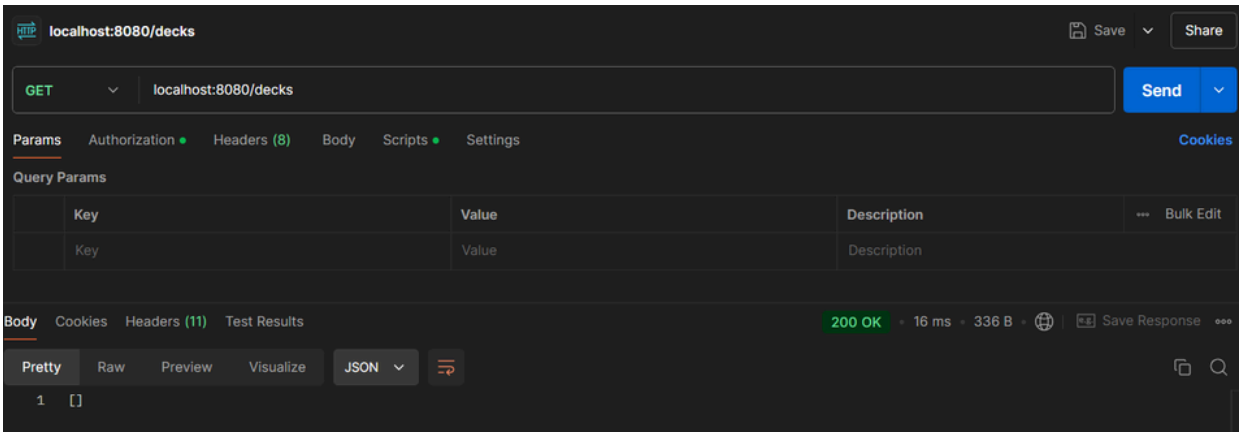
Now you have all the tools to manage your collection!

- # Deck Management
  - ## Display your decks

    `localhost:8080/decks`

By calling this endpoint, you can see all the decks you have ever created. This request is made via GET.

At the beginning, it will look empty, as you don't have decks. Your Postman interface will look like this:

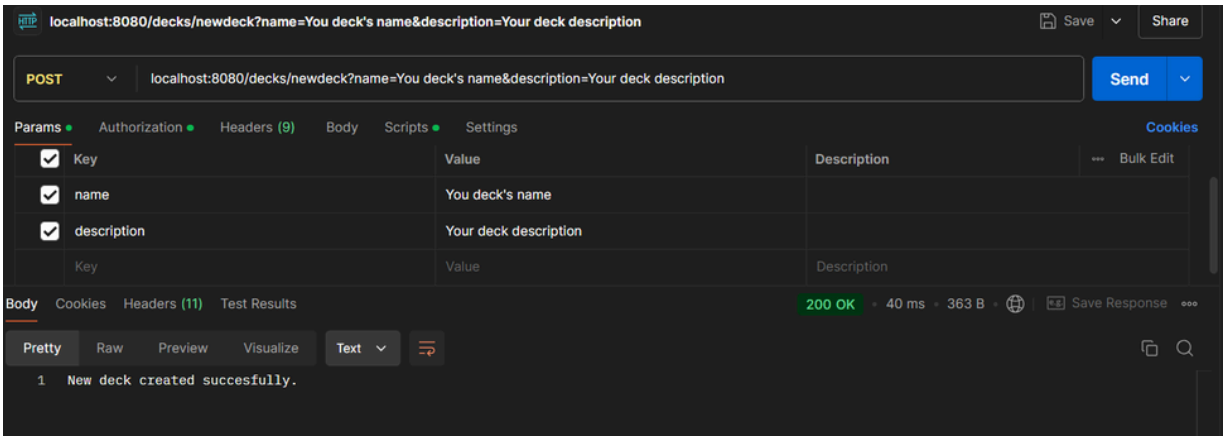


As you can see, your set of decks is empty, so let's create a new one!

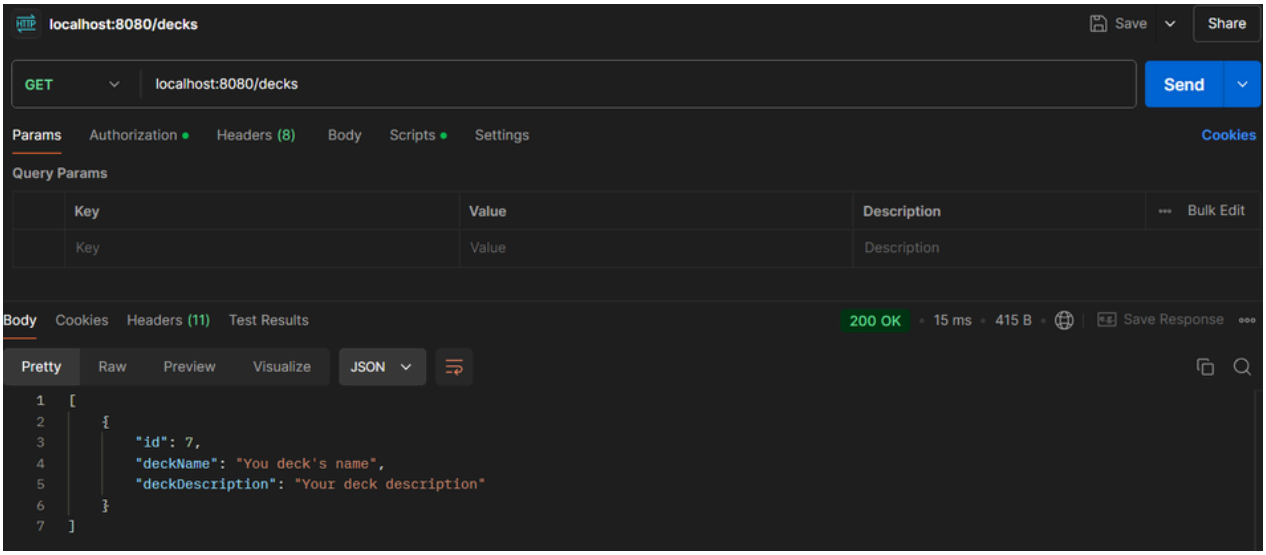
  - ## Create a new deck

    `localhost:8080/decks/newdeck`

To create a new deck, you will have to set the request method to POST and insert in the Parameter tab the name of your new deck and its description as parameters, just like this:



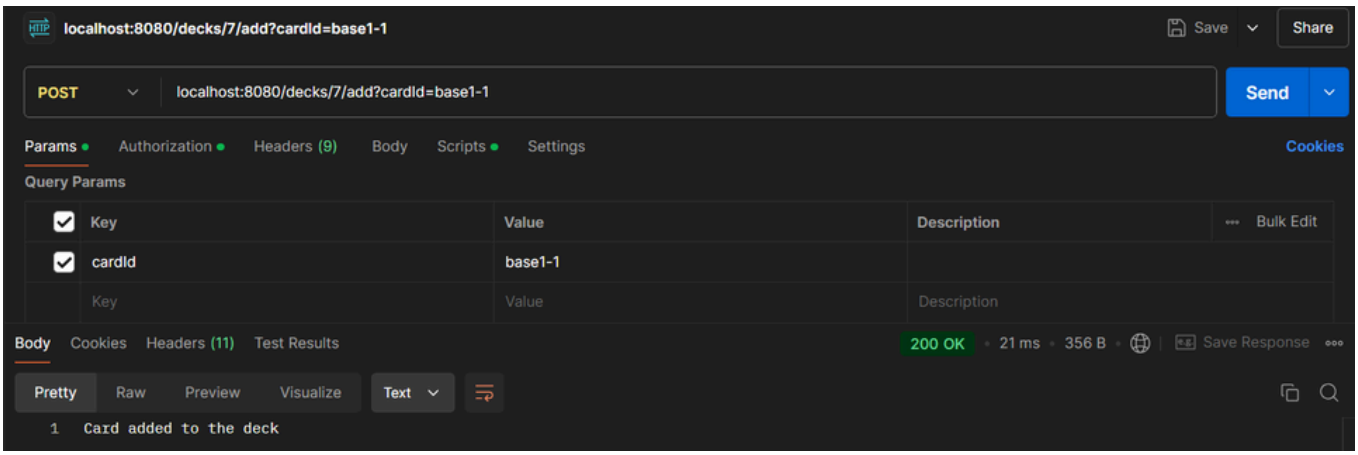
Once the request is sent, your new deck will be created and you will visualize it in your deck collection with the previous call. It will be something like this:



- ## Display of a single deck

  `localhost:8080/decks/{deckId}`

To see the content of a specific deck, you will need to know its id and write it in the path where {deckId} is. We previously created our new deck with id 7, so our new GET request will be:



As you can see, the deck is now empty, as we haven't added any cards to it, so let's populate our deck!

○ Adding a card to the deck

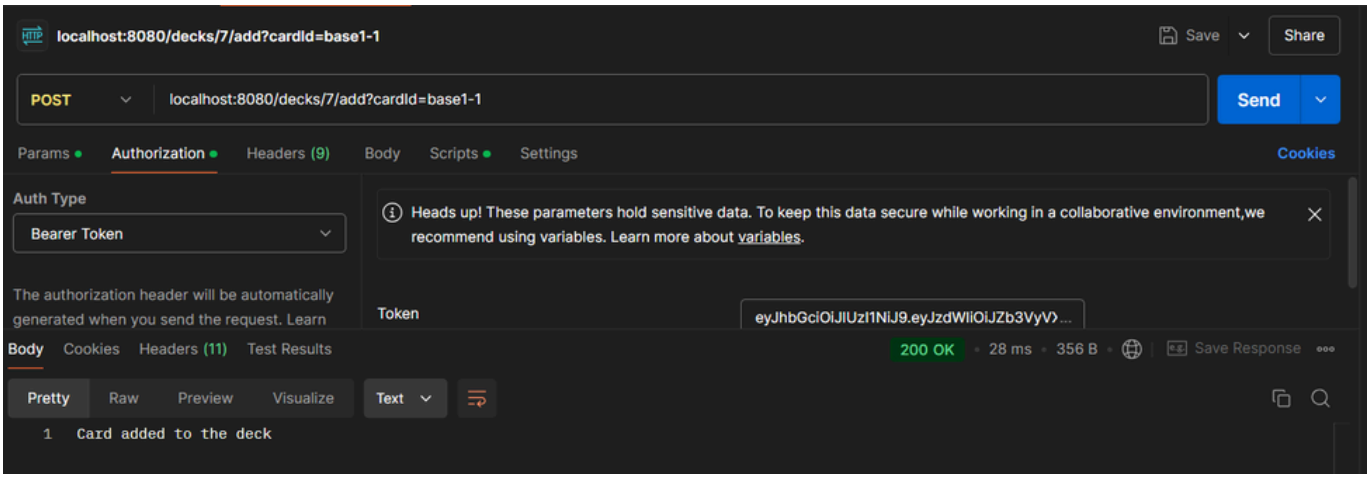
```
localhost:8080/decks/{deckId}/add
```

Similarly as the collection's add, we will need to call via POST this endpoint to add a new card, specifying the id of the deck we want to update in the path, and writing the card id in the parameters section.

> **Note**
> You will only be able to add a card to your deck only if you have it in your collection, so make sure u own the card.
>
> You can add a maximum of 4 copies of the same card in the deck, as per the Pokémon TCG rules.

Let's try adding that 'base1-1' card we already own!

If you followed all the steps, you may have removed it when we tested the collection features, so try adding it again to your collection if this doesn't work as below:

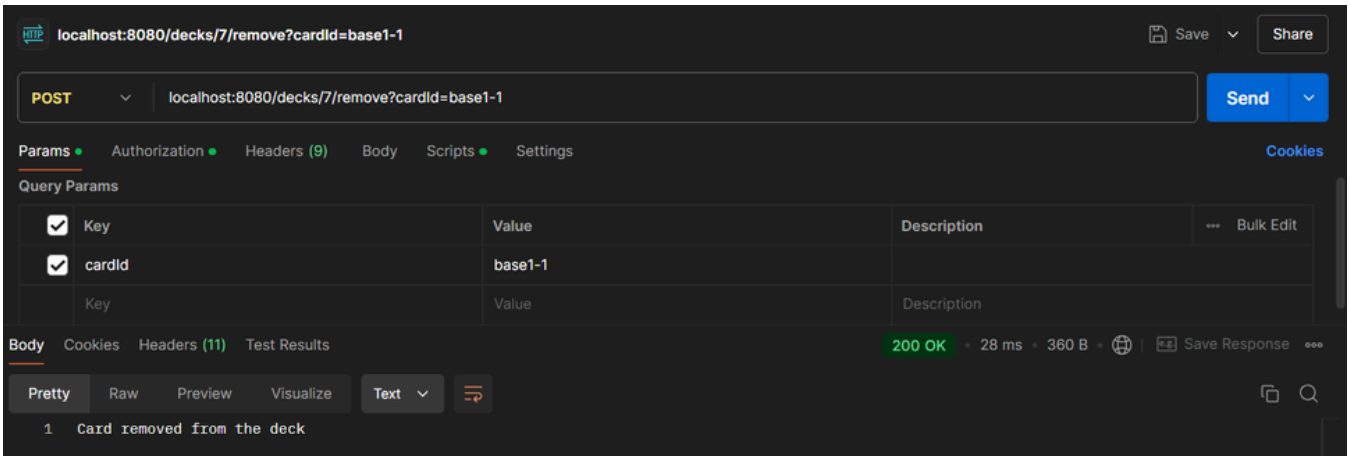


Once the card is added, we can now see a list of the cards we have in the deck by the previous display request.

○ Removing a card from the deck

```
localhost:8080/decks/{deckId}/remove
```

Similarly as the collection's remove, we will need to call via POST this endpoint to remove a card, specifying the id of the deck we want to update in the path, and writing the card id in the parameters section.



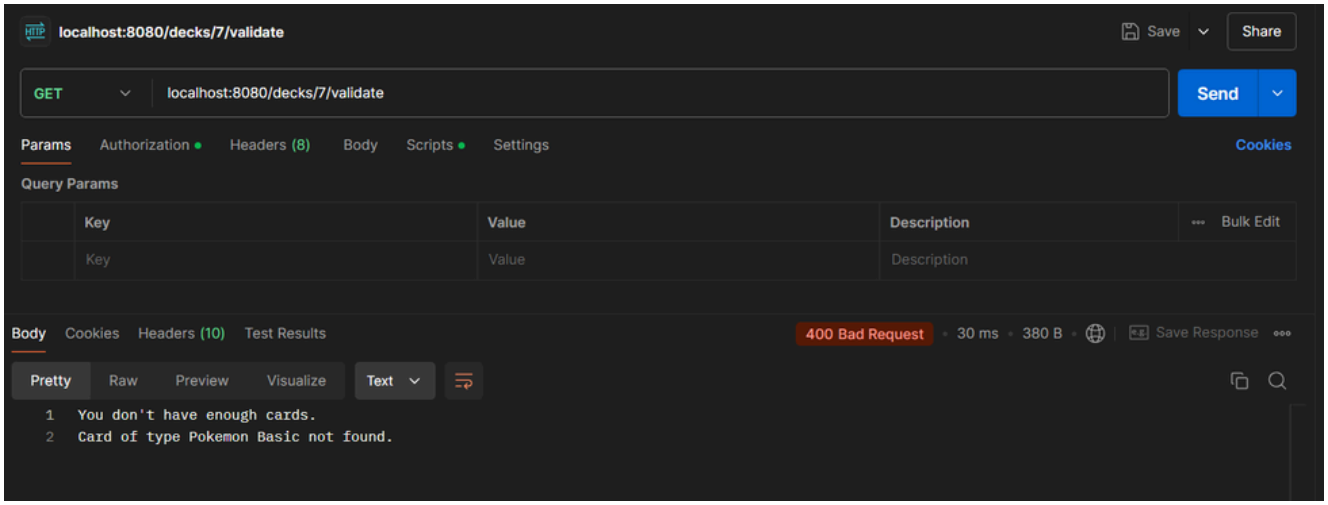
- ## Validating the deck

```
localhost:8080/decks/{deckId}/validate
```

To check if your deck follows the TCG rules, you can always call the validation endpoint via GET that will show you what the deck is missing to be valid.

To be valid, your deck should contain:

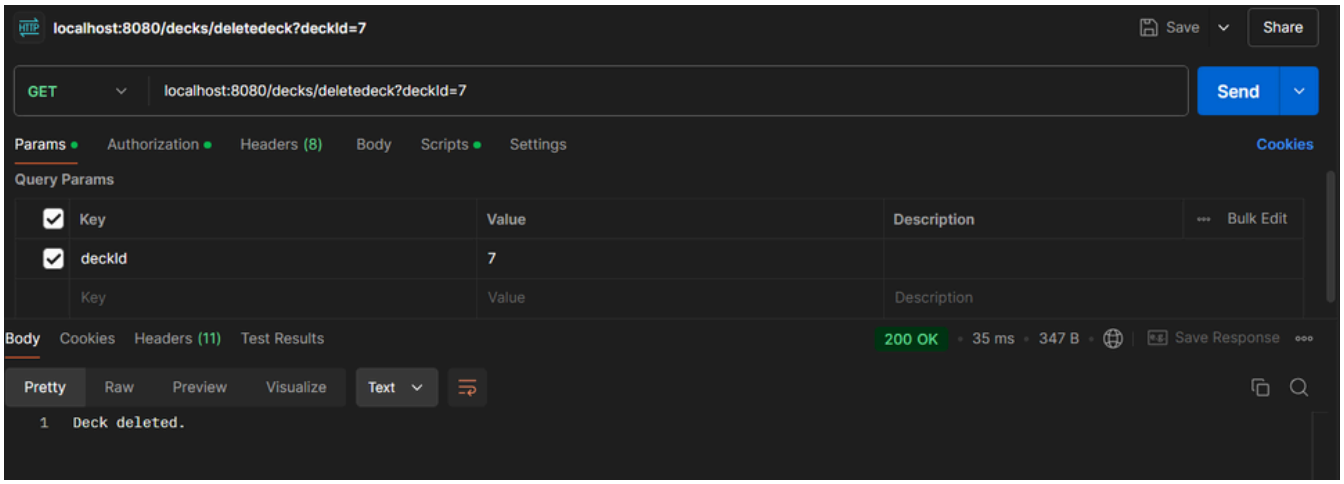
- exactly 60 cards
- at least one Basic Pokémon card
- maximum of 4 copies of a card



- ## Deleting the deck

```
localhost:8080/decks/deletedeck
```

If you want to delete a deck, you will have to call this endpoint via POST and specify the deckId of the deck you want to delete in the Params area. Let's try deleting the deck we just created:



And that's about it! Have fun playing Pokémine!

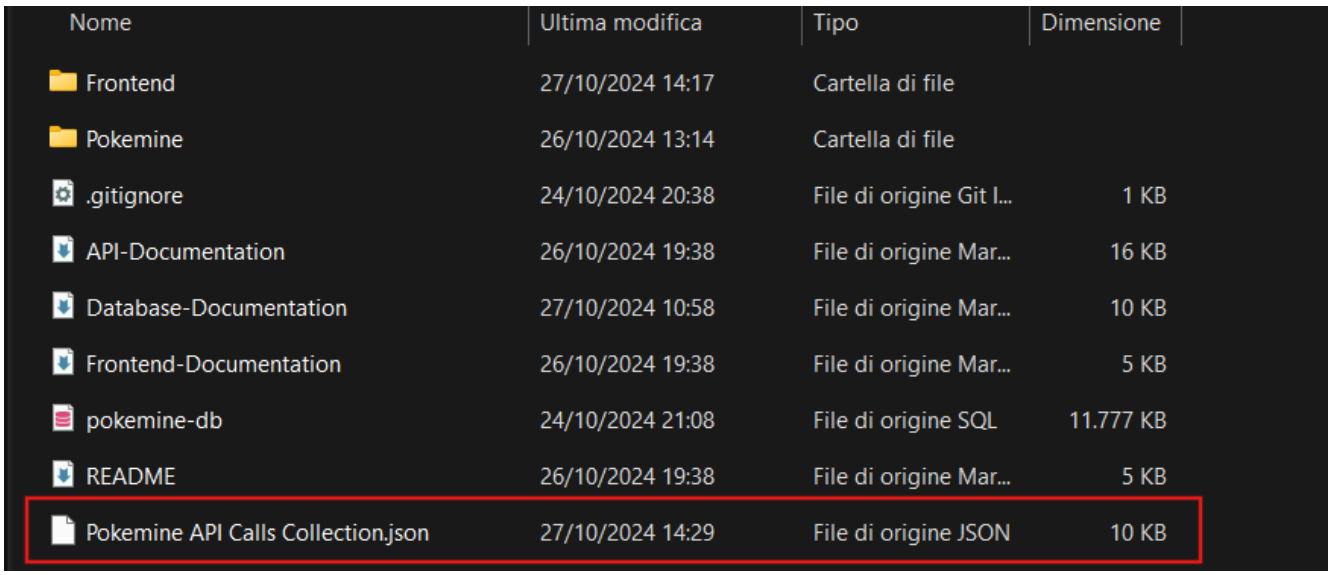
For further technical documentation on how our Pokémine API works, please read the documentation we offer on our repository (https://github.com/mariadoro0/A-strategy-Pokemine/blob/main/API-Documentation.md).

Optional Step:

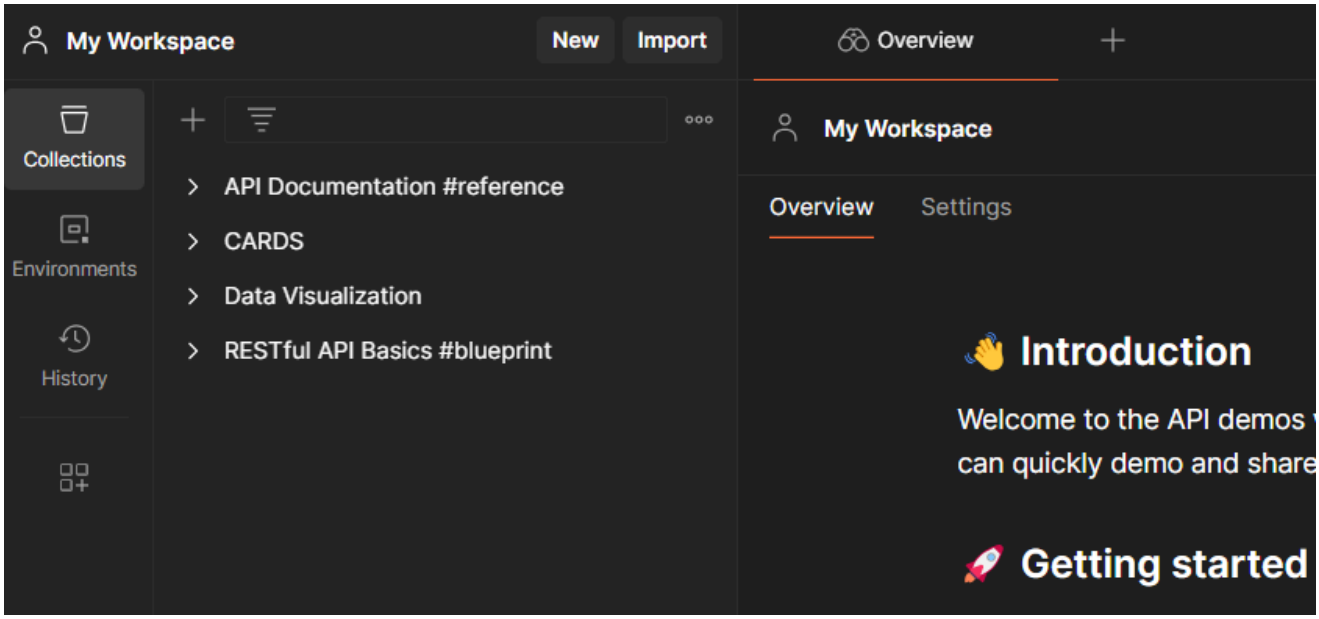
Import our call collection on Postman

To import our call collection on Postman, in order to try our API without writing all the calls by hand, you'll have to follow these steps:

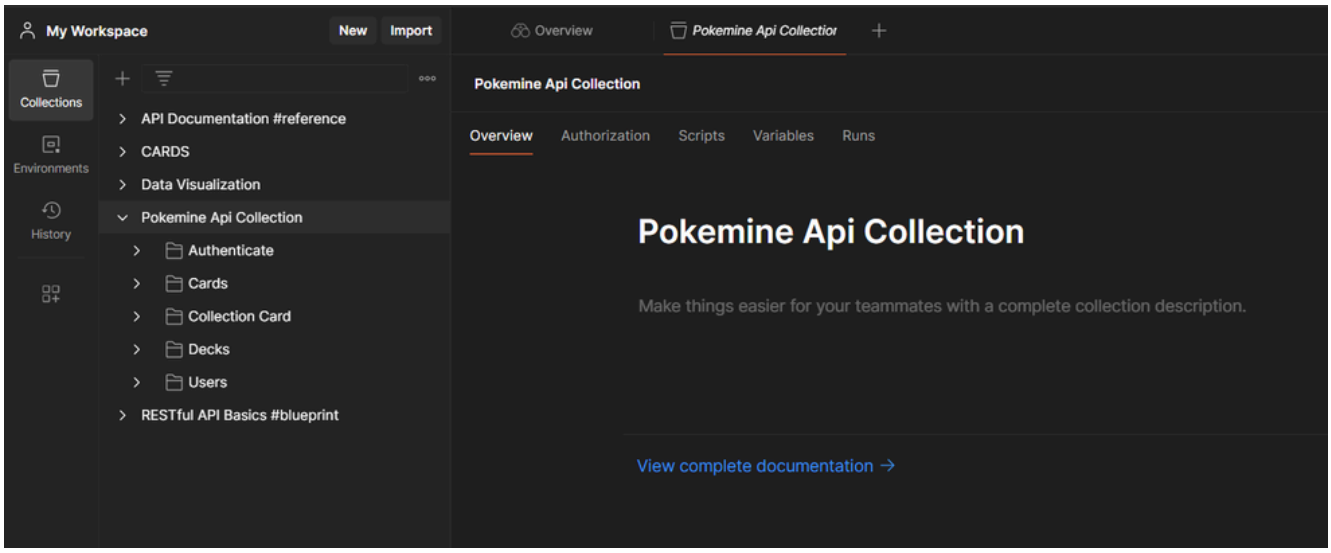
- Locate the Pokemine API Calls Collection.json file on your local rerpository.



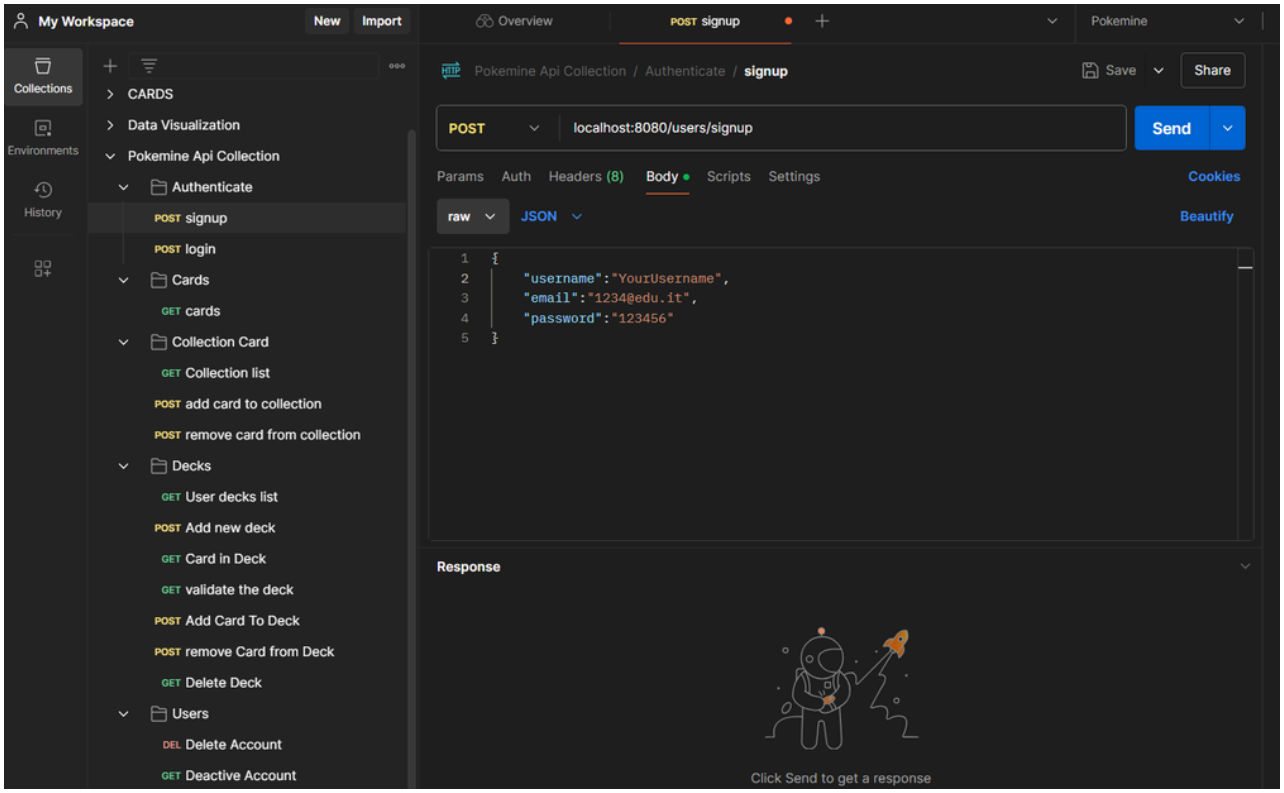
- Open up Postman, go on your Workspace and select Collections



- Click on the 'Import' tab, open the File Explorer and select the .json file. This will import our call collection in your collection, resulting in your collection to look like this:



- Opening the various folder, you will see the single calls, that you can use and modify as you like!



- You are all set, **have fun**!