API offices

Settings api office

Clone proyect from :

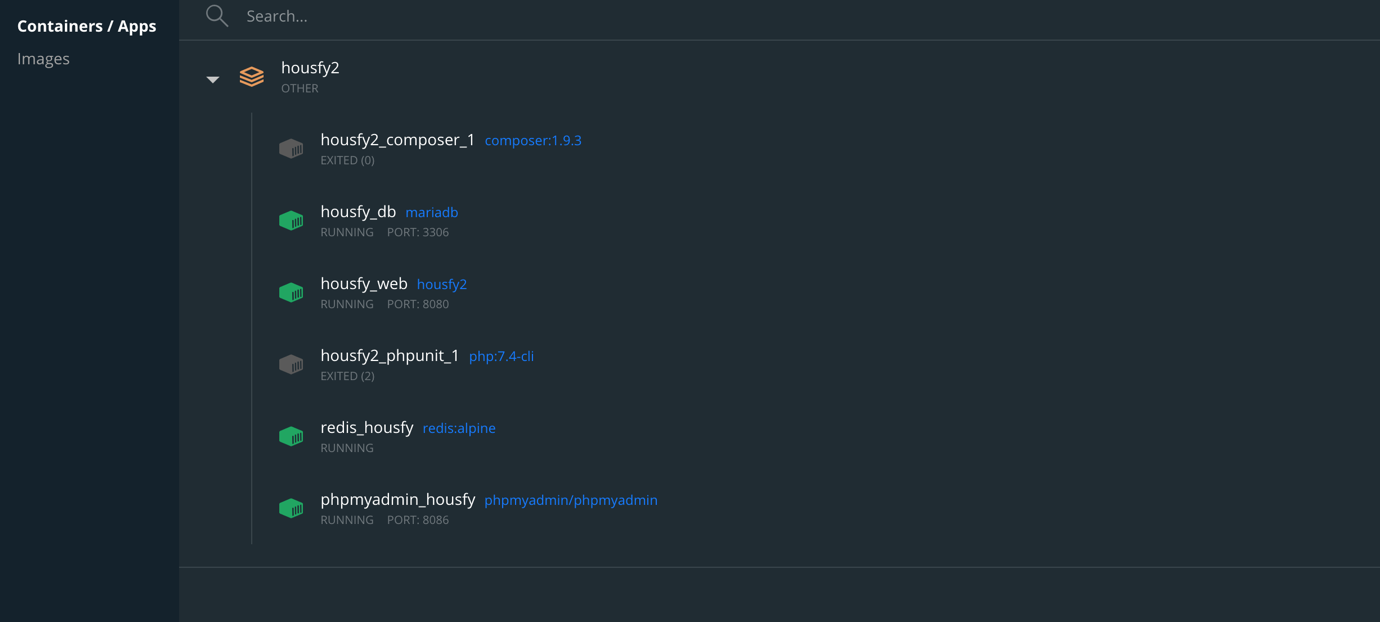
<https://github.com/felipemorastorm/housfyTest.git>

One time proyect download run via terminal :

Docker-compose up -d

This start services image housfy2 that have the follow container :

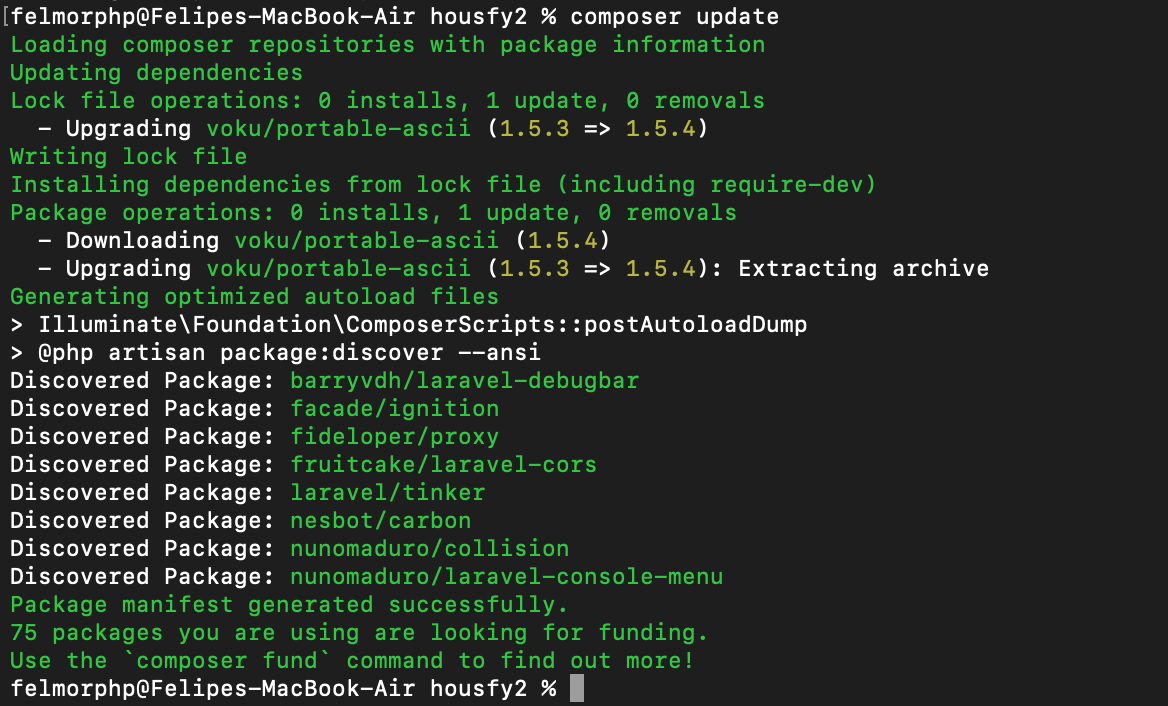
PHP apache 7.4 / mysql-mariadb / phpMyAdmin / redis / composer /PHP unit ->Cli interpreter 7.4



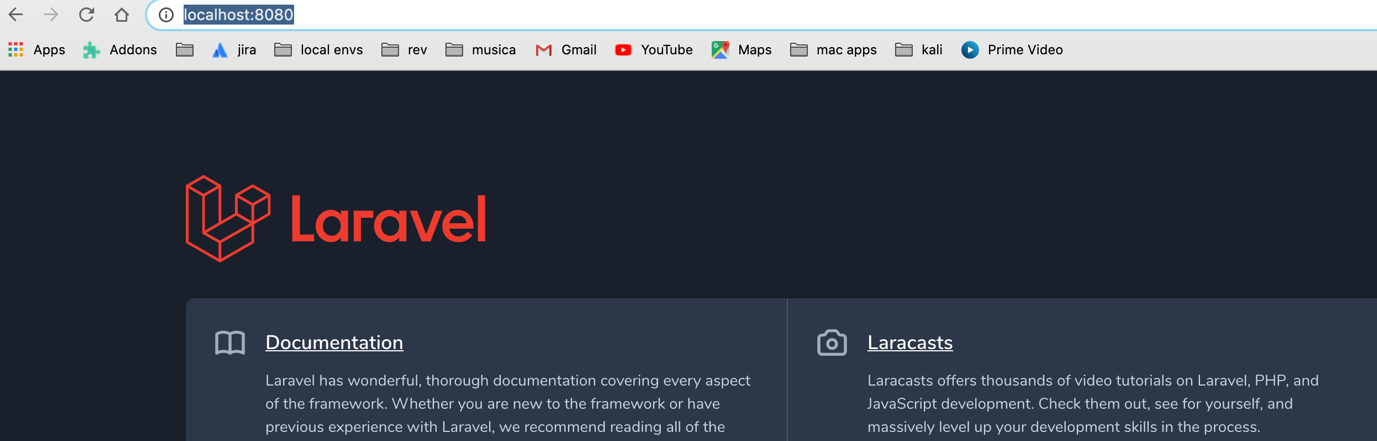
One time all services are running correctly we update all ibrary of Laravel proyect with:

composer update

This will update all Laravel library dependences in composer.json file pushing updates to vendor folder , await that update process arrive to end.



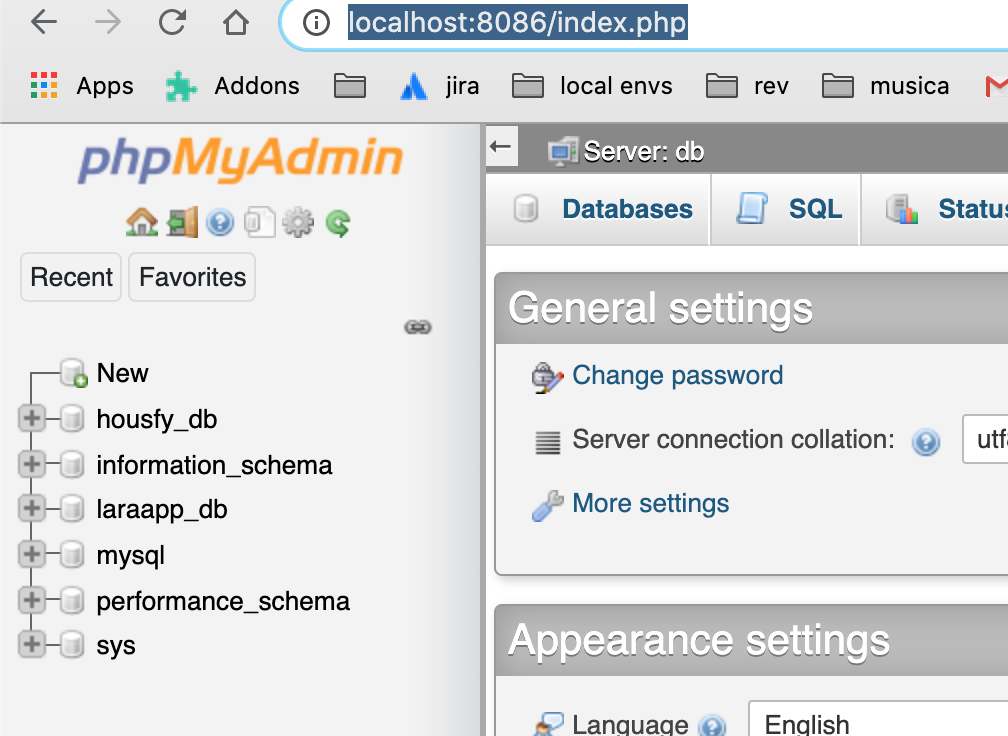
If all ok Laravel is preinstallated in this proyect in with the update of library the home page must be showed via browser : http://localhost:8080/



For this proyect a database has been created in first execution : housfy\_db how is the main schema database , for verify that this table has been created correctly we can access via phpMyAdmin on : <http://localhost:8086/index.php> with

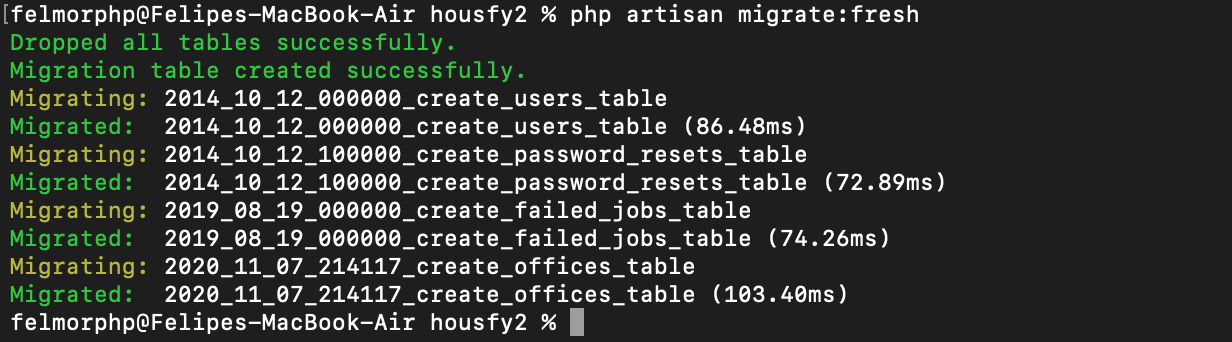
username :root

password :



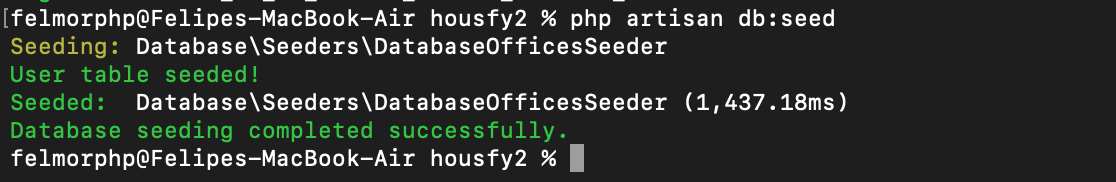
One time verified that schema is there we can begin to launch the migrations from create all tables needed from this proyect and later populate , for this we execute terminal (over the folder path route where proyect have been cloned ) the command :

php artisan migrate:fresh

- 

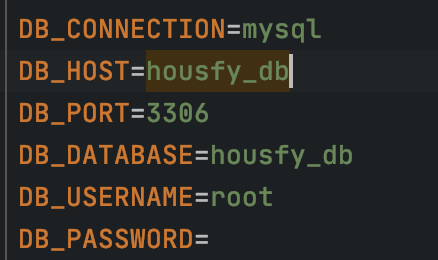
This create the default tables for application , now for populate the tables launch the command seeder :

php artisan db:seed

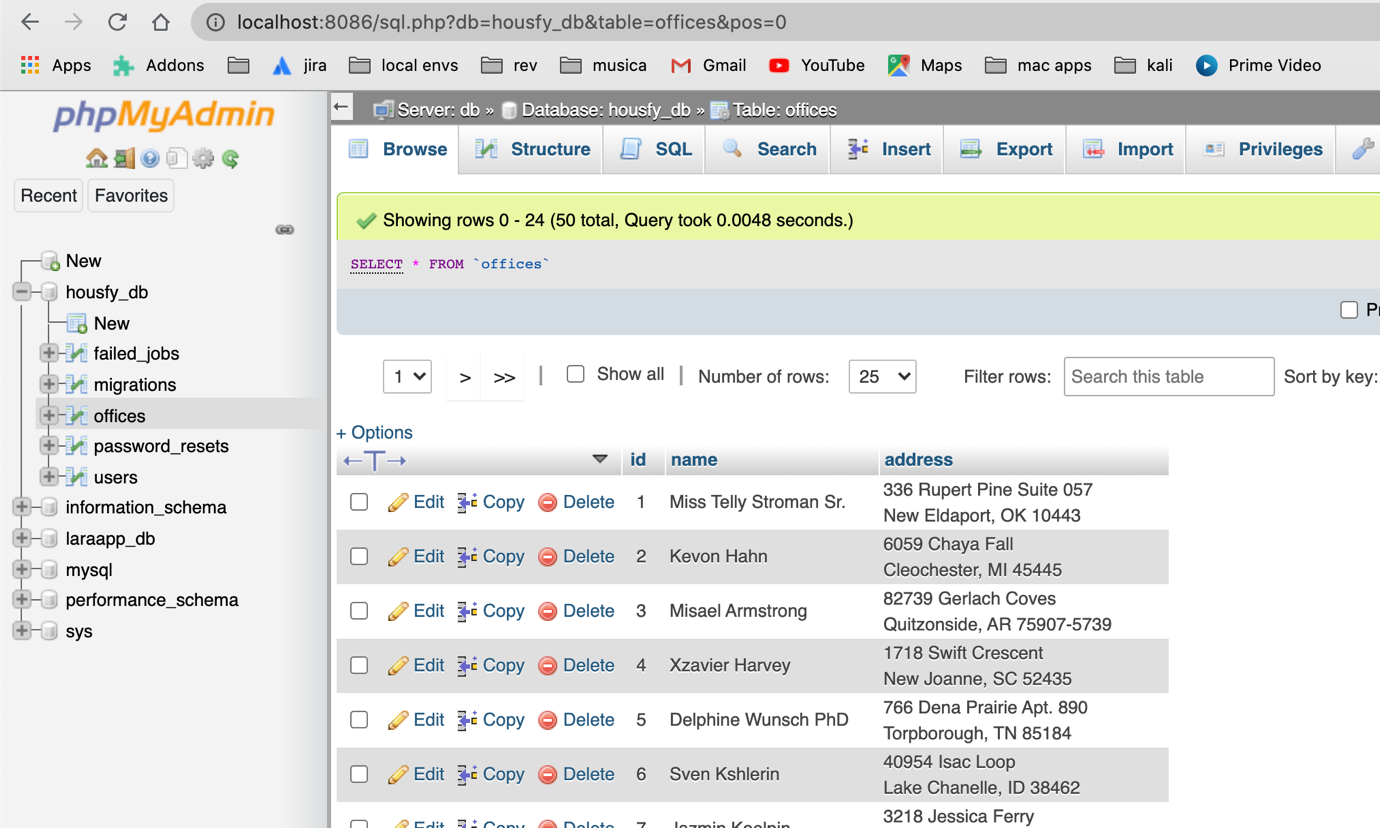


If find some problem in the execution of this command change the DB\_HOST variable

From housefly\_db (container name docker database to 127.0.0.1)



In this point database is ready , database housfy\_db is created, table for save the offices is created and populated , we can verify via phpMyAdmin that all is ok looking the table offices



The seeding of data assign a fake name and address to the field of office.

In this point we can begin to launch our api rest operation

API REST

All operations crud are implemented under controller : apiOfficesController

We have 5 operations :

getOffice -> Fetch data from office with id = x

getAllOffices(call offices)->Fetch all data from all offices

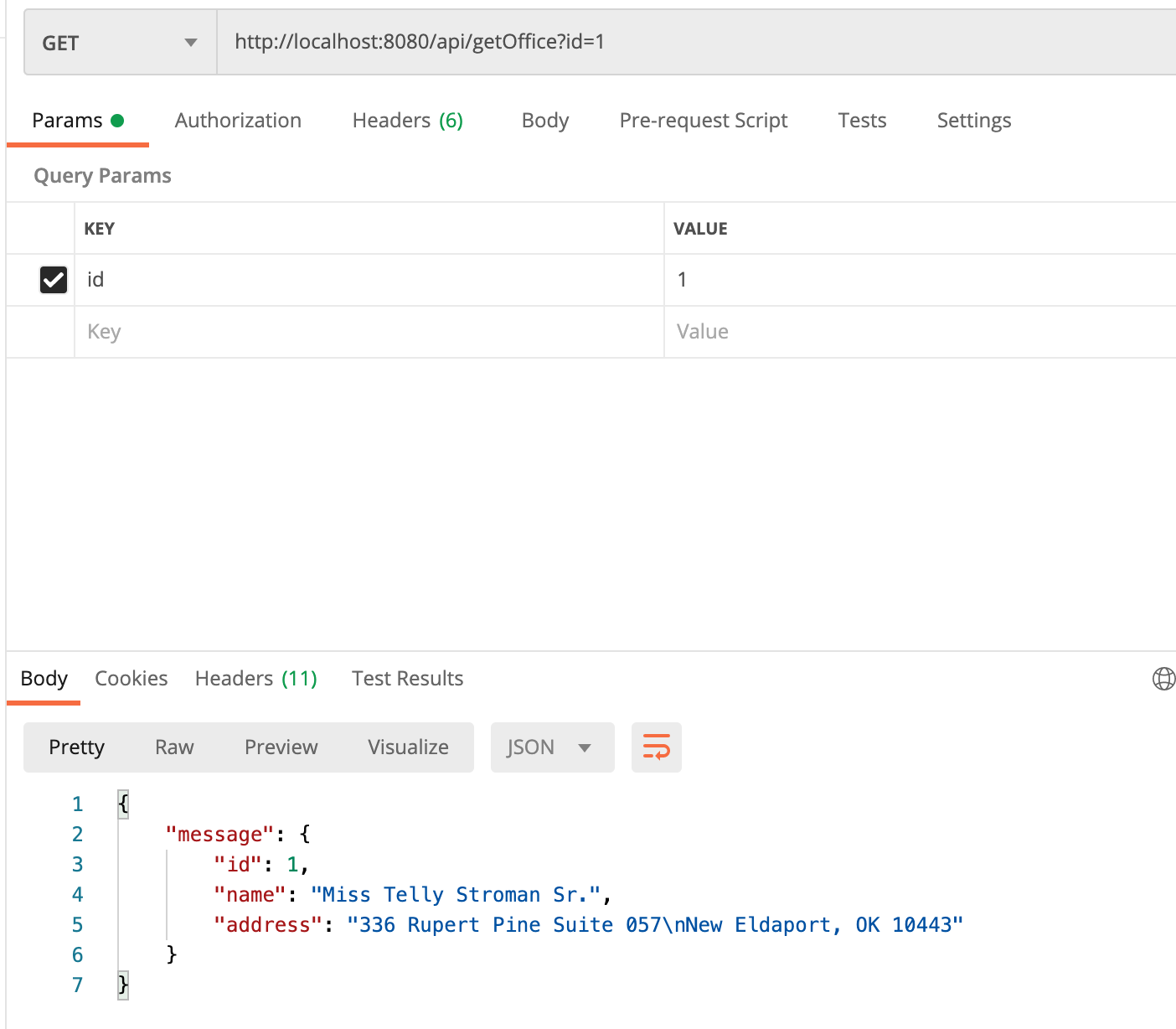
newOffice-> save a new office in dbbb with params name and address

updateOffice->update name and address for existing office in ddbb with id = x

deleteOffice -> delete office with id = x

For test all is ok, we can use postman :

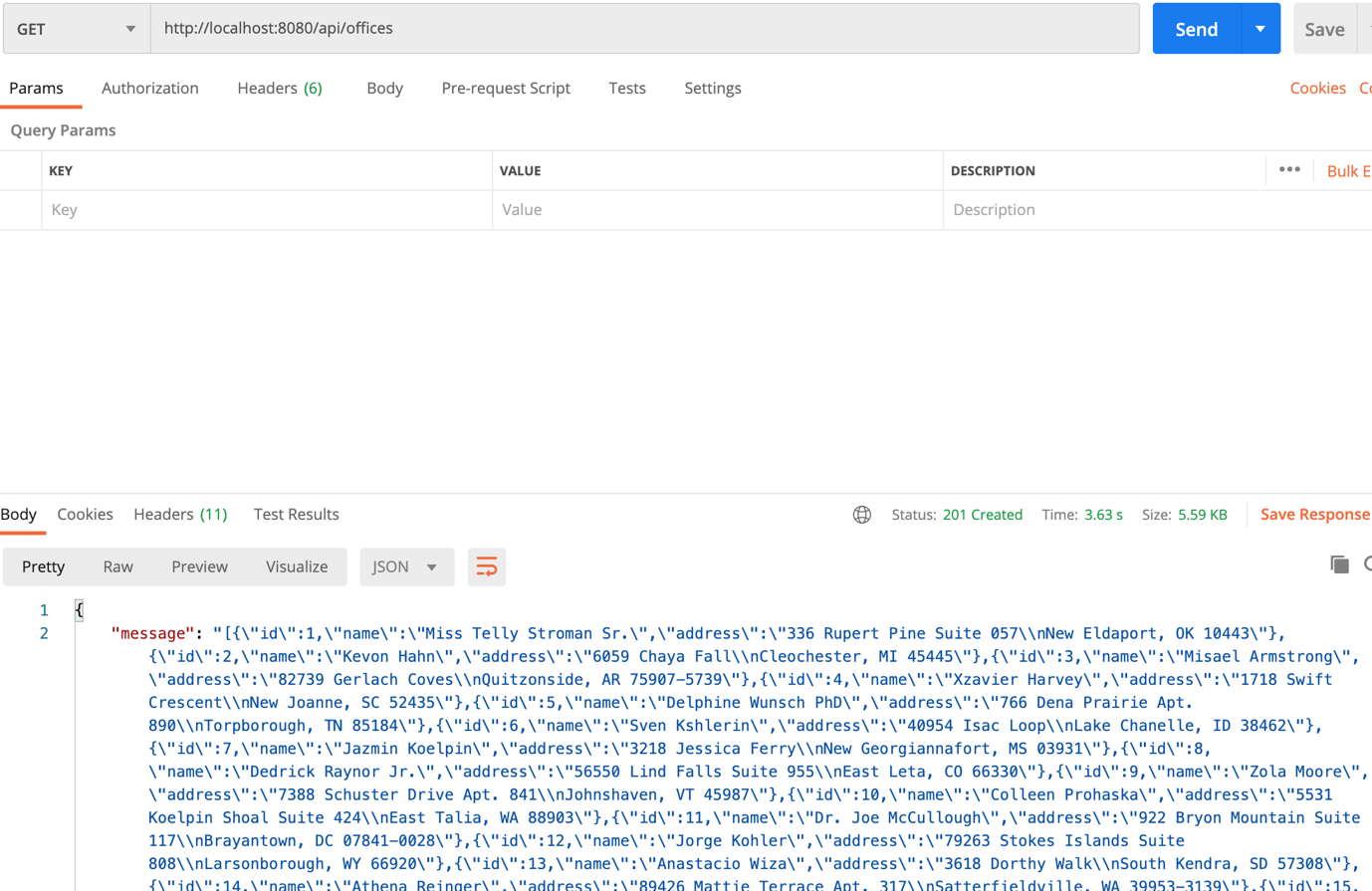
<http://localhost:8080/api/getOffice?id=1>



Call of getOffice (READ OPERATION/ GET ) with param id

-Get all offices: (READ METHOD / GET)

http://localhost:8080/api/offices



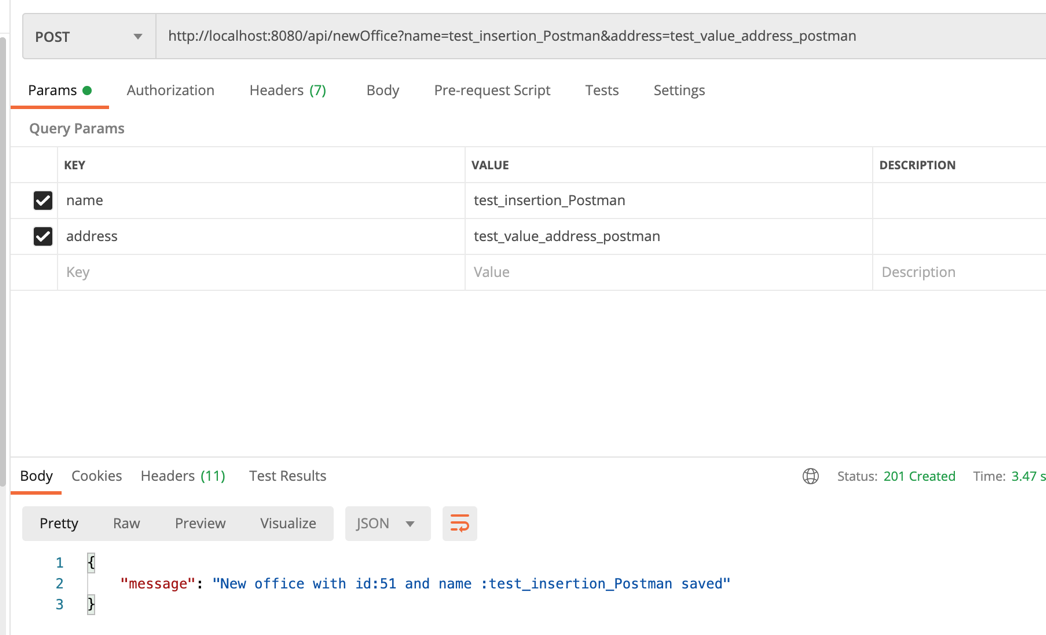
Retrieve all offices in ddbb (or CACHE if exist)

-newOffice

http://localhost:8080/api/newOffice?name=test\_insertion\_Postman&address=test\_value\_address\_postman

Params : name and address

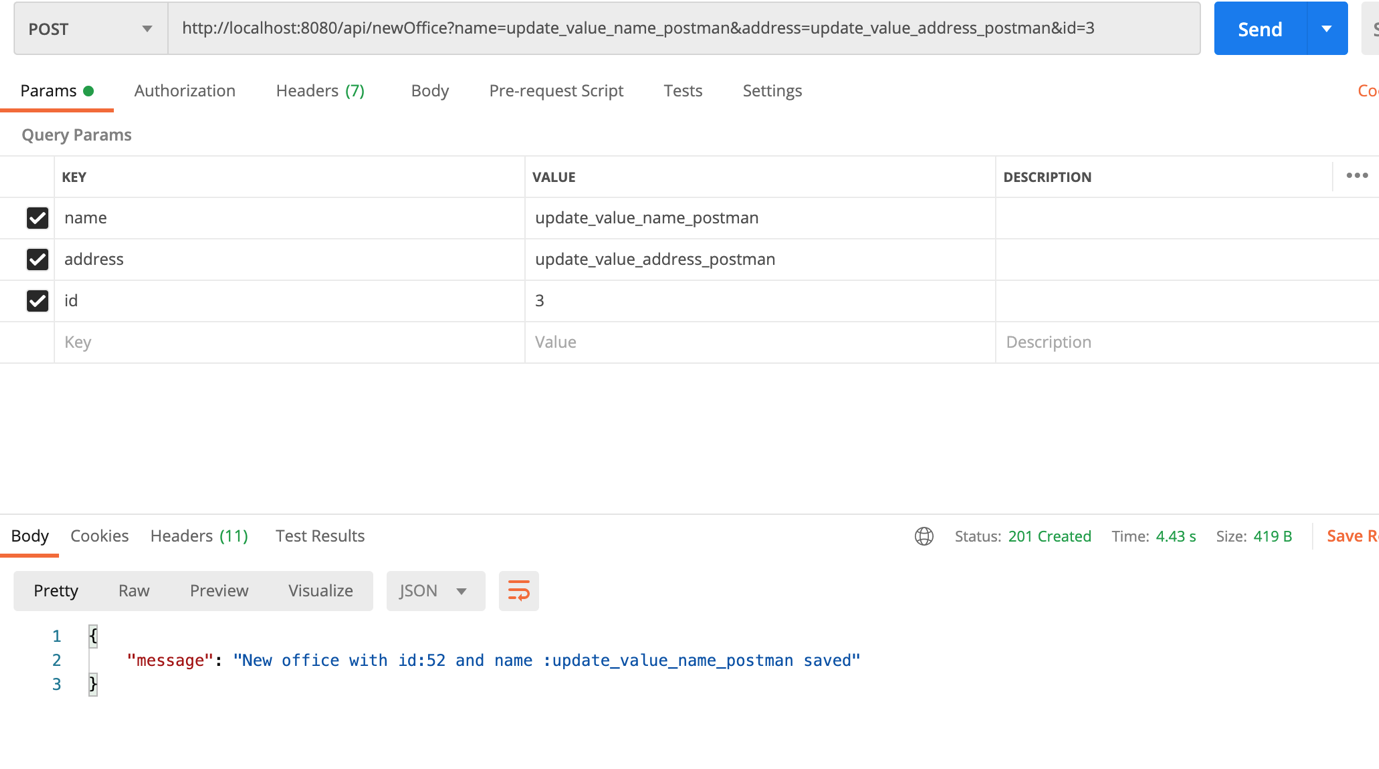
Create a new office and save in ddbb



POST , WRITE OPERATION

-updateOffice

http://localhost:8080/api/newOffice?name=test\_insertion\_Postman&address=test\_value\_address\_postman



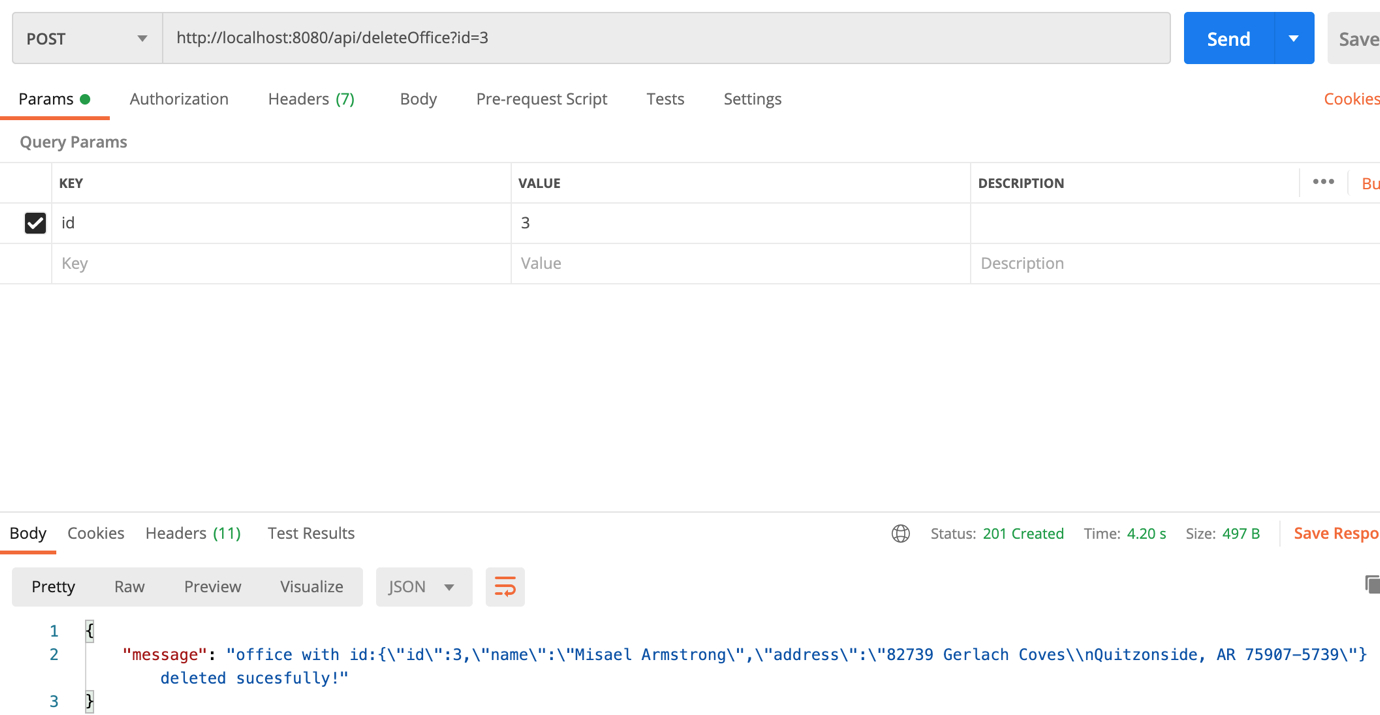
Update a Office that exist in ddbb

Params id -> id of office to update , name -> new name , address-> new address

WRITE OPERATION /POST

-deleteOffice

http://localhost:8080/api/newOffice?name=test\_insertion\_Postman&address=test\_value\_address\_postman



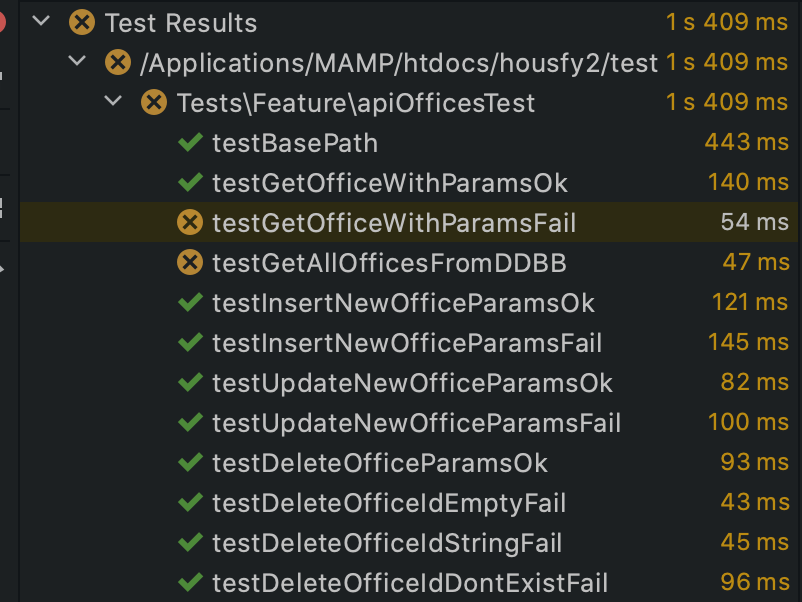
Delete office from ddbb

Params id -> id to office for deletion

WRITE OPERATION /POST

TEST :

All test can be runned vendor php unit / phpStorm environment linked to phpunit server / console application(custom)



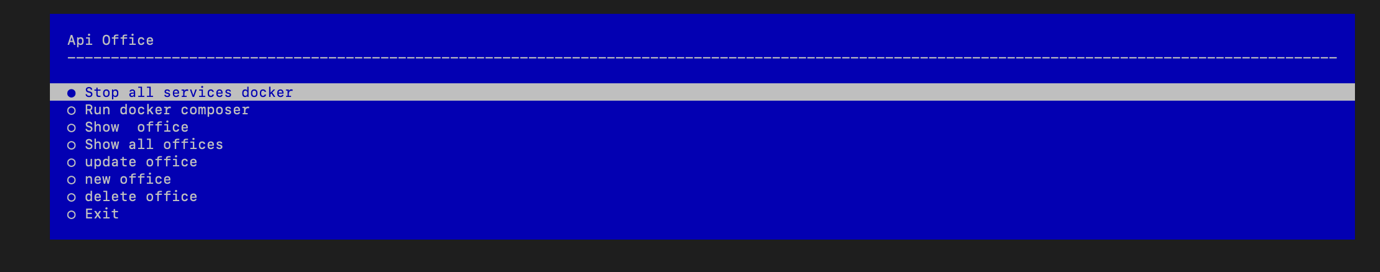
In this case all test are ok (less this that don’t arrive to get the offices cause test don’t arrive to connect with redis server for retrieve data)

Console Api

The new command is implemented on app/console/command/apiOfficeConsole and can be executed via terminal taping :

php artisan apiOffices

This show a new menu where we can choose some options



Docker -> start all services / stop all services

CRUD OPERATIONS :

Read ->

get all offices

get office (request id)

Write ->

New Office -> Write a new office (request name and address)

Update office->Update a exist office(request id , name and address)

Delete Office -> Delete a office by id

All operations return a message with status of operation

