Gustavo Gama

LudusCristaltec Backend Challenge

Instructions file.

PREREQUIREMENTS FOR BOTH EXERCICES :
- Docker (usage of Docker Desktop is recommended for both projects).
- Postman (check out the postman collections available and import them).

Challenge 1 : CAT API PROXY

How to run the code.

Get Docker Desktop running then:

Step 1: Download & Unzip the 1st project and go into the root folder of the project (the folder for the first project is : “chall_1_catapi”, inside you can fild the files related to Docker).

Step 2: Run the sh script to build and launch the container

→ ./script_run_docker.sh

After this, the server app will be running on
http://localhost:3000

NOTE IN CASE YOU CAN’T RUN BASH SCRIPTS (windows for example): with docker desktop installed, just copy paste the docker compose commands found in the script directly on the terminal, it should work

There are 3 endpoints available, please check the Postman collection to see the preset of requests available.

The postman collection can be found in the root of the project, the name of the file is :
“Postman_collection_CAT_API_PROXY.postman_collection.json”
Import it on Postman to perform the requests and tests. There is also a description of the collection that explains the different requests available.

Step 3 (Optional): Run the sh script for testing.
After running the step 2 (after lauching the container), open a second terminal, at the root of the project. Then run the script:

→ ./script_run_tests_running_docker.sh
This will launch the unit tests for the challenge 1.

Step 4: stop everything with “docker compose down”

Challenge 2 : CAT AS A SERVICE CRUD

PREREQUIREMENTS: docker is needed along docker compose. Again, usage of Docker Desktop is recommended.
How to run the app.
Step 1: Download & Unzip the 1st project and go into the root folder of the project (the folder for the first project is : "chall_2_CATAAS_crud").

Step 2: Run the sh script to build and launch all the services.

→ ./script_run_docker.sh

NOTE IN CASE YOU CAN'T RUN BASH SCRIPTS (windows for example): with docker desktop installed, just copy paste the docker compose commands found in the script directly on the terminal, it should work. Make sure to wait for the db to be up.

What this script does :
This script will run all the commands required to start all the services: the app itself, the database (mariadb), and an adminer (to check in a browser the database).

Please wait a bit for the dependencies to install, and for the database to start (should take around a minute to start the db).

Once the db is up & running, the app will become available too. A set of instructions are executed to perform migrations and seeding of the database (3 cats will be present on the db at launch).

The config uses the environment of the docker-compose.yml file.

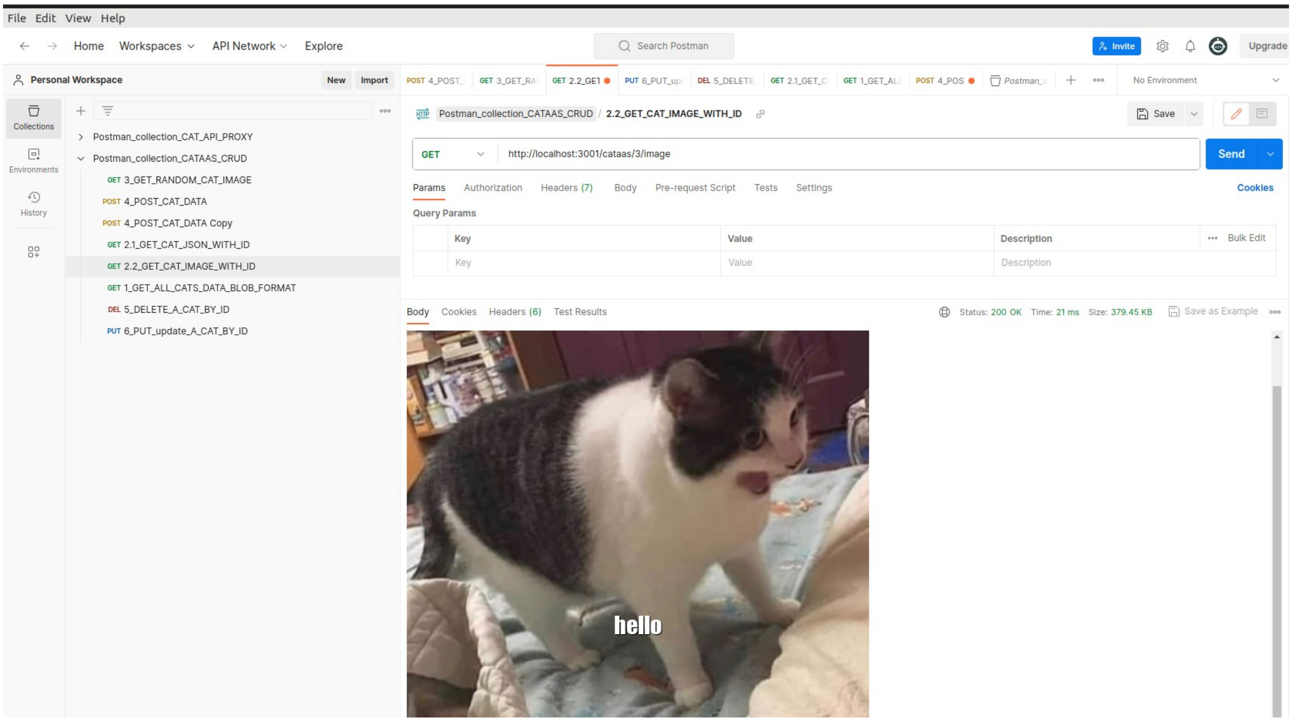
After this, the server app will be running on
http://localhost:3001

Step 3: Check the results

There are multiple requests available in the postman collection:
"Postman_collection_CATAAS_CRUD".

You should be able to run them after step 2 which was in charge of the migrations and seeding.
Please test them as you wish. I will describe (and show some screenshots) here those who have special characteristics.

- GET a cat IMAGE by id (in collection it is the request: "2.2_GET_CAT_IMAGE_WITH_ID").
(note: it's different from getting a cat data in json, which is in BLOB format).
Check out image 1 for more details, an example of url is: http://localhost:3001/cataas/3/image



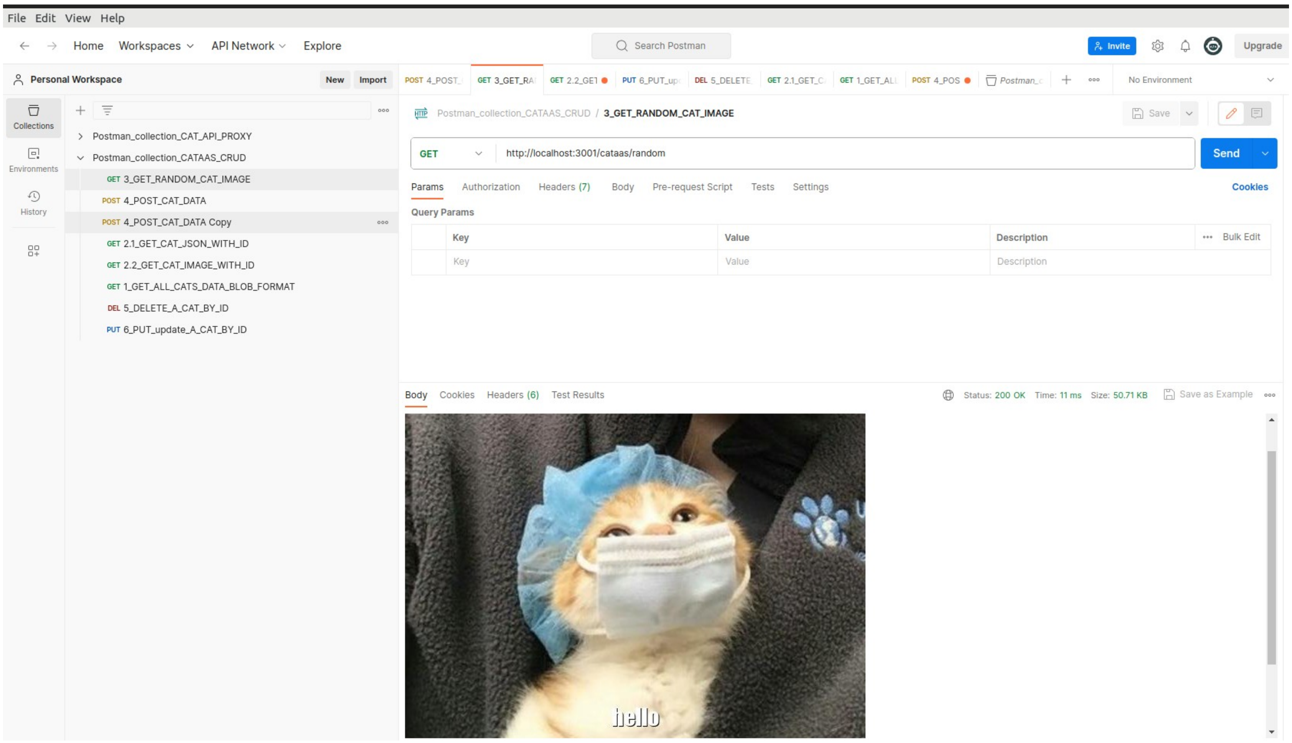
GET http://localhost:3001/cataas/3/image

Status: 200 OK Time: 21 ms Size: 379.45 KB

Image 1. GET method to fetch a cat IMAGE by id.

- GET a random cat IMAGE (in collection it is: “3_GET_RANDOM_CAT_IMAGE”).

This request will return a random cat (image), check Image 2. Example of url: http://localhost:3001/cataas/random



GET http://localhost:3001/cataas/random

Status: 200 OK Time: 11 ms Size: 50.71 KB

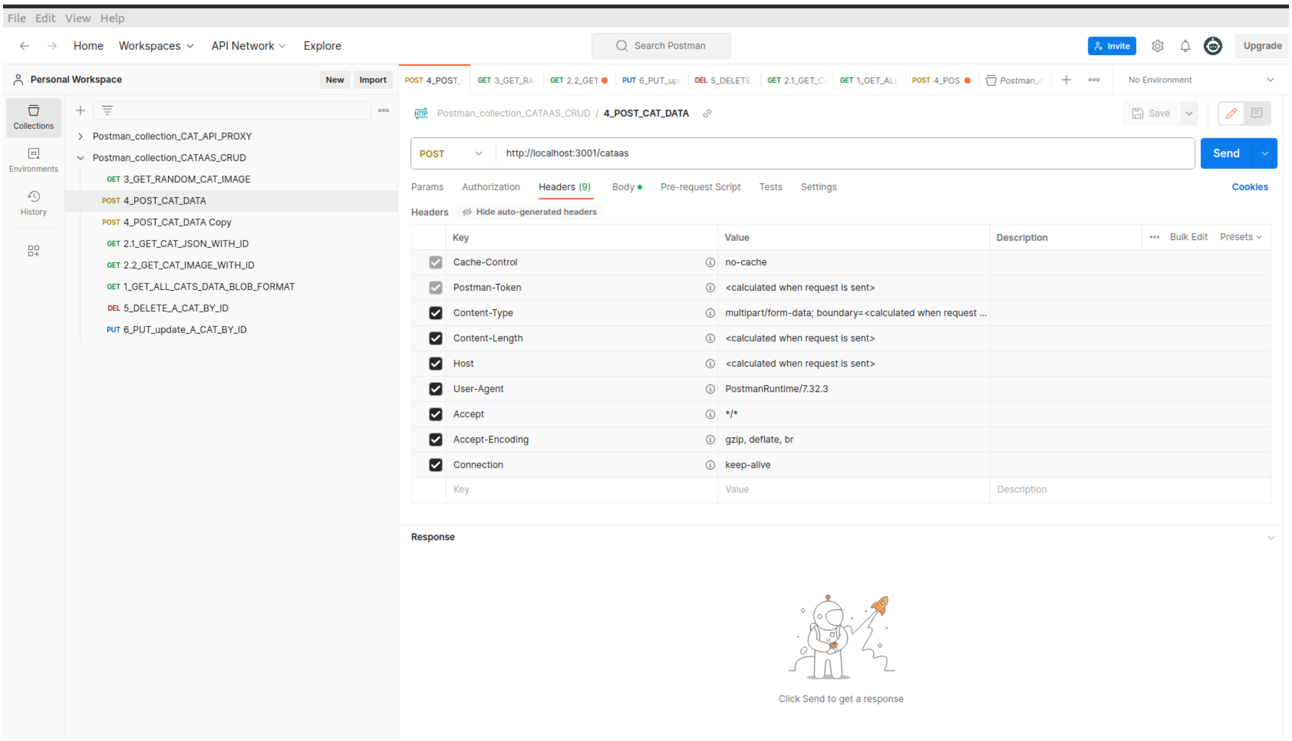
Image 2. GET method to fetch a random cat IMAGE.

- POST to create a cat (in collection it is: “4_POST_CAT_DATA”)

This request will create a cat only name field is required, but you can also provide other data such as metadata, imageData (that you can upload) etc.

It requires some extra information to perform this, you can find in that request (from the collection) the different headers required in “Headers” tab (check image 3).

Additionnaly, check out Image 4 for an example on how to input the data, including the image itself (in theory jpeg and png should work at least). You have to click on the button “Select files” and choose a picture to upload (I provided a few on catImages folder in the root folder), then click Send.



- Postman_collection_CAT_API_PROXY
- Postman_collection_CATAAS_CRUD
  - GET 3_GET_RANDOM_CAT_IMAGE
  - POST 4_POST_CAT_DATA
  - POST 4_POST_CAT_DATA Copy
  - GET 2.1_GET_CAT_JSON_WITH_ID
  - GET 2.2_GET_CAT_IMAGE_WITH_ID
  - GET 1_GET_ALL_CATS_DATA_BLOB_FORMAT
  - DEL 5_DELETE_A_CAT_BY_ID
  - PUT 6_PUT_update_A_CAT_BY_ID

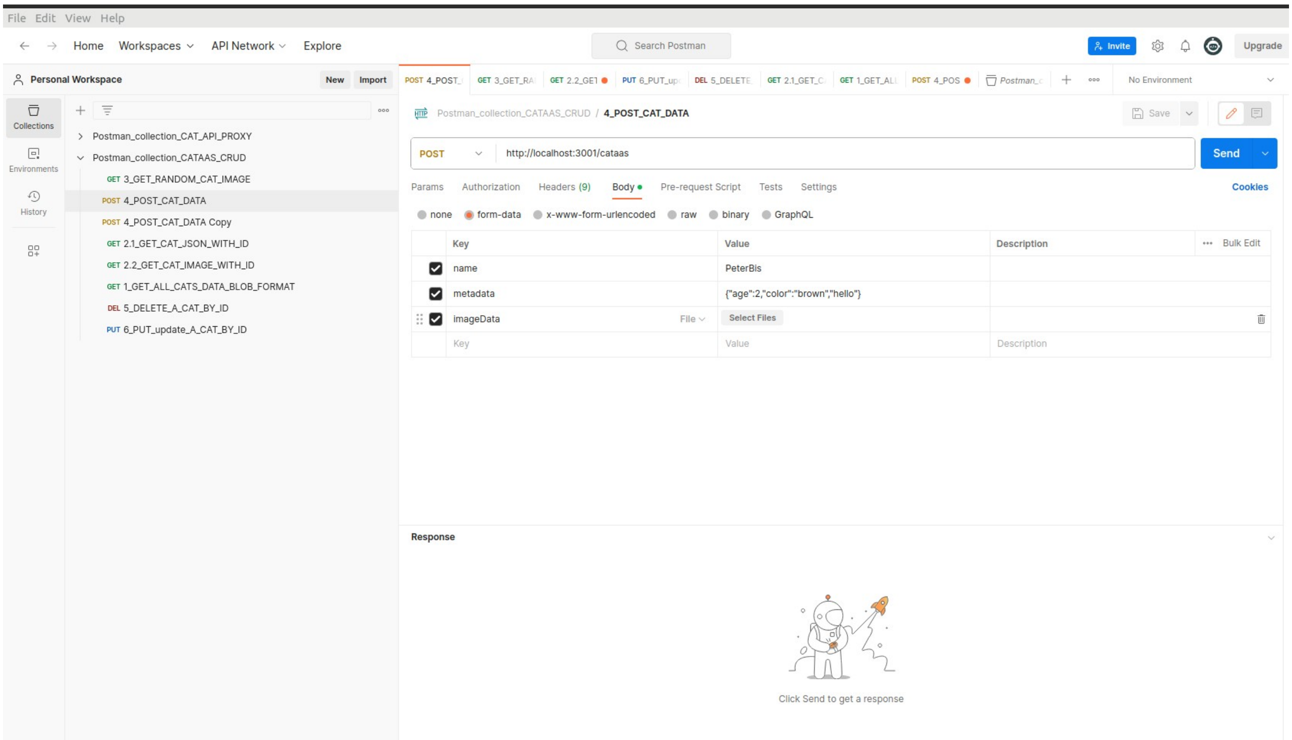
Postman_collection_CATAAS_CRUD / 4_POST_CAT_DATA

POST http://localhost:3001/cataas

Headers — Hide auto-generated headers

	Key	Value	Description
☑	Cache-Control	no-cache	
☑	Postman-Token	<calculated when request is sent>	
☑	Content-Type	multipart/form-data; boundary=<calculated when request ...	
☑	Content-Length	<calculated when request is sent>	
☑	Host	<calculated when request is sent>	
☑	User-Agent	PostmanRuntime/7.32.3	
☑	Accept	*/*	
☑	Accept-Encoding	gzip, deflate, br	
☑	Connection	keep-alive	
	Key	Value	Description

Image 3: POST method, headers tab.



Postman_collection_CAT_API_PROXY
Postman_collection_CATAAS_CRUD
GET 3_GET_RANDOM_CAT_IMAGE
POST 4_POST_CAT_DATA
POST 4_POST_CAT_DATA Copy
GET 2.1_GET_CAT_JSON_WITH_ID
GET 2.2_GET_CAT_IMAGE_WITH_ID
GET 1_GET_ALL_CATS_DATA_BLOB_FORMAT
DEL 5_DELETE_A_CAT_BY_ID
PUT 6_PUT_update_A_CAT_BY_ID

Postman_collection_CATAAS_CRUD / 4_POST_CAT_DATA

POST http://localhost:3001/cataas

○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	Key	Value	Description
☑	name	PeterBis	
☑	metadata	{"age":2,"color":"brown","hello"}	
☑	imageData File	Select Files	
	Key	Value	Description

Image 4: POST method, body tab.

Example: make Post on URL http://localhost:3001/cataas

After clicking on Send, the cat will be created with the data provided, and if you put an image there it should appear with the appropriate GET method (such as 2.2_GET_CAT_IMAGE_WITH_ID from the collection).

Step 4: Run the tests

To be honest I ran into some trouble with settings related to testing (while running all services at the same time from the sh script). So to run the unit tests I advice the following (check out src/__tests__ folder to find the tests js file):

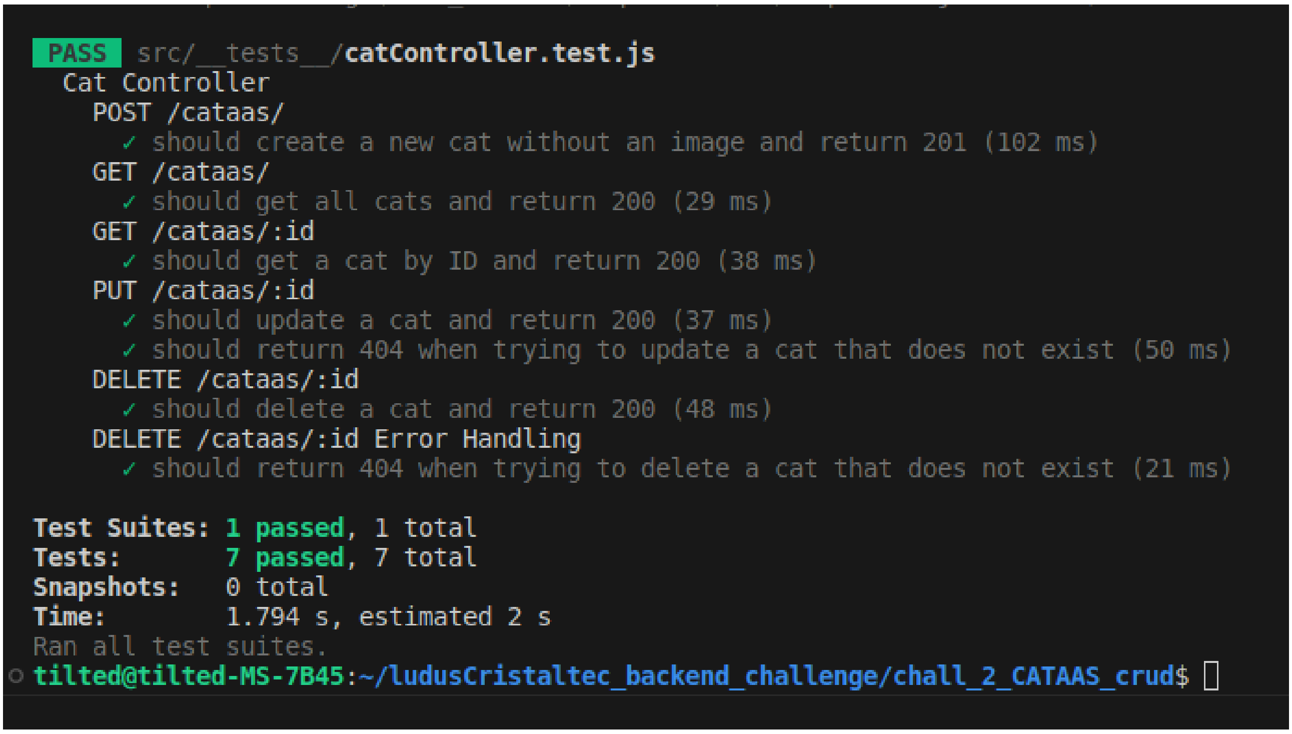
- Shutdown everything just in case and run all again with the commands:
→ docker compose down
→ docker compose build

→ docker compose up -d mariadb adminer

This will just start the Database service and the adminer (to check on browser the db).
Once the db is running we have to make the migrations:
→ npx sequelize-cli db:migrate --config config/env_config.js

Finally, run the tests command:
→ npm test

And the tests should appear on the console (image 5).



```
 PASS  src/__tests__/catController.test.js
  Cat Controller
    POST /cataas/
      ✓ should create a new cat without an image and return 201 (102 ms)
    GET /cataas/
      ✓ should get all cats and return 200 (29 ms)
    GET /cataas/:id
      ✓ should get a cat by ID and return 200 (38 ms)
    PUT /cataas/:id
      ✓ should update a cat and return 200 (37 ms)
      ✓ should return 404 when trying to update a cat that does not exist (50 ms)
    DELETE /cataas/:id
      ✓ should delete a cat and return 200 (48 ms)
    DELETE /cataas/:id Error Handling
      ✓ should return 404 when trying to delete a cat that does not exist (21 ms)

Test Suites: 1 passed, 1 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        1.794 s, estimated 2 s
Ran all test suites.
○ tilted@tilted-MS-7B45:~/ludusCristaltec_backend_challenge/chall_2_CATAAS_crud$ 
```

Image 5: tests OK.

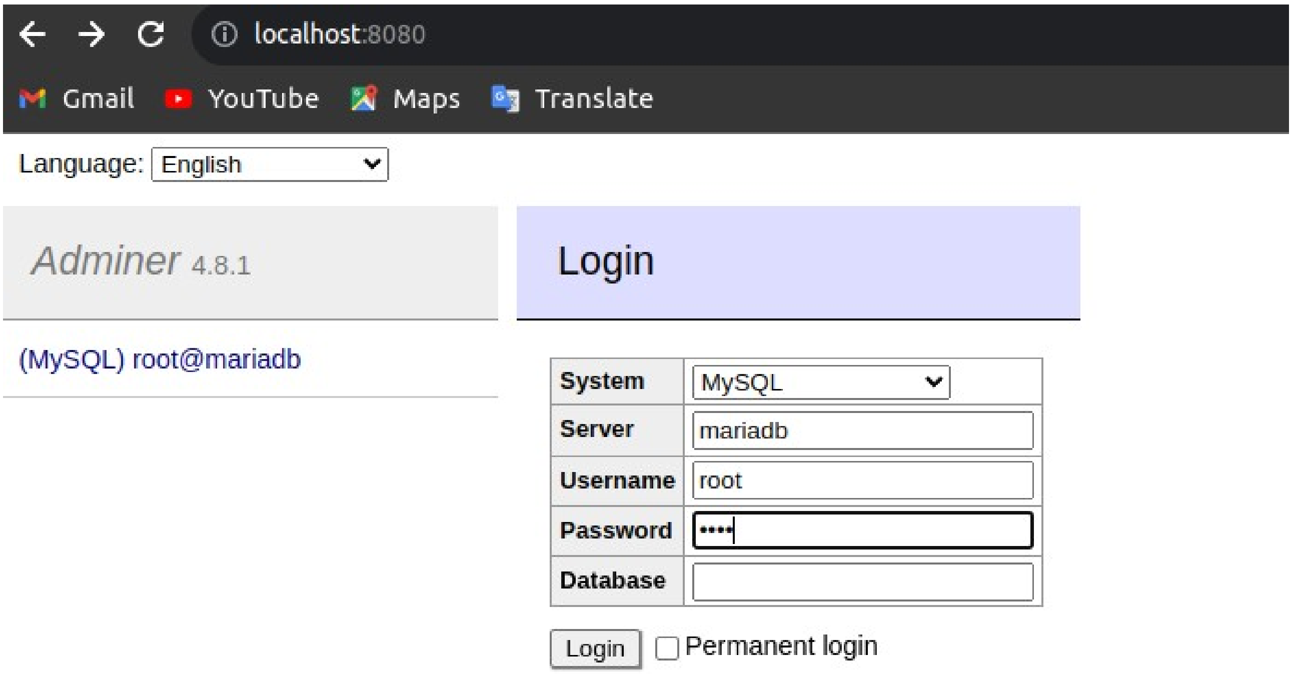
Step 5 (optional): check out adminer service on the browser

Go to the browser and enter http://localhost:8080/

Log in, check image 6, password is “root” .

Then click on: my_database → Cats → Select Data (check out image 7).

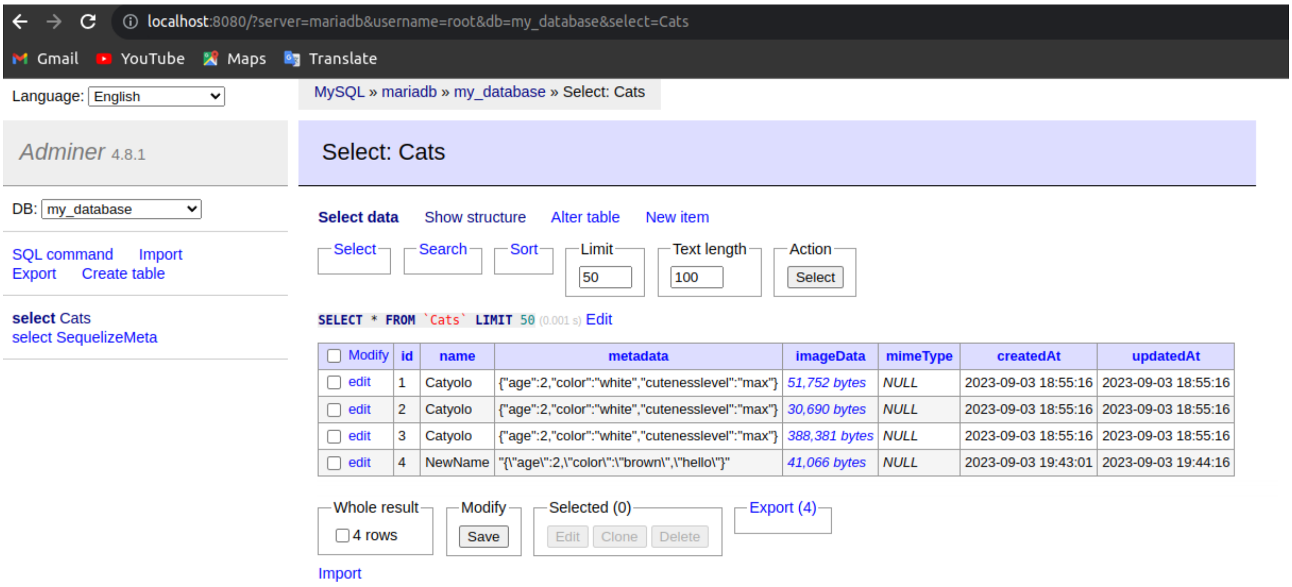
You can find the data present in the database (for visual purposes, verification/tracking).



localhost:8080

System	MySQL
Server	mariadb
Username	root
Password	••••
Database	

Image 6. Adminer interface login



localhost:8080/?server=mariadb&username=root&db=my_database&select=Cats

Limit: 50
Text length: 100

```
SELECT * FROM `Cats` LIMIT 50 (0.001 s) Edit
```

□ Modify	id	name	metadata	imageData	mimeType	createdAt	updatedAt
□ edit	1	Catyolo	{"age":2,"color":"white","cutenesslevel":"max"}	*51,752 bytes*	*NULL*	2023-09-03 18:55:16	2023-09-03 18:55:16
□ edit	2	Catyolo	{"age":2,"color":"white","cutenesslevel":"max"}	*30,690 bytes*	*NULL*	2023-09-03 18:55:16	2023-09-03 18:55:16
□ edit	3	Catyolo	{"age":2,"color":"white","cutenesslevel":"max"}	*388,381 bytes*	*NULL*	2023-09-03 18:55:16	2023-09-03 18:55:16
□ edit	4	NewName	"{\"age\":2,\"color\":\"brown\",\"hello\"}"	*41,066 bytes*	*NULL*	2023-09-03 19:43:01	2023-09-03 19:44:16

□ 4 rows

Image 7. Cats Table present in the mariadb.

Step 6: stop everything with “docker compose down”

Final note:
I hope everything was clear and worked out well, in case of doubt or problem, contact me please:
gustavo.p.gama@gmail.com

But as long as you have docker desktop there shouldnt be an issue. Both exercices were developed on Ubuntu and tested on Ubuntu AND Windows (for compatibility purposes).