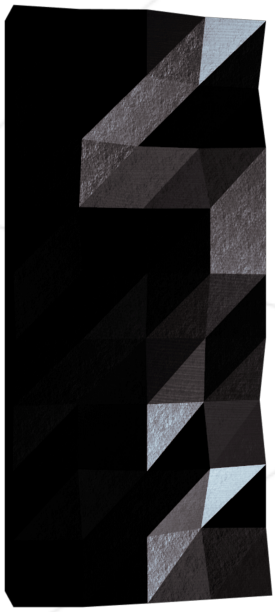

Building Applications with Python

Module 5

Summary: API + DynamoDB

Version: 1.1

Contents

Chapter I

	Exercise 01
NoSQL	
Turn-in directory: $ex01/$	
Files to turn in: `api.py, test_api.py, database.py, schemas.py`	
Allowed functions: `No restrictions`	

`To proceed, use the ‘database.py‘ file created in the previous module, and the latest version of the API from module 3.`

Let’s convert the most recent version of the API to use DynamoDB instead of PostgreSQL. To do this, use the database you created in the previous exercise.

All your endpoints, which previously communicated with the Postgres database through the `database.py` and `models.py` files, should have similar behavior but using DynamoDB.

Chapter II

	Exercise 02
Balance	
Turn-in directory: $ex02/$	
Files to turn in: `api.py, test_api.py *.py`	
Allowed functions: `No restrictions`	

1. Implement an endpoint that returns the account balance: `/account/:id/balance`;

Chapter III

	Exercise 03
Authentication	
Turn-in directory: $ex03/$	
Files to turn in: `api.py, test_api.py, database.py, schemas.py`	
Allowed functions: `Standard library, pydantic, fastapi, pynamodb, python-dotenv, uvicorn, pytest`	

`To proceed, copy all necessary files from the previous exercise.`

1. You must implement authentication on all endpoints of your API.
2. For this, you will use the `OAuth2PasswordBearer` from FastAPI.
3. Your tests should work with authentication.
4. To configure the `token` value, as well as a username and password, use the `python-dotenv` library, in both `api.py` and `test_api.py`.
5. You must implement the `post("/token")` route, which exchanges a username and password (from `OAuth2PasswordRequestForm`) for an access token.

`In the tests, pass the headers in this format to provide credentials and authenticate`
`‘response = client.get("/items/", headers={"Authorization": f"Bearer {token}"})‘`

To use OAuth2 authentication with FastAPI, you may also need to install the ‘python-multipart‘ library.

Ideally, the password used would be stored encrypted in the database. This is not necessary for the purposes of this exercise.

Chapter IV

	Exercise 04
Containerize your application	
Turn-in directory: $ex04/$	
Files to turn in: `api.py, test_api.py, Dockerfile, docker-compose.yml`	
Allowed functions: `Standard library, fastapi, psqlalchemy, psycopg2-binary, python-dotenv, uvicorn, pytest`	

In this exercise you should use `docker compose`

1. Containerize your application.

2. The `docker-compose.yml` file should contain the commands and services necessary to start the dynamodb-local database and your application.

3. The table, if it does not exist, should also be created automatically.

4. Your application should be visible to the host through port 8080.

Chapter V

Peer Review and Submission

- Submit your project to your *Git* repository available on the project page on the intranet.
- Only the work within your repository will be evaluated during the defense. Do not hesitate to check the names of your files and folders to ensure they are correct.
- At the time of evaluation, the evaluator will go to the workstation of the student being evaluated to perform the tests. A clone of the repository will be made in a new folder, and these are the files that will be evaluated.