Kazakh-British Technical University

Web Application Development

Assignment №2

Exploring Django with Docker

Full Name: Tamiris Abildayeva

ID: 23MD0503

Link to GitHub: https://github.com/TamirisK/university-web-application-development

13.10.2024

Table of Contents

Introduction

The goal of Assignment 2 is to configure a Django application using Docker Compose and general practice Docker Compose to see networking and volumes. Docker Compose is a tool that helps define and share multi-container applications. A container is a standard unit of software that packages up code and all its dependencies. Container allows the application to run quickly and reliably from one computing environment to another.

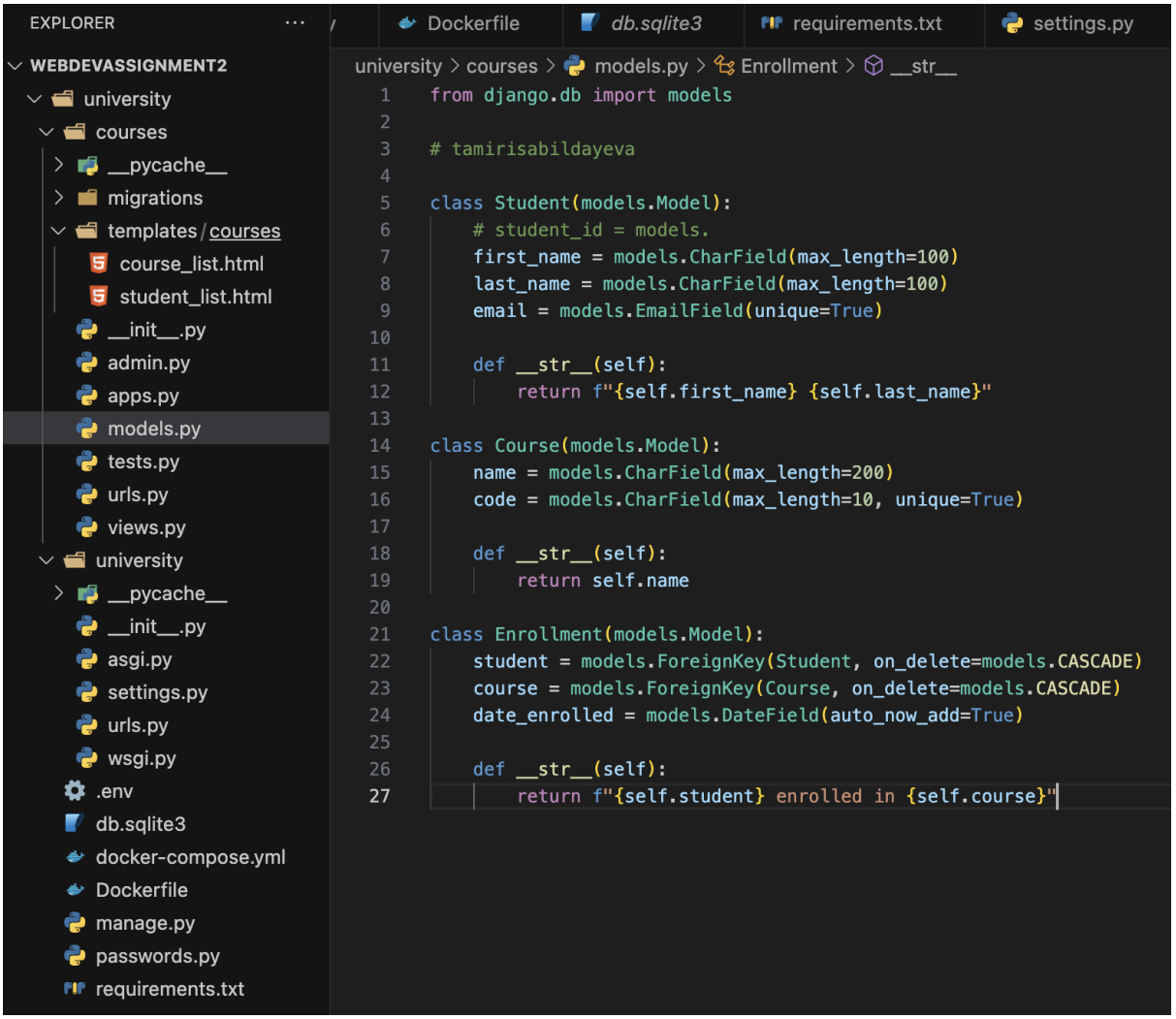


Fig. 1: Structure of Django application for this assignment.

Docker Compose

Configuration

Docker Compose uses a docker-compose.yml file to define services.

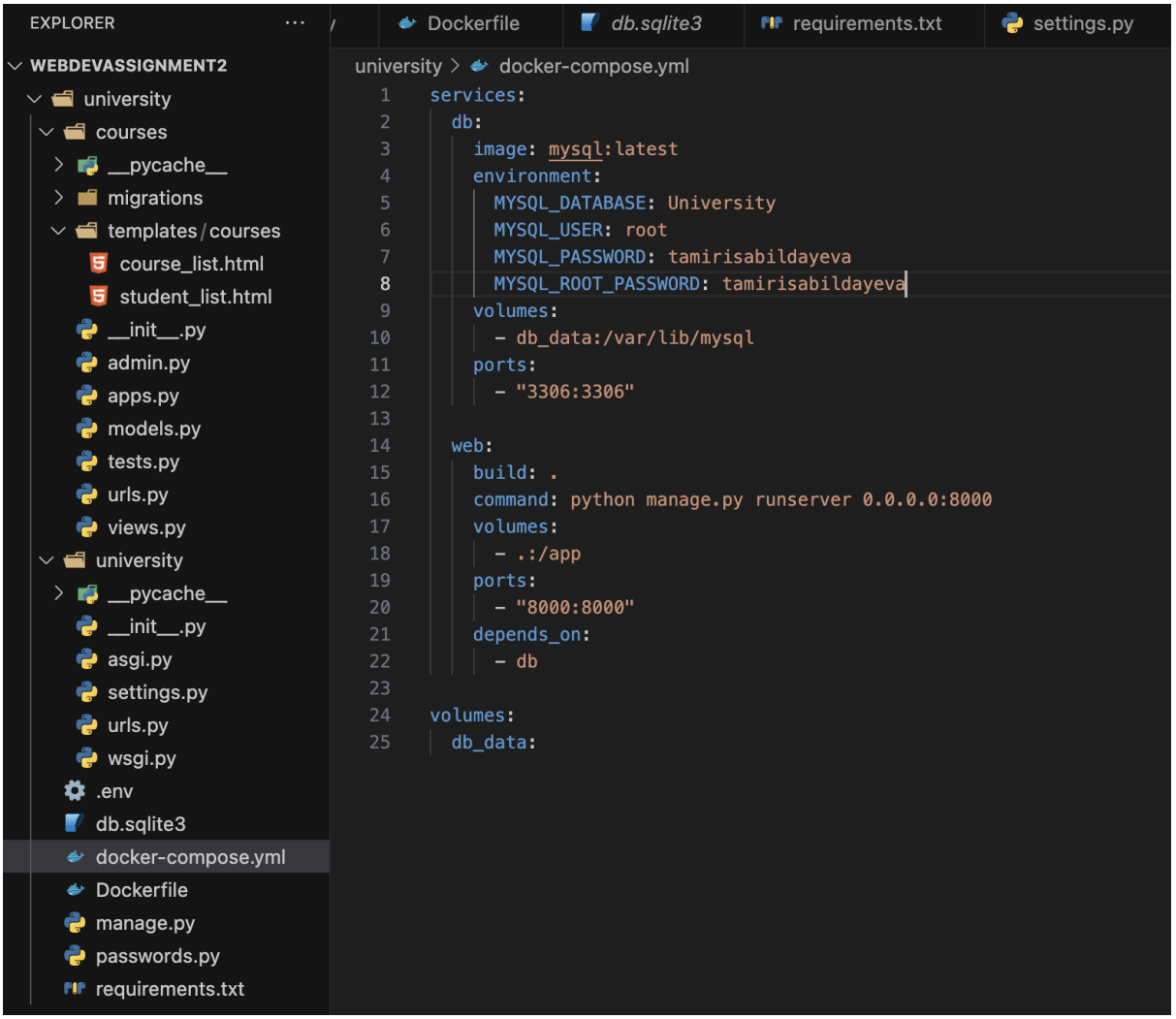


Fig. 2: Configuration in docker-compose.yml file.

Services:

- **db:**
  - **image:** The Docker image to be used in the project, in this case, the latest version of MySQL.
  - **environment**: Environment variables for the MySQL container that was selected previously need to specify the database title, username, user password, and root password.
  - **volumes:** To ensure that MySQL data is not lost when the container restarts, a volume (in this case, db_data) is defined to persist the data at the specified path inside the container, in this case, /var/lib/mysql.

    - **ports:** Maps port 3306 on the host to port 3306 on the container, allowing external access to the MySQL service. This configuration uses the default port mapping.
- **web:**
    - **build:** Configuration to build the Docker image from the current directory.
    - **command:** The command to run when the container starts, in this case, start all interfaces at port 8000.
    - **volumes:** Mounts the current directory to the /app directory inside the container.
    - **ports:** Similar to the ports in the db section, but with the port number set to 8000.
    - **depends-on:** Web service should start only after the db service is up and running. This ensures that the database is ready for connections when Django starts.
- **volumes:**
    - **db-data:** Name of the volume (db-data), which used by the db service.

Build and Run

A Dockerfile is used to build the Django container.

The Dockerfile starts from a base Python image, installs the project dependencies from requirements.txt, and then copies the Django project files into the container.

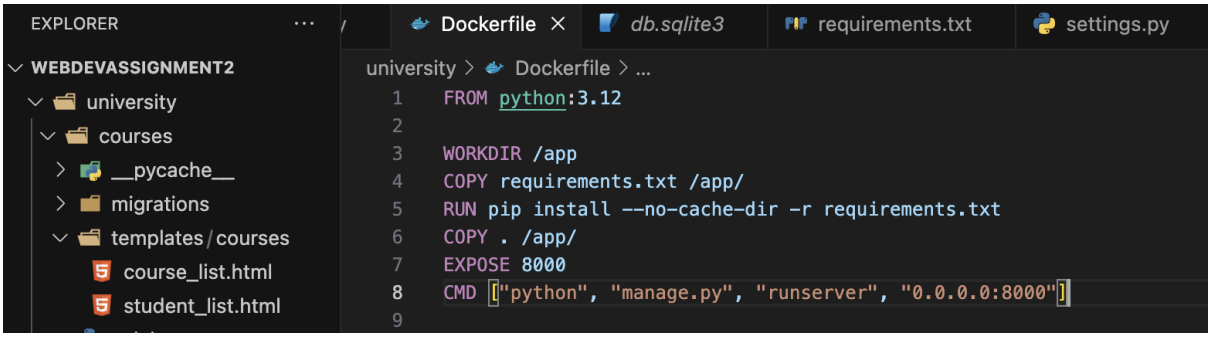


Fig. 3: Configuration in Dockerfile file.

The command django-admin startproject is used to create a new Django project. It sets up the necessary directory structure and files to begin a web application in Django

After creating the Django project, the next step is to add individual apps. In Django, apps are components of a project, each responsible for handling a specific part of the functionality

Fig. 4: Command startproject & startapp.

The "makemigrations" command creates migration files in the "migrations" directory for each app based on model changes. These files are applied to the database by using the "migrate" command. The "runserver" command starts the Django development server, allowing local access to the application.

```
tamirisabildayeva@T-MacBook-Pro university % python3 manage.py makemigrations
Migrations for 'courses':
  courses/migrations/0001_initial.py
    + Create model Course
    + Create model Student
    + Create model Enrollment
tamirisabildayeva@T-MacBook-Pro university % python3 manage.py migrate

Operations to perform:
  Apply all migrations: admin, auth, contenttypes, courses, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying courses.0001_initial... OK
  Applying sessions.0001_initial... OK
tamirisabildayeva@T-MacBook-Pro university % python3 manage.py createsuperuser

Username (leave blank to use 'tamirisabildayeva'):
Email address: tamiris.talgatovna.5@gmail.com
Password:
Password (again):
Superuser created successfully.
tamirisabildayeva@T-MacBook-Pro university % python3 manage.py runserver
```

Fig. 5: Command makemigrations, migrations & runserver.

The docker-compose up --build command starts the services defined in a Docker Compose application, rebuilding the service images before running them. Docker Compose is used to manage multi-container Docker applications defined in a docker-compose.yml file, allowing it to define services, networks, and volumes.

- **docker-compose:** to define services, networks, and volumes for your application in a simple YAML format.
- **up:** to start the services defined in the docker-compose.yml file.
- **-build:** to rebuild the images for the services before starting the containers.

```
tamirisabildayeva@T-MacBook-Pro university % docker-compose up --build

WARN[0000] /Users/tamirisabildayeva/Education/MasterDegree/WebAppDev/WebDevAssignment2/university/docker-compos
fusion
[+] Running 11/11
 ✔ db Pulled
   ✔ 8b4274ea61c5 Pull complete
   ✔ 65b13290a890 Pull complete
   ✔ 4a0edbf0dd13 Pull complete
   ✔ f8d978a6739e Pull complete
   ✔ 64da6e9087ad Pull complete
   ✔ d4f83991d877 Pull complete
   ✔ 9543fc93cec2 Pull complete
   ✔ 08f5c00014df Pull complete
   ✔ f88be74802d6 Pull complete
   ✔ 883258893faf Pull complete
[+] Building 0.0s (1/1) FINISHED
 => [web internal] load build definition from Dockerfile
 => => transferring dockerfile: 2B
failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory
tamirisabildayeva@T-MacBook-Pro university %
```

Fig. 6: Command docker-compose up –build.

The result in the browser after running docker-compose up --build.

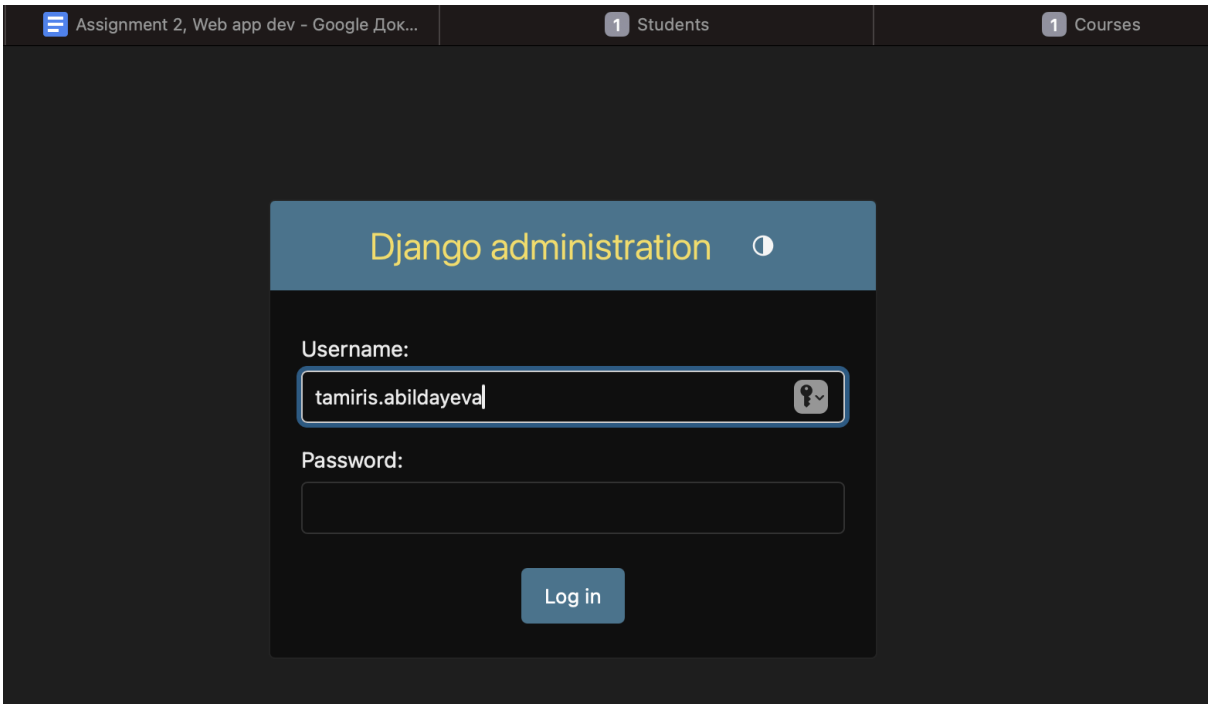


Fig. 7: Application runs.

Finding

I learned how to use Docker Compose to manage different parts of a computer program. I found out how to set up a database and a web service, and how to make them work together.

Docker Networking and Volumes

Networking

Creating a custom network in a Docker environment can greatly improve functionality and security. For example, adding a custom network net to the docker-compose.yml file using the bridge driver can benefit services like Django and MySQL, enabling them to communicate smoothly.

- **Isolation**: Creating specific networks for containers enhances security and ensures authorized services can communicate.
- **Service Discovery**: Docker provides automatic service discovery for easy communication between services.
- **Simplified Configuration**: Custom networks streamline communication between services and reduce configuration errors.
- **Balance**: Custom networks allow for load balancing and easier scaling of services.
- **Monitoring**: Custom networks can be configured with monitoring tools for easier issue identification.

Volumes

I used Docker volumes for data persistence in the application and learned about their benefits, including backup and restore capabilities. Also read about data sharing, version control, security, and performance optimization.

- **Backups and Restore**: Volumes allow easy data backup and restoration, crucial for maintaining data integrity, especially in databases.
- **Data Sharing**: Volumes enable multiple containers to share the same data, eliminating the need for data duplication.

Docker volumes store data outside the container for easy backup or sharing. They rely on Docker's file system and are the preferred way to store data for Docker containers and services.

Finding

During this part of the assignment, I acquired knowledge and experience in using Docker networks and volumes for the first time. I also learned about networking and volumes, which were mentioned earlier. This assignment gave me hands-on experience with Docker networking and volumes, improving my understanding of developing containerized applications.

Django Application Setup

Project Structure

In this Django project, there are three types of models: Student, Course, and Enrollment. Models play a crucial role in defining the structure of your application's data. Each model represents a database table, enabling the efficient storage and retrieval of information.

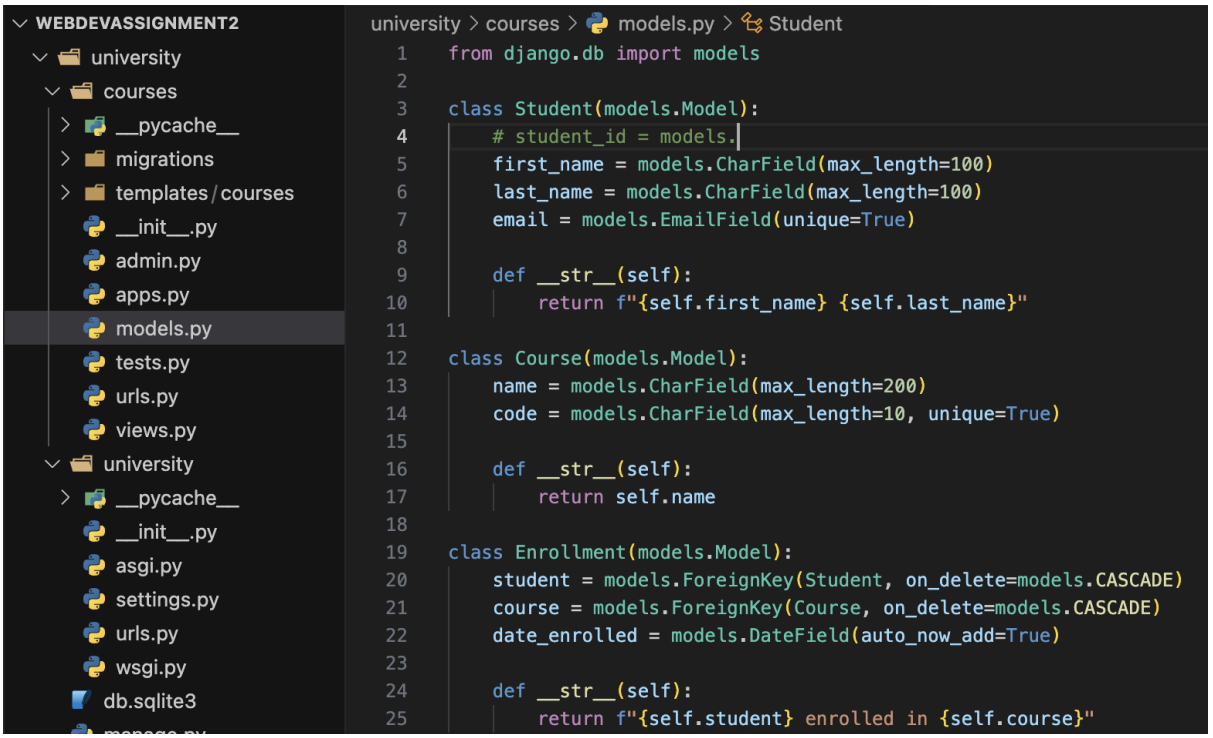


Fig. : Django models.

Registers the Student model with the Django admin site for managing student records.

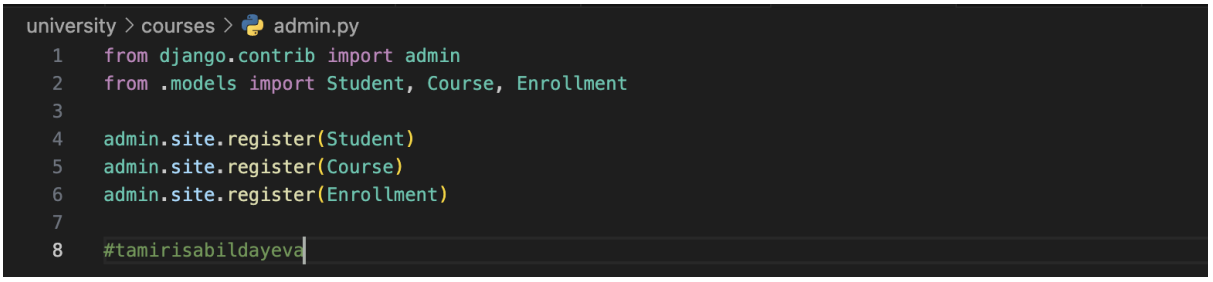


Fig. : admin.py file.

```
university > courses > urls.py > ...
from django.urls import path
from . import views

urlpatterns = [
    path('students/', views.student_list, name='student_list'),
    path('courses/', views.course_list, name='course_list'),
]
```

Fig. 8: urls.py file.

Database Configuration

After that, need to connect to the database.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': os.getenv('University'),
        'USER': os.getenv('root'),
        'PASSWORD': os.getenv('tamirisabildayeva'),
        'HOST': 'db',
        'PORT': '3306',
    }
}
```

Fig. : settings.py file.

The command mysql -u root -p is used to connect to a MySQL database server from the command line.

- **mysql**: Command-line client for MySQL.
- **-u root**: Specifies the MySQL username, in this case, root.
- **-p**: Prompts for the password associated with the specified username.

```
tamirisabildayeva@T-MacBook-Pro ~ % mysql -u root -p

Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 9.0.1 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE university_db;
Query OK, 1 row affected (0.00 sec)

mysql> EXIT;
Bye
tamirisabildayeva@T-MacBook-Pro ~ %
```

Fig. : Connection to the MySQL database.

Findings

When developing a Django application in Docker, I learned the importance of launching containers in the correct sequence in the docker-compose.yml file. During development, I found that performance could be improved by using Docker's caching mechanism. I adjusted the Dockerfile to optimize layer caching, especially for dependencies. By separating the installation of dependencies from the application code, I reduced the need to reinstall packages with every code change.

In a Dockerized application, managing environment variables is crucial. I store sensitive information in a .env file instead of hardcoding it into docker-compose.yml, which improves security and flexibility.

The database initialization can take longer than the application startup, so I ensured that the web service starts only after the database container is ready.

In summary, these practices improved application stability and provided a clearer structure for managing dependencies within the Docker environment. This experience enhanced my understanding of Docker.

Conclusion

To summarize, Assignment 2 gave me an introduction to setting up a Django application using Docker Compose. I learned how to effectively manage multiple containers, emphasizing the importance of networking, volumes, and running containers in the right order. Additionally, I optimized the Dockerfile for better performance, implemented best practices for managing environment variables, and gained practical skills that will be valuable for future projects using containerized applications.

Reference

- https://docs.djangoproject.com/en/5.1/intro/tutorial01/
- https://docs.djangoproject.com/en/5.1/intro/tutorial02/
- https://docs.docker.com/
- https://hub.docker.com/
- https://chatgpt.com/ (to fix the error with mysqlclient, didn't help)
- https://stackoverflow.com/questions/76585758/mysqlclient-cannot-install-via-pip-cannot-find-pkg-config-name-in-ubuntu
- https://stackoverflow.com/questions/76876823/cannot-install-mysqlclient-on-macos
- https://www.youtube.com/watch?v=WBqHr2kPc_A
- & other tutorial, but forget to save links

```
tamirisabildayeva@T-MacBook-Pro university % brew services list

Name          Status  User              File
mysql         none
nginx         none
postgresql@14 started tamirisabildayeva ~/Library/LaunchAgents/homebrew.mxcl.postgresql@14.plist
tamirisabildayeva@T-MacBook-Pro university %
```

```
tamirisabildayeva@T-MacBook-Pro university % docker-compose up --build

WARN[0000] /Users/tamirisabildayeva/Education/MasterDegree/WebAppDev/WebDevAssignment2/university/docker-compos
fusion
[+] Running 11/11
 ✔ db Pulled
   ✔ 8b4274ea61c5 Pull complete
   ✔ 65b13290a890 Pull complete
   ✔ 4a0edbf0dd13 Pull complete
   ✔ f8d978a6739e Pull complete
   ✔ 64da6e9087ad Pull complete
   ✔ d4f83991d877 Pull complete
   ✔ 9543fc93cec2 Pull complete
   ✔ 08f5c00014df Pull complete
   ✔ f88be74802d6 Pull complete
   ✔ 883258893faf Pull complete
[+] Building 0.0s (1/1) FINISHED
 => [web internal] load build definition from Dockerfile
 => => transferring dockerfile: 2B
failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory
tamirisabildayeva@T-MacBook-Pro university %
```

Build history | Active builds

Search | Show only

ID	Name	Builder	Status	Duration	Created
V9VHBI	university	desktop-...	Completed	1.4s	8 hours ago
TVSYZC	university	desktop-...	Completed	13.5s	8 hours ago
9JDDWM	university	desktop-...	Completed	1.6s	17 hours ago
T3IE9D	university	desktop-...	Completed	2m 04s	17 hours ago
YS5XP4	WebDevAssignment1	desktop-...	Completed	3.9s	21 days ago

[Assignment 2] Web Application Development by Abildayeva - Google Докумен... | Assignment 2, Web app

Django administration

Site administration

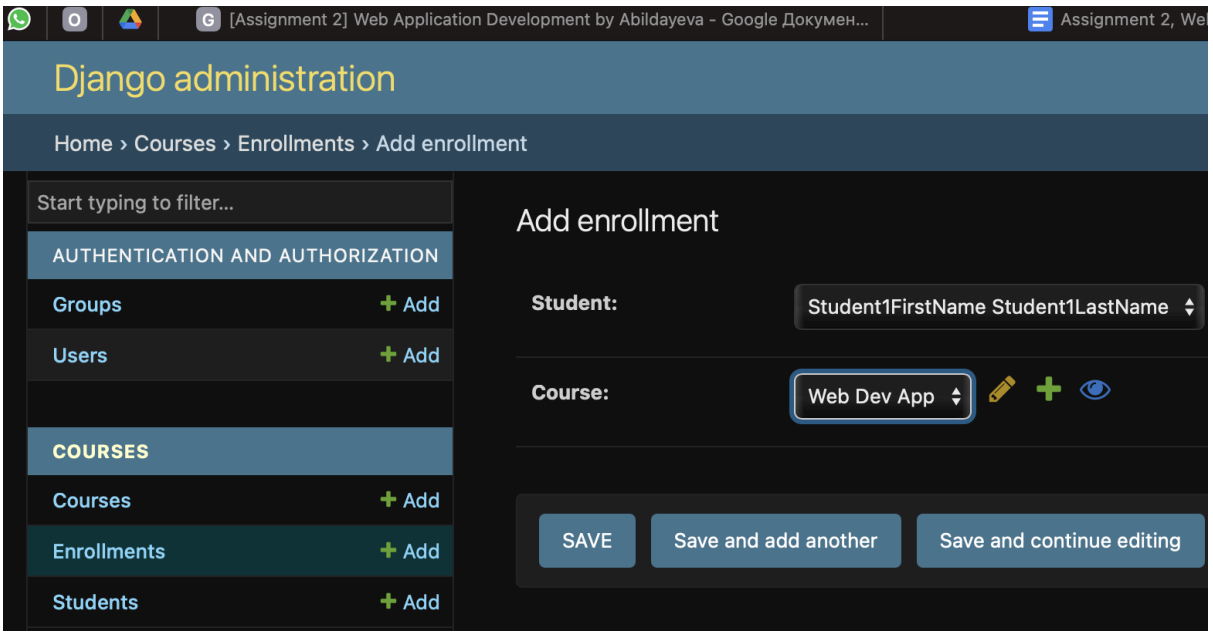
AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

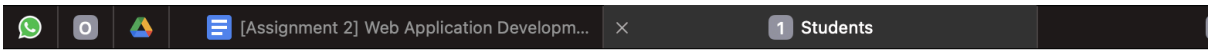
COURSES		
Courses	+ Add	Change
Enrollments	+ Add	Change
Students	+ Add	Change

Recent actions

My actions

None available



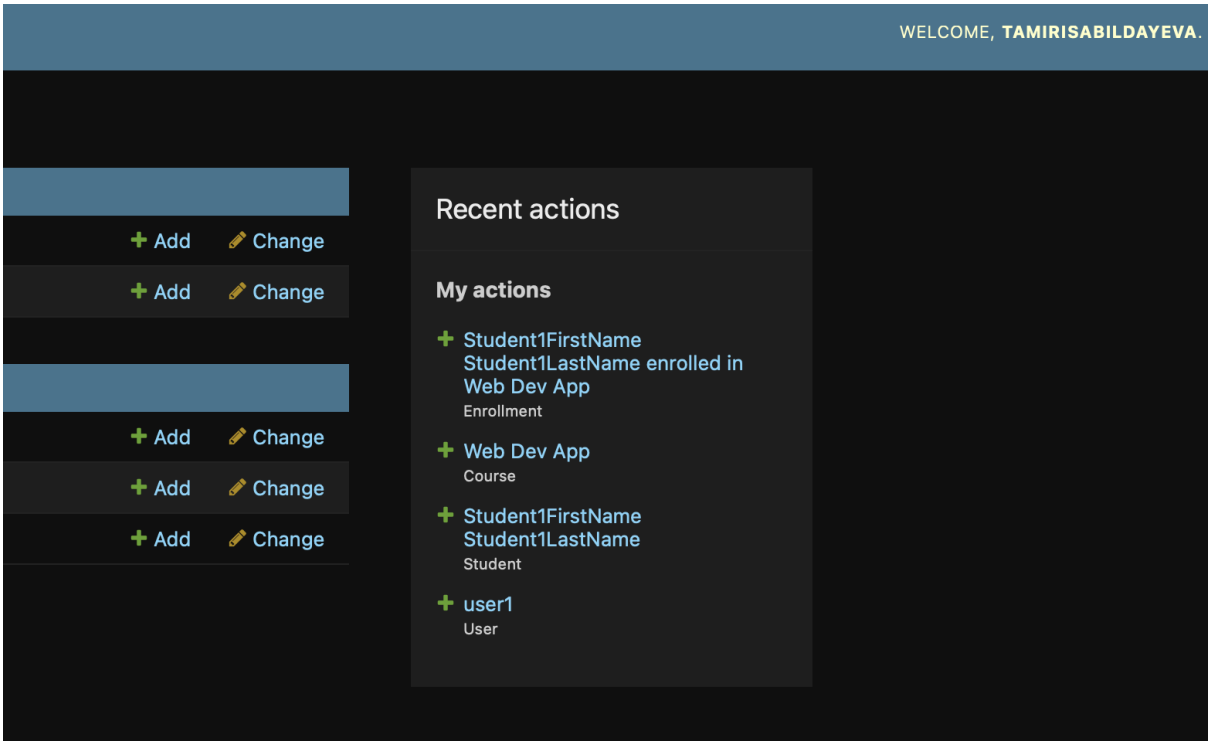


Student List

- Student1FirstName Student1LastName - Student1Email@gmail.com

Course List

- Web Dev App (24SE01)


WELCOME, TAMIRISABILDAYEVA.
Add
Change
Add
Change
Add
Change
Add
Change
Add
Change
Recent actions
My actions
Student1FirstName Student1LastName enrolled in Web Dev App
Enrollment
Web Dev App
Course
Student1FirstName Student1LastName
Student
user1
User