

# Assignment №2

Almabekuly Aibek  
Date: 13.10.2024

Almaty, 2024

## **Introduction**

This project is a web application developed for a fitness center, designed to provide management of gym facilities, trainers, and schedules. Built using Django and Docker, the platform allows users to explore available gyms, book sessions with trainers, and keep track of class schedules through an integrated calendar. The project aims to streamline the fitness center's operations by offering a user-friendly interface for both clients and administrators. Clients can view gym locations, browse trainers' profiles, and check real-time availability through the calendar. Trainers can manage their schedules, view upcoming bookings, and interact with clients. Administrators have full control over managing facilities, updating trainers' information, and scheduling classes or events.

With Docker and Docker Compose, the project is containerized to ensure consistency across different environments, making it easy to deploy and scale as the fitness center grows.

## Assignment Instructions

### 1. Docker Compose

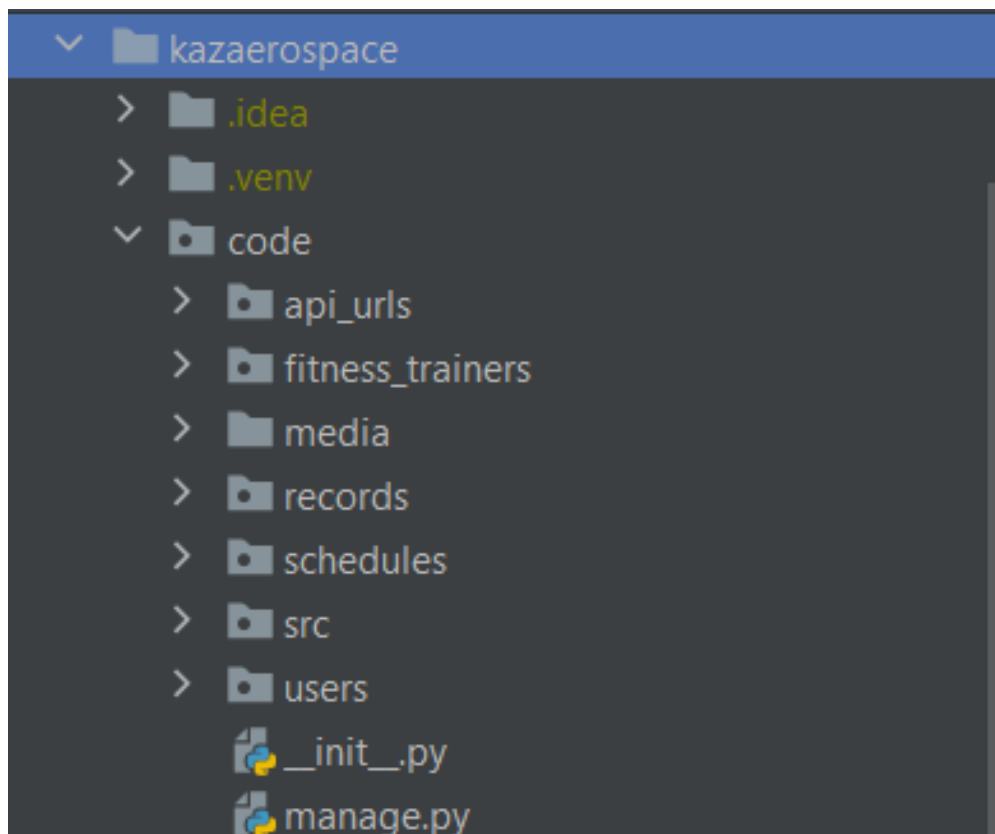
- **Create a Docker Compose File**
  - Create a docker-compose.yml file for your Django application.
  - Include services for:
    - Django web server
    - PostgreSQL database (or another database of your choice)
- **Define Environment Variables**
  - Use environment variables for database configuration (e.g., DB\_NAME, DB\_USER, DB\_PASSWORD).
- **Build and Run the Containers**
  - Use docker-compose up to build and run the application.
  - Ensure that the services are running correctly.
- **Document the Process**
  - Take screenshots of the Docker Compose file and the terminal output during the build and run process.
  - Write a brief explanation of the configurations used.

### Result:

I created some python project based on Django. It started like that:

```
\Master\3-semester\Web_app_dev_MD> django-admin startproject kazaerospace
```

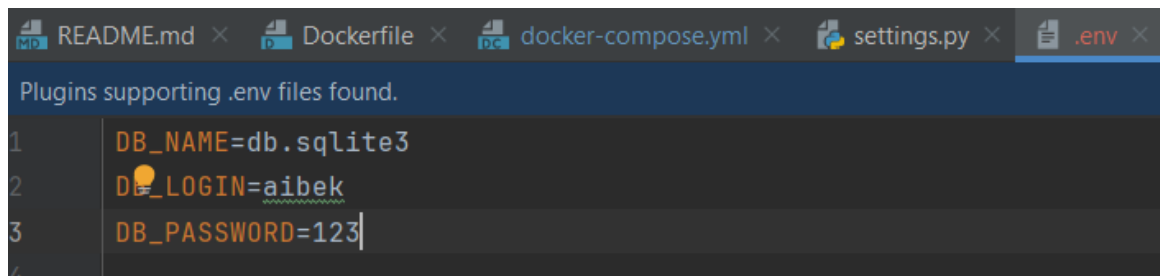
And we have some basic files, which created with our project:



After that we identified database to the project, and we used default database of Django SQLite3:

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': BASE_DIR / 'db.sqlite3',  
    }  
}
```

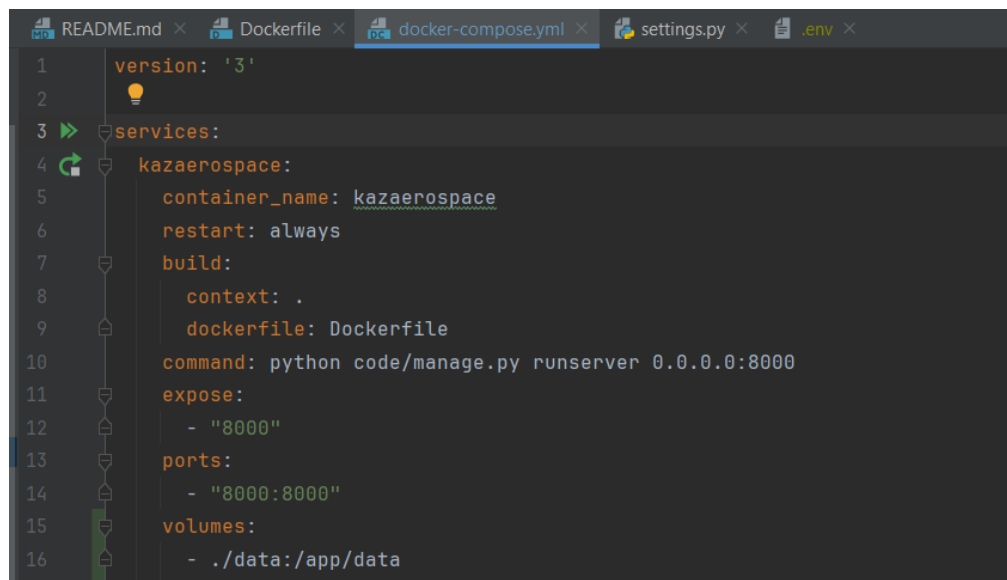
To define login and password to database we will use .env file. ENV file looks like that:



The screenshot shows a code editor with several tabs: README.md, Dockerfile, docker-compose.yml, settings.py, and .env. The .env file is active and contains the following text:

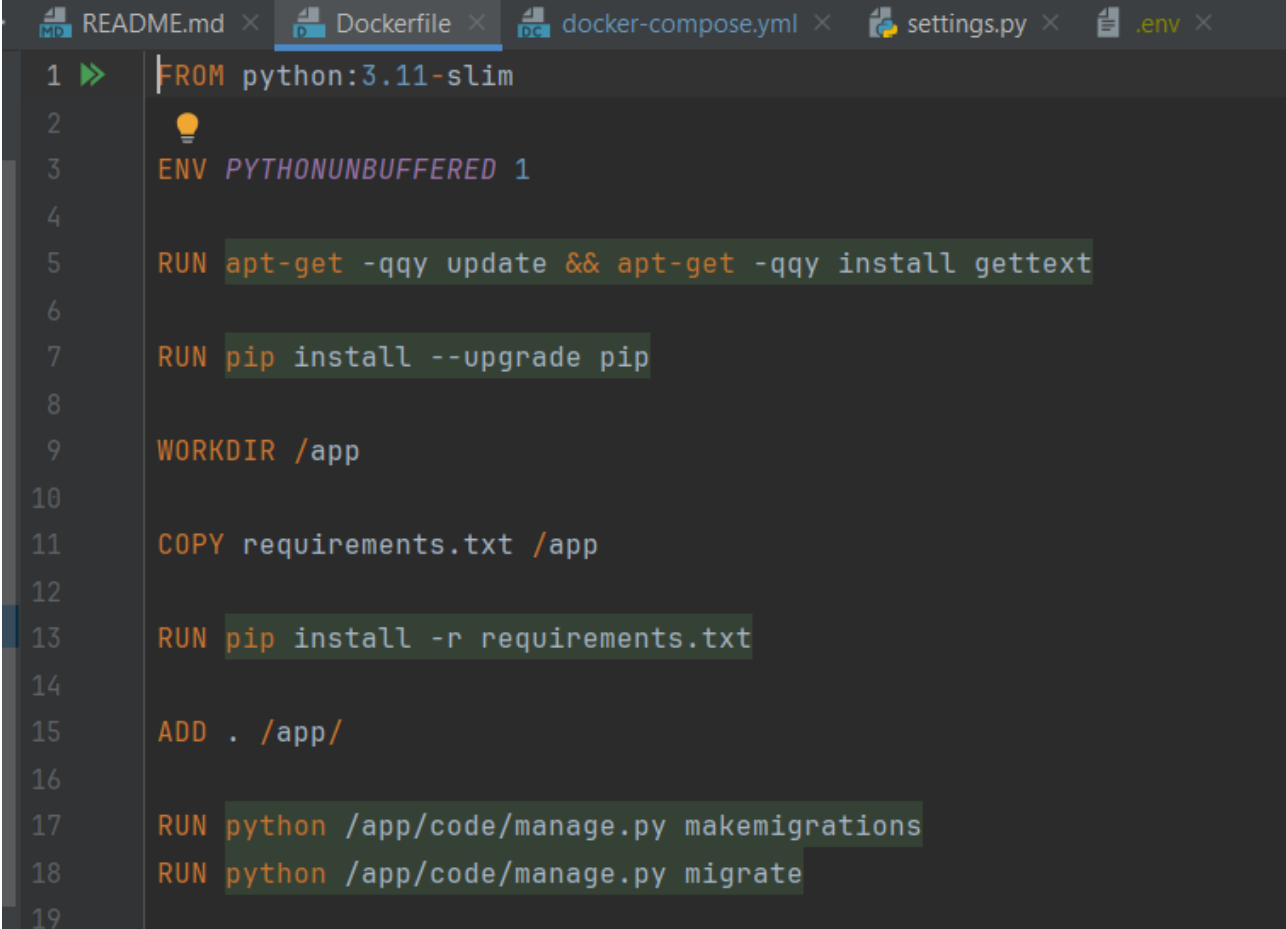
```
1 DB_NAME=db.sqlite3  
2 DB_LOGIN=aibek  
3 DB_PASSWORD=123
```

To launch project we need to create a dockerFile and Docker-compose.yml files. Both files we can see here:



The screenshot shows a code editor with several tabs: README.md, Dockerfile, docker-compose.yml, settings.py, and .env. The docker-compose.yml file is active and contains the following text:

```
1 version: '3'  
2  
3 services:  
4   kazaerospace:  
5     container_name: kazaerospace  
6     restart: always  
7     build:  
8       context: .  
9       dockerfile: Dockerfile  
10    command: python code/manage.py runserver 0.0.0.0:8000  
11    expose:  
12      - "8000"  
13    ports:  
14      - "8000:8000"  
15    volumes:  
16      - ./data:/app/data
```



```
1 FROM python:3.11-slim
2
3 ENV PYTHONUNBUFFERED 1
4
5 RUN apt-get -qq update && apt-get -qq install gettext
6
7 RUN pip install --upgrade pip
8
9 WORKDIR /app
10
11 COPY requirements.txt /app
12
13 RUN pip install -r requirements.txt
14
15 ADD . /app/
16
17 RUN python /app/code/manage.py makemigrations
18 RUN python /app/code/manage.py migrate
19
```

## 2. Docker Networking and Volumes

- **Set Up Docker Networking**

- Define a custom network in your docker-compose.yml file to allow communication between services.
- Verify that the Django app can connect to the database using the network.

- **Implement Docker Volumes**

- Configure a volume in the docker-compose.yml file to persist PostgreSQL data.
- Add a volume for Django to persist uploaded files and static files.

- **Document the Process**

- Take screenshots of the updated docker-compose.yml file, and explain how networking and volumes enhance your application.

### **Result:**

To allow between services we need to use port forwarding between local and global services. Here we used port forwarding:

```

command: python code/manage.py runserver 0.0.0.0:8000
expose:
  - "8000"
ports:
  - "8000:8000"
volumes:
  - ./data:/app/data

```

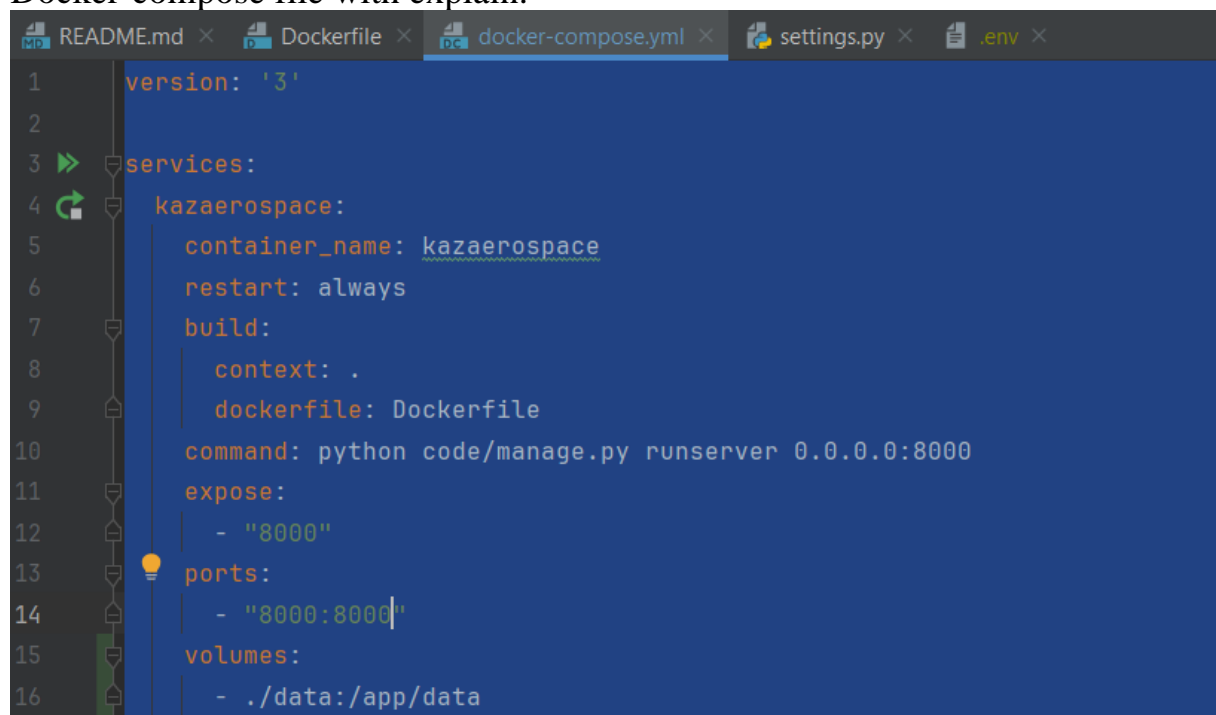
To mount project files, we should create volumes inside of docker-compose file. It helps us to mount certain files:

```

volumes:
  - ./data:/app/data

```

Docker-compose file with explain:



```

1  version: '3'
2
3  services:
4    kazaerospace:
5      container_name: kazaerospace
6      restart: always
7      build:
8        context: .
9        dockerfile: Dockerfile
10     command: python code/manage.py runserver 0.0.0.0:8000
11     expose:
12       - "8000"
13     ports:
14       - "8000:8000"
15     volumes:
16       - ./data:/app/data

```

These Docker settings define a service called kazaerospace:

- container\_name: Specifies the container's name as kaz aerospace.
- restart: Ensures the container always restarts automatically if it crashes.
- build: Build the container from the Dockerfile located in the current directory.
- command: Run the Django development server with the command python code/manage.py runserver 0.0.0.0:8000.
- expose: Expose port 8000 internally for intercontainer communication.
- Ports: Map sport 8000 of the container to port 8000 on the host machine.
- volumes: Mounts the local ./data directory to /app/data inside the container, ensuring data persistence.

### 3. Django Application Setup

- **Create a Django Project**

- Inside the Django service container, create a new Django project using the command `django-admin startproject myproject`.
- Create a simple app (e.g., blog) with at least one model and a corresponding view.

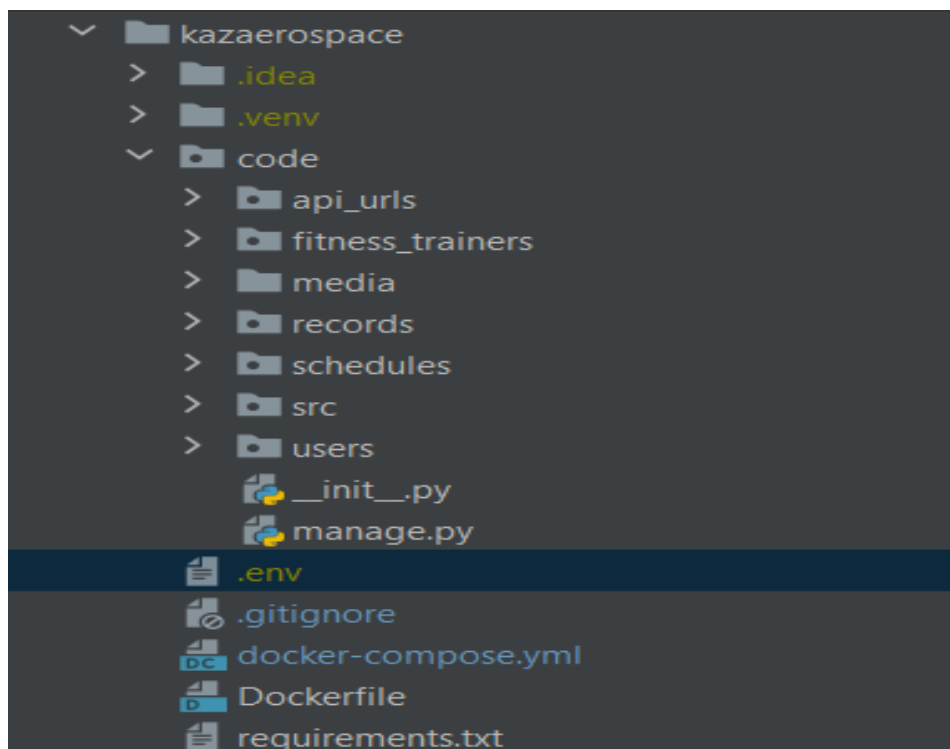
- **Configure the Database**

- Update the Django settings to use the PostgreSQL database configured in your Docker Compose setup.
- Run migrations to set up the database schema.

- **Document the Process**

- Take screenshots of the project structure, model definition, and any migrations.
- Explain how the Django application is structured and how it interacts with Docker.

**Project created for using in backend, and project files and project looks like that:**





Here we can see swagger documentation of our project.



## **Conclusion**

This project successfully meets the requirement of utilizing Docker and Docker Compose to containerize the fitness center management system. By leveraging these tools, the application ensures consistent and efficient deployment across different environments, simplifying the setup process for both development and production. Docker Compose streamlines the orchestration of services, enabling the seamless interaction between components such as the web server and the database. This containerized approach provides scalability, flexibility, and ease of maintenance, making the system adaptable to the evolving needs of the fitness center.