**Report**

**Title of the Assignment: Assignment 2, Web app dev**

**Student’s Name**: **Abylay Aitbanov 23MD0377**

**Date of Submission: 13.10.2024**

**Table of Contents**

**1. Introduction**

**2. Docker Compose**

**2.1. Breakdown of the Configuration**

**2.2. Configuration of Dockerfile**

**2.3. Build and Run**

**3. Docker Networking and Volumes**

**3.1. Networking**

**3.2. Volumes**

**3.3. Findings**

**4. Django Application Setup**

**4.1. Project Structure**

**4.2. Database Configuration**

**4.3. Findings**

**5. Conclusion**

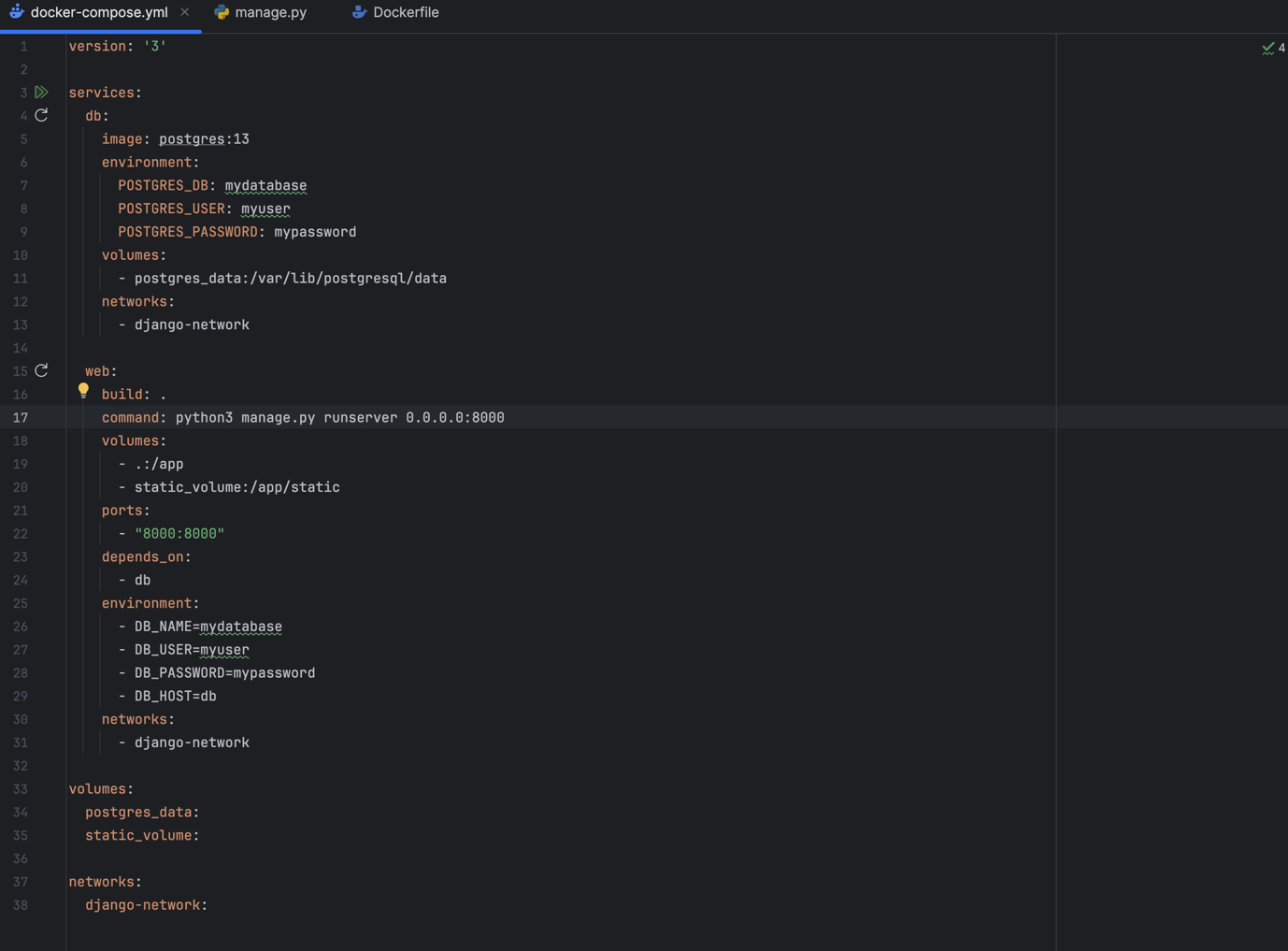
**6. References**

**Introduction**

**The goal of this assignment is to gain hands-on experience with Django and Docker, focusing on Docker Compose, Docker networking, and volumes. Students will set up a Django application within a Docker environment and document the process.**

**Docker Compose**

**Configuration**

****

**In this assignment, it includes two main services: db for PostgreSQL and web for the Django application. The configuration specifies environment variables, network settings, and volume mappings to ensure data persistence and proper communication between services.**

**Breakdown of the Configuration**

**db:**

**image: postgres:13 - > it uses the official PostgreSQL image with version 13**

**environment:**

**POSTGRES\_DB: mydatabase**

**POSTGRES\_USER: myuser**

**POSTGRES\_PASSWORD: mypassword**

**POSTGRES\_DB: The name of the database that will be created when the PostgreSQL container is initialized.**

**POSTGRES\_USER: The username for connecting to the PostgreSQL database.**

**POSTGRES\_PASSWORD: The password for the specified user. This is crucial for database authentication.**

**volumes:**

**- postgres\_data:/var/lib/postgresql/data -> Maps a named volume (postgres\_data) to the PostgreSQL data directory inside the container**

**networks:**

**- django-network - > Connects the db service to a custom network named django-network**

**web:**

**build: . -> Docker image for this service will be built from the current directory**

**command: python3 manage.py runserver 0.0.0.0:8000 -> command to run when the container starts**

**volumes:**

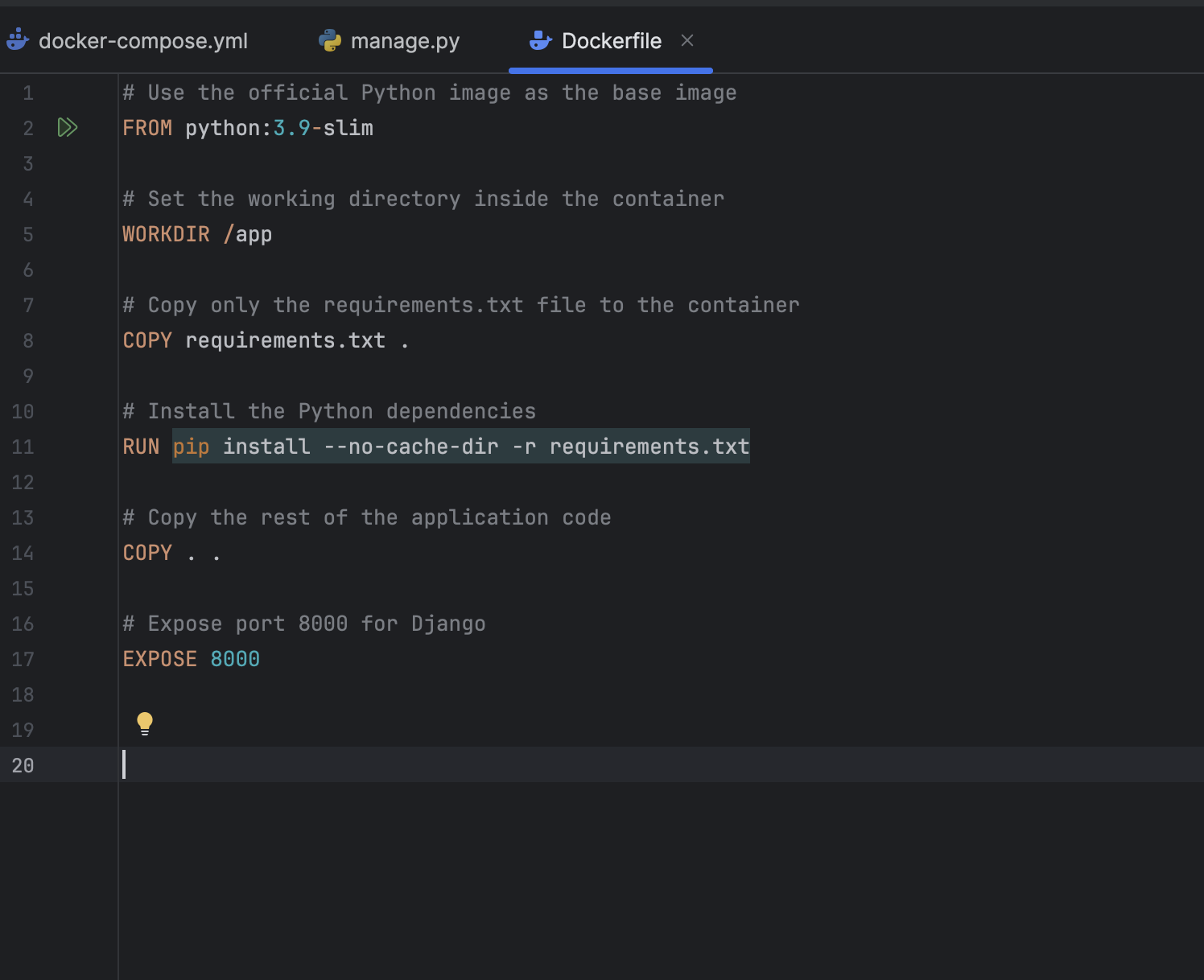
**- .:/app -> mounts the current directory**

**- static\_volume:/app/static -> maps a named volume to the static files**

**ports:**

**- "8000:8000" -> port 8000 on the host machine, mapping it to port 8000 on the container**

**Configuration of Dockerfile**

****

**FROM python:3.9-slim -> official Python image version 3.9, with a “slim” variant that is smaller in size**

**WORKDIR /app ->** **This directive sets the working directory inside the container to /app**

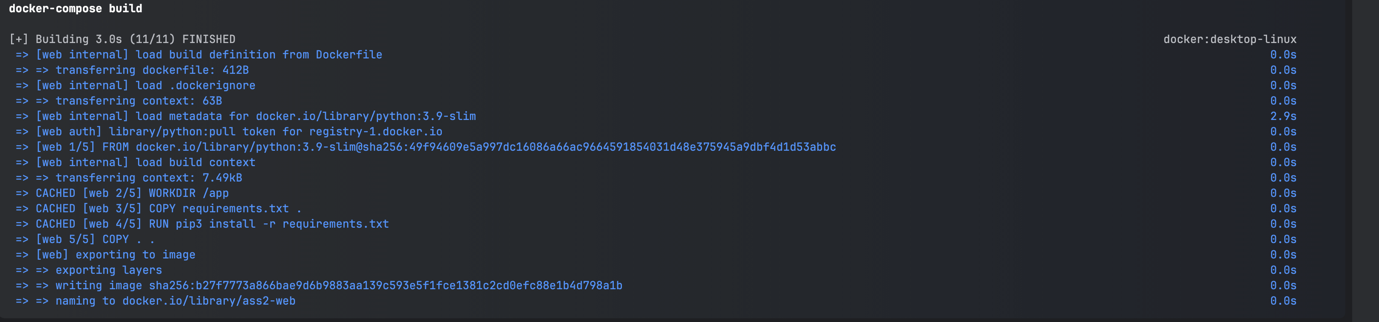
**COPY requirements.txt . -> This command copies files from the host machine into the Docker image**

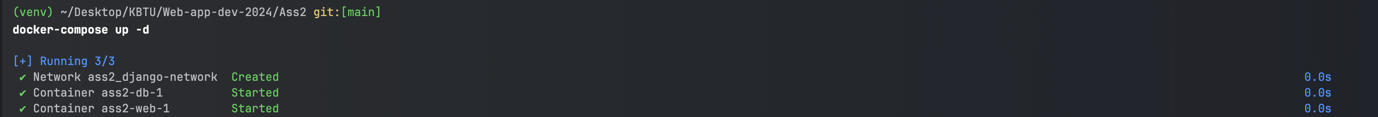
**RUN pip install --no-cache-dir -r requirements.txt -> to install the dependencies listed in requirements.txt. The --no-cache-dir option is used to prevent pip from caching**

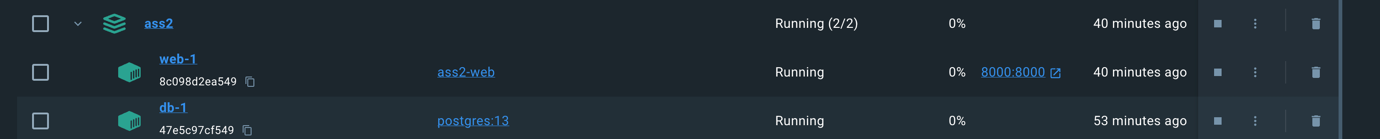
**COPY . . -> This command copies all files and directories from the current directory on the host machine into the current working directory**

**EXPOSE 8000 -> This is the port used by the Django development server**

**Build and Run**

****

****

****

**docker-compose up –build ->** **This command compiles the images defined in the Dockerfile and starts the services in the docker-compose.yml.**

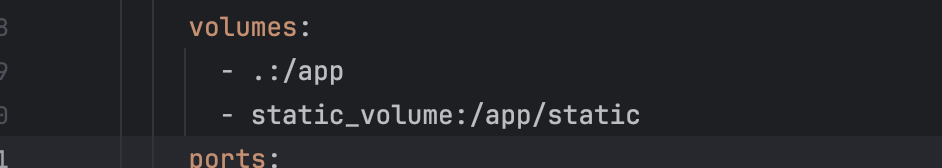
**Docker Networking and Volumes**

**Networking**

****

**Networking allows to communication between containers. In this setup, a network named django-network was created to communicate between the Django web service and the PostgreSQL database service. This setup enhances security and isolates container traffic from the host network.**

**Volumes**

****

****

**Volumes were implemented to ensure data persistence for the PostgreSQL database. The postgres\_data volume maps to the database’s data directory, ensuring that data remains intact across container restarts.**

**Findings**

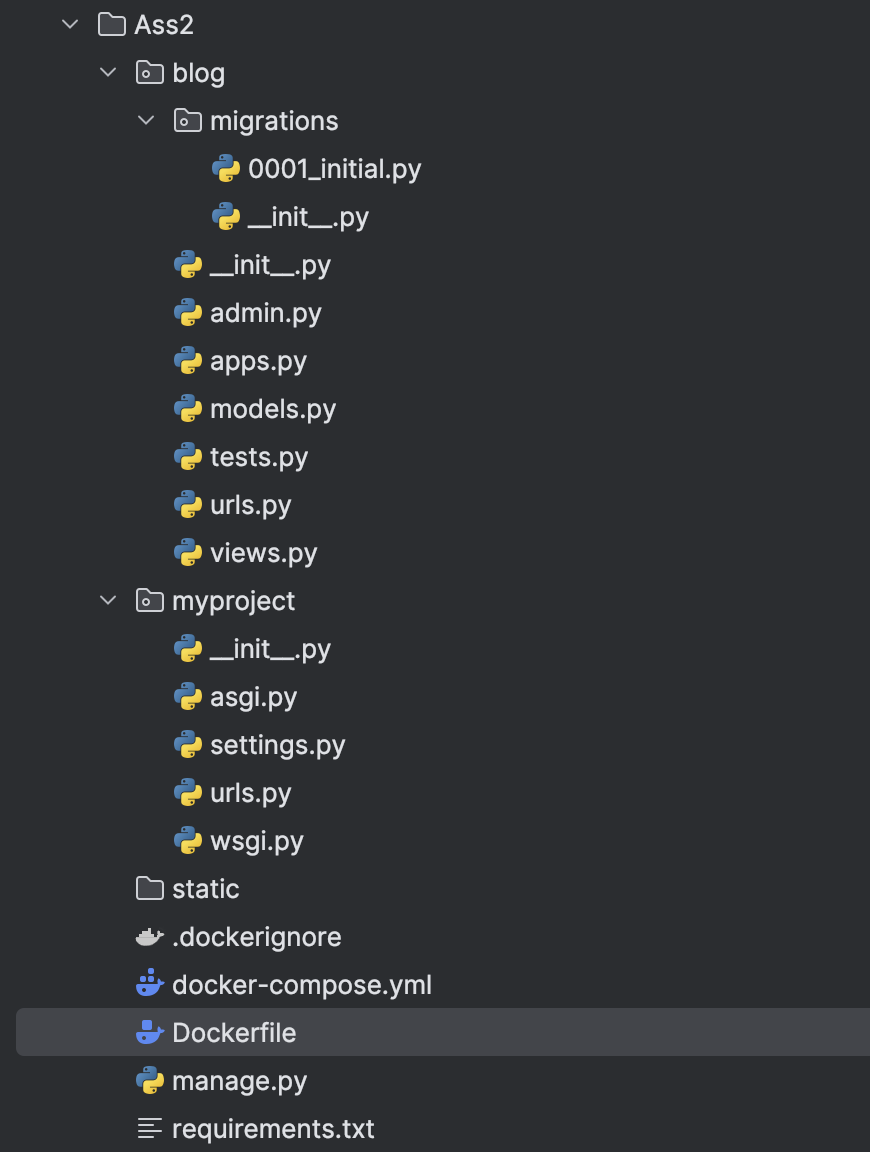
**The use of Docker networking and volumes provides significant advantages:**

**Data Persistence -> volumes ensure that the database retains data**

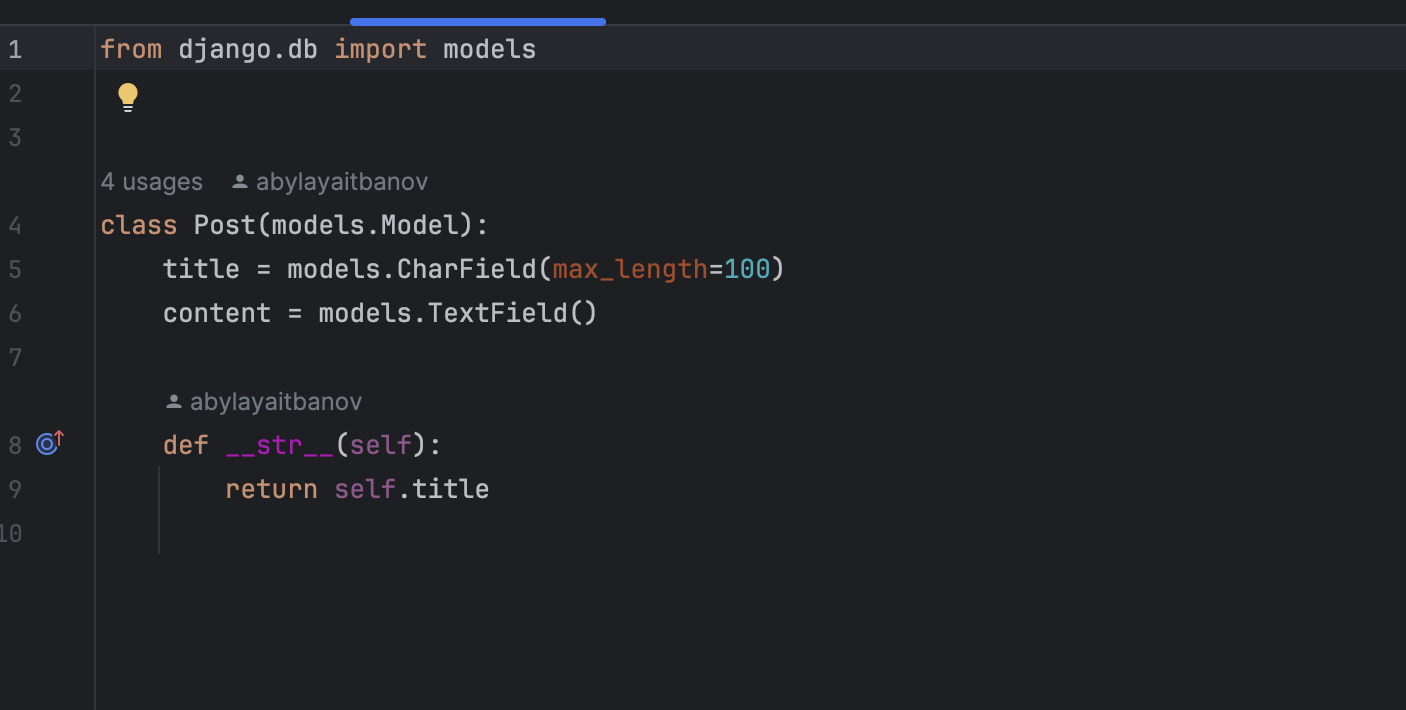
**Isolation - > custom networks enhance security by isolating communication between containers**

**Django Application Setup**

**Project Structure**

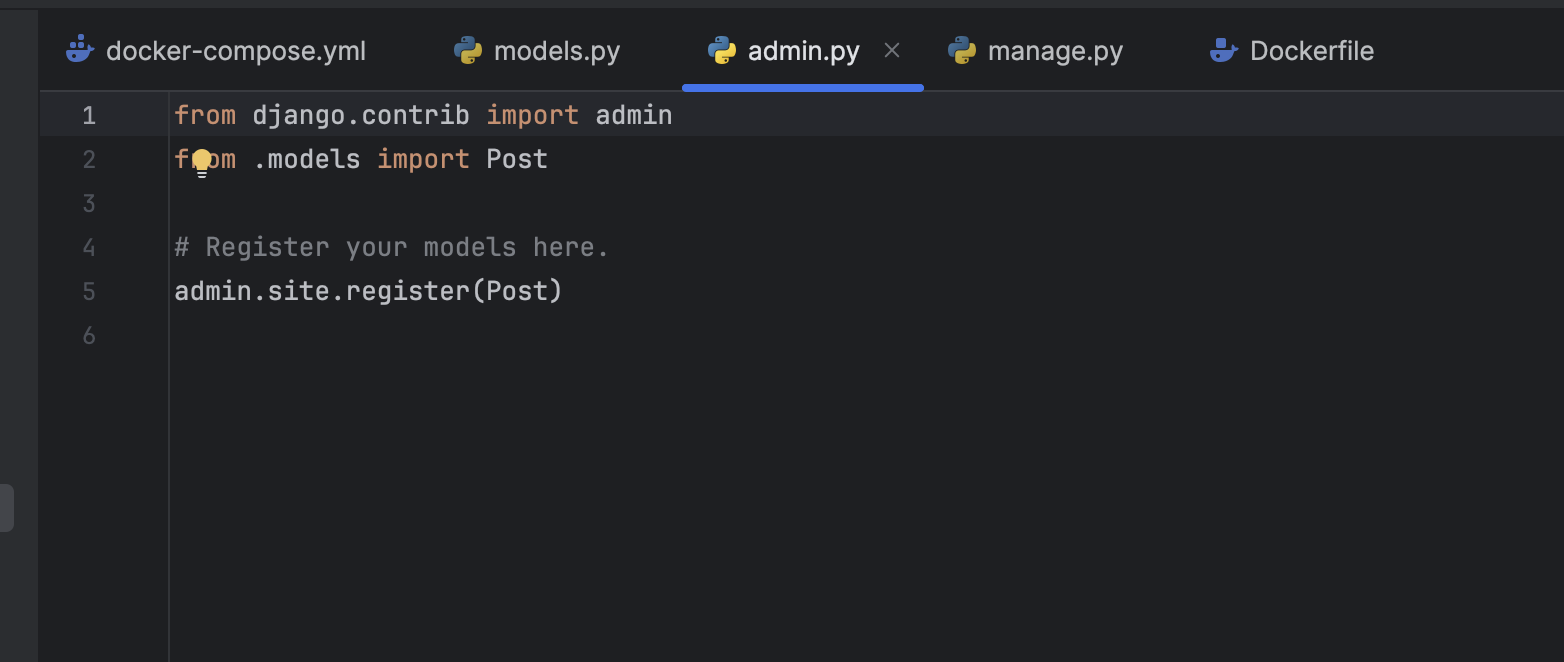
****

**Standard structure -> including essential directories for applications, static files, and templates. The main application, blog, contains models, views, and URLs for managing blog posts.**

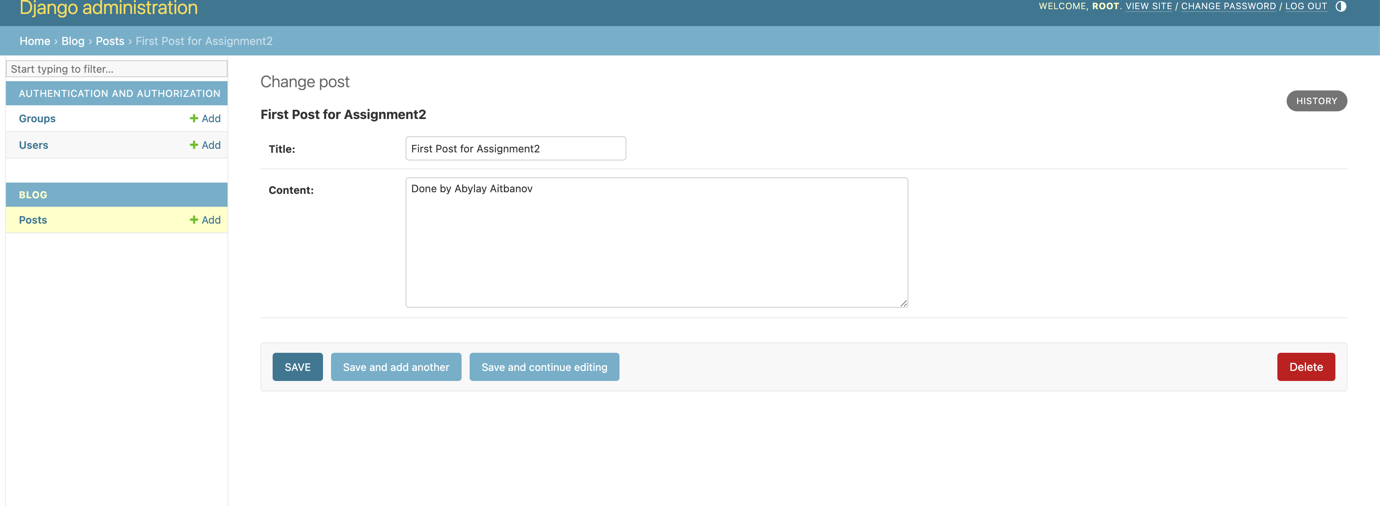
****

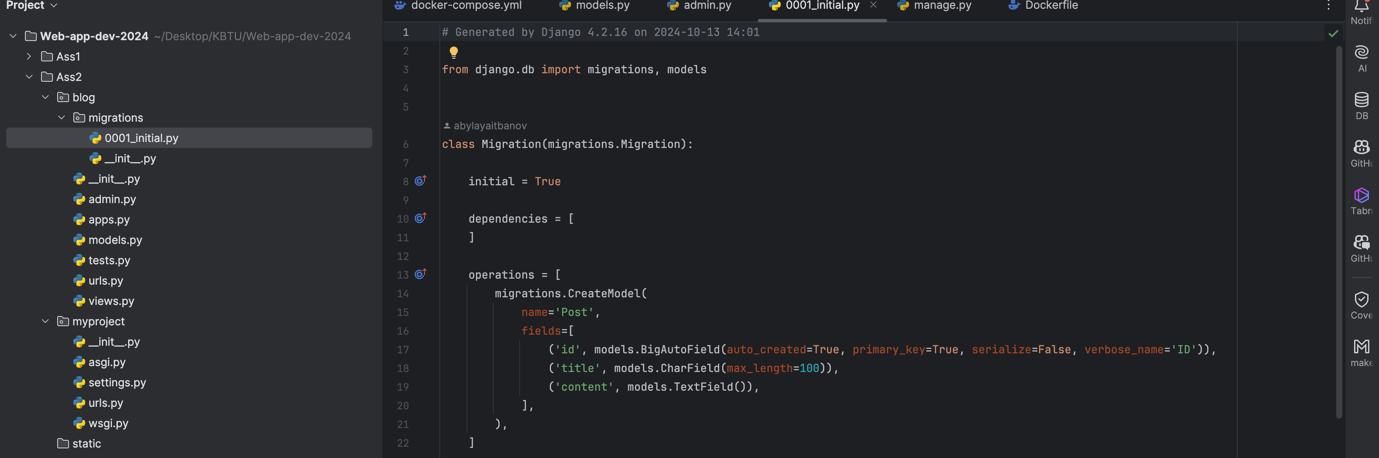
**In blog.models we have simple model Post that habe 2 attributes title and content,**

**And resulting self.title**

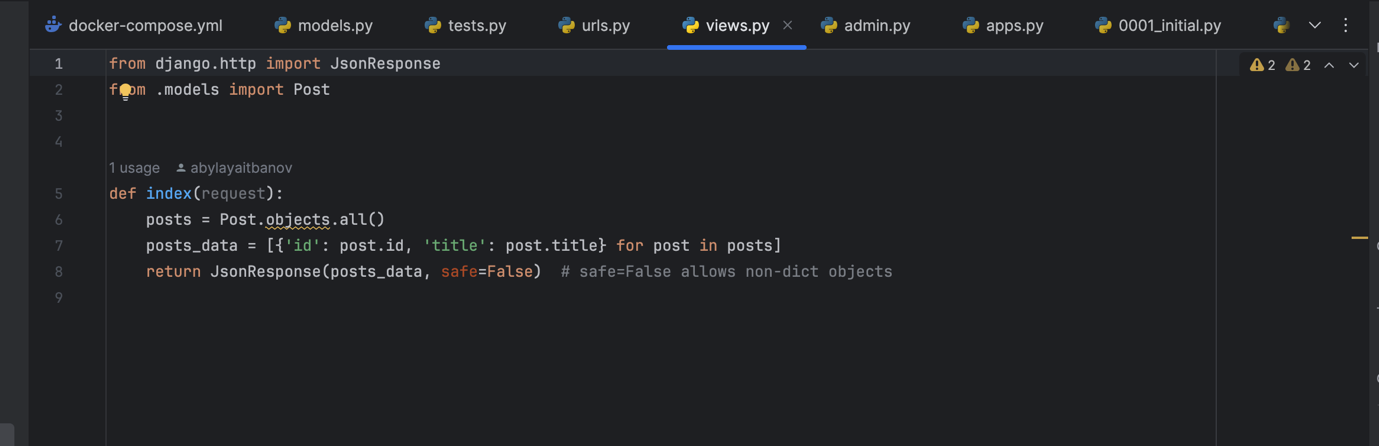
****

**In blog.admins I have register part, to create some posts in admin panel**

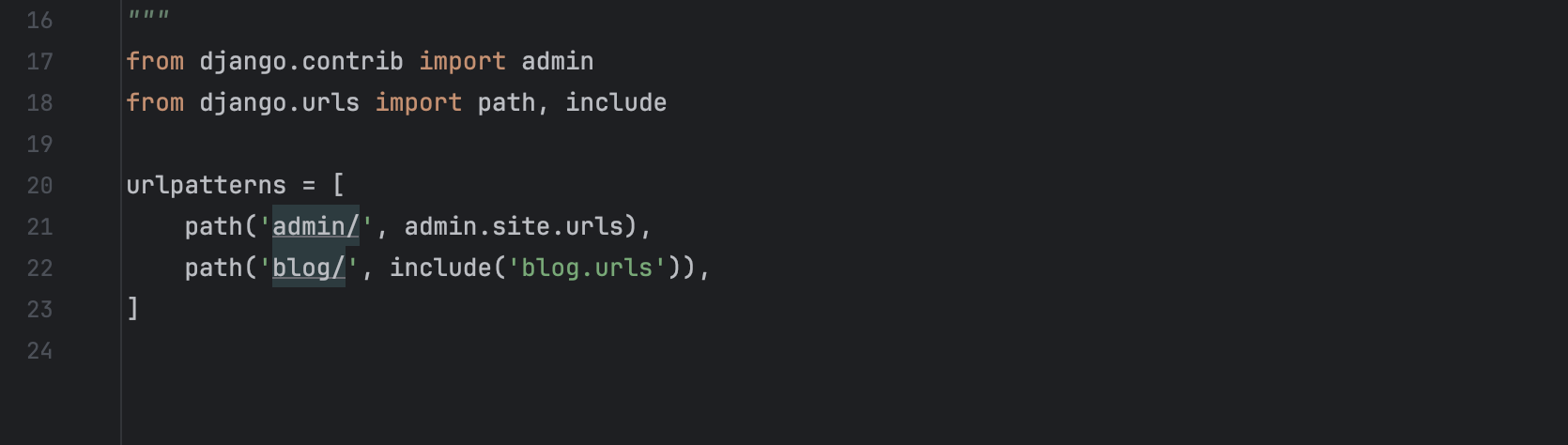
****

****

**Here we have migration files, that shows us the creation of model**

****

**Here we have blog.views file that take all Post object file from Db and return JsonResponse**

****

**Main urls.py file where we redirect out requests that we need, here we have admin/, and blog/ urls**

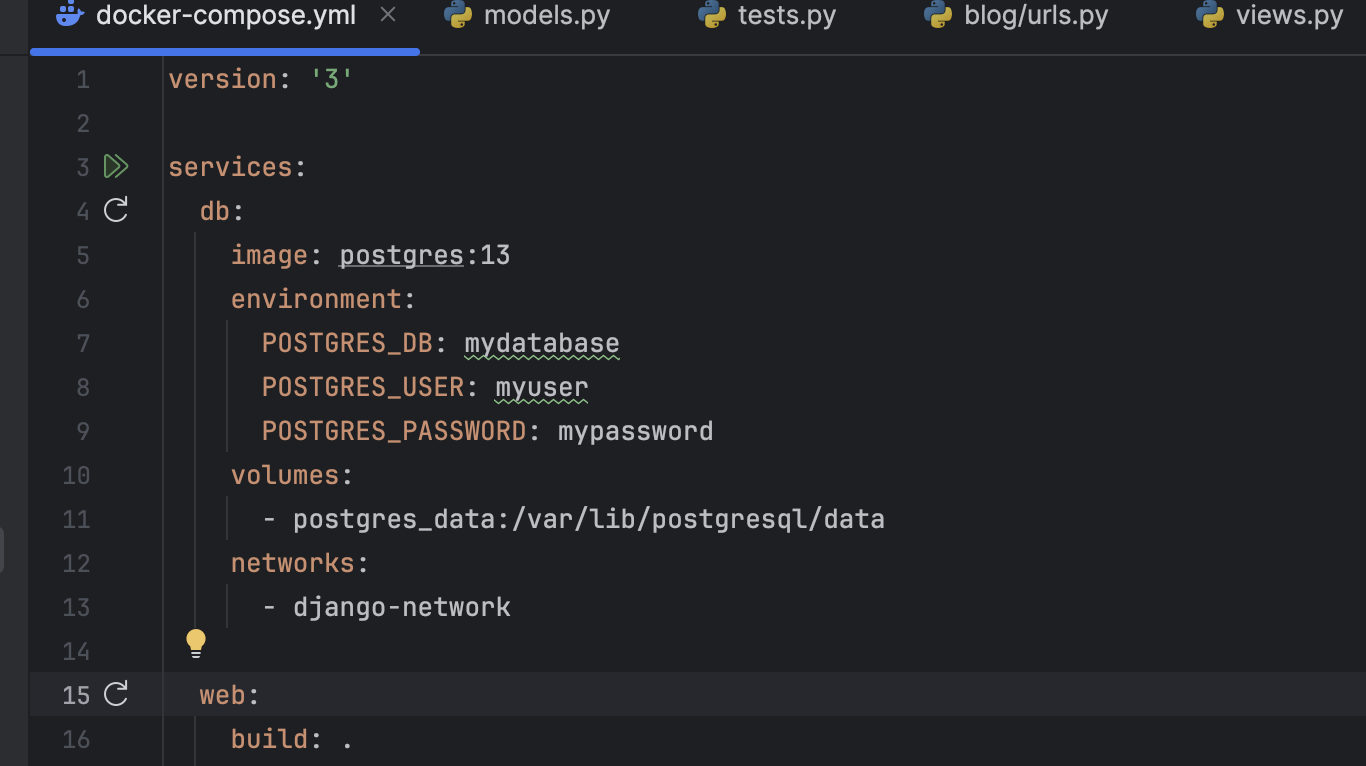
**docker-compose run web django-admin startproject myproject . -> command to start Django project**

**docker-compose run web python manage.py startapp blog -> to create app (blog)**

**Database Configuration**

**The PostgreSQL database is integrated with Django through the settings defined in settings.py**

****

****

**All the os.environ was define in docker-compose file**

**Findings**

**Developing a Django application within Docker has provided numerous benefits, including consistent development environments and** **ability to easily scale and manage services**

**Conclusion**

**This assignment shows that the key learnings from setting up a Django application using Docker. The combination of Docker and Django not only streamlines the development process but also improves application deployment and management.**

**References**

**Official Docker docs:** [**https://docs.docker.com/**](https://docs.docker.com/)

**Official Django docs:** [**https://www.djangoproject.com/**](https://www.djangoproject.com/)

**My github page: https://github.com/AA19BD**