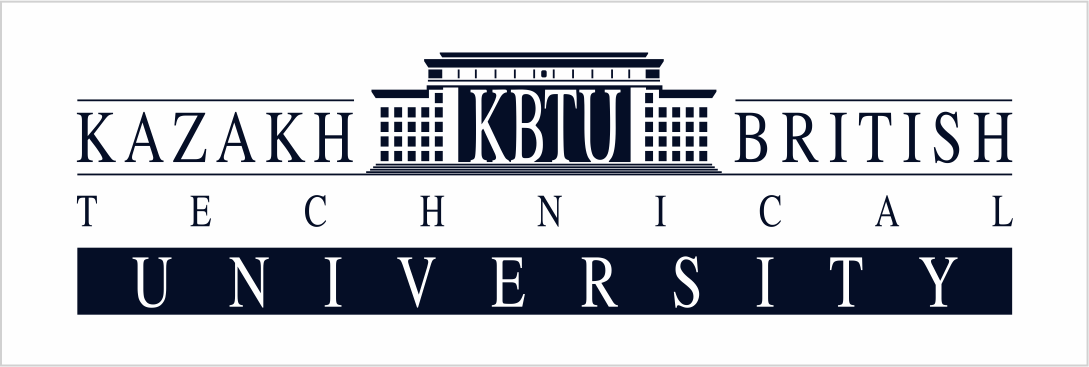
****

**Assignment №2**

**Docker compose**

**Prepared by Azimkhanov Bauyrzhan**

**Almaty, 13.10.2024**

**Table of contents**

Introduction 3

Docker compose 3

Docker networking and volumes 7

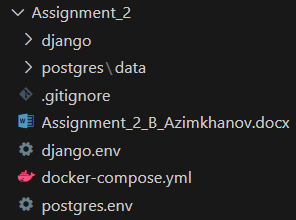
Django application setup 9

Conclusion 12

References 13

**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**

services:

  django:

    build: django/.

    ports:

      - 8000:8000

    env\_file:

      - ./django.env

    command:

    depends\_on:

      postgres:

        condition: service\_healthy

        restart: true

    networks:

      - net

    volumes:

      - ./django:/usr/src/app

  postgres:

    image: postgres:16.4-bullseye

    ports:

      - 5432:5432

    env\_file:

      - ./postgres.env

    healthcheck:

      test: ["CMD-SHELL", "pg\_isready -U bauyrzhan -d assignment2"]

      interval: 10s

      retries: 3

      start\_period: 30s

      timeout: 10s

    networks:

      - net

    volumes:

      - ./postgres/data:/var/lib/postgresql/data

networks:

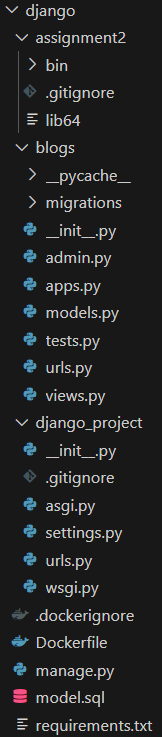
  net:

    driver: bridge

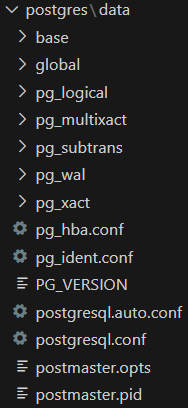
This docker-compose.yml file defines a multi-container Docker application with two services: django and postgres. It also sets up a custom network named net.

**Services**

1. **django**
   * **Build context**: The django service is built from the django/ directory.
   * **Ports**: Maps port 8000 on the host to port 8000 in the container.
   * **Environment variables**: Loaded from the ./django.env file.
   * **Dependencies**: Depends on the postgres service being healthy before starting.
   * **Restart Policy**: Always restart the container if it stops.
   * **Networks**: Connected to the net network.
   * **Volumes**: Mounts the ./django directory on the host to /usr/src/app in the container.



1. **postgres**
   * **Image**: Uses the postgres:16.4-bullseye image.
   * **Ports**: Maps port 5432 on the host to port 5432 in the container.
   * **Environment variables**: Loaded from the ./postgres.env file.
   * **Healthcheck**:
     + **Test**: Runs the command “pg\_isready -U bauyrzhan -d assignment2” to check if the database is ready.
     + **Interval**: Checks every 10 seconds.
     + **Retries**: Retries up to 3 times before considering the service unhealthy.
     + **Start Period**: Waits 30 seconds before starting health checks.
     + **Timeout**: Each health check command must complete within 10 seconds.
   * **Networks**: Connected to the net network.
   * **Volumes**: Mounts the ./postgres/data directory on the host to /var/lib/postgresql/data in the container.



**Networks**

* **net**: A custom bridge network used to facilitate communication between the django and postgres services.

This setup ensures that the Django application can communicate with the PostgreSQL database, and the health of the PostgreSQL service is monitored before the Django service starts.

The entire database state is preserved by mounting volumes in the Postgres service. Similarly, mounting helps dynamically apply and reflect changes in the Django project code.

**Build and run**

After writing Dockerfiles and other code, you need to run the "docker compose build" command, and then the "docker compose up" command.

**Docker networking and volumes**

**1. Custom network setup and its benefits**

The docker-compose.yml file defines a custom network named net using the bridge driver. This network facilitates communication between the django and postgres services. Here are the benefits of this setup:

* **Isolation**: The custom network isolates the services from other containers running on the host, enhancing security.
* **Service discovery**: Docker automatically assigns each service a hostname based on the service name, making it easy for services to discover and communicate with each other.
* **Simplified configuration**: By using a custom network, you avoid the need to manually configure IP addresses and ports for inter-service communication.

**2. Volumes for data persistence**

Volumes are used in the docker-compose.yml file to ensure data persistence for both the django and postgres services:

* **Django service**:
  + The volume ./django:/usr/src/app mounts the ./django directory on the host to /usr/src/app in the container. This allows the Django application code to be accessible and editable from the host machine.
* **Postgres service**:
  + The volume ./postgres/data:/var/lib/postgresql/data mounts the ./postgres/data directory on the host to /var/lib/postgresql/data in the container. This ensures that the PostgreSQL database data is stored persistently on the host, even if the container is stopped or removed.

**3. Advantages of using Docker networking and volumes**

Using Docker networking and volumes in your application offers several advantages:

* **Networking**:
  + **Enhanced security**: Custom networks isolate containers, reducing the risk of unauthorized access.
  + **Ease of communication**: Services can easily communicate using service names, simplifying the configuration.
  + **Scalability**: Networks can be easily scaled to include more services or containers as needed.
* **Volumes**:
  + **Data persistence**: Volumes ensure that data is not lost when containers are stopped or removed.
  + **Separation of concerns**: Application code and data are managed separately, making it easier to update and maintain the application.
  + **Performance**: Volumes are optimized for performance, providing faster data access compared to bind mounts.

These features make Docker a powerful tool for developing, deploying, and managing applications in a consistent and efficient manner.

**Django application setup**

**Project structure**

The Django project is created using two articles from the official Django documentation. Links to both articles are provided in the corresponding section.

This application has a blogs app with corresponding views and models. It is connected to the main module (django\_project directory) via the urls.py file in blog.

from django.contrib import admin

from django.urls import include, path

urlpatterns = [

    path("blogs/", include("blogs.urls")),

    path('admin/', admin.site.urls),

]

from django.urls import path

from . import views

urlpatterns = [

    path("", views.index, name="index"),

]

from django.shortcuts import render

from django.http import HttpResponse

def index(request):

    return HttpResponse("Hello, world. You're at the blogs index.")

from django.db import models

class Question(models.Model):

    question\_text = models.CharField(max\_length=200)

    pub\_date = models.DateTimeField("date published")

class Choice(models.Model):

    question = models.ForeignKey(Question, on\_delete=models.CASCADE)

    choice\_text = models.CharField(max\_length=200)

    votes = models.IntegerField(default=0)

BEGIN;

--

-- Create model Question

--

CREATE TABLE "blogs\_question" ("id" bigint NOT NULL PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY, "question\_text" varchar(200) NOT NULL, "pub\_date" timestamp with time zone NOT NULL);

--

-- Create model Choice

--

CREATE TABLE "blogs\_choice" ("id" bigint NOT NULL PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY, "choice\_text" varchar(200) NOT NULL, "votes" integer NOT NULL, "question\_id" bigint NOT NULL);

ALTER TABLE "blogs\_choice" ADD CONSTRAINT "blogs\_choice\_question\_id\_c774bda9\_fk\_blogs\_question\_id" FOREIGN KEY ("question\_id") REFERENCES "blogs\_question" ("id") DEFERRABLE INITIALLY DEFERRED;

CREATE INDEX "blogs\_choice\_question\_id\_c774bda9" ON "blogs\_choice" ("question\_id");

COMMIT;

**Database configuration**

The PostgreSQL database is integrated with Django via the postgresql-client library in the Django Docker image and the Python3 pip module psycopg2-binary. They are installed at the image build stage (docker compose build) on the 3rd and 10th lines, respectively.

FROM python:3.11

RUN apt-get update \

    && apt-get install -y --no-install-recommends \

        postgresql-client \

    && rm -rf /var/lib/apt/lists/\*

WORKDIR /usr/src/app

COPY requirements.txt ./

RUN pip install -r requirements.txt

COPY . .

EXPOSE 8000

CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

Further configuration is performed in the settings.py file in the DATABASES structure (line 89). Data on the database name and other things are taken from the environment variable of the PostgreSQL container. These variables are set at the Docker compose level (docker compose up) in the env\_file fields.

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.postgresql',

        'NAME': os.environ.get('POSTGRES\_NAME'),

        'USER': os.environ.get('POSTGRES\_USER'),

        'PASSWORD': os.environ.get('POSTGRES\_PASSWORD'),

        'HOST': os.environ.get('POSTGRES\_HOST'),

        'PORT': os.environ.get('POSTGRES\_PORT')

    }

}

**Findings**

You need to pay close attention to the order of launching containers in Docker compose. Otherwise, errors may occur when starting Django.

To avoid such incidents, you need to declare checks of the state and readiness (healthcheck field in docker-compose.yml) of the container with the database for work. After all, database initialization can take longer than a similar process for the application. Since a ready and running database is required to launch the application (depends\_on field in docker-compose.yml), I included the healthcheck field.

**Conclusion**

**Key learnings from the assignment**

This assignment highlighted the importance of using Docker Compose to manage multi-container applications. By defining services, networks, and volumes in a single file, we can streamline the setup and deployment process, ensuring consistency across different environments.

**Significance of using Docker with Django for application development**

Using Docker with Django simplifies the development and deployment process by encapsulating the application and its dependencies in containers. This approach ensures that the application runs consistently, regardless of the underlying environment. It also facilitates easier scaling and management of the application components.

**Significance of healthchecks for containers with databases**

Implementing healthchecks for containers, especially those running databases, is crucial. Healthchecks ensure that the database service is fully operational before dependent services, like Django, start. This prevents potential errors and ensures a smooth startup process, enhancing the reliability and stability of the application.

**References**

Links to Django tutorials from official documentation:

1. <https://docs.djangoproject.com/en/5.1/intro/tutorial01/>
2. <https://docs.djangoproject.com/en/5.1/intro/tutorial02/>

Link to official Docker documentation:

* <https://docs.docker.com/>

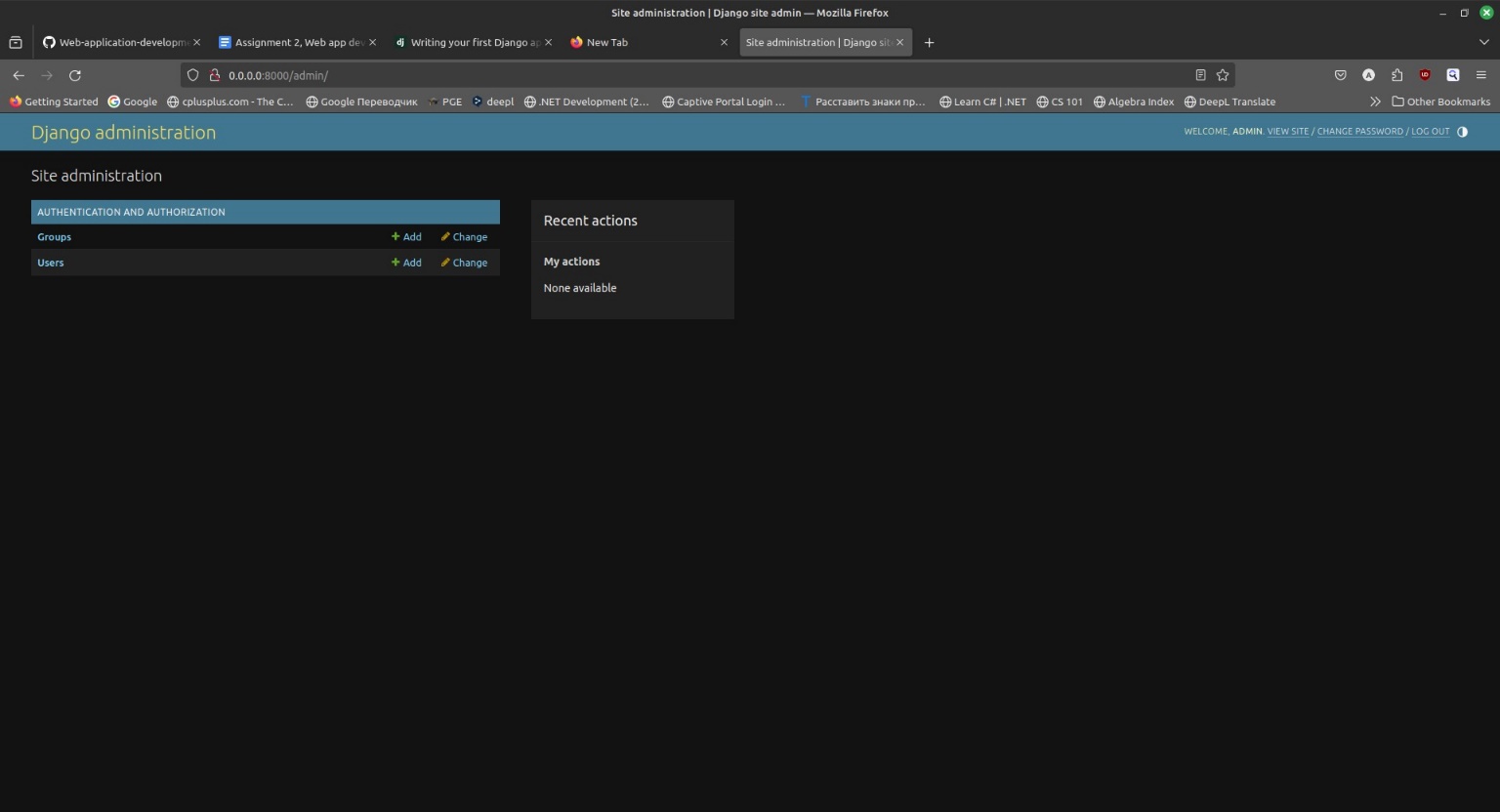
Link to official Docker image registry where from I took base images for compose:

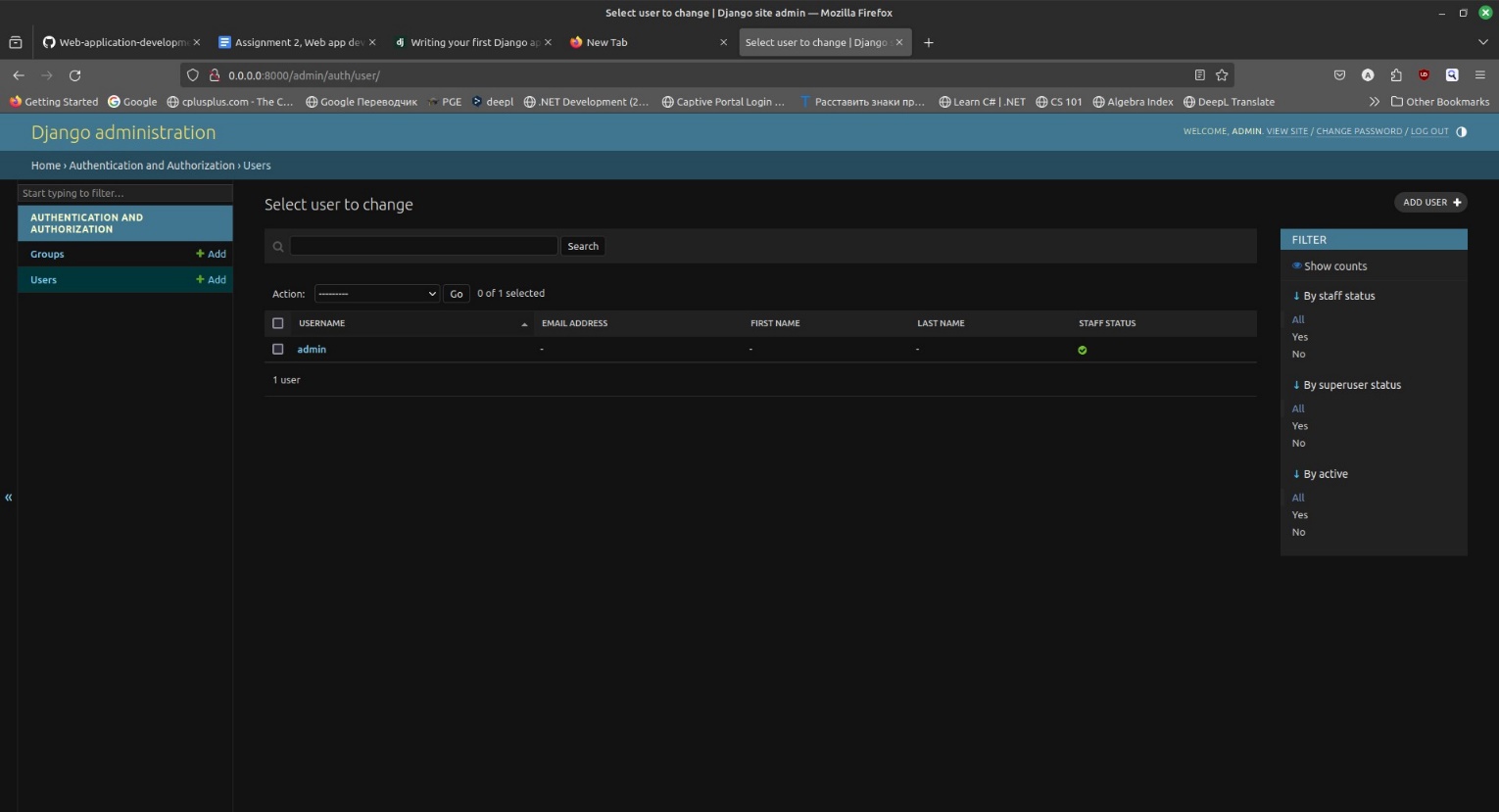
* <https://hub.docker.com/>

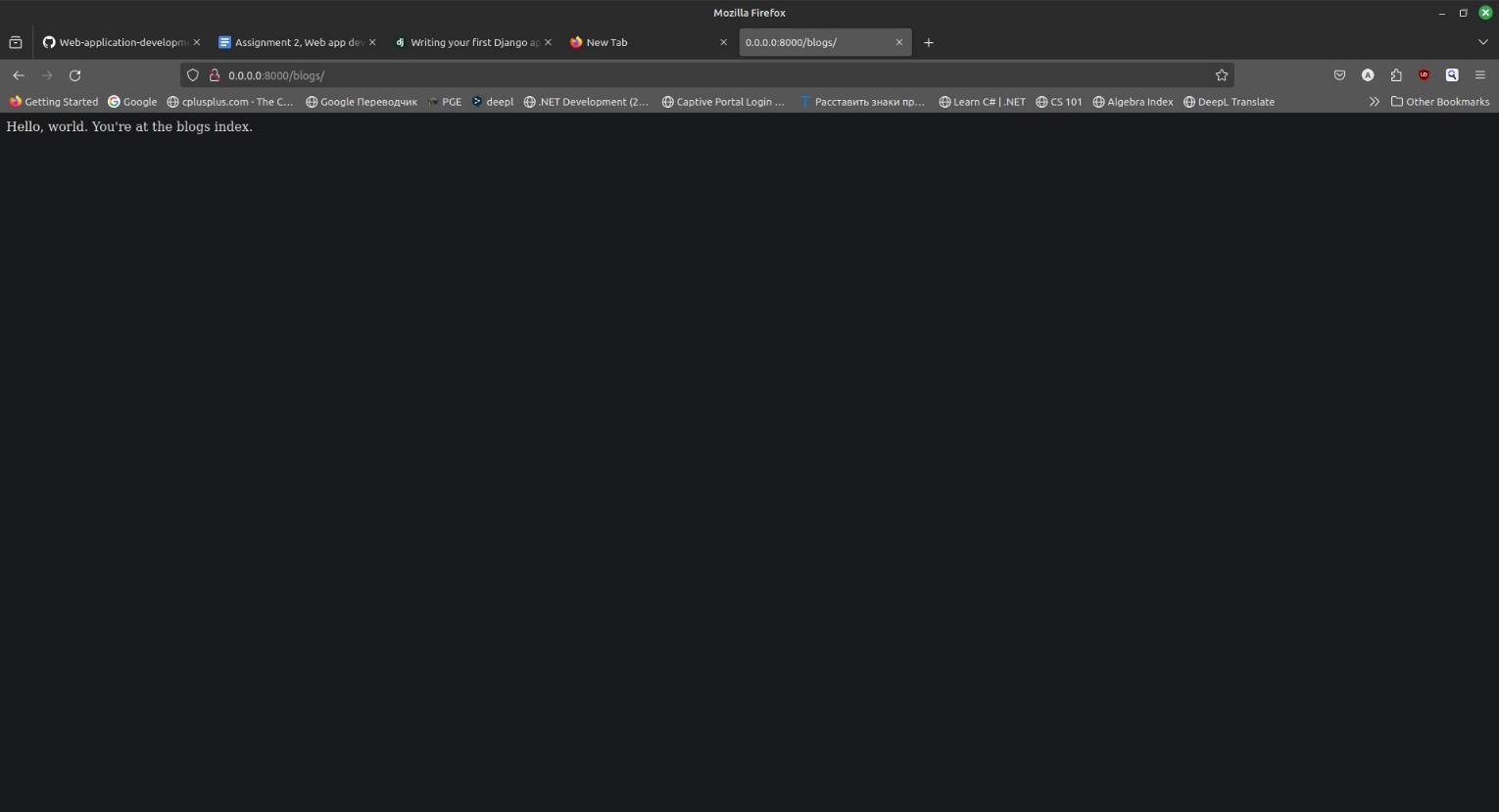
Link to my GitHub repository (screenshots, logs and SQL structure included):

* <https://github.com/BauyrzhanAzimkhanov/Web-application-development-MSc->

**Screenshots**







**Logs**

bauyrzhan@bauyrzhan-laptop:~/Desktop/web-dev/Web-application-development-MSc-/Assignment\_2$ docker compose up

[+] Running 2/0

 ✔ Container assignment\_2-postgres-1  Created                                                                                                              0.0s

 ✔ Container assignment\_2-django-1    Recreated                                                                                                            0.0s

Attaching to django-1, postgres-1

postgres-1  |

postgres-1  | PostgreSQL Database directory appears to contain a database; Skipping initialization

postgres-1  |

postgres-1  | 2024-10-11 14:18:16.365 UTC [1] LOG:  starting PostgreSQL 16.4 (Debian 16.4-1.pgdg110+2) on x86\_64-pc-linux-gnu, compiled by gcc (Debian 10.2.1-6) 10.2.1 20210110, 64-bit

postgres-1  | 2024-10-11 14:18:16.366 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432

postgres-1  | 2024-10-11 14:18:16.366 UTC [1] LOG:  listening on IPv6 address "::", port 5432

postgres-1  | 2024-10-11 14:18:16.368 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"

postgres-1  | 2024-10-11 14:18:16.372 UTC [30] LOG:  database system was interrupted; last known up at 2024-10-10 11:52:06 UTC

postgres-1  | 2024-10-11 14:18:16.891 UTC [30] LOG:  database system was not properly shut down; automatic recovery in progress

postgres-1  | 2024-10-11 14:18:16.894 UTC [30] LOG:  redo starts at 0/19F39A0

postgres-1  | 2024-10-11 14:18:16.895 UTC [30] LOG:  invalid record length at 0/19F4428: expected at least 24, got 0

postgres-1  | 2024-10-11 14:18:16.895 UTC [30] LOG:  redo done at 0/19F43F0 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s

postgres-1  | 2024-10-11 14:18:16.901 UTC [28] LOG:  checkpoint starting: end-of-recovery immediate wait

postgres-1  | 2024-10-11 14:18:16.915 UTC [28] LOG:  checkpoint complete: wrote 8 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.004 s, sync=0.005 s, total=0.016 s; sync files=7, longest=0.002 s, average=0.001 s; distance=2 kB, estimate=2 kB; lsn=0/19F4428, **redo lsn=0/19F4428**

postgres-1  | 2024-10-11 14:18:16.922 UTC [1] LOG:  database system is ready to accept connections

django-1    | Watching for file changes with StatReloader

django-1    | Performing system checks...

django-1    |

django-1    | System check identified no issues (0 silenced).

django-1    | October 11, 2024 - 14:18:22

django-1    | Django version 5.1.1, using settings 'django\_project.settings'

django-1    | Starting development server at http://0.0.0.0:8000/

django-1    | Quit the server with CONTROL-C.

django-1    |

django-1    | Not Found: /

django-1    | [11/Oct/2024 14:18:28] "GET / HTTP/1.1" 404 2312

django-1    | [11/Oct/2024 14:18:34] "GET /admin/ HTTP/1.1" 200 5852

django-1    | [11/Oct/2024 14:18:34] "GET /static/admin/js/theme.js HTTP/1.1" 200 1653

django-1    | [11/Oct/2024 14:18:34] "GET /static/admin/css/dark\_mode.css HTTP/1.1" 200 2804

django-1    | [11/Oct/2024 14:18:34] "GET /static/admin/css/dashboard.css HTTP/1.1" 200 441

django-1    | [11/Oct/2024 14:18:34] "GET /static/admin/css/base.css HTTP/1.1" 200 22092

django-1    | [11/Oct/2024 14:18:34] "GET /static/admin/css/nav\_sidebar.css HTTP/1.1" 200 2810

django-1    | [11/Oct/2024 14:18:34] "GET /static/admin/js/nav\_sidebar.js HTTP/1.1" 200 3063

django-1    | [11/Oct/2024 14:18:34] "GET /static/admin/css/responsive.css HTTP/1.1" 200 17972

django-1    | [11/Oct/2024 14:18:34] "GET /static/admin/img/icon-changelink.svg HTTP/1.1" 200 380

django-1    | [11/Oct/2024 14:18:34] "GET /static/admin/img/icon-addlink.svg HTTP/1.1" 200 331

django-1    | [11/Oct/2024 14:20:53] "GET /admin/auth/user/ HTTP/1.1" 200 11051

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/css/base.css HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/js/theme.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/css/dark\_mode.css HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/css/nav\_sidebar.css HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/js/nav\_sidebar.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/css/changelists.css HTTP/1.1" 200 6878

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/js/jquery.init.js HTTP/1.1" 200 347

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/js/core.js HTTP/1.1" 200 6208

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/js/admin/RelatedObjectLookups.js HTTP/1.1" 200 9097

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/js/actions.js HTTP/1.1" 200 8076

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/js/urlify.js HTTP/1.1" 200 7887

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/js/prepopulate.js HTTP/1.1" 200 1531

django-1    | [11/Oct/2024 14:20:53] "GET /admin/jsi18n/ HTTP/1.1" 200 3342

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/js/filters.js HTTP/1.1" 200 978

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/js/vendor/jquery/jquery.js HTTP/1.1" 200 285314

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/css/responsive.css HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/js/vendor/xregexp/xregexp.js HTTP/1.1" 200 325171

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/img/icon-yes.svg HTTP/1.1" 200 436

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/img/search.svg HTTP/1.1" 200 458

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/img/icon-addlink.svg HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/img/tooltag-add.svg HTTP/1.1" 200 331

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/img/icon-viewlink.svg HTTP/1.1" 200 581

django-1    | [11/Oct/2024 14:20:53] "GET /static/admin/img/sorting-icons.svg HTTP/1.1" 200 1097

django-1    | [11/Oct/2024 14:21:32] "GET /admin/auth/user/1/change/ HTTP/1.1" 200 21815

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/css/nav\_sidebar.css HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/css/base.css HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/theme.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/nav\_sidebar.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/css/dark\_mode.css HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/vendor/jquery/jquery.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/jquery.init.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/css/forms.css HTTP/1.1" 200 8710

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/admin/DateTimeShortcuts.js HTTP/1.1" 200 19319

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/core.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/calendar.js HTTP/1.1" 200 9141

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/admin/RelatedObjectLookups.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /admin/jsi18n/ HTTP/1.1" 200 3342

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/SelectBox.js HTTP/1.1" 200 4530

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/actions.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/css/responsive.css HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/urlify.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/vendor/xregexp/xregexp.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/prepopulate.js HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/SelectFilter2.js HTTP/1.1" 200 15502

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/change\_form.js HTTP/1.1" 200 606

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/js/prepopulate\_init.js HTTP/1.1" 200 586

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/css/widgets.css HTTP/1.1" 200 11564

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/img/icon-addlink.svg HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/img/icon-calendar.svg HTTP/1.1" 200 1086

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/img/search.svg HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/img/icon-unknown.svg HTTP/1.1" 200 655

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/img/icon-unknown-alt.svg HTTP/1.1" 200 655

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/img/icon-clock.svg HTTP/1.1" 200 677

django-1    | [11/Oct/2024 14:21:32] "GET /static/admin/img/selector-icons.svg HTTP/1.1" 200 3291

django-1    | [11/Oct/2024 14:21:37] "GET /admin/ HTTP/1.1" 200 5852

django-1    | [11/Oct/2024 14:21:37] "GET /static/admin/css/dashboard.css HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:37] "GET /static/admin/img/icon-changelink.svg HTTP/1.1" 304 0

django-1    | [11/Oct/2024 14:21:42] "GET /blogs/ HTTP/1.1" 200 40