Assignment 2, Web app dev
Tussupbay Daulet
13.10.2024

Contents:

1. **Django application setup**
2. **Docker**

1.Django application setup.

1. First created django project with to-do app. Set virtual environent and install django.

```
PowerShell
~\KBTU M\Web Fall\Assignments\Assignment 2 main
 django-admin startproject to_do_django_docker_compose_app
~\KBTU M\Web Fall\Assignments\Assignment 2 main
 cd to_do_django_docker_compose_app
~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
 python manage.py startapp to_do
~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
 py -3 -m venv .venv
~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
 .venv\scripts\activate
(.venv) ~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
 code .
(.venv) ~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
 pip freeze > requirements.txt
(.venv) ~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
 python -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\dtussupbayev\kbtu m\web fall\assignments\assignment 2\to_do_djang
o_docker_compose_app\.venv\lib\site-packages (24.2)
(.venv) ~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
 python -m pip install django
Collecting django
  Using cached Django-5.1.2-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.8.1 (from django)
  Using cached asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from django)
  Using cached sqlparse-0.5.1-py3-none-any.whl.metadata (3.9 kB)
Collecting tzdata (from django)
  Using cached tzdata-2024.2-py2.py3-none-any.whl.metadata (1.4 kB)
Using cached Django-5.1.2-py3-none-any.whl (8.3 MB)
Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)
Using cached sqlparse-0.5.1-py3-none-any.whl (44 kB)
Using cached tzdata-2024.2-py2.py3-none-any.whl (346 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-5.1.2 sqlparse-0.5.1 tzdata-2024.2
(.venv) ~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
 pip freeze > requirements.txt
(.venv) ~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
```

2. Create database to_do in postgresql

```
~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main xUSAGE
▶ psql -h localhost -p 5432 -U postgres
Password for user postgres:
psql (17.0)
WARNING: Console code page (866) differs from Windows code page (1251)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# CREATE DATABASE to_do;
CREATE DATABASE
postgres=# \q
~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
▶ psql -U postgres -d to_do -W
Password:
psql (17.0)
WARNING: Console code page (866) differs from Windows code page (1251)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

to_do=# \q
```

3. Create .env file, and set postgresql credentials:

```
Help                                  to_do_django_docker_compose_app
requirements.txt    .env
.env
1  DATABASE_NAME=to_do
2  DATABASE_USER=postgres
3  DATABASE_PASSWORD=postgres
4  DATABASE_HOST=localhost
5  DATABASE_PORT=5432
```

4. Installed postgresql adapter for python

```
(.venv) ~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
▶ pip install psycopg2-binary
Collecting psycopg2-binary
  Downloading psycopg2_binary-2.9.9-cp312-cp312-win_amd64.whl.metadata (4.6 kB)
Downloading psycopg2_binary-2.9.9-cp312-cp312-win_amd64.whl (1.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 2.5 MB/s eta 0:00:00
Installing collected packages: psycopg2-binary
Successfully installed psycopg2-binary-2.9.9
(.venv) ~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
▶
```

5. In settings.py set database settings to use postgresql and env vars

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.environ.get('DATABASE_NAME'),
          'USER': os.environ.get('DATABASE_USER'),
          'PASSWORD': os.environ.get('DATABASE_PASSWORD'),
          'HOST': os.environ.get('DATABASE_HOST'),
          'PORT': os.environ.get('DATABASE_PORT'),
      }
  }
```

6. Add to_do app in settings.py, and its url in urls.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
        'to_do.apps.TodoConfig',
    ]
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('todo.urls'))
]
```

7. Define to do list model

```
from django.db import models
```

```python
# Create your models here.
class ToDoItem(models.Model):
    text = models.CharField(max_length=150)
    completed = models.BooleanField(default=False)
```

8. Create views, urls for to-do app

```python
def to_do_list(request):
    items = ToDoItem.objects.all()
    return render(request, 'to_do_list.html', {'items': items})

def add_to_do_item(request):
    if request.method == 'POST':
        text = request.POST['text']
        ToDoItem.objects.create(text=text)
        return redirect('to_do_list')
    else:
        return render(request, 'add_to_do_item.html')

def toggle_to_do_completed(_, item_id):
    item = get_object_or_404(ToDoItem, id=item_id)
    item.completed = not item.completed
    item.save()
    return redirect('to_do_list')

def delete_to_do_item(_, item_id):
    item = get_object_or_404(ToDoItem, id=item_id)
    item.delete()
    return redirect('to_do_list')
```

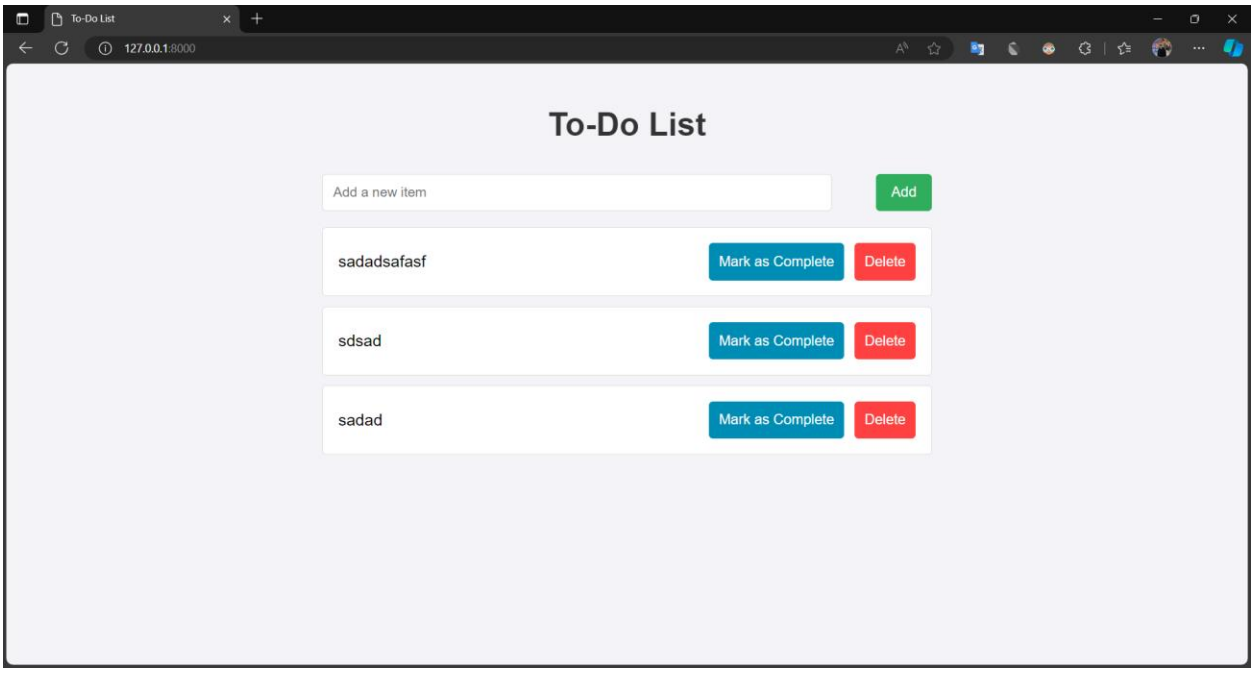
```python
urlpatterns = [
    path('', views.to_do_list, name='to_do_list'),
    path('add/', views.add_to_do_item, name='add_to_do_item'),
    path('toggle_to_do_completed/<int:item_id>/', views.toggle_to_do_completed, name='toggle_to_do_completed'),
    path('delete/<int:item_id>/', views.delete_to_do_item, name='delete_to_do_item'),
]
```

9. Make migrations and migrate

```
~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app  main
▶ python manage.py makemigrations
Migrations for 'to_do':
  to_do\migrations\0001_initial.py
    + Create model ToDoItem
```

```
~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app  main
> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, to_do
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 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 admin.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 sessions.0001_initial... OK
  Applying to_do.0001_initial... OK
```

10. Try to runserver:



Items were added in postgresql database:

```
to_do=# \dt
                    List of relations
 Schema |            Name            | Type  |  Owner
--------+----------------------------+-------+----------
 public | auth_group                 | table | postgres
 public | auth_group_permissions     | table | postgres
 public | auth_permission            | table | postgres
 public | auth_user                  | table | postgres
 public | auth_user_groups           | table | postgres
 public | auth_user_user_permissions | table | postgres
 public | django_admin_log           | table | postgres
 public | django_content_type        | table | postgres
 public | django_migrations          | table | postgres
 public | django_session             | table | postgres
 public | to_do_todoitem             | table | postgres
(11 rows)


to_do=# SELECT * FROM to_do_todoitem;
 id |    text     | completed
----+-------------+-----------
  4 | sadadsafasf | f
  1 | sdsad       | f
  2 | sadad       | f
(3 rows)


to_do=#
```

11. Freezed current state of venv in requirements.txt

```
(.venv) ~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
pip freeze > requirements.txt
(.venv) ~\KBTU M\Web Fall\Assignments\Assignment 2\to_do_django_docker_compose_app main
```

2.Docker compose setup.

Created docker-compose.yml:

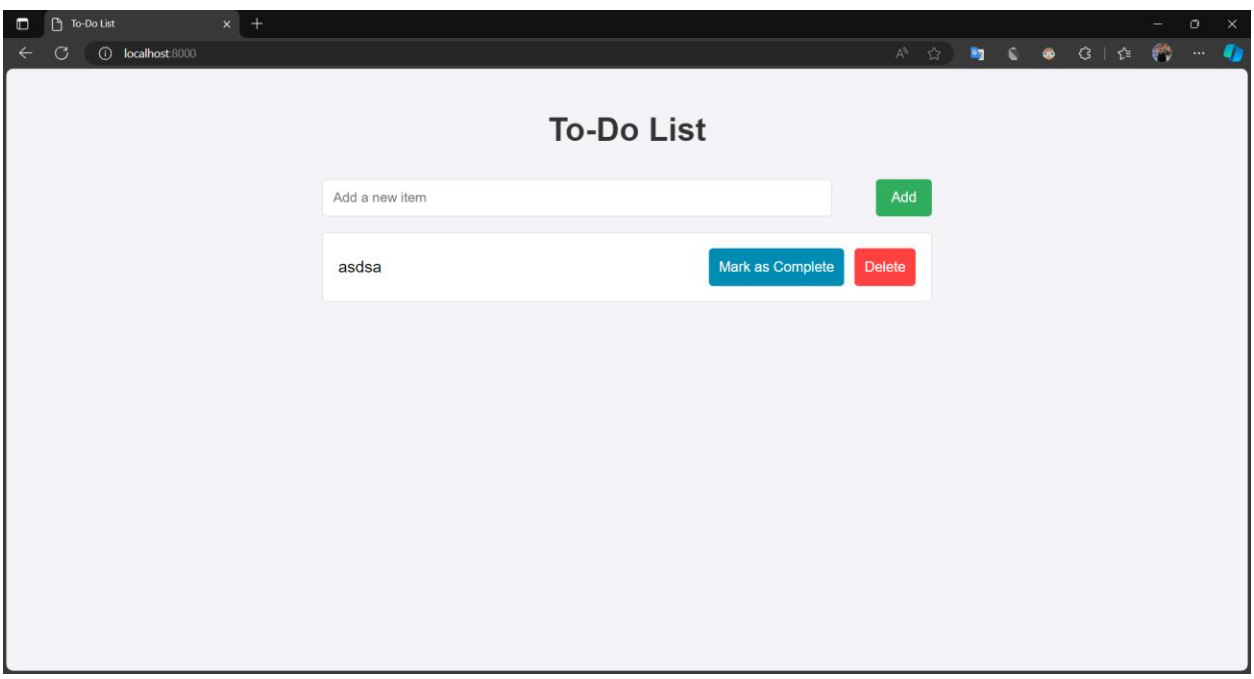
```yaml
services:
  db:
    image: postgres:17
    container_name: db
    environment:
     - POSTGRES_DB=${DATABASE_NAME}
     - POSTGRES_USER=${DATABASE_USER}
     - POSTGRES_PASSWORD=${DATABASE_PASSWORD}
    ports:
    - '5432:5432'
    volumes:
    - pg_data:/var/lib/postgresql/data

  web:
    build: .
    container_name: django
    ports:
      - '8000:8000'
    volumes:
      - .:/app
    environment:
      - DATABASE_NAME=${DATABASE_NAME}
      - DATABASE_USER=${DATABASE_USER}
      - DATABASE_PASSWORD=${DATABASE_PASSWORD}
      - DATABASE_HOST=db
      - DATABASE_PORT=${DATABASE_PORT}
    depends_on:
      - db

volumes:
  pg_data:
```

Define two services: web (for Django) and app-postgres (for PostgreSQL). Use environment variables. Map volume pg_data to postgresql data in container. web service depends on app-postgres, so postgres starts first.

```
▶ docker-compose up --build
[+] Building 4.4s (11/11) FINISHED                                                                                  docker:desktop-linux
 => [web internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 241B                                                                                              0.0s
 => [web internal] load metadata for docker.io/library/python:3.12.6                                                              1.6s
 => [web internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                                   0.0s
 => [web 1/5] FROM docker.io/library/python:3.12.6@sha256:14f073695854184b65a82808ea2baa352d49c1a86442f06a90d9c236861c7a8f        0.0s
 => [web internal] load build context                                                                                             2.5s
 => => transferring context: 810.73kB                                                                                             2.5s
 => CACHED [web 2/5] WORKDIR /app                                                                                                 0.0s
 => CACHED [web 3/5] COPY requirements.txt .                                                                                      0.0s
 => CACHED [web 4/5] RUN pip install --no-cache-dir -r requirements.txt                                                           0.0s
 => CACHED [web 5/5] COPY . .                                                                                                     0.0s
 => [web] exporting to image                                                                                                      0.0s
 => => exporting layers                                                                                                           0.0s
 => => writing image sha256:54fa69580d09eba9a7e4ea9d215ee59dbdfd59ee7820993414e96f7a03e3344f                                      0.0s
 => => naming to docker.io/library/to_do_django_docker_compose_app-web                                                            0.0s
 => [web] resolving provenance for metadata file                                                                                  0.0s
[+] Running 2/2
 ✓ Container db      Created                                                                                                      0.1s
 ✓ Container django  Recreated                                                                                                    0.2s
Attaching to db, django
db      |
db      | PostgreSQL Database directory appears to contain a database; Skipping initialization
db      |
db      | 2024-10-13 19:02:11.882 UTC [1] LOG:  starting PostgreSQL 17.0 (Debian 17.0-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2
.0-14) 12.2.0, 64-bit
db      | 2024-10-13 19:02:11.883 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db      | 2024-10-13 19:02:11.883 UTC [1] LOG:  listening on IPv6 address "::", port 5432
db      | 2024-10-13 19:02:11.893 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db      | 2024-10-13 19:02:11.907 UTC [29] LOG:  database system was shut down at 2024-10-13 19:00:46 UTC
db      | 2024-10-13 19:02:11.925 UTC [1] LOG:  database system is ready to accept connections
django  | Watching for file changes with StatReloader
```



3.Docker compose networking and volumes.

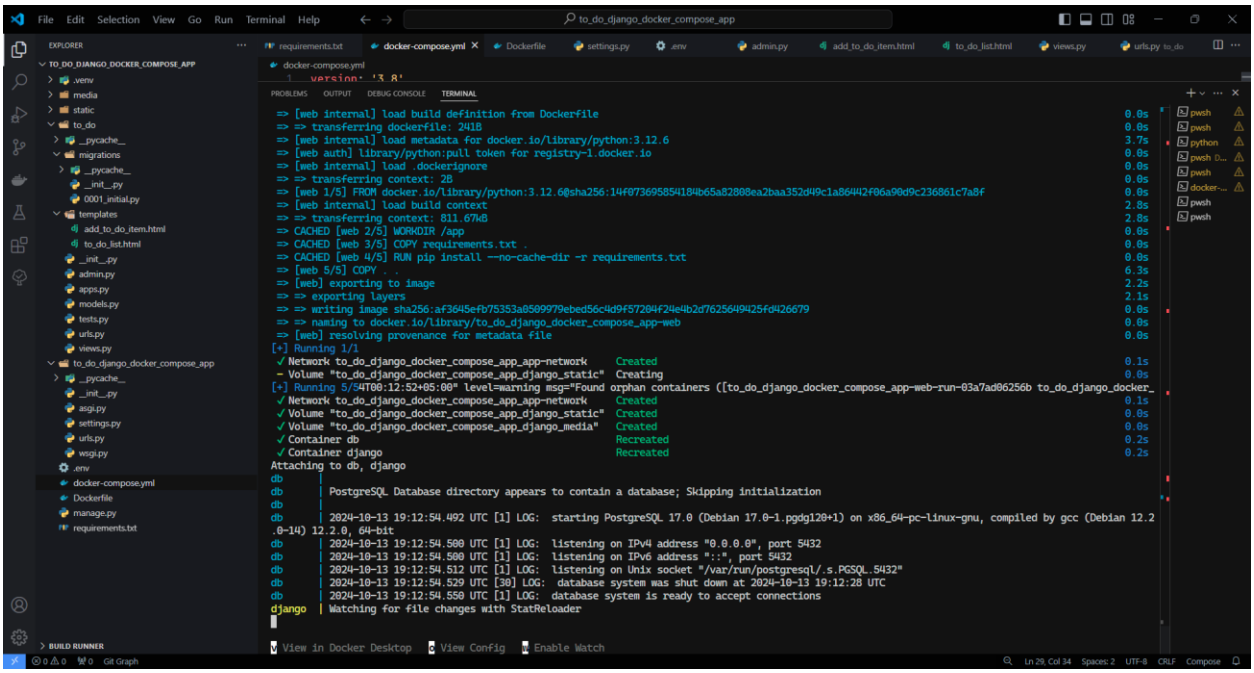
```yaml
networks:
  app-network:
    driver: bridge

services:
  db:
    image: postgres:17
    container_name: db
    environment:
      - POSTGRES_DB=${DATABASE_NAME}
      - POSTGRES_USER=${DATABASE_USER}
      - POSTGRES_PASSWORD=${DATABASE_PASSWORD}
    ports:
      - '5432:5432'
    volumes:
      - pg_data:/var/lib/postgresql/data
    networks:
      - app-network

  web:
    build: .
    container_name: django
    ports:
      - '8000:8000'
    volumes:
      - .:/app
      - django_static:/app/static
      - django_media:/app/media
    environment:
      - DATABASE_NAME=${DATABASE_NAME}
      - DATABASE_USER=${DATABASE_USER}
      - DATABASE_PASSWORD=${DATABASE_PASSWORD}
      - DATABASE_HOST=db
      - DATABASE_PORT=${DATABASE_PORT}
    depends_on:
      - db
    networks:
```

```
      - app-network

volumes:
  pg_data:
  django_static:
  django_media:
```



Defined a bridge network named app-network. Both services were attached to the app-network. pg_data volume is for to persist postgresql data. django_static and django_media, are defined to persist uploaded files and static files of Django app.By networking we can ensure that our services networking isolated from other containers. Volumes help to ensure that our data not lost by container restarts.