

# Administrarea Sistemelor de Operare

## Proiect

Documentatie

Student: Cîrciu Mihnea Teodor

Grupa: 30644

In prima etapa a proiectului am configurat environment-ul (venv\_django),am instalat software-ul necesar (Pycharm), am instalat Python si Django. In acest process am urmat tutorialele celor de la Django, propuse si in Descrierea Proiectului pusa la dispozitie de profesor:

- <https://docs.djangoproject.com/en/3.2/intro/install/>
- <https://docs.djangoproject.com/en/3.2/intro/>
- <https://docs.djangoproject.com/en/3.2/intro/tutorial01/#write-your-first-view>
- <https://docs.djangoproject.com/en/3.2/intro/tutorial02/#introducing-the-django-admin>

Django REST framework

mihneateodor

Api Root

Api Root

OPTIONSGET

The default basic root view for DefaultRouter

GET /scrumboard/

HTTP 200 OK  
Allow: GET, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "lists": "http://127.0.0.1:8000/scrumboard/lists/",
  "cards": "http://127.0.0.1:8000/scrumboard/cards/"
}
```

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

+ AddChange

Users

+ AddChange

SCRUMBOARD

Cards

+ AddChange

Lists

+ AddChange

Recent actions

My actions

+ Card object (2)  
Card

+ Card object (1)  
Card

+ List object (1)  
List

In cea de-a doua faza am inceput implementarea propriu-zisa a aplicatiei. Aceasta va consta intr-o platforma ce contine utilizatori care pot comunica intre ei. Subiectul acestei aplicatii este prezentarea unor obiective turistice din diferite tari ale globului, avand de asemenea posibilitatea de a trimite imagini in conversatii.

Aplicatia are o baza de date care contine: utilizatori, camere pentru conversatii si mesaje.

Utilizatorii sunt caracterizati prin numele de utilizator si parola.

Camerele sunt caracterizate prin denumire, descriere, proprietar si participanti.

Mesajele sunt caracterizate prin utilizator, camera si continut.

Autentificare, deconectare si autorizare

La accesarea aplicatiei utilizatorii care nu sunt conectati au posibilitatea sa vada camerele existente alaturi de proprietar si descriere inasa nu pot vedea conversatiile dintre participanti.

Odata conectat la aplicatie, utilizatorul poate accesa fiecare camera, inasa poate vedea mesajele doar in conversatiile la care el este participant. De asemenea camerele pot fi sterse sau pot avea parte de modificari (descriere, denumire, participanti) inasa doar proprietarul poate accesa aceste setari.

---

## Main

Hello mihneateodor

[Logout](#)

---

### HomePage

---

[Create room](#)

---

[Edit](#) [Delete](#)

@mihneateodor

1 -- [Romania](#)

---

[Edit](#) [Delete](#)

@mihneateodor

2 -- [Spain](#)

---

@marian.user

5 -- [France](#)

---

# Main

Hello mihneateodor

[Logout](#)

## France

Population: 67,5 mil Capital: Paris Location: Western Europe

You are not a participant of this conversation!

### Participants

@marian.user

@dragos.user

@dragos.user 0 minutes ago

Wow, great :DDD



@mihneateodor 2 minutes ago

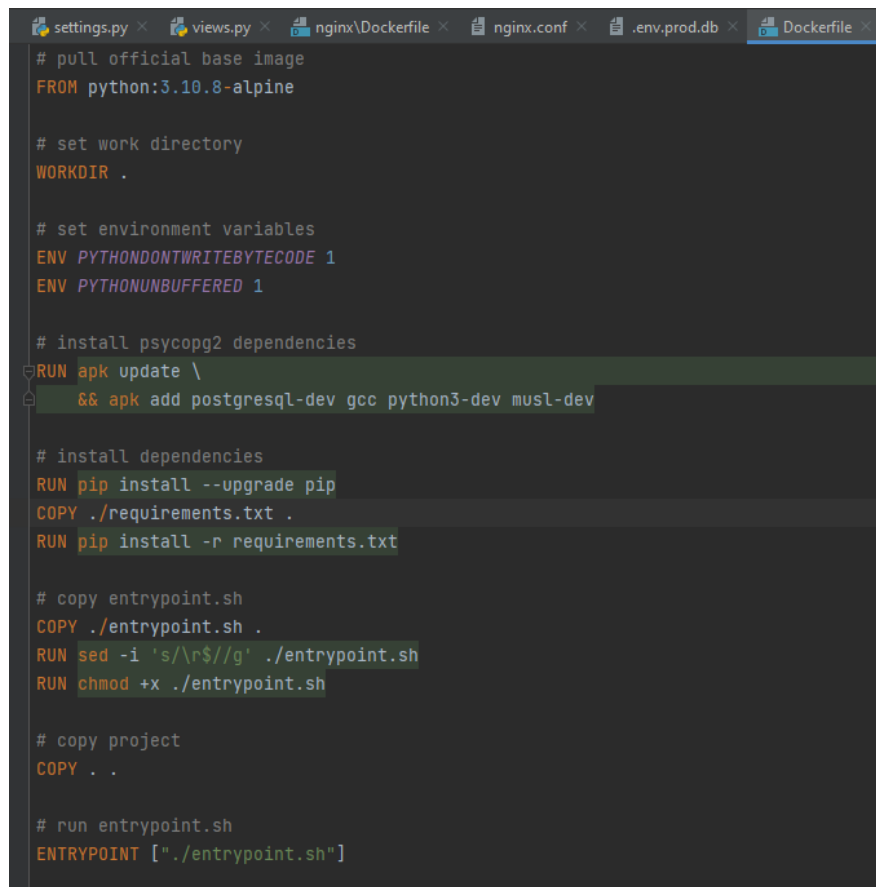
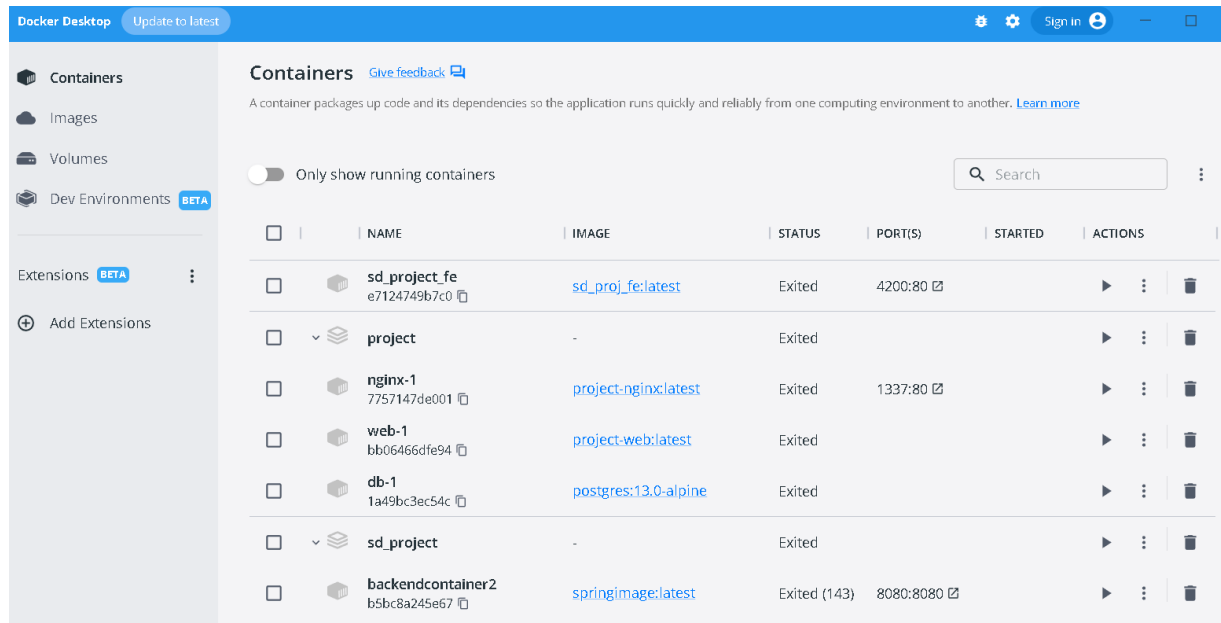
@mihneateodor 12 hours, 6 minutes ago

I recommend you visiting Cluj-Napoca's Botanical Garden

No file chosen

In cea de-a treia faza a proiectului a fost realizat deploy-ul aplicatiei prin containere Docker.

Initial am instalat Docker si am creat fisierul Dockerfile:



Urmatorul punct vizat in faza a treia a proiectului a fost trecerea de la baza de date SQLite la cea Postgres dupa care am configurat fisierul docker-compose.yml:

```
views.py x nginx\Dockfile x nginx.conf x .env.prod.db x Dockerfile x docker-compose.yml
version: '3.8'

services:
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - ./app:/usr/src/app/
    ports:
      - 8000:8000
    env_file:
      - ./.env.dev
    depends_on:
      - db
  db:
    image: postgres:13.0-alpine
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    environment:
      - POSTGRES_USER=mihneateodor
      - POSTGRES_PASSWORD=parola1234
      - POSTGRES_DB=aso_proj_dev
volumes:
  postgres_data:
```

Apoi am folosit gunicorn si nginx pentru a utiliza un server WSGI real. Pentru acesta am configurat si un fisier docker-compose.prod.yml:

```
requirements.txt x docker-compose.prod.yml x settings.py x views.py x nginx\Dockfile
version: '3.8'

services:
  web:
    build:
      context: .
      dockerfile: Dockerfile.prod
    command: gunicorn project.wsgi:application --bind 0.0.0.0:8000
    volumes:
      - static_volume:/project_dock/web/staticfiles
      - media_volume:/project_dock/web/mediafiles
    expose:
      - 8000
    env_file:
      - .env.prod
    depends_on:
      - db
  nginx:
    build: ./nginx
    volumes:
      - static_volume:/project_dock/web/staticfiles
      - media_volume:/project_dock/web/mediafiles
    ports:
      - 1337:80
    depends_on:
      - web
  db:
    image: postgres:13.0-alpine
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    env_file:
      - .env.prod.db
volumes:
  postgres_data:
  static_volume:
  media_volume:
```

Am configurat de asemenea si fisiere specifice pentru variabilele de mediu care vor fi folosite in de docker-compose pentru developement si productie:

File	Line	Content
.env.dev	1	DEBUG=1
	2	SECRET_KEY=foo
	3	DJANGO_ALLOWED_HOSTS=localhost 127.0.0.1 [::1]
	4	SQL_ENGINE=django.db.backends.postgresql
	5	SQL_DATABASE=aso_proj_dev
	6	SQL_USER=mihneateodor
	7	SQL_PASSWORD=parola1234
	8	SQL_HOST=db
	9	SQL_PORT=5432
	10	DATABASE=postgres
.env.prod	1	DEBUG=0
	2	SECRET_KEY=secret
	3	DJANGO_ALLOWED_HOSTS=localhost 127.0.0.1 [::1]
	4	SQL_ENGINE=django.db.backends.postgresql
	5	SQL_DATABASE=aso_proj_prod
	6	SQL_USER=mihneateodor
	7	SQL_PASSWORD=parola1234
	8	SQL_HOST=db
	9	SQL_PORT=5432
	10	DATABASE=postgres