



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICA ȘI CALCULATOARE

ADMINISTRAREA SISTEMELOR DE OPERARE

Deploying Django and Docker

Student: Ioana Vîrnă

Profesor îndrumător: Andrei Bogdan
Leucuta

Introducere

În acest proiect, va fi realizat un web-site minimal și se va simula lansarea acestuia în producție.

Site-ul va fi realizat folosind framework-ul Django, bazat pe Python.

Realizarea sa va cuprinde 3 etape și anume:

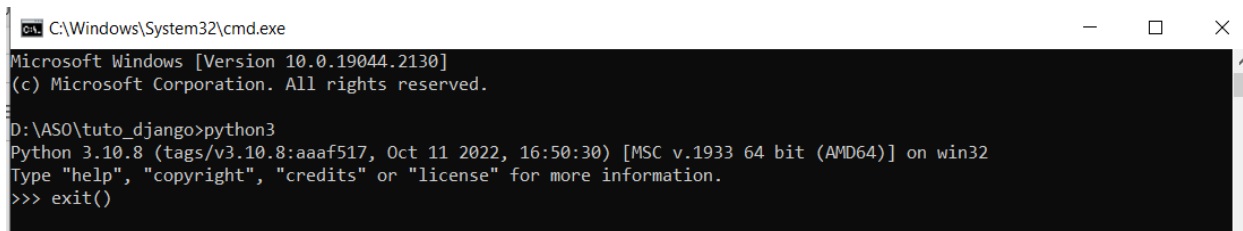
- Etapa 1: Instalare Python, Django și crearea unui site minimalist
- Etapa 2: Crearea unui chat minimalist
- Etapa 3: Presupunând că site-ul este funcțional, îl pregătim de lansarea în piață

Etapa 1

În această etapă se vor instala resursele necesare în realizarea temei și se vor implementa cerințele minime, având astfel o bază a proiectului.

Pasi pentru crearea proiectului de baza:

1. Verificarea existentei **python3**. Dacă nu există, vom fi ghidați să îl instalăm.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.


D:\AS0\tuto_django>python3
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
```

2. Deschidem linia de comandă din locația unde vom dori să cream proiectul




UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

3. Cream un director numit `venv_django`, utilizand comanda **`python3 -m venv venv_django`**

 C:\Windows\System32\cmd.exe

```
D:\ASO>cd tuto_django
```

```
D:\ASO\tuto_django>python3 -m venv venv_django
```

 venv_django

24 oct. 2022 18:50

4. Activam virtual environment-ul creat. Se poate observa ca este activat atunci cand in fata prompt-ului obisnuit de CMD, avem intre paranteze numele al virtual environmentului.

```
D:\ASO\tuto_django>venv_django\Scripts\activate
```

```
(venv_django) D:\ASO\tuto_django>pip list
```

```
Package      Version
```

```
-----
```

```
pip          22.2.2
```

```
setuptools  63.2.0
```

```
[notice] A new release of pip available: 22.2.2 -> 22.3
```

```
[notice] To update, run: python.exe -m pip install --upgrade pip
```

5. Instalam modulele Django si djangoestframework utilizand comanda **`pip`**

```
(venv_django) D:\ASO\tuto_django>cd venv_django
```

```
(venv_django) D:\ASO\tuto_django\venv_django>pip install django djangoestframework
```

```
Collecting django
```

```
  Using cached Django-4.1.2-py3-none-any.whl (8.1 MB)
```

```
Collecting djangoestframework
```

```
  Using cached djangoestframework-3.14.0-py3-none-any.whl (1.1 MB)
```

```
Collecting tzdata
```

```
  Using cached tzdata-2022.5-py2.py3-none-any.whl (336 kB)
```

```
Collecting asgiref<4,>=3.5.2
```

```
  Using cached asgiref-3.5.2-py3-none-any.whl (22 kB)
```

```
Collecting sqlparse>=0.2.2
```

```
  Using cached sqlparse-0.4.3-py3-none-any.whl (42 kB)
```

```
Collecting pytz
```

```
  Using cached pytz-2022.5-py2.py3-none-any.whl (500 kB)
```

```
Installing collected packages: pytz, tzdata, sqlparse, asgiref, django, djangoestframework
```

```
Successfully installed asgiref-3.5.2 django-4.1.2 djangoestframework-3.14.0 pytz-2022.5 sqlparse-0.4.3 tzdata-2022.5
```

```
[notice] A new release of pip available: 22.2.2 -> 22.3
```

```
[notice] To update, run: python.exe -m pip install --upgrade pip
```

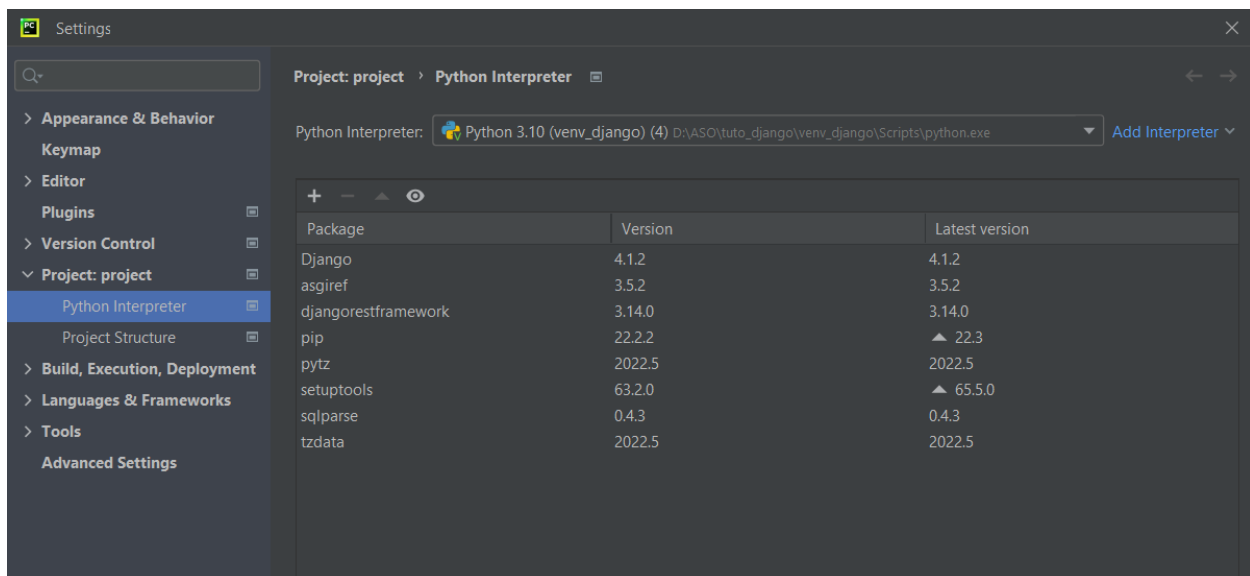


UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

6. Din directorul care contine directorul cu virtual environmental creat, initiem un proiect Django cu ajutorul comenzii **Django-admin startproject project**

```
(venv_django) D:\ASO\tuto_django\venv_django>cd ..  
  
(venv_django) D:\ASO\tuto_django>django-admin startproject project
```

7. Folosind mediul de dezvoltare **PyCharm**, deschidem directorul tuto_django/project. Configuram din setari interpretorul de python (mediul virtual creat mai devreme)



8. Invocam pornirea serverului utilizand comanda **python manage.py runserver** din terminalul PyCharm. Se va crea o aplicatie web Django, insa fara o baza de date.

```
PS D:\ASO\tuto_django\project> python manage.py runserver  
Watching for file changes with StatReloader  
Performing system checks...  
  
System check identified no issues (0 silenced).  
  
You have 18 unapplied migration(s). Your project may not work properly until you apply t  
he migrations for app(s): admin, auth, contenttypes, sessions.  
Run 'python manage.py migrate' to apply them.  
October 24, 2022 - 18:57:03  
Django version 3.2, using settings 'project.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CTRL-BREAK.
```



9. Cream aplicatia
python

scumboard. Dupa ce s-a creat, in sectiunea **INSTALLED_APPS** din **setting.py** vom adauga si 'scumboard'.

```
PS D:\AS0\tuto_django\project> python.exe .\manage.py startapp scumboard
PS D:\AS0\tuto_django\project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
```

Name	Date modified	Type
Quick access		
Desktop		
Downloads		
.idea	24 oct. 2022 19:02	File folder
project	24 oct. 2022 18:58	File folder
scumboard	24 oct. 2022 19:02	File folder

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

10. Definim modelele aplicatiei, **List** si **Card** in **models.py** din directorul **scumboard**.

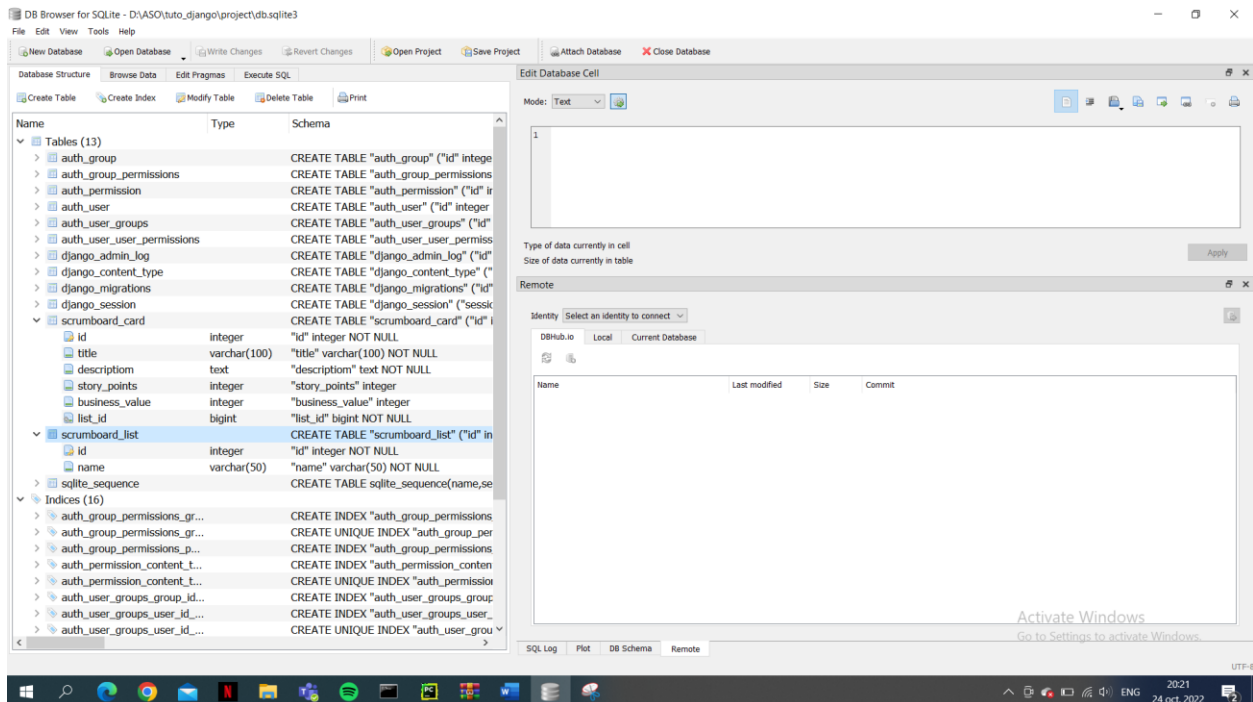
```
settings.py x models.py x
1 from django.db import models
2
3 # Create your models here.
4 class List(models.Model):
5     name=models.CharField(max_length=50)
6
7 class Card(models.Model):
8     title = models.CharField(max_length=100)
9     description=models.TextField(blank=True)
10    # relation with foreign key below
11    # Each Card must belong to a list
12    list=models.ForeignKey(List, related_name="cards", on_delete=models.CASCADE)
13    story_points = models.IntegerField(null=True, blank=True)
14    business_value=models.IntegerField(null=True, blank=True)
15
```

11. Apelam comanda **python manage.py makemigrations**, urmată de **python manage.py migrate**. Makemigrations doar pregătește baza de date pentru manipulare, migrate o modifică efectiv.

```
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

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 scrumboard.0001_initial... OK
Applying sessions.0001_initial... OK
PS D:\ASO\tuto_django\project>
```

12. Dacă ne uităm în SQLite, vom putea vedea tabelele `scrumboard_card` și `scrumboard_list` unde vom putea vedea câmpurile definite în clasele `List` și `Card` din `models.py`



The screenshot displays the DB Browser for SQLite interface. The left sidebar shows the 'Database Structure' pane with 'Tables (13)' and 'Indices (16)' expanded. The 'Tables (13)' section lists the following tables and their schemas:

- `auth_group`: CREATE TABLE "auth_group" ("id" integer NOT NULL, "name" varchar(150) NOT NULL)
- `auth_group_permissions`: CREATE TABLE "auth_group_permissions" ("group_id" integer NOT NULL, "permission_id" integer NOT NULL)
- `auth_permission`: CREATE TABLE "auth_permission" ("id" integer NOT NULL, "content_type_id" integer NOT NULL, "codename" varchar(100) NOT NULL)
- `auth_user`: CREATE TABLE "auth_user" ("id" integer NOT NULL, "username" varchar(150) NOT NULL, "password" varchar(128) NOT NULL, "email" varchar(254) NOT NULL)
- `auth_user_groups`: CREATE TABLE "auth_user_groups" ("user_id" integer NOT NULL, "group_id" integer NOT NULL)
- `auth_user_user_permissions`: CREATE TABLE "auth_user_user_permissions" ("user_id" integer NOT NULL, "permission_id" integer NOT NULL)
- `django_admin_log`: CREATE TABLE "django_admin_log" ("id" integer NOT NULL, "user_id" integer NOT NULL, "action_time" datetime NOT NULL, "action_object" varchar(255) NOT NULL, "action_object_repr" varchar(255) NOT NULL, "change_message" text NOT NULL)
- `django_content_type`: CREATE TABLE "django_content_type" ("id" integer NOT NULL, "app_label" varchar(100) NOT NULL, "model" varchar(100) NOT NULL)
- `django_migrations`: CREATE TABLE "django_migrations" ("id" integer NOT NULL, "app" varchar(100) NOT NULL, "name" varchar(200) NOT NULL, "timestamp" datetime NOT NULL)
- `django_session`: CREATE TABLE "django_session" ("session_key" varchar(40) NOT NULL, "session_data" text NOT NULL, "expire_date" datetime NOT NULL)
- `scrumboard_card`: CREATE TABLE "scrumboard_card" ("id" integer NOT NULL, "title" varchar(100) NOT NULL, "description" text NOT NULL, "story_points" integer NOT NULL, "business_value" integer NOT NULL, "list_id" bigint NOT NULL)
- `scrumboard_list`: CREATE TABLE "scrumboard_list" ("id" integer NOT NULL, "name" varchar(50) NOT NULL)
- `sqlite_sequence`: CREATE TABLE "sqlite_sequence" (name, seq)

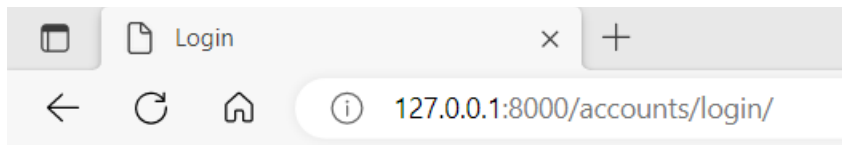
The 'Indices (16)' section shows the following indices:

- `auth_group_permissions_gr...`: CREATE INDEX "auth_group_permissions_group_id" ON "auth_group_permissions" ("group_id")
- `auth_group_permissions_p...`: CREATE INDEX "auth_group_permissions_permission_id" ON "auth_group_permissions" ("permission_id")
- `auth_group_permissions_p...`: CREATE INDEX "auth_group_permissions_group_id_permission_id" ON "auth_group_permissions" ("group_id", "permission_id")
- `auth_permission_content_t...`: CREATE INDEX "auth_permission_content_type_id_codename" ON "auth_permission" ("content_type_id", "codename")
- `auth_permission_content_t...`: CREATE UNIQUE INDEX "auth_permission_content_type_id_codename" ON "auth_permission" ("content_type_id", "codename")
- `auth_user_groups_group_id...`: CREATE INDEX "auth_user_groups_user_id_group_id" ON "auth_user_groups" ("user_id", "group_id")
- `auth_user_groups_group_id...`: CREATE INDEX "auth_user_groups_group_id" ON "auth_user_groups" ("group_id")
- `auth_user_groups_user_id...`: CREATE INDEX "auth_user_groups_user_id" ON "auth_user_groups" ("user_id")
- `auth_user_groups_user_id...`: CREATE UNIQUE INDEX "auth_user_groups_user_id" ON "auth_user_groups" ("user_id")

Etapa II

Scopul acestei etape este de a realiza un chat minimalist unde se pot conecta utilizatori si pot trimite si primi mesaje.

Am inceput cu crearea sistemului de conectare la chat. Utilizatorul este creat doar de un admin, ulterior putandu-se sa se conecteze accesand <http://127.0.0.1:8000/accounts/login/>



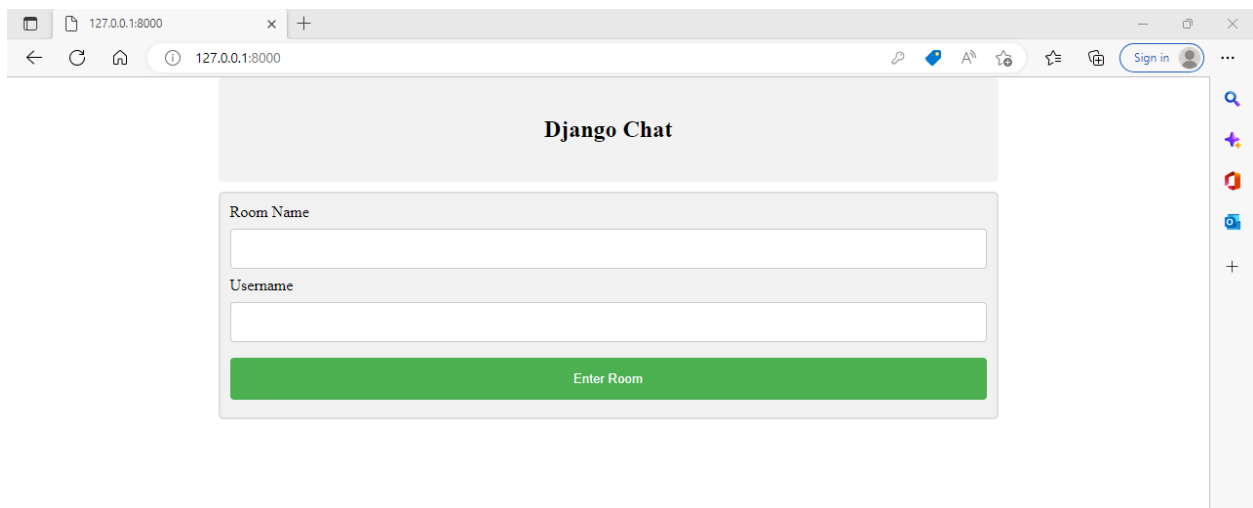
Log In

Username:

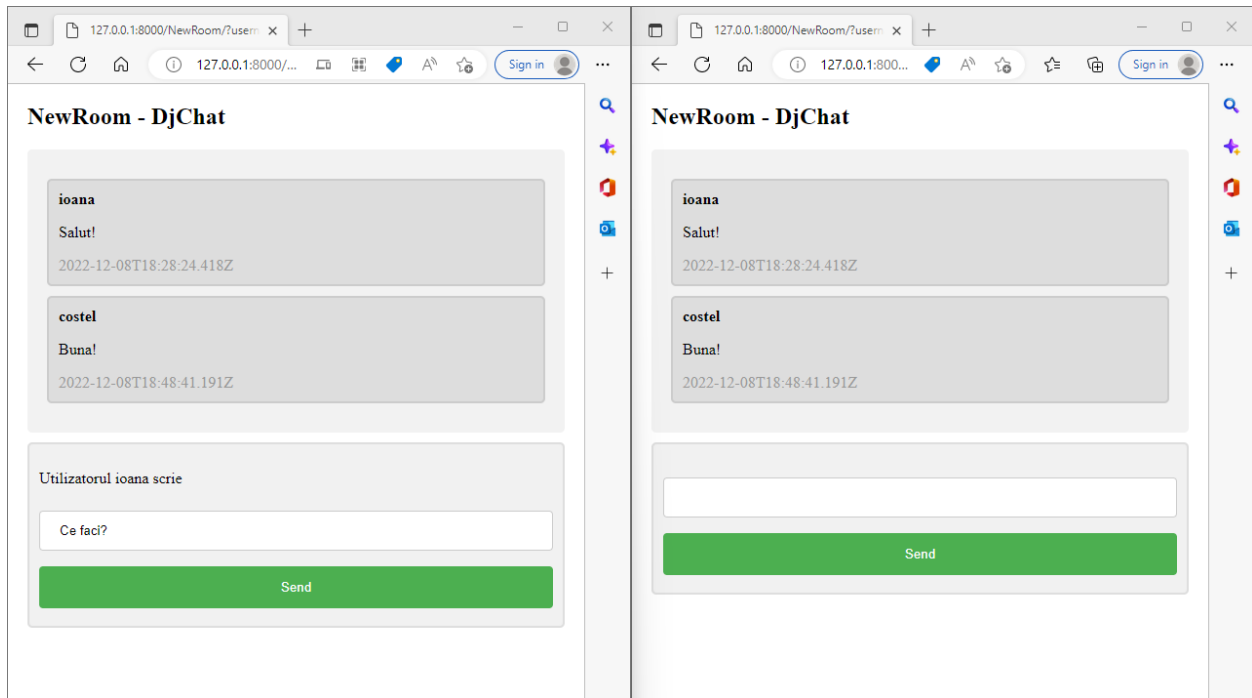
Password:

Dupa conectare, acesta poate intra intr-o camera de chat. Camera poate exista sau nu, insa in cazul in care camera ceruta nu e create, in momentul in care se introduce un nume nou de camera, aceasta se creaza.

De asemenea, utilizatorul poate alege numele sub care sa se conecteze la chat.



Camera de chat este prezentata in urmatoarea poza, unde se poate vedea ca in momentul in care un utilizator scrie, este semnalat acest lucru.



Rularea aplicatiei se va face cu comanda: `python manage.py runserver`

Etapa III

Aceasta etapa presupune deploy-ul aplicatiei noastre intr-un container de Docker. Pentru a putea realiza, am instalat Docker Desktop si am creat un Dockerfile.

Am actualizat variabilele SECRET_KEY, DEBUG si ALLOWED_HOSTS din fisierul settings.py, variabile ce au fost stocate in fisierul .env.dev


```
SECRET_KEY = os.environ.get("SECRET_KEY")

DEBUG = int(os.environ.get("DEBUG", default=0))

# 'DJANGO_ALLOWED_HOSTS' should be a single string of hosts with a space between each.
# For example: 'DJANGO_ALLOWED_HOSTS=localhost 127.0.0.1 [::1]'
ALLOWED_HOSTS = os.environ.get("DJANGO_ALLOWED_HOSTS").split(" ")
```

```
DEBUG=1
SECRET_KEY=django-insecure-&)yyv8a9d--xu7)64bbja-j6e)@t^(xnye2i!pqm0t%nx$utj5
DJANGO_ALLOWED_HOSTS=localhost 127.0.0.1 [::1]
```

De asemenea, am migrat pe o baza de date reala (Postgres), adaugand un nou serviciu in docker-compose.yml. Mai mult, variabila DATABASES, adaugand in .env.dev noile variabile pentru database

```
DATABASES = {
    "default": {
        "ENGINE": os.environ.get("SQL_ENGINE", "django.db.backends.sqlite3"),
        "NAME": os.environ.get("SQL_DATABASE", BASE_DIR / "db.sqlite3"),
        "USER": os.environ.get("SQL_USER", "user"),
        "PASSWORD": os.environ.get("SQL_PASSWORD", "password"),
        "HOST": os.environ.get("SQL_HOST", "localhost"),
        "PORT": os.environ.get("SQL_PORT", "5432"),
    }
}
```

```
SQL_ENGINE=django.db.backends.postgresql
SQL_DATABASE=hello_django_dev
SQL_USER=hello_django
SQL_PASSWORD=hello_django
SQL_HOST=db
SQL_PORT=5432
DATABASE=postgres
```

Aplicatia se poate rula atat in mod de productie, cat si development. Diferenta dintre cele 2 sta in variabila DEBUG, in modul prod fiind setata FALSE, astfel fisierele statice nu vor fi disponibile.

Am folosit de asemenea si un server WSGI real, numit gunicorn si un reverse proxy (nginx) care redirectioneaza requesturile catre site-ul web, fie va servi fisierele statice.

Pentru a rula in varianta de development, folosim urmatoarele comenzi:

```
$ docker-compose down -v  
$ docker-compose up -d --build  
$ docker-compose exec web python manage.py migrate --noinput
```

Pentru a rula in varianta de productie, folosim urmatoarele comenzi:

```
$ docker-compose -f docker-compose.prod.yml up -d --build  
$ docker-compose -f docker-compose.prod.yml exec web python manage.py migrate -  
noinput
```

Doar pt fisierele statice:

```
$ docker-compose -f docker-compose.prod.yml exec web python manage.py collectstatic -  
-no-input --clear
```

Probleme intalnite si modul de rezolvare

M-am ghidat dupa 2 tutoriale gasite pe internet (se va face referire la ele in bibliografie). Am avut probleme cu cateva dintre metodele de trimis/primit mesaje, insa si cu partea de login.

Am incercat sa realizez sa apara la ambii utilizatori cand cineva scrie ceva, insa nu am reusit sa leg prin websocket-uri asta.

In etapa 3 am intampinat foarte multe erori. Unele au fost din cauza ca nu era bun path-ul catre un anumit fisier, altele din lipsa rularii comenzii de migrare a bazei de date. Pentru a le rezolva, am apelat la ajutorul colegilor sau a indrumatorului de laborator, in momentul in care sursele de pe internet nu ma ajutau cu informatiile oferite.

Concluzii

Am dobandit cunostinte stabile despre rutare si implementarea unei aplicatii de chat in timp real 😊 , dar si despre deploy-ul acesteia pe Docker.