Graphical user interface, text, application, email

Description automatically generated

Proiect de semestru la disciplina *Administrarea Sistemelor De Operare*

*Ionescu Maria-Andreea*

*An universitar: 2022-2023*

*Semestrul: 1*

Cuprins

1. Etapa 1 ......................................................................2
2. Etapa 2 .......................................................................
3. Etapa 3 .......................................................................
4. Etapa 1

In aceasta prima etapa am instalat Python3 ( in aplicatia Pycharm preinstalata) , framework-ul Django si am creat un site minimalist, toate acestea fiind realizate urmarind tutorialele primate la laborator.

Dupa instalarea acestora am rulat urmatoarele comenzi :

* In folderul in care am vrut sa creez proiectul Django



Dupa rularea acestei comenzi mi s-a creat un script *manage.py* care va ajuta la interactionarea cu proiectul, si un folder cu pachetul Python pentru site.

* In consola din Pycharm



Dupa rularea acestei comenzi am putut accesa site-ul in browser la locatia http://127.0.0.1:8000. 127.0.0.1 .

* In consola din Pycharm



Dupa rularea acestei comenzi am creat o aplicatie pentru site numita *aso*.

* In consola din Pycharm



Dupa rularea acestei comenzi am initiat o baza de date de tipul SqLite cu setari implicite.

Ulterior am activat interfata de administrator din Django, urmarind tutorialul:

[Writing your first Django app, part 2 | Django documentation | Django (djangoproject.com)](https://docs.djangoproject.com/en/3.2/intro/tutorial02/#introducing-the-django-admin)

A screenshot of a computer

Description automatically generated

1. Etapa 2

* Logarea utilizatorilor

Am create un director nou numit “templates” in care am introdus documente html pentru pagina de logare. In documentul “login.html” am utilizat standardul Django utilizand POST pentru a trimite date si tagul {% csrf\_token %} pentru Securitate.

Duap aceea am updatat fisierul settings.py pentru a spune lui Django sa caute folderul templates la nivelul proiectului.

Utilizatorul se poate loga accesand url-ul: 127.0.0.1:8000/accounts/login

Graphical user interface, text, application

Description automatically generated

In cazul in care nu se introduc credentiale corecte vom fi anuntati prin intermediul unui mesaj ca mai jos:

Graphical user interface, text, application, email

Description automatically generated

* Utilizatorii pot posta mesaje si pot vedea mesajele postate:

Dupa intrarea intr-un room, utilizatorii isi pot trimite mesaje reciproc.

Graphical user interface

Description automatically generated with medium confidence

* Exista mai multe camere de chat. Un utilizator poate crea camere de chat in care sa invite alti utilizatori.

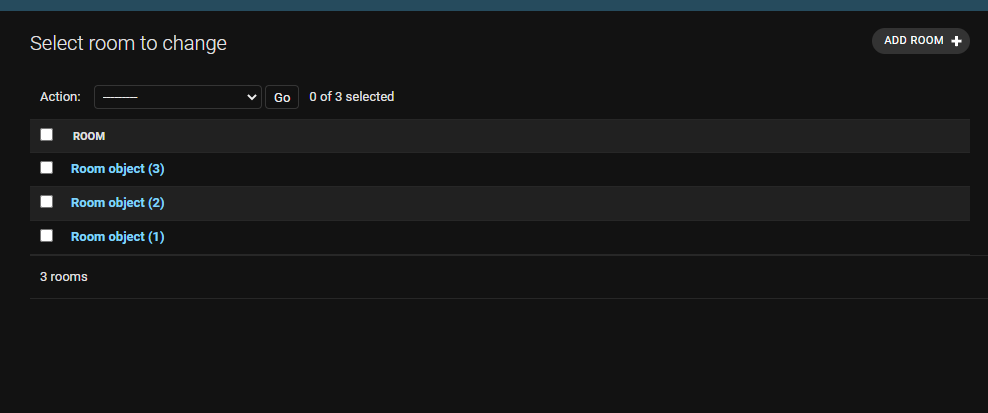
Accesand link-ul 127.0.0.1:8000/admin se poate observa o sectiune Rooms.

A screenshot of a computer

Description automatically generated with medium confidence



Aici se pot vizualiza camera de chat deja existente sau se poate crea una noua apasand butonul Add room.



Inainte de a intra intr-o camera de chat utilizatorul trebuie sa introduca numele acesteia si numele cu care va fi identificat.

Graphical user interface, text, application

Description automatically generated

* Pe langa text, utilizatorii pot posta si imagini.

Am adaugat in settings.py 2 variabile (MEDIA\_ROOT si MEDIA\_URL) care reprezinta calea unde pozele vor fi stocate, respective url-ul pentru browser pentru a accesa pozele prin intermediul Http.

Atunci cand se face un request de tipul POST trebuie sa encodam datele sub forma multipart/form-data.

Graphical user interface

Description automatically generated with medium confidence

1. Etapa 3

Pentru realizarea acestei etape am urmarit tutorialul : [Dockerizing Django with Postgres, Gunicorn, and Nginx | TestDriven.io](https://testdriven.io/blog/dockerizing-django-with-postgres-gunicorn-and-nginx/)

Primul pas a fost sa imi instalez aplicatia Docker Desktop. Am continuat prin crearea unui fisier Dockerfile si a unuia docker-compose.yml.

Am facut update la variabilele  din settings.py.

Text

Description automatically generated

Apoi, am create un fisier .end.dev pt a stoca environment variables.

Text

Description automatically generated

Pentru configurarea Postgres am adaugat un serviciu nou in docker-compose.yml si am instalat Psycopg2.

Text

Description automatically generated

Am facut update la variabila Databases din settings.py:

Text

Description automatically generated

Am create un fisier requirements.txt care contine versiunile de Django, gunicorn si psycopg2.  
Text

Description automatically generated

Am adaugat un fisier entrypoint.sh pentru a verifica ca Postgres este healthy inainte de aplicarea migratiilor.

Text

Description automatically generated

Am create un fisier docker-compose-prod.yml pentru productie:

Text

Description automatically generated

De asemenea si fisiere .env.prod si env.prod.db :

A screenshot of a computer

Description automatically generated Text

Description automatically generated

Urmatorul pas important a fost adaugarea Nginx care sa functioneze care un proxy reverse pt Gunicorn pentru a primi request urile de la clienti.

Text

Description automatically generated

Am pornit applicatia prin rularea urmatoarelor comenzi:

Graphical user interface, text

Description automatically generated