

# Chat System

Student: Alexandru Iulian  
Grupa: 30641

<b>1. Cerințe realizate și utilitatea lor în situații reale</b>	<b>3</b>
<b>2. Modul de rezolvare</b>	<b>3</b>
<b>3. Probleme intalnite</b>	<b>4</b>
<b>4. Concluzii</b>	<b>4</b>

## 1. Cerințe realizate și utilitatea lor în situații reale

Acest sistem de chat oferă utilizatorilor posibilitatea de a comunica între ei cu ajutorul unei aplicații web. Aceștia au posibilitatea de a crea camere noi de chat unde alți utilizatori pot intra și trimite mesaje în timp real. De asemenea, atunci când o persoană scrie în chat, celelalte persoane sunt notificate prin afișarea unui mesaj pe interfața grafică. Fiecare utilizator își poate crea cont și se poate loga cu un nume de utilizator și parola.

Intrat pe prima pagină a platformei, utilizatorul are posibilitatea de a se înregistra cu un cont nou sau de a intra într-un cont deja existent. După acest pas, el este redirectionat către pagină unde sunt listate toate camerele existente. Acesta are posibilitatea fie de a intra într-o cameră existentă, fie de a crea o nouă cameră completând câmpul pentru numele camerei. Odată intrat într-o cameră, acesta poate vedea mesajele deja trimise și poate trimite un mesaj nou.

În această fază a proiectului, am pregătit deploy-ul în Docker pentru acest proiect. De asemenea, am migrat baza de date la PostgreSQL.

## 2. Modul de rezolvare

Pentru realizarea proiectului am urmărit mai multe tutoriale. A fost nevoie instalarea plugin-ului “channels” pentru conectarea prin web socket-uri pentru trimiterea mesajelor.

Aplicația este împărțită în trei aplicații django. Prima este modulul principal făcut atunci când am creat proiectul. Al doilea modul este modulul “principal” care se ocupă cu login/sign up pentru user. Aici am definit și un director “templates” unde am definit un fișier html base pe care îl putem extinde în fiecare altă pagină creată. Acesta conține navigation bar-ul și toată logica aferentă pentru logout, login, signup și navigare la pagină de camere. Al treilea modul este cel care se ocupă de sistemul de chat. Aici am definit o pagină html pentru a afișa lista de camere și a crea o cameră nouă și o altă cameră html care este camera de chat unde utilizatorii pot trimite și vizualiza mesaje. În acest “modul” am creat și o clasă de consumer pentru websocket. Aceasta are rolul de a conecta utilizatorul la websocket și de a realiza acțiunile atunci când se primește un mesaj nou. Aici se creează un canal pentru camera aferentă, iar fiecare utilizator care intră în acest canal este adăugat unui grup. Astfel, atunci când un nou mesaj este postat, toți grupul

este notificat. Mesajele sunt stocate în baza de date, astfel acestea sunt încărcate atunci când intrăm într-o cameră. Fiecare mesaj este aferent unei camere.

Atunci când un utilizator scrie un mesaj într-o cameră, ceilalți utilizatori sunt anunțați pe chat că acesta scrie. Pentru a implementa această cerință, am folosit același sistem ca cel de trimis mesaje, însă am făcut o diferențiere între ele, adică un câmp de tipul "message\_type". Acesta poate să fie "new\_message" atunci când se trimite un mesaj nou sau "type\_event" atunci când un utilizator scrie un mesaj. Pentru evenimentul "type\_event" se afișează pe ecran mesajul "User X is typing" timp de 1 secundă, astfel dacă acesta se oprește din scris, mesajul va dispărea.

Pentru a rula sistemul:

- Activare virtual environment: **source venv\_django/bin/activate**
- Navigare în directorul proiectului: **cd project**
- Rularea server-ului: **python manage.py runserver**
- Sistemul este disponibil la adresa <http://127.0.0.1:8000/admin/>

Pentru a putea face migrare a bazei de date pe Postgres și pentru a face deploy în Docker am urmărit un tutorial. Pentru migrarea pe Postgres am modificat în settings.py db engine și am adăugat credențiale pentru baza de date Postgres.

Pentru a face deploy pe Docker, am creat un fișier Dockerfile, un fișier docker-compose.yml și un .env.dev. În docker-compose am definit un serviciu pentru baza de date, folosit un imagine de postgres și un service pentru serverul de Django. Am specificat faptul că serverul de Django trebuie să depindă de baza de date, astfel baza de date Postgres va trebui să pornească înaintea serverului de Django. În Dockerfile am instalat o versiune de python și am setat un director de lucru. Am copiat fișierul requirements și am instalat toate pachetele de care depinde proiectul.

Fișierul .env.dev detine variabilele de mediu pentru credențialele bazei de date.

Pentru a rula containerul de Docker:

- **docker-compose up —build**

### 3. Probleme intalnite

Una dintre probleme intalnite in realizarea proiectului a fost conectarea prin web socket. După implementarea codului cu ajutorul unui tutorial, am intampinat o problema cu plugin-ul “channels” versiunea 4.0.0. Conectarea la websocket nu se putea realiza cu aceasta versiune de plugin din cauza unei erori a librăriei, astfel a fost nevoie sa fac downgrade la versiunea 3.0.5. Am ajuns la aceasta actiune după mai multe căutări îndelungate despre eroarea primită la conectare.

O alta problema intalnite a fost la crearea containerului de Docker, prin care serverul de Django nu se conecta la baza de date Postgres, deoarece serverul

### 4. Concluzii

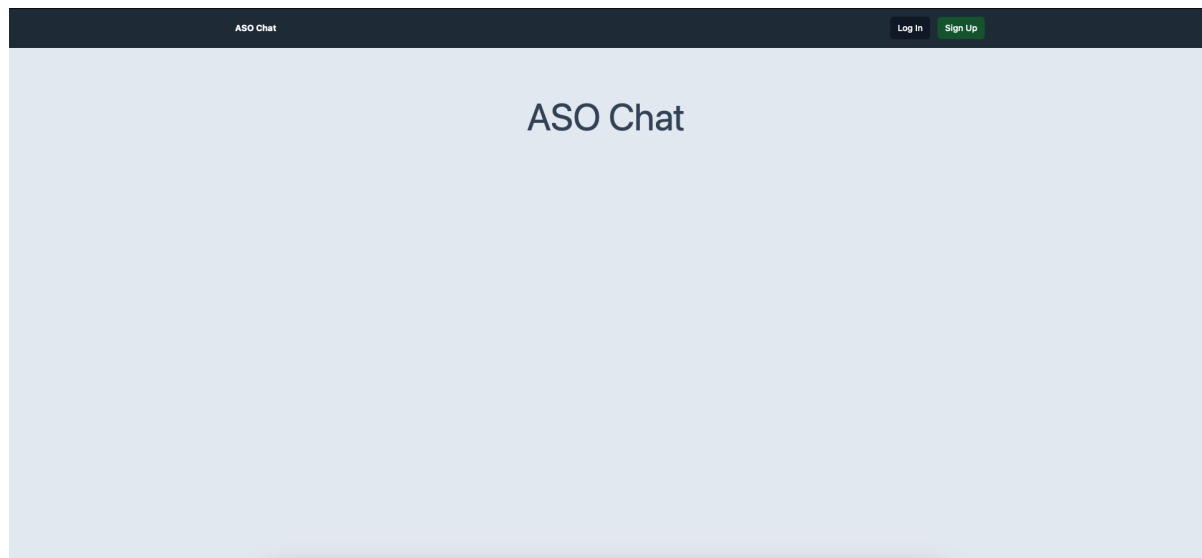
În urma realizării acestui proiect, consider ca am invatat principiile de bază pentru framework-ul Django. Consider ca acesta reprezinta un mediu de dezvoltare avansat care oferă o dezvoltare rapidă pentru aplicațiile web.

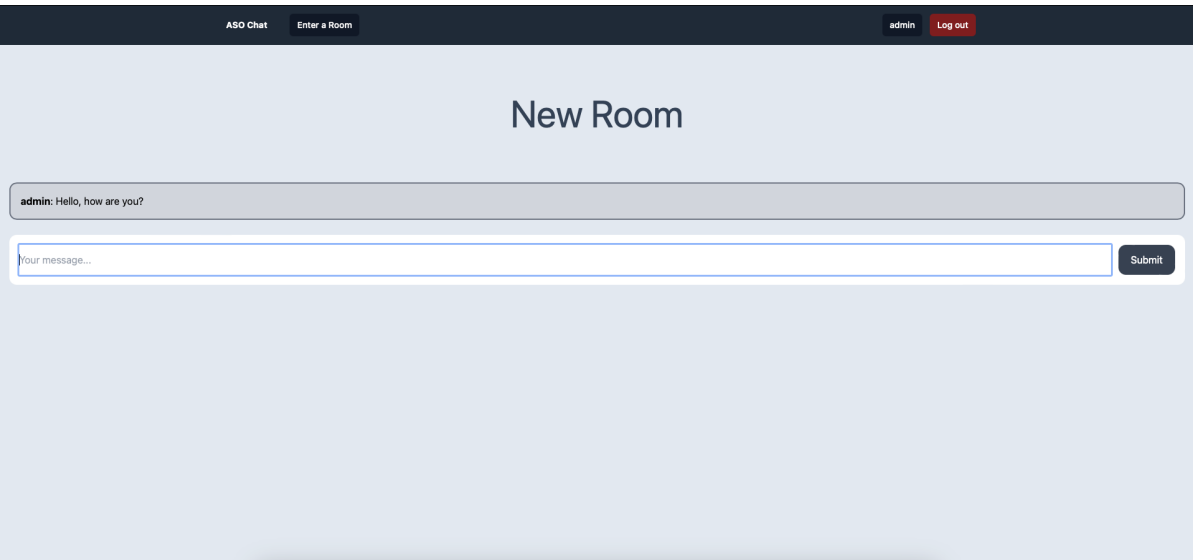
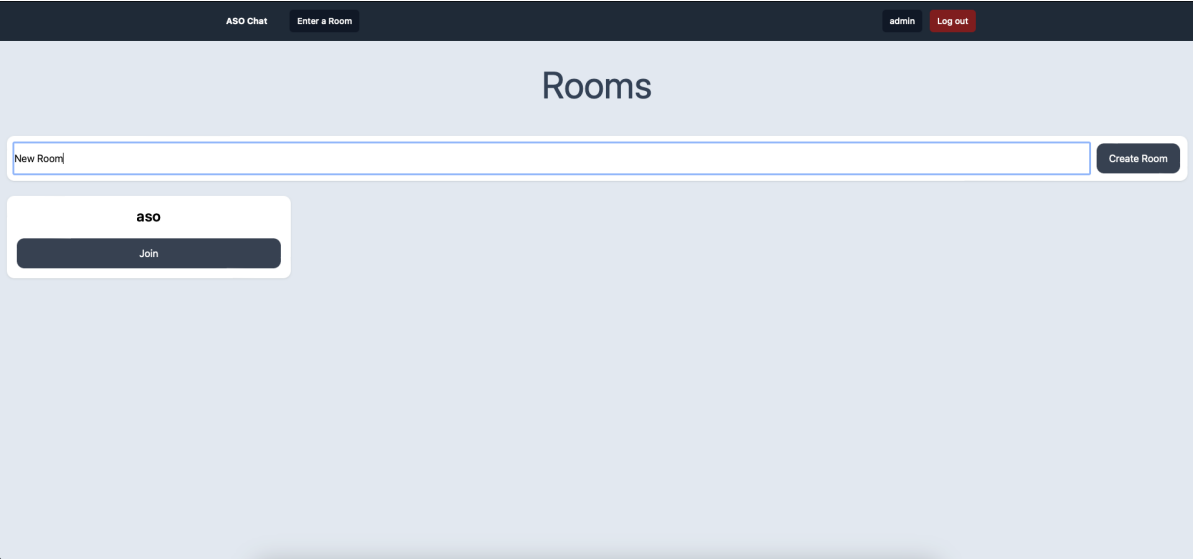
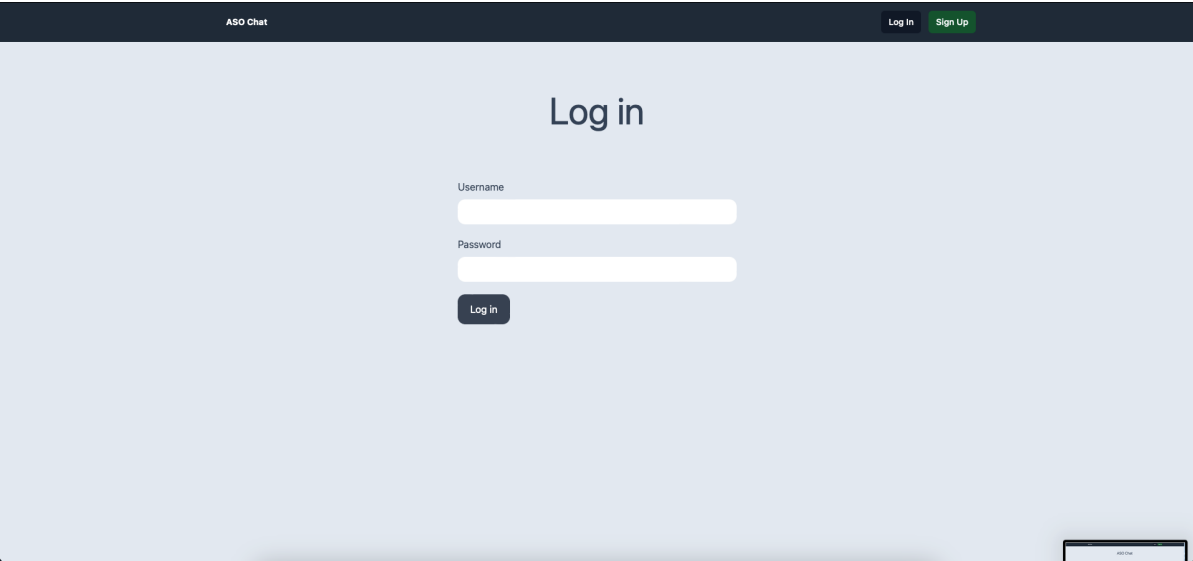
Cel mai important lucru pe care l-am invatat în realizarea acestui proiect este de bază lucru cu websocket. Consider ca mai am multe de invatat, fiind un mod de comunicare complex, însă am putut învăța bazele și am putut implementa cerințele pentru proiect.

Tutorial urmarit: [https://www.youtube.com/watch?v=SF1k\\_Twr9cg](https://www.youtube.com/watch?v=SF1k_Twr9cg)

Tutorial Docker si Postgres:







<https://testdriven.io/blog/dockerizing-django-with-postgres-gunicorn-and-nginx/>





☐ Only show running containers

Search

<input type="checkbox"/>		NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input type="checkbox"/>	▼	 project	-	Running (2/2)			<div><div></div><div></div><div></div></div>
<input type="checkbox"/>		 db-1 d8d982f06fda 	<a href="#">postgres:13.0-alpine</a>	Running		7 seconds ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>		 web-1 a7dddfbf1248 	<a href="#">project-web:latest</a>	Running	<a href="#">8000:8000</a> 	6 seconds ago	<div><div></div><div></div><div></div></div>