Proiect ASO

Etapa 2

Timaru Florinel-Vasile

1. Cerinte realizate si utilitatea lor in situatii reale

In aceasta etapa a fost implementata o aplicatie web, in framework-ul Django, utilizand modul de lucru cu template-uri. Fiecare utilizator isi poate crea un cont, se poate autentifica, poate vizualiza toate camerele create de ceilalti utilizatori si de asemenea poate crea la randul lui o camera. Acesta poate intra intr-o camera de chat, si poate trimite mesaje si de asemenea poate vizualiza mesajele de ceilalti utilizatori in acea camera. Mesajele sunt stocate in baza de date din memoria aplicatiei, asadar toate mesajele vor fi disponibile pana la inchiderea acesteia.

1. Modul de rezolvare

Pentru rezolvarea aplicatiei a stat la baza o singura aplicatie de tip Django, si de asemenea instalat plugin-ul Channels v.3.0.5, pentru lucrul cu WebSocket-uri. De asemenea in proiect se afla directorul templates, unde se pot gasi toate fisierele de tip Django HTML.

De asemenea in fisierul de configurare(settings.py), au fost adaugate aplicatia, plugin-ul channels, si ASGI.

La baza fisierelor Django HTML, sta un fisier intitulat base.html, pe care toate celelalte fisiere HTML il mostenesc. Acesta e compus dintr-un navbar si intreaga fereastra de sub acesta, ce coloreaza background-ul si adauga un titlu paginii. Pentru tot ce tine de user, login, logout, register a fost mostenit din auth-ul Django, oferind o productivitate ridicata implementarii acestuia.

Pentru login, in baza de date user-ul este stocat cu un username, password1 si password2.

Odata autentificat, user-ul este redirectionat pe o pagina de camere, unde este folosit un tempate ce afiseaza toate camerele din baza de date. In baza de date tabela de camere numita Room contine un nume si un „slug” ce este unic, pe post de ID. Fiecare utilizator poate crea o camera folosind butonul din navbar.

Utilizatorul poate alege o camera si va fi redirectionat pe alta pagina html ce va contine mesajele trimise de toti ceilalti utilizatori din acea camera. Mesajele sunt stocate in baza de date, care ca si field-uri are urmatoarele: o referinta catre camera, o referinta catre utilizatorul ce a trimis mesajul, un content si o data, pentru a putea fi ordonate.

Tot aici se creeaza un websocket, la care se conecteaza utilizatorul. Prin clasa de consumer, ce faciliteaza conectarea acestuia la socket, se realizeaza si trimiterea si receptia de mesaje. Clasa de consumer creeaza cate un canal separat pentru fiecare camera, iar utilizatorul mereu cand incearca sa se conecteze la websocket, acesta incearca sa se conecteze canalului respectiv, urmand ca apoi sa fie adaugat intr-un grup.

1. Probleme intalnite

Una din problemele majore in implementarea aplicatiei a fost implementarea clasei Consumer si creearea WebSocket-ului, acesta fiind un concept relativ nou. Utilizatorul realiza „handshake-ul”, insa nu se putea conecta la canal si era imediat deconectat.

De asemenea limbajul JavaScript m-a incurcat destul de mult, deoarece text editor-ul nu-mi oferea sugestii iar debugging-ul a fost greu de realizat si inteles.

1. Concluzii

In final consider ca aplicatia este o aplicatie stabila, datorita modului rapid de lucru cu framework-ul Django, in comparatie cu framework-ul Java Spring. De asemenea cred ca ar fi benefic documentarea mai in detaliu in legatura cu WebSocket-urile si sintaxa JavaScript.

Capturi de ecran:

Graphical user interface, website

Description automatically generated

Graphical user interface, application, website

Description automatically generated