Chat App – Faza 2 Proiect ASO

Diaconu Andrei

In cea de-a doua parte a proiectului am avut de implementat o aplicatie de tip chat cu ajutorul acealuiasi framework – Django. Desi tema 1 nu se lega direct cu aceasta noua tema in dezbatare, mi-a fost de folos pentru a inteleg bazele si modul de functionare a acestui framework/environment nou de lucru.

Ca orice aplicatie aceasta trebuie sa urmeze modelul de register(sign up) si login (sign in). Pe langa aceasta, ca orice aplicatie de tip chat, userii pot sa comunice direct intre ei. Ca puncte bonus am atins posibilitatea userilor sa incarce pe langa mesaje de tip text si imagini fara a fi salvate in baza de date si crearea de camere noi pe chat cu posibilitate de invitatie a unor noi useri prin trimiterea unor email-uri.

Poate nu este neaparat considerata o biblioteca, dar initial as vrea sa declar ca am folosti tailwind css pentru o stilizare mai rapida si frumoasa a aplicatie. Am adaugat in settings.py resursele necesare, iar mai apoi in fiecare template create in head trebuie sa exista referinta la scriptul de tailwind ca acesta sa-si aplice modificarile. Tot pentru tailwind am folosit un middleware care da reset la pagina la fiecare modificare facuta (in scopul vederii schimbarii unor modificari de aspect/clase).

Primele mele interactiuni cu requesturile si form-urile in django au fost acestea de login si sign up. Fiind deja din librariile lor majoritatea logicii implementate mi-a fost mai usor sa inteleg exact cum functioneaza si sa pot mai incolo face propriile mele form-uri. Asadar majoritatea bibliotecilor folosite apartin de django si le-am importat pentru a avea niste form-uri deja definite, sau restrictiona accesul unor useri neconectati (login decorators, authenticate, login, logut...).

Alte biblioteci uzuale adaugate sunt cele de redirect sau render care ne permit alegerea caror pagini sa fie returnate in urma exacutarii requsturilor. Mana in mana merg si Http/Json response care ne permit interactiunea cu raspunsul acestor requesturi de care ziceam mai sus.

Pentru trimiterea de imagini si nesalvarea lor in baza de date a fost nevoie de adugarea default storageului atat ca librarie in views cat si in settings sau urls. Tot in settings si in views au fost folosite send_email helpers from django.core care ne permite setarea de email a unui host si trimiterea de catrea acesta de mailuri foarte usor si rapid, setand cativa paramterii precum subject, body sau destinatiarii.

Desi ideal la aceata tema era integrarea unui websocket am folosit jquery/ ajax requesturi. Este putin mai lent, nu asa eficient insa in cadrul unei aplicatii de genul acesta mai mica ca si accesibilitate mi-am permis folosirea de asemenea requesturi.

Pentru obtinerea de mesaje noi am facut un script care se executa la fiecare secunda si extrage mesajele existnte in baza de date pentru camera pe care un user se afla in acel moment. Goleste #chat-area care este un tag de lista neordonata si o populeaza la fiecare secunda cu noile mesaje venite. Mentionz ca aspectul difera pentru mesajele trimise de userul conectat fata de celelalte mesaje. Asadar a fost nevoie de putin structurare.

Pe acelasi principiu, un ajax request se executa si cand se trimite un mesaj. Fara a se da refresh, un mesaj nou se creaza .Si aici a fost nevoie de o structurare pentru ca am integrat si optiune de incarcare de imagine si ajax requestul a avut nevoie de mai multi parametrii.

Legat de problemele intampinate de-a lungul implementarii pot aminti cea mai mare dintre ele si anume verisunile de channels si django. Ultima versiune de django cu ultima versiune de channels nu functiona deloc. Am downgradat pe rand si channels si django si am ajuns sa porneasca aplicatia cu channels insa paginile se incarcau la infinit. Asadar am fost nevoit sa renunt la ideea de websocket si channels in favoarea ajax requesturilor. Lucrez in Ruby on Rails si mi-sunt la indemana jquery-urile asadar nu ma declar neaparat nemultumit, insa ar fi frumos sa putem beneficia de un demo de integrare a websocket-ului

Alte mici probeleme intampinate au fost la send_email, cand google cere folosirea unei parole speciale pentru aplicatii, asadar cu un indrumator bun scurtam tipul de implementare a acestui feature care nu tine neaparat de cod ci mai mult de politica de lucru workspace-ului Google.

In concluzie mi-a placut sa implementez chat-ul, este un task interesant si as putea veni cu imbunatatire si sa inlocuiesc cu un websocket. Django este un framework foarte organizat si destul de straight forward, asadar voi avea in vedere folosirea lui pe mai departe.

Bibliografie:

https://www.youtube.com/watch?v=tUqUdu0Sjyc

https://www.youtube.com/watch?v=IpAk1Eu52GU