UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Chat minimalist Django
Proiect ASO 2

Student: Alexuc Tudor
Grupa: 30642
Îndrumător de laborator: Andrei Bogdan Leucuta

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Cerințele rezolvate

Realizarea unui chat minimalist cu ajutorul Django in care:

1. utilizatorii se pot loga
2. utilizatorii pot posta mesaje si vizualiza mesajele postate.

Mesajele se vor salva in baza de date (SQLite), folosind ORM-ul din Django . Paginile vor fi realizate cu ajutorul template-urilor. Unul din punctele optionale implementate a fost existenta a mai multe camere de chat si abilitatea fiecarui utilizator de a crea o camera.

Modul de rezolvare

Unul din pachetele folosite in lucrare este AJAX(**Asynchronous JavaScript And XML)**. Acest pachet a fost folosit pentru a procesa trimiterea mesajelor de catre utilizator si query-urile o data la 1000ms asupra bazei de date pentru a lua noile mesaje, in cazul in care exista. Avantajul folosirii AJAX este ca pagina nu trebuie reincarcata, libraria avand ca scop updatarea anumitor parti din pagina.

Modul ideal de folosire AJAX este impreuna cu jQuery. jQuery ajuta la scrierea scripturilor Javascript in cadrul paginilor HTML.

Pentru utilizare a fost nevoie de urmatoarea linie:

```
<script   src="https://code.jquery.com/jquery-3.1.1.min.js"
integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="
crossorigin="anonymous"></script>
```

De asemenea, cele doua script-uri realizate in cadrul paginii de chat sunt:

```
setInterval(function(){
    $.ajax({
        type: 'GET',
        url : "getMessages/{{room}}/",
        success: function(response){
            console.log(response);
            $("#display").empty();
            for (var key in response.messages)
            {
                var temp="<div class='container darker'>" +
                    "<b>"+response.messages[key].user+"</b>" +
                    "<p>"+response.messages[key].value+"</p>" +
                    "<span class='time-left'>"+response.messages[key].date+"</span>" +
                    "</div>";
                $("#display").append(temp);
            }
        },
        error: function(response){
            alert('An error occured')
        }
    });
},1000);
})
</script>
```

Scriptul de preluare a mesajelor pe interval

```
<script type="text/javascript">
  $(document).on('submit','#post-form',function(e){
    e.preventDefault();

    $.ajax({
      type:'POST',
      url:'/chatrooms/send',
      data:{
          username:$('#username').val(),
          room_id:$('#room_id').val(),
          // isTyping:$(false),
          message:$('#message').val(),
        csrfmiddlewaretoken:$('input[name=csrfmiddlewaretoken]').val(),
      },
      success: function(data){
         //alert(data)
      }
    });
    document.getElementById('message').value = ''
  });
</script>
```

Scriptul pentru trimiterea mesajelor

Pentru realizarea cerintelor am creat doua aplicatii noi in cadrul proiectului si anume **accounts** si **chatrooms** (Ambele au fost adaugate in sectiunea INSTALLED_APPS din settings.py).

Accounts contine toate functionalitatile de login, logout, registre si reset password(cu trimitere de email) , toate realizate cu ajutorul librariei auth din django.
Alte linii necesare in settings.py pentru implementarea sistemului de autentificare sunt:

```
LOGIN_REDIRECT_URL = 'chathome'
LOGOUT_REDIRECT_URL = 'home'
```

```
EMAIL_BACKEND = "django.core.mail.backends.filebased.EmailBackend"
EMAIL_FILE_PATH = str(BASE_DIR.joinpath('sent_emails'))
```

Chatrooms este partea principala a acestei etape, fiind aplicatia pentru chat.
Pentru implementarea acesteia am creeat 3 modele noi :

```
class User(models.Model):
    username = models.CharField(max_length=100)
    password = models.CharField(max_length=100)
    email = models.CharField(max_length=100)
```

```
class Room(models.Model):
    name = models.CharField(max_length=1000)
```

```
class Message(models.Model):
    value = models.CharField(max_length=10000000)
    date = models.DateTimeField(default=datetime.now, blank=True)
    room = models.CharField(max_length=1000000)
    user = models.CharField(max_length=1000000)
```

User - pentru a gestiona persoanele care trimit mesaje
Room - pentru a gestiona chat room-urile
Message - contine atat un id de room cat si un id de user pentru a sti ce user a scris in ce room

Modul in care functioneaza aplicatia este urmatorul:
Utilizatorul se logheaza pe site.
Pe pagina principala are optiunea de a crea o camera sau de a intra intr-o camera existenta.

Pentru procesarea acestor optiuni avem functia **checkroom** din views.py

```
def checkroom(request):
    room = request.POST['roomId']

    if 'create' in request.POST:
        if Room.objects.filter(name=room).exists():
            messages.error(request, "Room already exists.")
            return redirect('chathome')
        else:
            new_room = Room.objects.create(name=room)
            new_room.save()
            return redirect('chatroom/'+room)

    if 'join' in request.POST:
        if Room.objects.filter(name=room).exists():
            return redirect('chatroom/'+room)
        else:
            messages.error(request, "Room does not exist.")
            return redirect('chathome')

    messages.error(request, "Why?")
    return redirect('chathome')
```

Aceasta functie verifica daca utilizatorul doreste sa intre sau sa creeze o camera si realizeaza optiunea ceruta sau afiseaza un mesaj de eroare corespunzator.

Probleme întâlnite și modul de rezolvare

Pentru aplicatia de chat simpla realizata nu au fost multe probleme care merita mentionate, in schimb in incercarea de realizare a altor obiective optionale am intampinat cateva probleme, motiv pentru care nu au fost realizate.

Nefiind foarte familiar cu jquery nu am reusit sa introduc in scriptul de preluare a mesajelor si optiunea de imagini dar am reusit trimiterea acestora. Problema a fost la crearea obiectului html pentru mesaj, ceva din modul in care preia jQuery datele pentru script facea un request de POST in plus pe un link neexistent.

O alta problema similara a fost la cerinta optionala de a aparea mesajul “<User> is typing…” pentru ceilalti utilizatori din camera cand unul din ei scrie. Problema intampinata la aceasta cerinta a fost optinuile de listeneri care putea fi pusi pe input si modul in care acestia interactionau cu AJAX.

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Mod de utilizare al site-ului

Cont admin:
- Username: adminuser
- Password: adminpassword

Cont simplu creat cu flow-ul de inregistrare implementat:
- Username: simpleuser
- Password: simplepassword

Cel mai usor mod de a testa functionalitatea e prin a deschide o pagina normala de browser si o pagina incognito deoarece nu poti fi logat pe mai multe conturi in cadrul aceluiasi browser.
Apoi de pe ambele ferestre se intra pe aceeasi camera si se scrie mesaje.
De asemenea, se pot crea doua conturi noi in cazul in care se doreste sa se testeze aplicatia cu tot cu flow-ul de autentificare.

Link Github:
https://github.com/DMyrm/ASO-Django-Chat-App

Concluzii

Prin realizarea acestei lucrari mi-am reimprospatat cunostiintele legate de sistemul si libraria de autentificare integrat in Django si modul de utilizare al acestuia si am invata cateva metode de lucru cu AJAX si jQuery care sunt doua unelte foarte importante pentru HTML.