Partea 2

Proiect ASO 2021

Rus Mihai Tudorel

Grupa 30643

1.Cerinte rezolvate:

Crearea unui mediu de dezvoltare pentru a creea o aplicatie simpla, un chat public si anume instalarea unei masini virtuale, pe care am instalat sistemul de operare Ubuntu 20.04. Instalarea mediului Django, pentru a putea modela aplicatia si toate metodele din aceasta. Pornirea serverului destinate aplicatiei si rularea catorva comenzi simple prezente pe pagina de ajutor a mediului Django.

Pentru cea de a doua parte a proiectului, am realizat un chat minimalist, unde utilizatorii se pot loga cu username ul propriu intr o anumita camera de chat, pentru a putea comunica cu ceilalti.

2.Modul de rezolvare:

In prima faza, am verificat sa fie prezent pe sistem, biblioteca python fara de care nu am reusi sa rulam codul in Django prin scrierea linii python3 –version. Dupa ce ne am asigurat ca avem prezent mediul python instalat, putem incepe prin a instala mediul Django, dar doar intr un virtual environment, pentru a nu instala toate bibliotecile si dependintele Django, daca nu avem nevoie de toate acestea, prin urmatoarele comenzi :

-pip install virtualenv

-python -m virtualenv env

-env\Scripts\activate

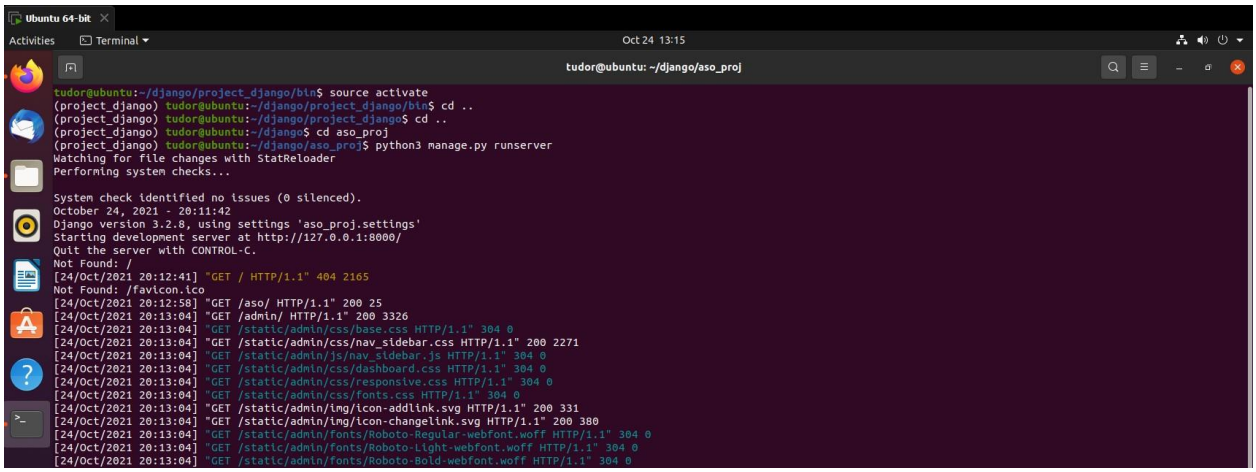
-pip install django

Odata rulate aceste comenzi, am instalat Django in environment ul env si l am activat. Pentru a creea un nou proiect gol, care are ca si functionalitate pornirea unui server si a unei aplicatii web simpliste este nevoie de urmatoarele comenzi :

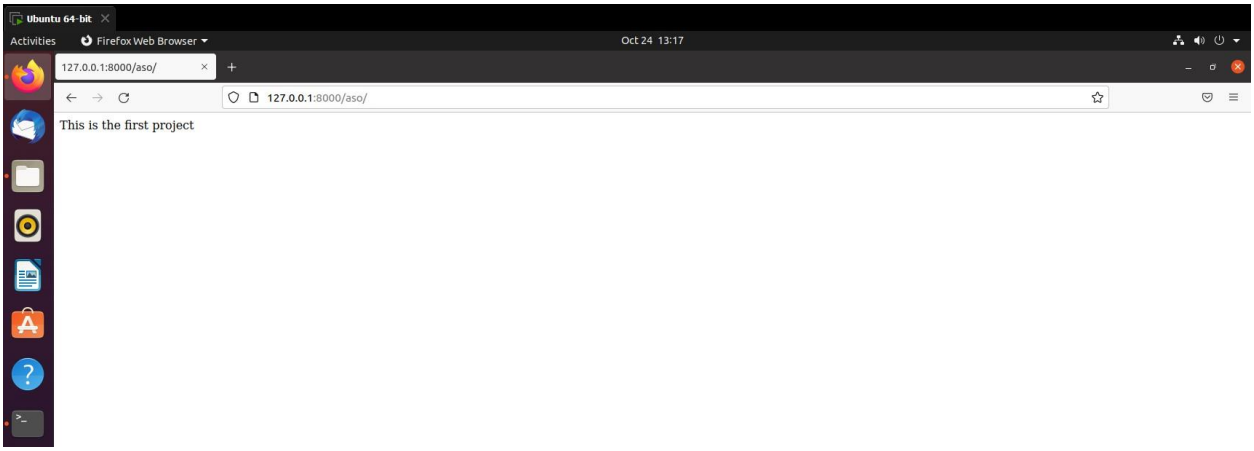
-django-admin startproject asoProject

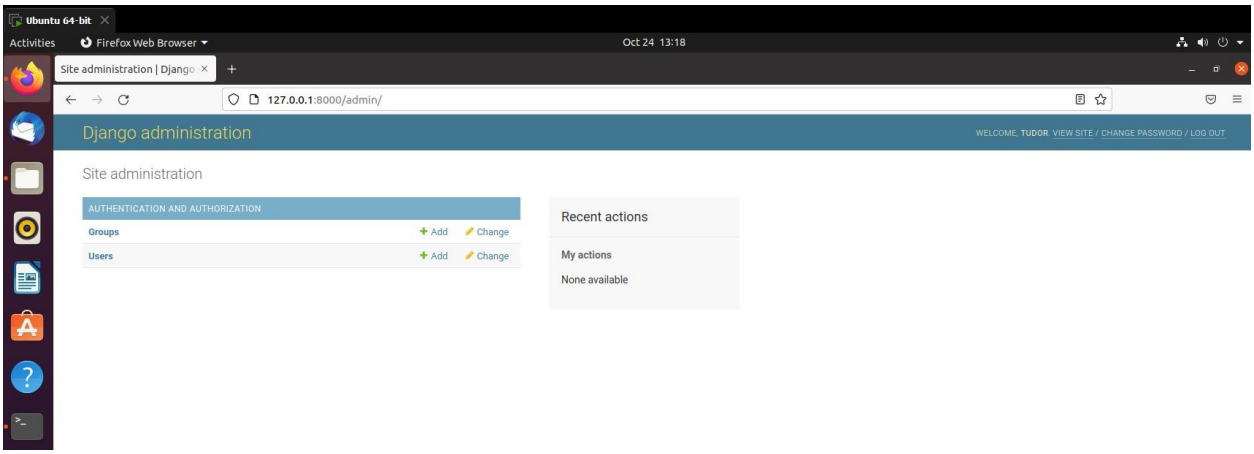
-source activate(in folderul environmentului deja instalat)

-python manage.py runserver(in folderul aplicatiei deja create)



Odata ce serverul a pornit, putem sa accesam un browser web, si sa ne conectam la adresa de ip: http://127.0.0.1:8000/ cu local-host pe portul 8000, unde putem vedea un site minimalist, pentru gestiunea unui site web.





In final pentru a putea avea toate setarile la proiect prezente pe site dar si pe local, ar trebui rulata comanda :

-python manage.py migrate (ca toate datele ce au fost introduce in browser sa fie prezente si in cod)

3.Pentru a putea realiza proiectul am reusit sa creez o serie de urls care sa fie caile catre paginile care se pot deschide in cadrul aplicatiei:

chatapp > chat > urls.py > ...

```python
from django.urls import path
from . import views

urlpatterns =[
    path('',views.home,name="home"),
    path('<str:room>/',views.room,name="room"),
    path('checkview',views.checkview,name="checkview"),
    path('send',views.send,name="send"),
    path('getMessages/<str:room>/',views.getMessages,name="getMessages")
]
```

Pentru fiecare URL in parte exista o metoda care implementeaza apelurile de GET,POST si UPDATE pentru useri si camera de chat:

```python
def home(request):
    return render(request, 'home.html')


def room(request, room):
    username = request.GET.get('username') # henry
    room_details = Room.objects.get(name=room)
    return render(request, 'room.html', {

        'username': username,
        'room': room,
        'room_details': room_details,
    })


def checkview(request):
    room = request.POST['room_name']
    username = request.POST['username']

    if Room.objects.filter(name=room).exists():
        return redirect('/'+room+'/?username='+username)
    else:
        new_room = Room.objects.create(name=room)
        new_room.save()
        return redirect('/'+room+'/?username='+username)
```

4.Stilizarea aplicatiei am realizat o in fisierele home.html si room.html unde am imbinat notiuni de html css si javascript pentru a putea face aplicatia una responsive si user-friendly.