Inginerie software

Sistem de cinematografie folosind Python Django

Facut de: Miklos Balazs și Ursu Razvan

Grupa 30234

Universitatea Technica din Cluj Napoca

Facultatea de automatica si calculatoare

Rezumat

1. Referinte principale utilizate:

1.1 Rezumat

Pentru proiectul final, ne-am decis sa continuam functionalitatiile implementate de la miniproiect adaugand functionalitati precum cea de log in, log out, operatii de adaugare, stergere si update in functie de tipul utilizatorului, precum si un chat in care mai multi utilizatori logati pot comunica intre ei, sau pe propiul lor chat pot posta diferite lucruri.

2. Descriere project

2.1 Miniproiect initial (cod sursa)

myrottenpotatoes/models.py

```
from django.db import models

# Create your models here.

class Movie(models.Model):


  title = models.CharField(max_length = 50)

  rating = models.CharField(max_length = 50)

  description = models.TextField()

  release_date = models.DateField()
```

myrottenpotatoes/virews.py

```
from myapp.models import Movie

from django.http import HttpResponse

from django.shortcuts import render



def indeex(request):

        return render(request, "index.html")


def creeate(request):

        ti = request.POST.get('title')

        rat = request.POST.get('rating')

        des = request.POST.get('description')

        dat = request.POST.get('release')
        if ti is '' :
                return HttpResponse('<br>Title cannot be empty<br>')


        movie=Movie(
                title=ti, rating=rat,description=des,release_date=dat
        )
        res='<br> Adding a film <br>'
        res+=movie.title
        movie.save();
        return HttpResponse(res)

def deleete(request):
        ti = request.POST.get('title')
        if ti is None :
                return HttpResponse('<br>Title cannot be empty<br>')
        movie=Movie.objects.get(title=ti)
        movie.delete()
        res='<br> Deleting a film <br>'
        res+=movie.title
        return HttpResponse(res)

def updaate(request):
        ti = request.POST.get('title')
        newrat = request.POST.get('rating')
        newdes = request.POST.get('description')
        if ti is None :
                return HttpResponse('<br>Title cannot be empty<br>')
        movie=Movie.objects.get(title=ti)
        movie.rating=newrat
        movie.description=newdes
        movie.save();
        res='<br> Updating a film <br>'
        res+=movie.title
        return HttpResponse(res)
```

myrottenpotatoes/templates/index.html

```
        <html>
  <body>
    <form name = "form">{% csrf_token %} Insert information

      <div style = "max-width:470px;">
                                <label> title </label>
          <input type = "text" style = "margin-left:10%;"
            placeholder = "title" name = "title" />
      </div>

      <br>

      <div style = "max-width:470px;">
                                <label> rating </label>
          <input type = "text" style = "margin-left:10%;"
            placeholder = "rating" name = "rating" />
      </div>
```

```
<br>

            <div style = "max-width:470px;">
                      <label> description </label>
    <input type = "text" style = "margin-left:10%;"
      placeholder = "description" name = "description" />
</div>

<br>
            <div style = "max-width:470px;">
                      <label> release_date </label>
    <input type = "text" style = "margin-left:10%;"
      placeholder = "release date" name = "release" />
</div>

<br>

     <button  type = "submit" formaction = "{% url "Create_movie" %}"
                     formmethod = "POST"
             value = "Create" >
             <strong>Create movie</strong>
     </button>
     <button  type = "submit" formaction = "{% url "Delete_movie" %}"
                     formmethod = "POST"
             value = "Delete" >
             <strong>Delete movie</strong>
     </button>
```

```
            <button  type = "submit" formaction = "{% url "Update_movie" %}"
                            formmethod = "POST"
                    value = "Update" >
                    <strong>Update movie</strong>
            </button>


      </form>


   </body>
</html>
```

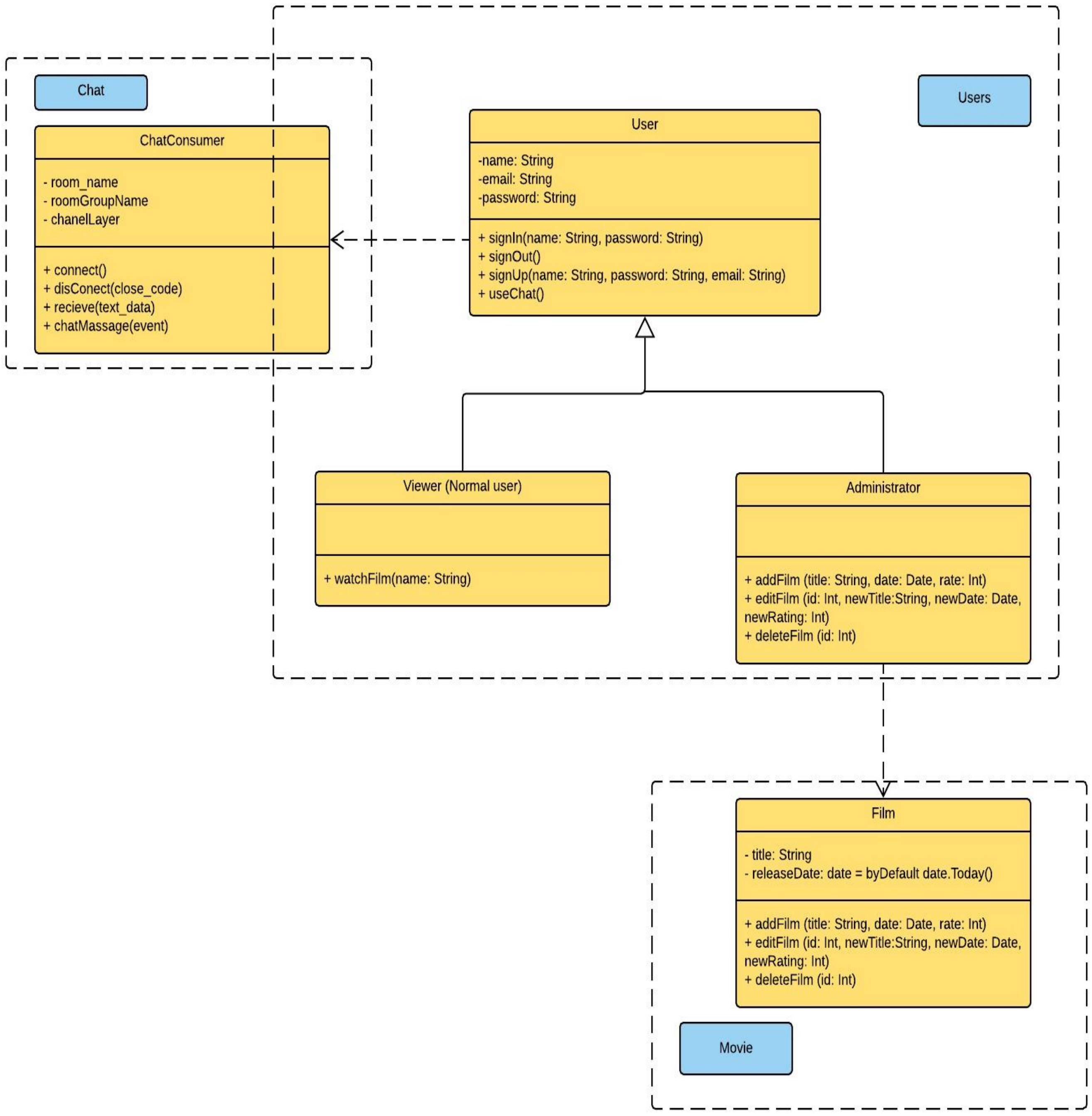
myrottenpotatoes/url.py

```
#from django.conf.urls import patterns, include, url
from myapp.views import creeate
from myapp.views import deleete
from myapp.views import updaate
from django.urls import path
urlpatterns = [
        path('create/', creeate, name = 'Create_movie'),
        path('delete/', deleete, name = 'Delete_movie'),
        path('update/', updaate, name = 'Update_movie'),
]
```

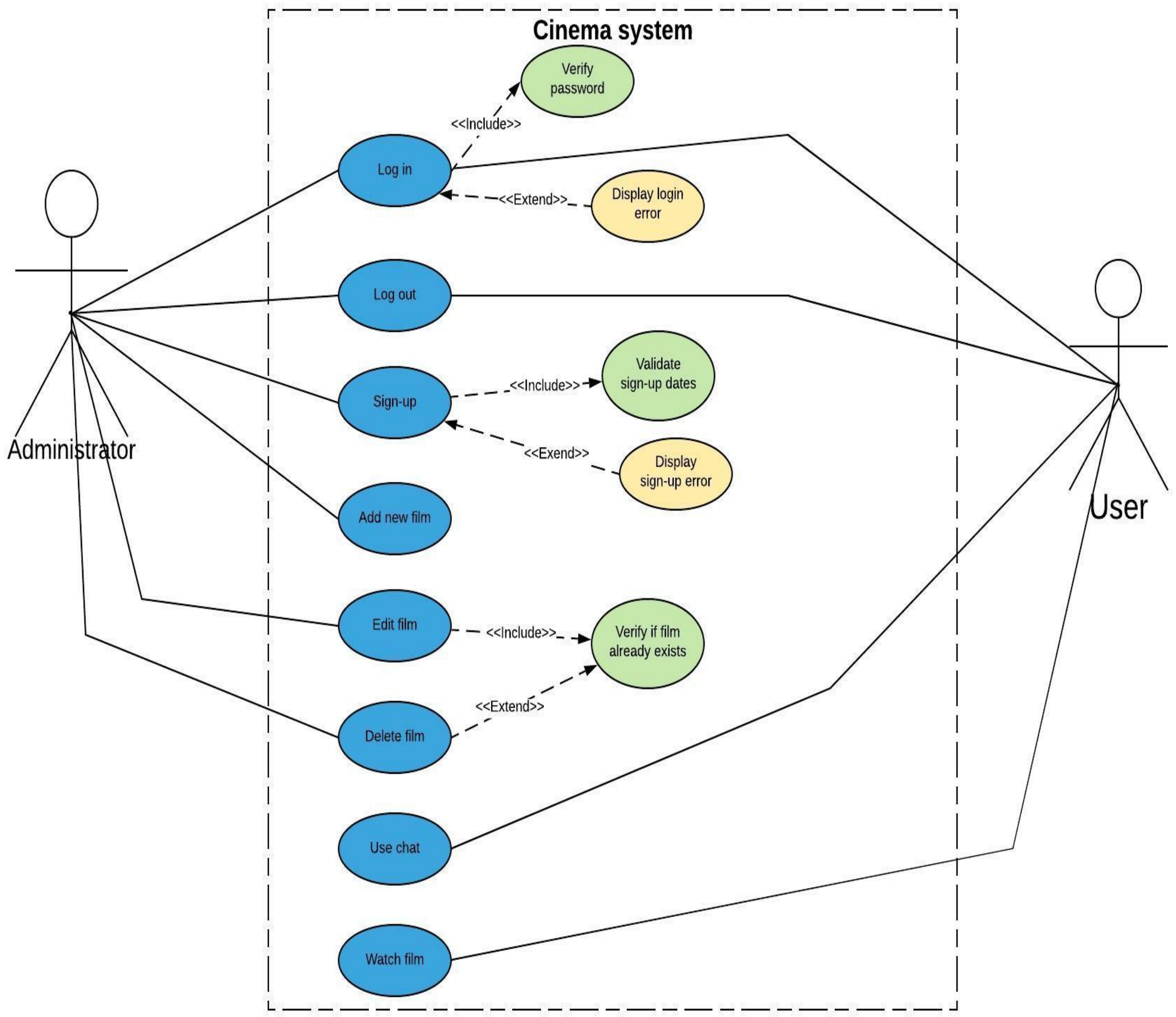
2.2 Proiect final

2.2.1 Diagrame UML

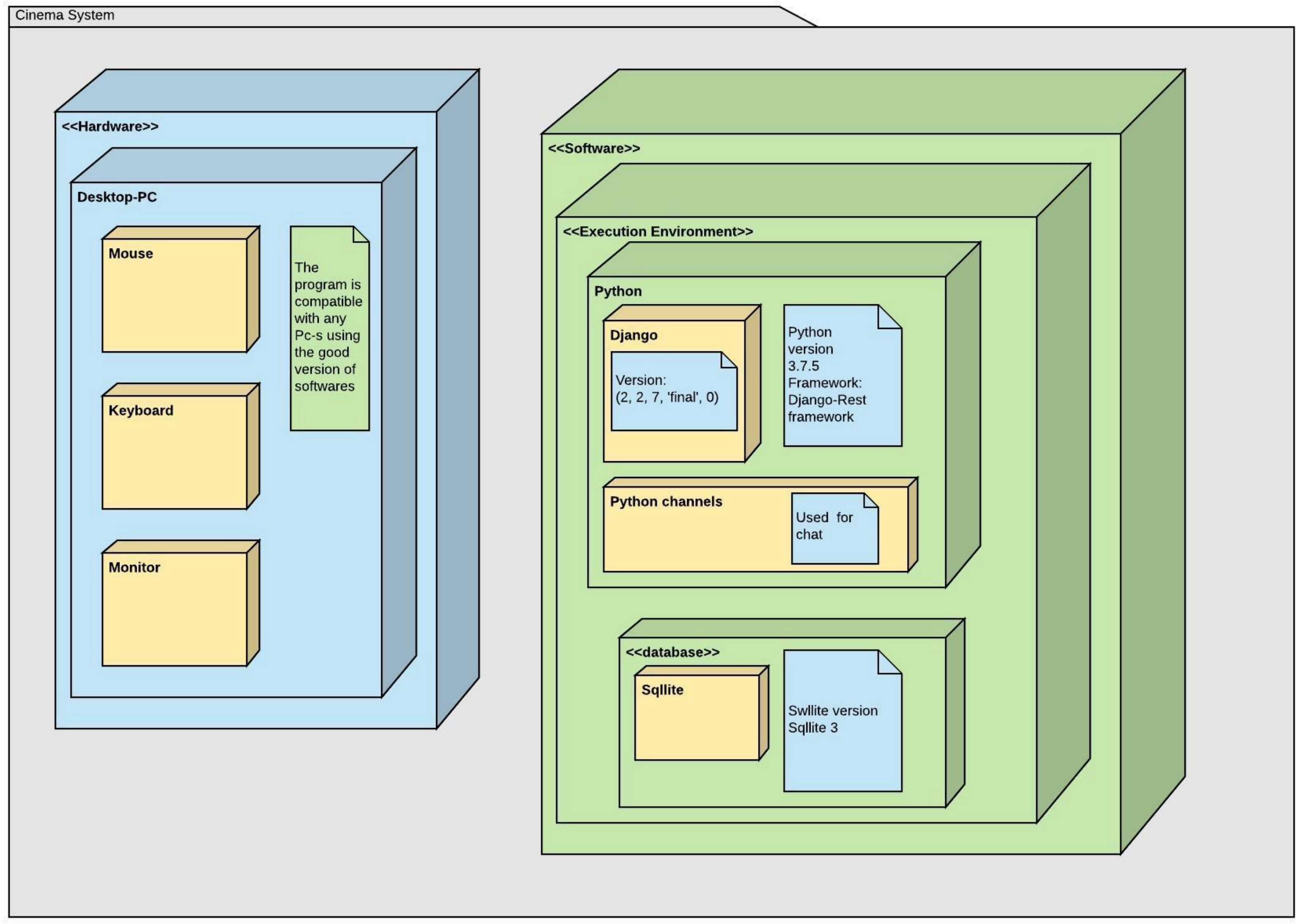
1. Diagrama de clasa:


Chat
ChatConsumer
- room_name
- roomGroupName
- chanelLayer
+ connect()
+ disConect(close_code)
+ recieve(text_data)
+ chatMassage(event)
User
-name: String
-email: String
-password: String
+ signIn(name: String, password: String)
+ signOut()
+ signUp(name: String, password: String, email: String)
+ useChat()
Users
Viewer (Normal user)
+ watchFilm(name: String)
Administrator
+ addFilm (title: String, date: Date, rate: Int)
+ editFilm (id: Int, newTitle:String, newDate: Date, newRating: Int)
+ deleteFilm (id: Int)
Film
- title: String
- releaseDate: date = byDefault date.Today()
+ addFilm (title: String, date: Date, rate: Int)
+ editFilm (id: Int, newTitle:String, newDate: Date, newRating: Int)
+ deleteFilm (id: Int)
Movie

2. Diagrama Use-Case



3. Diagrama deployment


Cinema System
<<Hardware>>
Desktop-PC
Mouse
Keyboard
Monitor
The program is compatible with any Pc-s using the good version of softwares
<<Software>>
<<Execution Environment>>
Python
Django
Version: (2, 2, 7, 'final', 0)
Python version 3.7.5 Framework: Django-Rest framework
Python channels
Used for chat
<<database>>
Sqllite
Swllite version Sqllite 3

2.2.2 Cod sursa complet

```
from django.apps import AppConfig



class UsersConfig(AppConfig):

    name = 'users'



from django import forms

from django.contrib.auth.models import User

from django.contrib.auth.forms import UserCreationForm



class UserRegisterForm(UserCreationForm):

    email = forms.EmailField()


    class Meta:

        model = User

        fields = ['username', 'email', 'password1', 'password2']

from django.shortcuts import render,redirect

from django.contrib.auth.forms import UserCreationForm

from django.contrib import messages

from django.contrib.auth.decorators import login_required

from .forms import UserRegisterForm

from django.contrib.auth import login, authenticate


def register(request):
```

```python
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username = username, password = 
raw_password)
            login(request,user)
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html', {'form': form})

@login_required
def profile(request):
    return render(request, 'users/profile.html')
```

```html
{% extends "movie/base.html" %}


{% block content %}
    <div class="content-section">
        <form method="POST">
            {% csrf_token %}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Log in</legend>
                {{form.as_p}}

            </fieldset>

            <div class="form-group">

                <button class="btn btn-outline-info"
type="submit">Login</button>

            </div>

        </form>

        <div class="border-top pt-3">

            <small class="text-muted">

                Need an Account? <a class="ml-2" href="{% url 'register'
%}">Sign In</a>

            </small>

        </div>

    </div>

{% endblock content %}

{% extends "movie/base.html" %}


{% block content %}

    <h2>You have been loged out</h2>


        <div class="border-top pt-3">

            <small class="text-muted">

                <a class="ml-2" href="{% url 'register' %}">Sign In</a>

            </small>

        </div>


{% endblock content %}

{% extends "movie/base.html" %}
{% block content %}
    <h1>
        {{ user.username}}
    </h1>
{% endblock content %}
{% extends "movie/base.html" %}

{% block content %}
    <div class="content-section">
        <form method="post">
            {% csrf_token %}
            {% for field in form %}
              <p>
                {{ field.label_tag }}<br>
                {{ field }}
                {% if field.help_text %}
                  <small style="color: grey">{{ field.help_text }}</small>
                {% endif %}
                {% for error in field.errors %}
                  <p style="color: red">{{ error }}</p>
                {% endfor %}
              </p>
            {% endfor %}
            <button type="submit">Sign up</button>
          </form>
    </div>
```

```
{% endblock content %}
```

Movie:

```
from django.contrib import admin

from .models import Post


admin.site.register(Post)

# Register your models here.


from django.apps import AppConfig



class MovieConfig(AppConfig):

    name = 'movie'





from django.db import models

from django.contrib.auth.models import User

from django.utils import timezone

from django.urls import reverse

from datetime import date


class Post(models.Model):

    title = models.CharField(max_length=100)

    rating = models.CharField(max_length=100)

    release_date = models.DateField(default = date.today )
    author = models.ForeignKey(User, on_delete = models.CASCADE)


    def __str__(self):

        return self.title


    def get_absolute_url(self):

        return reverse('post-detail', kwargs={'pk':self.pk})




from rest_framework import serializers

from .models import Post



class PostSerializer(serializers.ModelSerializer):

    class Meta:

        model = Post

        fields = ('id','title', 'rating', 'release date','author')



from django.urls import path,include

from . import views

from .views import PostListView, PostDetailView, PostCreateView,
PostUpdateView, PostDeleteView

from rest_framework import routers

from users import views as user_views

from django.conf.urls import include
from django.contrib import admin


"""router = routers.DefaultRouter()

router.register('post',views.PostView)

"""

urlpatterns = [

    path('', PostListView.as_view(),name = 'movie-home'),

    path('post/<int:pk>/', PostDetailView.as_view(),name = 'post-detail'),

    path('post/<int:pk>/update/', PostUpdateView.as_view(),name = 'post-
update'),

    path('post/<int:pk>/delete/', PostDeleteView.as_view(),name = 'post-
delete'),

    path('post/new/', PostCreateView.as_view(),name = 'post-create'),

    path('about/', views.about,name = 'movie-about')

]from django.shortcuts import render

from django.views.generic import ListView,DetailView,
CreateView,UpdateView,DeleteView


"""from rest_framework import viewsets,status,Response,APIView"""

from .models import Post

from .serializer import PostSerializer


#from django.http import HttpResponse


"""

class PostView(viewsets.ModelViewSet):

    queryset = Post.objects.all

    serializer_class = PostSerializer
```

```
"""
def home(request):
    context = {
        'posts' : Post.objects.all()
    }
    return render(request, 'movie/home.html', context)

class PostListView(ListView):
    model = Post
    template_name = 'movie/home.html'
    context_object_name = 'posts'
    ordering = ['-release_date']

class PostDetailView(DetailView):
    model = Post

class PostCreateView(CreateView):
    model = Post
    fields = ['title', 'rating', 'release_date']

    def form_valid(self, form):
        form.instance.rating = "PG"
        form.instance.author = self.request.user
        return super().form_valid(form)

class PostUpdateView(UpdateView):
    model = Post
    fields = ['title', 'rating','release_date']


    def form_valid(self, form):

        form.instance.author = self.request.user

        return super().form_valid(form)


class PostDeleteView(DeleteView):

    model = Post

    success_url = '/'


def about(request):

    return render(request, 'movie/about.html')


# Create your views here.
```

```
{% extends "movie/base.html" %}


{% block content %}

    <h1>about page</h1>

{% endblock content %}

{% load static %}

<html>

    <head>

        <!-- Required meta tags -->
        <meta charset="utf-8">

        <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">


        <!-- Bootstrap CSS -->

        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.cs
s" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">


        <link rel = "stylesheet" type = "text/css" href = "{% static
'movie/main.css' %}">

        {% if title %}

            <title>Django - {{title}} </title>

        {% else %}

            <title>Django Movie</title>

        {% endif %}

    </head>

    <body>

            <header class="site-header">

                    <nav class="navbar navbar-expand-md navbar-dark bg-
steel fixed-top">

                      <div class="container">

                        <a class="navbar-brand mr-4" href="/">Django
Movie</a>

                        <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarToggle" aria-controls="navbarToggle"
aria-expanded="false" aria-label="Toggle navigation">

                        <span class="navbar-toggler-icon"></span>
                        </button>

                        <div class="collapse navbar-collapse"
id="navbarToggle">

                          <div class="navbar-nav mr-auto">

                            <a class="nav-item nav-link" href="{% url
'movie-home' %}">Home</a>

                            {% if user.is_authenticated %}

                            <a class="nav-item nav-link" href="{% url
'post-create' %}">Add new Movie</a>

                            {% endif%}

                            <!--a class="nav-item nav-link" href="{% url
'movie-about' %}">Add new Movie</a-->

                          </div>

                          <!-- Navbar Right Side -->

                          <div class="navbar-nav">



                            {% if user.is_authenticated %}

                            <a class="nav-item nav-link" href="{% url
'index'%}">Chat</a>

                            <a class="nav-item nav-link" href="{% url
'room' user.username%}">My Chat</a>

                            <a class="nav-item nav-link" href="{% url
'profile'%}">Profile</a>

                            <a class="nav-item nav-link" href="{% url
'logout'%}">Logout</a>

                            {% else %}

                              <a class="nav-item nav-link" href="{% url
'login'%}">Login</a>
```

```
                              <a class="nav-item nav-link" href="{% url
'register'%}">Register</a>

                            {% endif%}

                          </div>

                        </div>

                      </div>

                    </nav>

                  </header>

                  <main role="main" class="container">

                        <div class="row">

                          <div class="col-md-8">

                            {% if messages %}

                            {% for message in messages %}

                              <div class="alert alert-{{ message.tags }}">

                                {{ message }}

                              </div>

                            {% endfor %}

                          {% endif %}

                          {% block content %}{% endblock %}

                          </div>

                          <div class="col-md-4">


                          </div>

                        </div>

                      </main>

         <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-
```

```
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>

        <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.mi
n.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>

        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>

    </body>

</html>

{% extends "movie/base.html" %}


{% block content %}

    {% for post in posts %}

    <article class="media content-section">

    <div class="media-body">

        <div class="article-metadata">

        <a class="mr-2" href="#">{{ post.author }}</a>

        <small class="text-muted">{{ post.release_date|date:"F d, Y"
}}</small>

        </div>

        <h2><a class="article-title" href="{% url 'post-detail' post.id
%}">{{ post.title }}</a></h2>

        <p class="article-content">{{ post.rating }}</p>

    </div>

    </article>

    {% endfor %}
{% endblock content %}

<!-- chat/templates/chat/index.html -->

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8"/>

    <title>Chat Rooms</title>

</head>

<body>

    What chat room would you like to enter?<br/>

    <input id="room-name-input" type="text" size="100"/><br/>

    <input id="room-name-submit" type="button" value="Enter"/>


    <script>

        document.querySelector('#room-name-input').focus();

        document.querySelector('#room-name-input').onkeyup = function(e) {

            if (e.keyCode === 13) {  // enter, return

                document.querySelector('#room-name-submit').click();

            }

        };


        document.querySelector('#room-name-submit').onclick = function(e)
{

            var roomName = document.querySelector('#room-name-
input').value;

            window.location.pathname = '/chat/' + roomName + '/';

        };

    </script>
```

```
</body>

</html>{% extends "movie/base.html" %}


{% block content %}

    <div class="content-section">

        <form method="POST">

            {% csrf_token %}

            <fieldset class="form-group">

                <legend class="border-bottom mb-4">Delete post</legend>

                <h2>Are you sure you want to delete
"{{object.title}}"</h2>

            </fieldset>

            <div class="form-group">

                <button class="btn btn-outline-danger"
type="submit">Yes,delete</button>

                <a class="btn btn-outline-secondary" href = "{% url 'post-
detail' object.id%}"> Cancel </a>

            </div>

        </form>

    </div>

{% endblock content %}

{% extends "movie/base.html" %}


{% block content %}

    <article class="media content-section">

    <div class="media-body">

        <div class="article-metadata">

        <a class="mr-2" href="#">{{ object.author }}</a>
        <small class="text-muted">{{ object.date_posted|date:"F d, Y"
}}</small>

        <div>

            <a class="btn btn-secondary btn_sm mt-1 mb-1" href = "{% url
'post-update' object.id %}"> Update</a>

            <a class="btn btn-danger btn_sm mt-1 mb-1" href = "{% url
'post-delete' object.id %}"> Delete</a>

        </div>

        </div>

        <h2 class="article-title"> {{ object.title }} </h2>

        <p class="article-content">{{ object.rating }}</p>

    </div>

    </article>

{% endblock content %}

{% extends "movie/base.html" %}


{% block content %}

    <div class="content-section">

        <form method="POST">

            {% csrf_token %}

            <fieldset class="form-group">

                <legend class="border-bottom mb-2">Add bew movie</legend>

                {{ form.as_p }}

            </fieldset>

            <div class="form-group">

                <button class="btn btn-outline-info"
type="submit">Add</button>

            </div>
        </form>
    </div>
{% endblock content %}


<!-- chat/templates/chat/room.html -->
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>Chat Room</title>
</head>
<body>
    <textarea id="chat-log" cols="100" rows="20"></textarea><br/>
    <input id="chat-message-input" type="text" size="100"/><br/>
    <input id="chat-message-submit" type="button" value="Send"/>
</body>
<script>
    var roomName = "{{ room_name|escapejs }}";

    var chatSocket = new WebSocket(
        'ws://' + window.location.host +
        '/ws/chat/' + roomName + '/');

    chatSocket.onmessage = function(e) {
        var data = JSON.parse(e.data);
        var message = data['message'];
        document.querySelector('#chat-log').value += (message + '\n');
    };


    chatSocket.onclose = function(e) {
        console.error('Chat socket closed unexpectedly');
    };


    document.querySelector('#chat-message-input').focus();
    document.querySelector('#chat-message-input').onkeyup = function(e) {
        if (e.keyCode === 13) {  // enter, return
            document.querySelector('#chat-message-submit').click();
        }
    };


    document.querySelector('#chat-message-submit').onclick = function(e) {
        var messageInputDom = document.querySelector('#chat-message-
input');
        var message = messageInputDom.value;
        chatSocket.send(JSON.stringify({
            'message': message
        }));


        messageInputDom.value = '';
    };
</script>
</html>
body {
    background: #fafafa;
```

```
  color: #333333;

  margin-top: 5rem;

}


h1, h2, h3, h4, h5, h6 {

  color: #444444;

}


ul {

  margin: 0;

}


.bg-steel {

  background-color: #5f788a;

}


.site-header .navbar-nav .nav-link {

  color: #cbd5db;

}


.site-header .navbar-nav .nav-link:hover {

  color: #ffffff;

}


.site-header .navbar-nav .nav-link.active {

  font-weight: 500;

}
```

```css
.content-section {
  background: #ffffff;
  padding: 10px 20px;
  border: 1px solid #dddddd;
  border-radius: 3px;
  margin-bottom: 20px;
}

.article-title {
  color: #444444;
}

a.article-title:hover {
  color: #428bca;
  text-decoration: none;
}

.article-content {
  white-space: pre-line;
}

.article-img {
  height: 65px;
  width: 65px;
  margin-right: 16px;
}
```

```css
.article-metadata {
  padding-bottom: 1px;
  margin-bottom: 4px;
  border-bottom: 1px solid #e3e3e3
}

.article-metadata a:hover {
  color: #333;
  text-decoration: none;
}

.article-svg {
  width: 25px;
  height: 25px;
  vertical-align: middle;
}

.account-img {
  height: 125px;
  width: 125px;
  margin-right: 20px;
  margin-bottom: 16px;
}

.account-heading {
  font-size: 2.5rem;
  }
```

Chat:

```
from django.apps import AppConfig



class ChatConfig(AppConfig):

    name = 'chat'



# chat/consumers.py

from channels.generic.websocket import AsyncWebsocketConsumer

import json

from channels.db import database_sync_to_async


from channels.db import database_sync_to_async



import django.contrib.auth


class ChatConsumer(AsyncWebsocketConsumer):

    async def connect(self):

        self.room_name = self.scope['url_route']['kwargs']['room_name']

        self.room_group_name = 'chat_%s' % self.room_name


        # Join room group

        await self.channel_layer.group_add(
            self.room_group_name,
            self.channel_name
        )

        await self.accept()

    async def disconnect(self, close_code):
        # Leave room group
        await self.channel_layer.group_discard(
            self.room_group_name,
            self.channel_name
        )

    # Receive message from WebSocket
    async def receive(self, text_data):
        text_data_json = json.loads(text_data)
        message = text_data_json['message']

        # Send message to room group
        await self.channel_layer.group_send(
            self.room_group_name,
            {
                'type': 'chat_message',

                'message': self.scope["user"].username + ':' + message

            }
        )

    # Receive message from room group
    async def chat_message(self, event):
        message = event['message']

        # Send message to WebSocket
        await self.send(text_data=json.dumps({
            'message': message
        }))
# chat/routing.py
from django.urls import re_path

from . import consumers

websocket_urlpatterns = [
    re_path(r'ws/chat/(?P<room_name>\w+)/$', consumers.ChatConsumer),
]# chat/tests.py
from channels.testing import ChannelsLiveServerTestCase
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.wait import WebDriverWait

from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager

class ChatTests(ChannelsLiveServerTestCase):
    serve_static = True  # emulate StaticLiveServerTestCase


    @classmethod

    def setUpClass(cls):

        super().setUpClass()

        try:

            # NOTE: Requires "chromedriver" binary to be installed in
$PATH

            cls.driver = webdriver.Chrome(ChromeDriverManager().install())

        except:

            super().tearDownClass()

            raise


    @classmethod

    def tearDownClass(cls):

        cls.driver.quit()

        super().tearDownClass()


    def
test_when_chat_message_posted_then_seen_by_everyone_in_same_room(self):

        try:

            self._enter_chat_room('room_1')


            self._open_new_window()

            self._enter_chat_room('room_1')


            self._switch_to_window(0)

            self._post_message('hello')
            WebDriverWait(self.driver, 2).until(lambda _:
                'hello' in self._chat_log_value,
                'Message was not received by window 1 from window 1')
            self._switch_to_window(1)
            WebDriverWait(self.driver, 2).until(lambda _:
                'hello' in self._chat_log_value,
                'Message was not received by window 2 from window 1')
        finally:
            self._close_all_new_windows()


    def
test_when_chat_message_posted_then_not_seen_by_anyone_in_different_room(se
lf):
        try:
            self._enter_chat_room('room_1')

            self._open_new_window()
            self._enter_chat_room('room_2')

            self._switch_to_window(0)
            self._post_message('hello')
            WebDriverWait(self.driver, 2).until(lambda _:
                'hello' in self._chat_log_value,
                'Message was not received by window 1 from window 1')

            self._switch_to_window(1)
            self._post_message('world')
            WebDriverWait(self.driver, 2).until(lambda _:
                'world' in self._chat_log_value,
                'Message was not received by window 2 from window 2')
            self.assertTrue('hello' not in self._chat_log_value,
                'Message was improperly received by window 2 from window
1')
        finally:
            self._close_all_new_windows()

    # === Utility ===

    def _enter_chat_room(self, room_name):
        self.driver.get(self.live_server_url + '/chat/')
        ActionChains(self.driver).send_keys(room_name + '\n').perform()
        WebDriverWait(self.driver, 2).until(lambda _:
            room_name in self.driver.current_url)

    def _open_new_window(self):
        self.driver.execute_script('window.open("about:blank",
"_blank");')
        self.driver.switch_to_window(self.driver.window_handles[-1])

    def _close_all_new_windows(self):
        while len(self.driver.window_handles) > 1:
            self.driver.switch_to_window(self.driver.window_handles[-1])
            self.driver.execute_script('window.close();')
        if len(self.driver.window_handles) == 1:
            self.driver.switch_to_window(self.driver.window_handles[0])
    def _switch_to_window(self, window_index):

self.driver.switch_to_window(self.driver.window_handles[window_index])


    def _post_message(self, message):

        ActionChains(self.driver).send_keys(message + '\n').perform()


    @property

    def _chat_log_value(self):

        return self.driver.find_element_by_css_selector('#chat-
log').get_property('value')


# chat/urls.py

from django.urls import path


from . import views


urlpatterns = [

    path('', views.index, name='index'),

     path('<str:room_name>/', views.room, name='room'),

]

# chat/views.py

from django.shortcuts import render


def index(request):

    return render(request, 'movie/index.html', {})


def room(request, room_name):
    return render(request, 'movie/room.html', {

        'room_name': room_name

    })
```

Project

```
from channels.routing import ProtocolTypeRouter, URLRouter

import chat.routing

from channels.auth import AuthMiddlewareStack


application = ProtocolTypeRouter({

    # Empty for now (http->django views is added by default)

    'websocket': AuthMiddlewareStack(

        URLRouter(

            chat.routing.websocket_urlpatterns

        )

    ),

})

"""

Django settings for project project.


Generated by 'django-admin startproject' using Django 2.2.6.


For more information on this file, see

https://docs.djangoproject.com/en/2.2/topics/settings/


For the full list of settings and their values, see
```

```
https://docs.djangoproject.com/en/2.2/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'c)!h$b9g=4ds4o#*+^mwxc0n&ieqbg)5j53w676ih8h%7t4z@y'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'movie.apps.MovieConfig',
    'users.apps.UsersConfig',
    'django.contrib.admin',
    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'rest_framework',

    'channels',

    'chat'


]


MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]


ROOT_URLCONF = 'project.urls'


TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [],
```

```python
        'APP_DIRS': True,

        'OPTIONS': {

            'context_processors': [

                'django.template.context_processors.debug',

                'django.template.context_processors.request',

                'django.contrib.auth.context_processors.auth',

                'django.contrib.messages.context_processors.messages',

            ],

        },

    },

]


WSGI_APPLICATION = 'project.wsgi.application'



# Database

# https://docs.djangoproject.com/en/2.2/ref/settings/#databases


DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),

        'TEST': {

            'NAME': os.path.join(BASE_DIR, 'db_test.sqlite3')

        }


    }
```

```
}



# Password validation
# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-
validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator'
,
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]



# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/


LANGUAGE_CODE = 'en-us'


TIME_ZONE = 'UTC'


USE_I18N = True


USE_L10N = True


USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/
STATIC_ROOT = os.path.join(BASE_DIR,'staticfiles')
STATIC_URL = '/static/'

LOGIN_REDIRECT_URL='movie-home'
LOGIN_URL = 'login'
ASGI_APPLICATION = "project.routing.application"
CHANNEL_LAYERS={
    "default": {
        "BACKEND": "channels.layers.InMemoryChannelLayer"
     }
from django.contrib import admin
from django.contrib.auth import views as auth_views
from django.urls import path, include

from users import views as user_views

from django.conf.urls import include


urlpatterns = [

    path('admin/', admin.site.urls),

    path('chat/', include('chat.urls'), name = 'chat'),

    path('register/', user_views.register, name = 'register'),

    path('profile/', user_views.profile, name = 'profile'),

    path('login/',
auth_views.LoginView.as_view(template_name='users/login.html'),name =
'login'),

    path('logout/',
auth_views.LogoutView.as_view(template_name='users/logout.html'),name =
'logout'),

    path('', include('movie.urls'))

]




"""

WSGI config for project project.


It exposes the WSGI callable as a module-level variable named
``application``.


For more information on this file, see

https://docs.djangoproject.com/en/2.2/howto/deployment/wsgi/

"""
```

```
import os


from django.core.wsgi import get_wsgi_application


os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'project.settings')


application = get_wsgi_application()
```

3. Bibliografie

- https://www.djangoproject.com/
- https://channels.readthedocs.io/en/latest/
- https://www.heroku.com/
- https://www.docker.com/
- https://realpython.com/python3-object-oriented-programming/