História de Usuários Streaming

Como um consumidor, gostaria de visualizar a listar de filmes cadastrados para que eu possa saber quais estão disponíveis

Como um consumidor, gostaria de pesquisar pelo nome do filme para que eu possa visualizar.

Como um consumidor, gostaria de poder visualizar os detalhes das informações de um filme escolhido para decidir se desejo assisti-lo.

Como um consumidor gostaria de poder marcar um filme como favorito para me lembrar dele no futuro.

Casos de Uso Streaming

Caso de Uso	Lista de Filmes Disponíveis
Resumo	O usuário após logar, visualizará a lista de filmes disponíveis
Ator Principal	Consumidor\Administrador
Interessados	Usuários que terão acesso a lista de filmes
Pré-condição	O usuário deve estar na home com a grid de filmes
Fluxo Principal	1- O usuário digita o conjunto de palavras-chave no campo pesquisar 2- O usuário clica no botão pesquisar filmes 3- O sistema avalia quais os filmes estão relacionados com as palavras-chave 4- O sistema retorna com a lista atualizada com base
Fluxo Alternativo	Evento 1: Ao buscar o filme e a lista retornar vazia. • A1 Caso o texto de palavras-chave seja maior do que dois, o sistema elimina a última e realiza a pesquisa novamente
Fluxo de Exceção	Evento 1: A API do sistema retornou um erro. • E1 o sistema exibe uma mensagem para o usuário informando que ocorreu uma falha técnica e pede para tentar novamente após alguns segundos
Requisitos	Navegadores: Chrome, FireFox, Safari Tempo de resposta: O tempo de retorno não deverá ultrapassar 500 ms

Caso de Uso	Pesquisar por palavra chave
Resumo	O usuário irá pesquisar filmes, pelo campo busca informando um conjunto de palavras. O sistema irá atualizar e exibir a lista de filmes relacionados.
Ator Principal	Consumidor
Interessados	Consumidor com objetivo de encontrar o filme desejado
Pré-condição	O usuário deve estar na home com a grid de filmes
Fluxo Principal	1- O usuário digita o conjunto de palavras-chave no campo pesquisar 2- O usuário clica no botão pesquisar filmes 3- O sistema avalia quais os filmes estão relacionados com as palavras-chave 4- O sistema retorna com a lista atualizada com base
Fluxo Alternativo	Evento 1: Ao buscar o filme e a lista retornar vazia. • A1 Caso o texto de palavras-chave seja maior do que dois, o sistema elimina a última e realiza a pesquisa novamente
Fluxo de Exceção	Evento 1: A API do sistema retornou um erro. • E1 o sistema exibe uma mensagem para o usuário informando que ocorreu uma falha técnica e pede para tentar novamente após alguns segundos
Requisitos	Navegadores: Chrome, FireFox, Safari Tempo de resposta: O tempo de retorno não deverá ultrapassar 500 ms

Caso de Uso	Visualizar Detalhes do filme escolhido
Resumo	O usuário irá clicar na imagem será direcionado para a tela de detalhes do filme.
Ator Principal	Consumidor
Interessados	Consumidor com objetivo de ler as informações do trailer
Pré-condição	O usuário deve estar na home com a grid de filmes
Fluxo Principal	1- O usuário clica na imagem do filme escolhido 2- O usuário deve visualizar a tela de detalhes como imagem descrição, nome, ano e o trailer do filme
Fluxo Alternativo	Clicar no nome do filme para direcionar a tela de detalhes
Fluxo de Exceção	Evento 1: A API do sistema retornou um erro. • E1 o sistema exibe uma mensagem para o usuário informando que ocorreu uma falha técnica e pede para tentar novamente após alguns segundos
Requisitos	Navegadores: Chrome, FireFox, Safari Tempo de resposta: O tempo de retorno não deverá ultrapassar 500 ms

Caso de Uso	Favoritar Filmes
Resumo	O usuário poderá clicar na imagem para favoritar.
Ator Principal	Consumidor
Interessados	Consumidor com objetivo de adicionar aos favoritos os filmes escolhidos
Pré-condição	O usuário deve estar na home com a grid de filmes
Fluxo Principal	1- O Acessa a lista de filmes 2- Clica para ver o detalhe dos filmes 3- Clica na Imagem para favoritar
Fluxo Alternativo	Evento: Imagem não favorita • Clico na Imagem da tela de detalhes e nada acontece
Fluxo de Exceção	Evento 1: A API do sistema retornou um erro. • Ao clicar fui direcionada para tela de erro. • Ao clicar a tela fica carregando
Requisitos	Navegadores: Chrome, FireFox, Safari Tempo de resposta: O tempo de retorno não deverá ultrapassar 500 ms

Arquitetura do Sistema Streaming

<Controle.py>

```
def list_movies():
    """
    List all movies in the catalog.

    :return: queryset with all movies
    """
    movies = []
    for movie in Movie.objects.all():
        movies.append({
            'id': movie.id,
            'title': movie.title,
            'year': movie.year
        })
    return movies



def is_movie_favourite(movie, user):
    """
    List all films marked as favourite for a given user.

    :param movie: instance of Movie
    :param user: instance of User
    :return: True if the movie is favourite for the user
    """
```

```
    return Favourite.objects.filter(user=user, movie=movie).count() != 0


def mark_movie_as_favourite(movie, state, user):
    """
    Set or unset a movie as favourite for the given user.

    :param movie: instance of Movie
    :param state: True if should be favourite
    :param user: instance of User
    """
    if state:
        Favourite.objects.get_or_create(
            movie=movie,
            user=user
        )
    else:
        Favourite.objects.filter(
            movie=movie,
            user=user
        ).delete()
```

<Models>

```
class User(AbstractUser):
    """
    User model

    Please note that fields first_name, last_name and email are defined in the superclass (AbstractUser).
    """

    class Type(models.TextChoices):
        REGULAR = 'Regular', _('Regular')
        ADMIN = 'Admin', _('Admin')

    type = models.CharField(max_length=50, choices=Type.choices, default=Type.REGULAR)
    birth_date = models.DateField(null=True, blank=True)
    address = models.TextField(max_length=500, blank=True)


@receiver(post_save, sender=settings.AUTH_USER_MODEL)
```

```
def create_auth_token(sender, instance=None,
created=False, **kwargs):
    if created:
        Token.objects.create(user=instance)



class Movie(models.Model):
    """
    Movie model
    """
    title = models.CharField(max_length=50, blank=True)
    year = models.IntegerField()
    cover = models.URLField(blank=True)
    description = models.TextField(blank=True)
    midia = models.URLField(blank=True)



class Favourite(models.Model):
    """
    Favourite relation model
    """
    user = models.ForeignKey(User,
on_delete=models.CASCADE)
```

```
    movie = models.ForeignKey(Movie,
on_delete=models.CASCADE)



class MovieWatched(models.Model):
    """
    Movie watched relation model
    """
    user = models.ForeignKey(User,
on_delete=models.CASCADE)
    movie = models.ForeignKey(Movie,
on_delete=models.CASCADE)
        date = models.DateTimeField(auto_now=True)


<Views>
       ef exibir_imagem(request, pk):
    instancia = get_object_or_404(MinhaModel, pk=pk)
    return render(request,
'template_para_exibir_imagem.html', {'instancia':
instancia})



class UserViewSet(viewsets.ModelViewSet):
```

```
    """
    User viewset.
    """
    queryset = get_user_model().objects.all()
    serializer_class = UserSerializer
    permission_classes = [permissions.IsAuthenticated]


class MovieViewSet(viewsets.ModelViewSet):
    """
    Movie viewset.
    """
    queryset = Movie.objects.all()
    serializer_class = MovieSerializer
    permission_classes = [permissions.IsAuthenticated]


class FavouriteViewSet(viewsets.ModelViewSet):
    """
    Favourite viewset.
    """
    queryset = Favourite.objects.all()
    serializer_class = FavouriteSerializer
```

```
    permission_classes = [permissions.IsAuthenticated]


class MovieWatchedViewSet(viewsets.ModelViewSet):
    """
    MovieWatched viewset.
    """
    queryset = MovieWatched.objects.all()
    serializer_class = MovieWatchedSerializer
    permission_classes = [permissions.IsAuthenticated]


class MovieView(views.APIView):
    """
    Movies view
    """
    permission_classes = [permissions.IsAuthenticated]

    def get(self, request):
        movies = list_movies()
        return Response(movies)
```

```
class FavouriteView(views.APIView):
    """
    Favourite view
    """
    permission_classes = [permissions.IsAuthenticated]

    def get(self, request, *args, **kwargs):
        """
        Check if a given movie is favourite
        """
        # Get the movie
        if 'id' not in request.GET:
            raise APIException('The `id` must be informed')
        movie_id = request.GET['id']
        movie = Movie.objects.get(id=movie_id)

        # Check if is favourite
        is_favourite = is_movie_favourite(movie,
request.user)

        # Return the response data
        response_data = {
            'is_favourite': is_favourite
```

```
        }
        return Response(response_data)

    def post(self, request, *args, **kwargs):
        """
        Mark a movie as favourite (or not).
        """
        # Get the movie
        if 'id' not in request.POST:
            raise APIException('The `id` must be informed')
        movie_id = request.POST['id']
        movie = Movie.objects.get(id=movie_id)

        # Get the desired state
        if 'state' not in request.POST:
            raise APIException('The `state` must be informed')
        state = request.POST['state'] == 'true'

        # Mark as favourite
        mark_movie_as_favourite(movie, state,
request.user)

        # Return the response
```

```
return Response({})
```