

**Visoka škola strukovnih studija za informacione i
komunikacione tehnologije**

DOKUMENTACIJA ZA SAJT



Predmet: WEB Programiranje PHP2

Student: Nikola Kalinčević

Broj Indeksa: 40/16

Sadržaj

1.Opis funkcionalnosti	3
2.Slike stranica	6
2.1 Inicijalna stranica	6
2.2 Login stranica	7
2.3 Kontakt stranica	8
2.4 Stranica sa svim filmovima	9
2.5 Stranica sa vestima	11
2.6 Pretraga filmova	12
2.7 Admin Panel	13
3.Dizajn baze podataka	14
4.MVC Organizacija	16
4.1 Kontroleri	16

4.2 Admin strana-Kontroleri	17
4.3 Modeli	18
4.4 Admin strana- Modeli	18
4.5 View	19
5. JavaScript kodovi.....	26
5.1 script.js.....	26
5.2 adminPanel.js.....	60
6. PHP kodovi (Laravel)	78
6.1 routes/web.php	78
6.2 Kontroleri	79
6.3 Modeli	171
6.4 Klasa za slanje mail-a	214
6.5 Middleware	220
6.6 Request-ovi	224

1.Opis funkcionalnosti

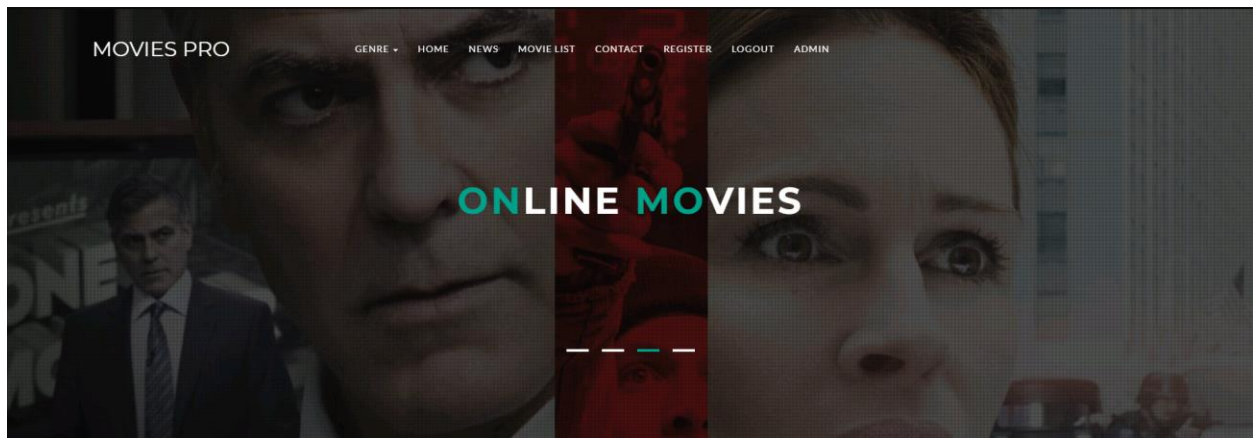
Aplikacija je realizovana kao sajt za online pregled filmova. Pri dolasku na sajt, korisniku se nudi mogućnost da se registruje, a potom uloguje da bi mogao da vidi deo sajta namenjen ulogovanim korisnicima. Takođe, registrovani korisnik može na kontakt stranici poslati email administratoru za slučaj nekih primedbi i sugestija vezanih za sam rad aplikacije. Ulogovani korisnik može da pregleda filmove koji su paginirani na "home" stranici, da izabere filmove iz određenog žanra kada klikne na neki žanr u dropdown meniju. Kada se klikne na neki film, na novoj stranici se prikazuju dodatne informacije o tom filmu, između ostalog može se pogledati trailer tog filma i videti IMDB ocena. Na posebnoj stranici se prikazuju paginirane vesti i klikom na neku vest se prikazuje nova stranica sa detaljnijim opisom te vesti. Na istoj toj stranici ulogovani korisnik može komentarisati tu vest. Komentarisanje vesti je realizovano putem ajax-a.

Na stranici gde su izlistani svi filmovi u formi tabele, realizovana je funkcionalnost pretrage koja radi sinhronizovano sa paginacijom. Ova funkcionalnost realizovana je takođe ajax-om. Filmovi se mogu pretraživati po nazivu- rezultat pretrage se paginira u zavisnosti od toga koliko postoji zapisa u bazi podataka za unetu ključnu reč. Korisnik sa administratorskim privilegijama ima posebnu stranicu kojoj samo on može pristupiti(admin panel). Preko admin panela administrator

upravlja celokupnom aplikacijom, tj. Vršiti Insert, Update i Delete nad dinamičkim sadržajem. Login i Registracija su takođe urađeni preko ajax-a. U aplikaciji postoje 2 napravljena middleware-a koji sprečavaju nedozvoljen pristup aplikaciji. Jedan middleware proverava **ip adresu** sa koje se pokušava ulazak u aplikaciju, i ta ip adresa se upisuje u bazu podataka. Drugi middleware proverava da li sajtu pristupa korisnik koji ima sesiju, tj. Onaj koji je prošao proces registracije i logovanja, a ako ne ispunjava date uslove, biva redirektovan na register stranicu sa porukom da pokušava da pristupi neautorizovano. Pri radu sa bazom podataka (upitima), svaka greška koja se desi upisuje se u bazu podataka u tabelu za greške. Svaka ključna aktivnost korisnika se takođe beleži u bazi podataka - logovanje, odjava, komentarisanje vesti, pretraživanje filmova itd. Ova statistika se izlistava na posebnoj stranici na admin panelu, pri čemu je te aktivnosti moguće sortirati po datumu (realizovano ajax-om). Sve rute koje su bitne za zaštitu aplikacije su zaštićene pomenutim middleware-ima. U footer-u sajta se nalaze linkovi do **autorske stranice** i **dokumentacije** sajta.

2.Slike stranica

2.1 Inicijalna stranica



Registration Form

First Name

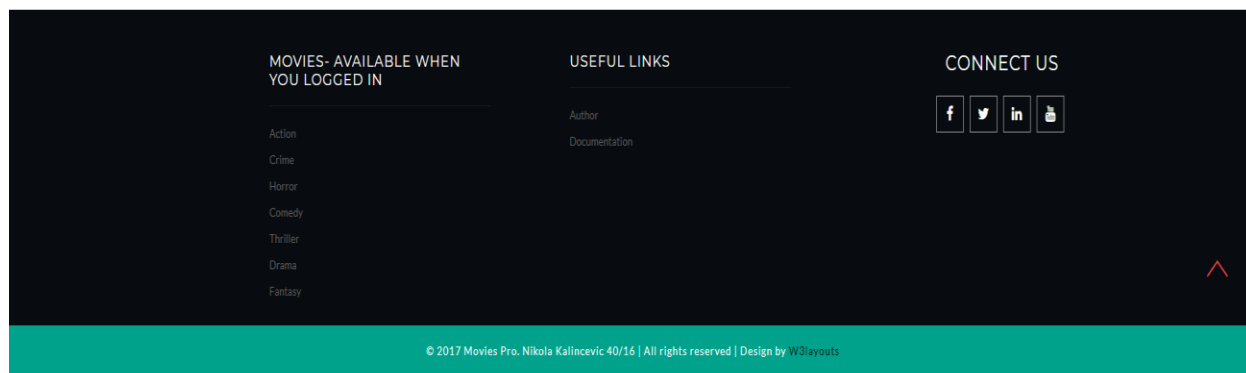
Last Name

Email

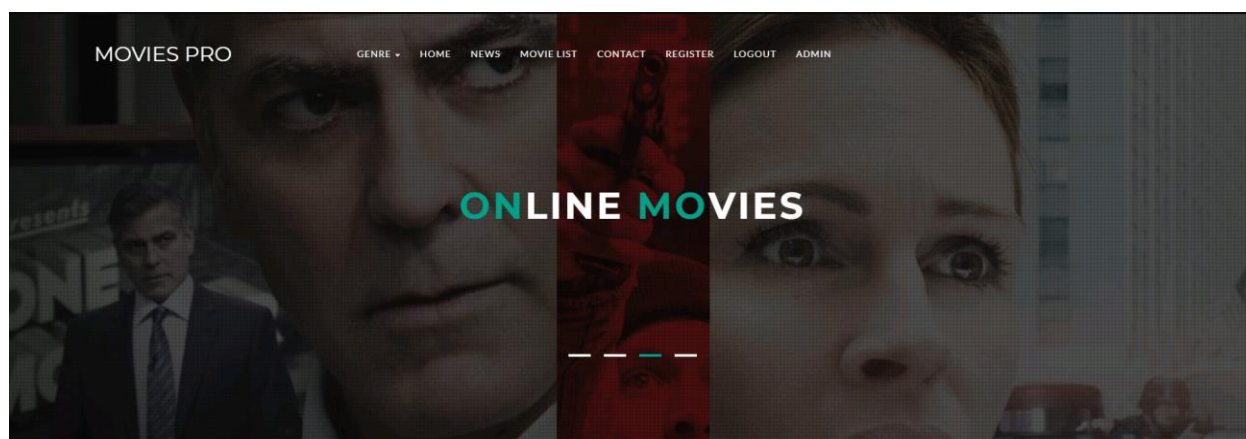
Username

Password





2.2 Login stranica

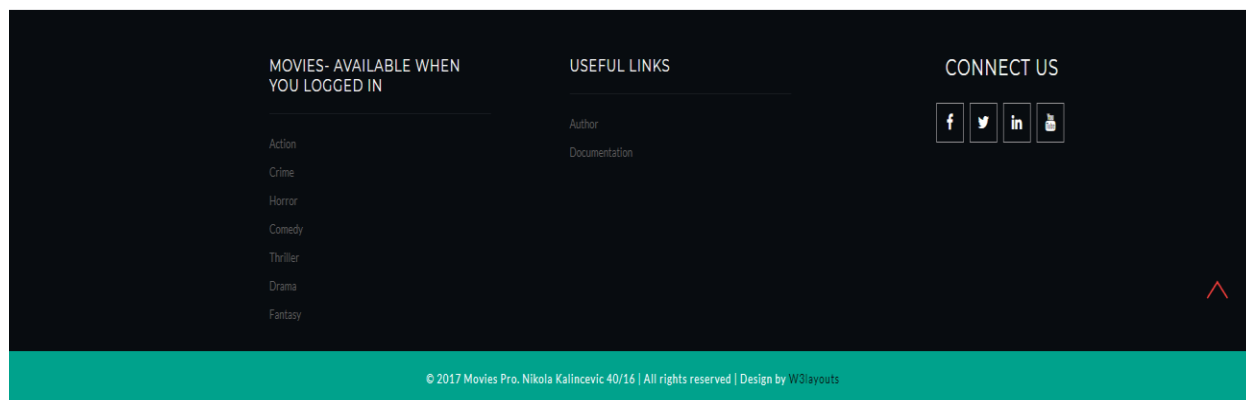


Login Form

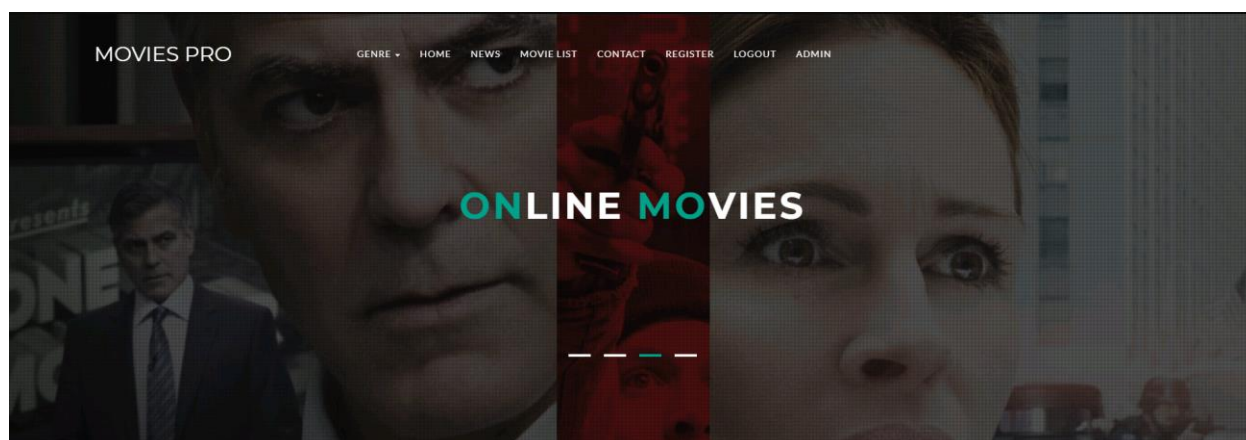
Email

Password

Login



2.3 Kontakt stranica



Contact our Administrator

Subject

You must type a subject!

Message

You must type a comment!

Send message



MOVIES- AVAILABLE WHEN YOU LOGGED IN

Action
Crime
Horror
Comedy
Thriller
Drama
Fantasy

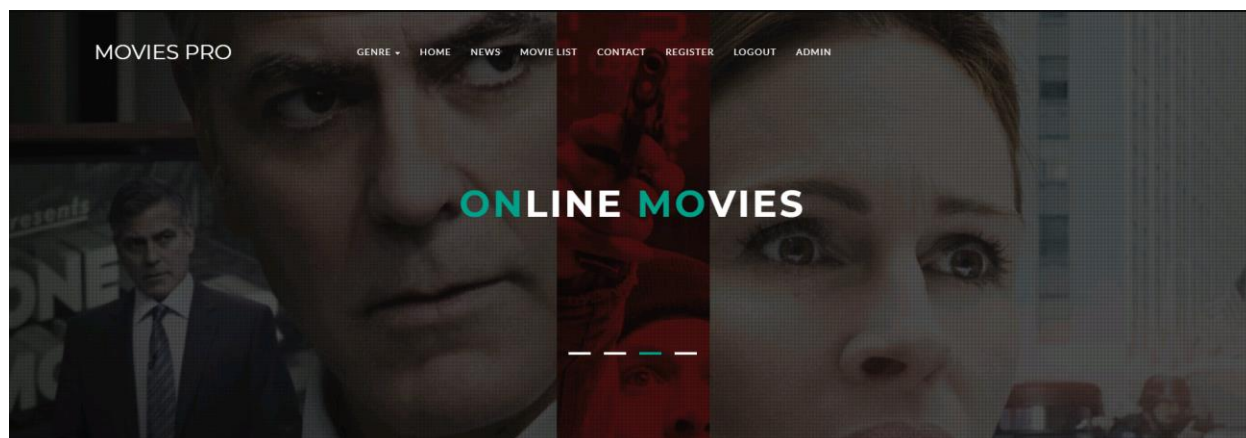
USEFUL LINKS

Author
Documentation

CONNECT US



2.4 Stranica sa svim filmovima



Movie: The Equalizer
IMDB Rating: ★7.2 /10

Movie: Mad Max
IMDB Rating: ★8.1 /10

Movie: Black Panther
IMDB Rating: ★7.3 /10

Movie: The Avengers
IMDB Rating: ★8.1 /10

Movie: Avatar
IMDB Rating: ★7.8 /10

Movie: Driver
IMDB Rating: ★7.6 /10

1 2 3 4 5 6

Lightshot
Screenshot is saved to Screenshot_3.png. Click here to open in the folder.

MOVIES- AVAILABLE WHEN YOU LOGGED IN

- Action
- Crime
- Horror
- Comedy
- Thriller
- Drama
- Fantasy

USEFUL LINKS

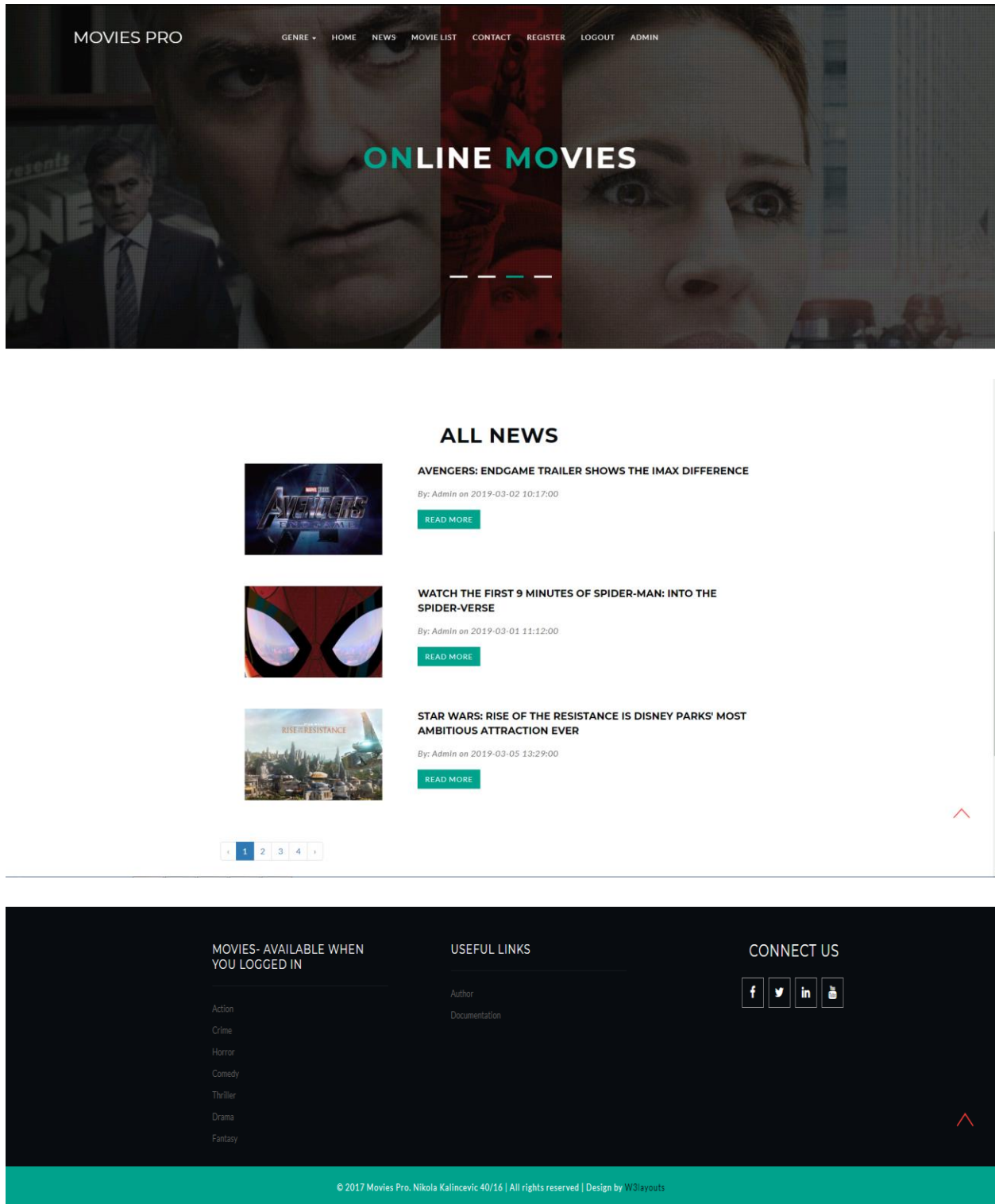
- Author
- Documentation

CONNECT US

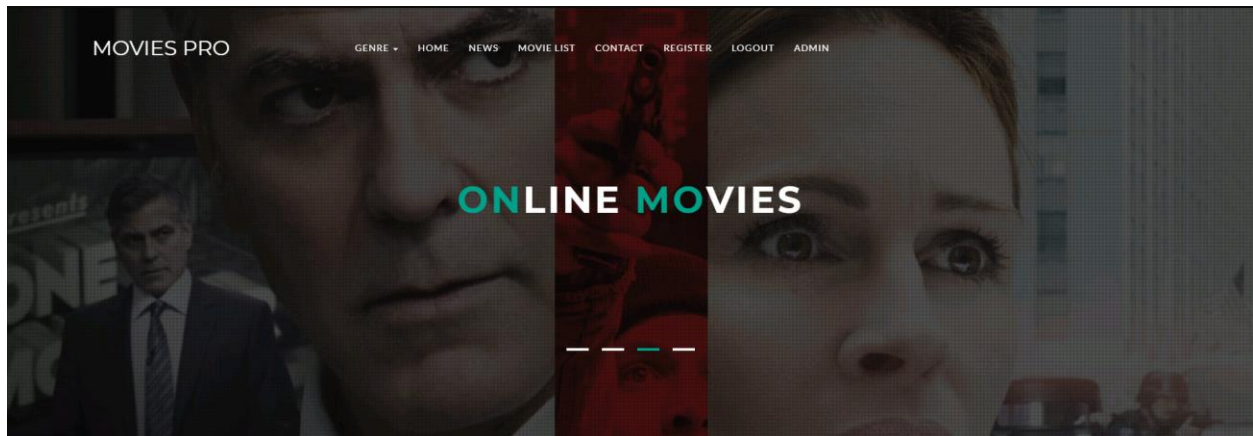
- f
- twitter
- in
- youtube

© 2017 Movies Pro. Nikola Kalincevic 40/16 | All rights reserved | Design by W3layouts

2.5 Stranica sa vestima







2.6 Pretraga filmova



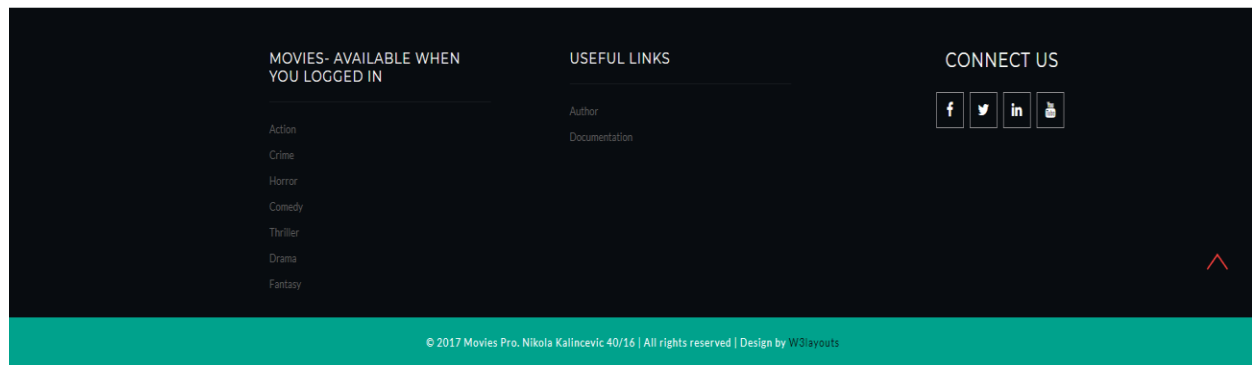
Search by name



MOVIE LIST

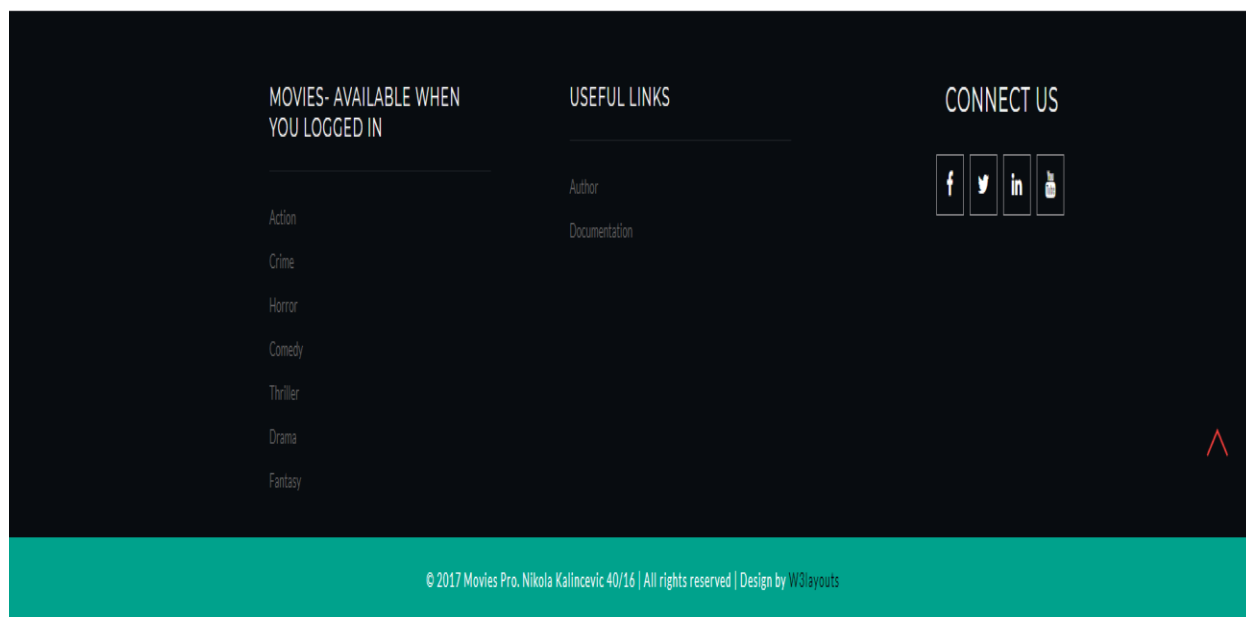
NO.	MOVIE NAME	PICTURE	LENGTH	GENRE	STARS	RATING
1	The Equalizer		132 Minutes	Action	Denzel Washington, Marton Csokas, Chloë Grace Moretz	★7.2
2	Mad Max		120 Minutes	Action	Tom Hardy, Charlize Theron, Nicholas Hoult	★8.1
3	Black Panther		134 Minutes	Action	Chadwick Boseman, Michael B. Jordan, Lupita Nyong'o	★7.3
4	The Avengers		143 Minutes	Action	Robert Downey Jr., Chris Evans, Scarlett Johansson	★8.1



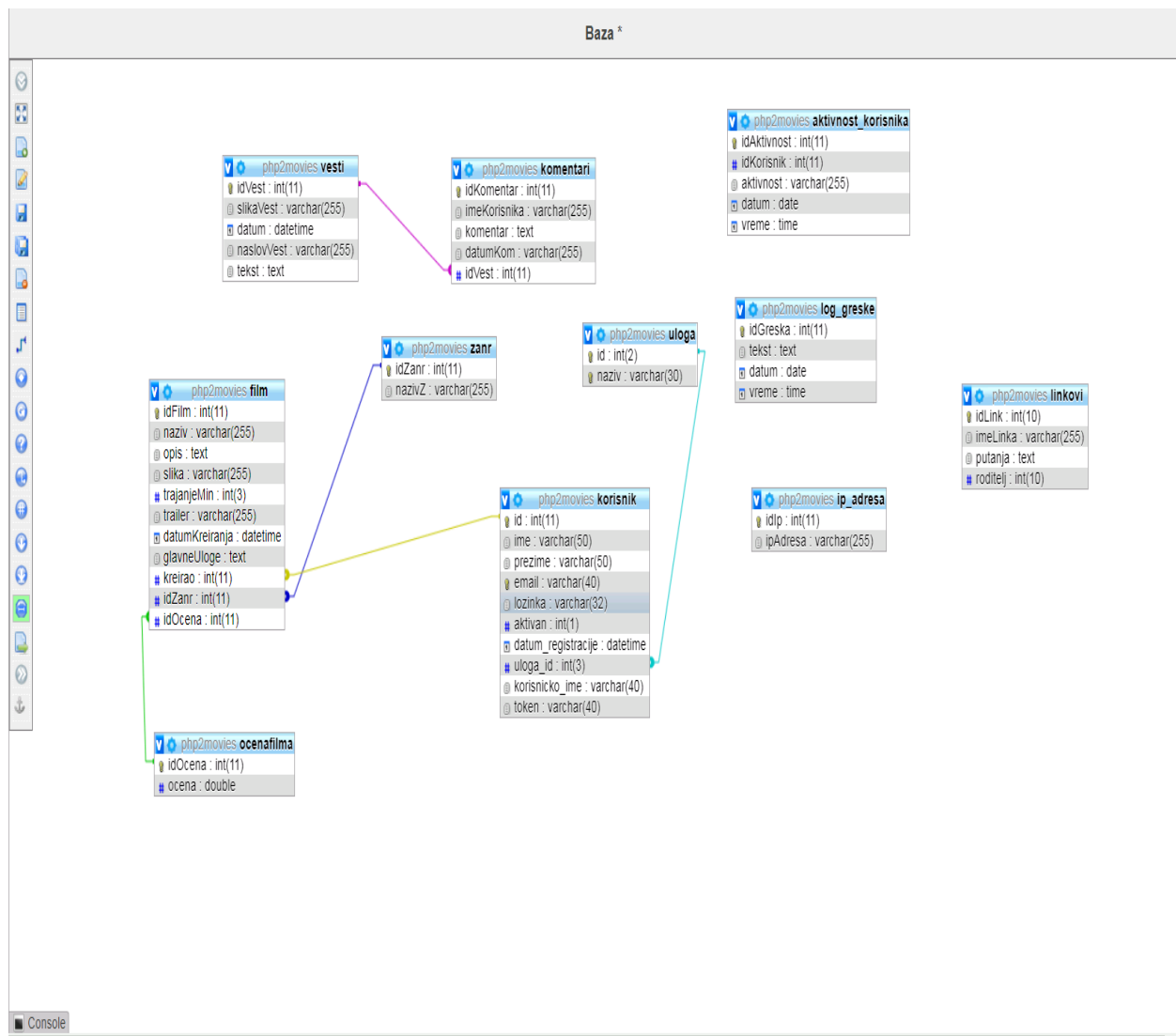


2.7 Admin Panel



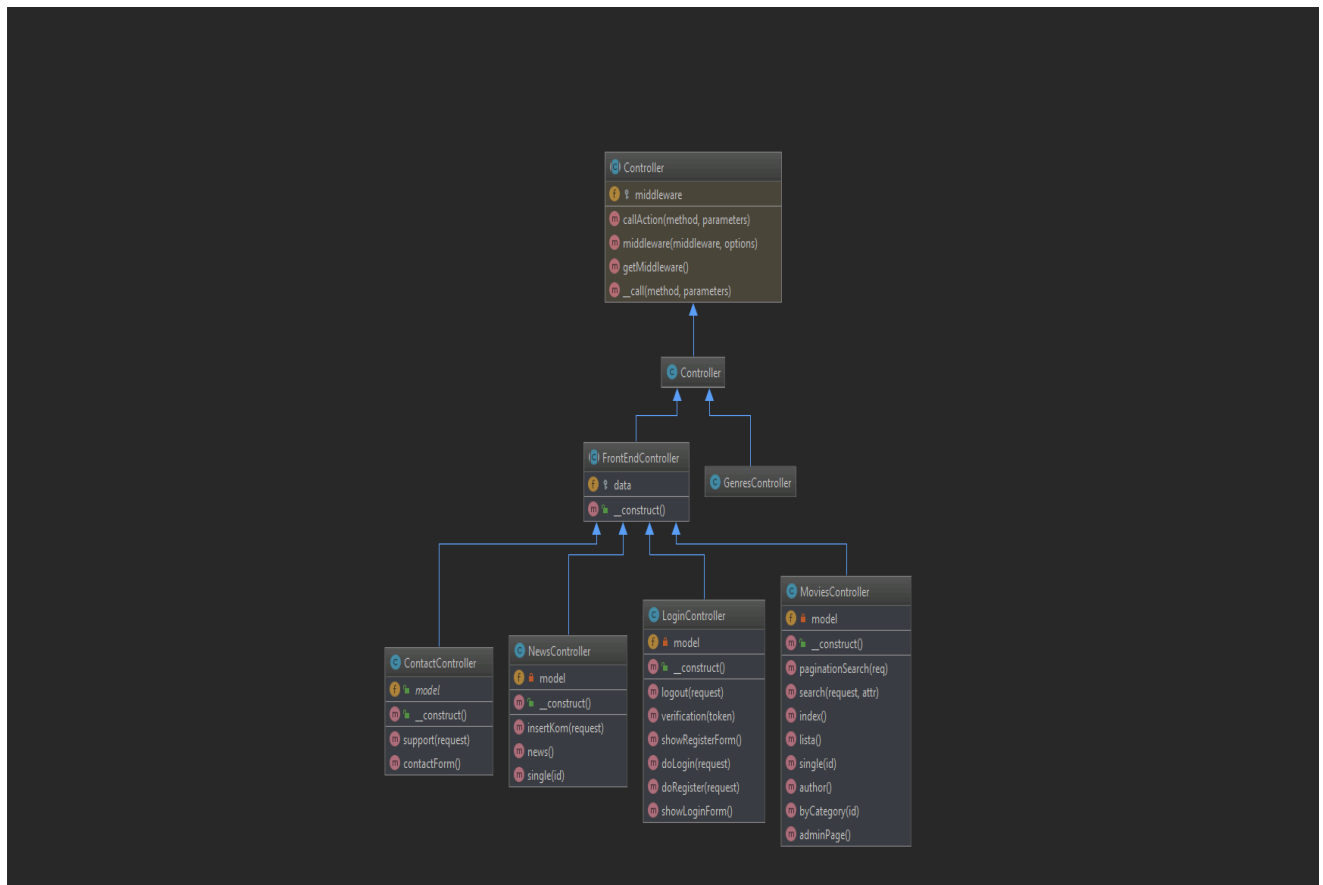


3.Dizajn baze podataka

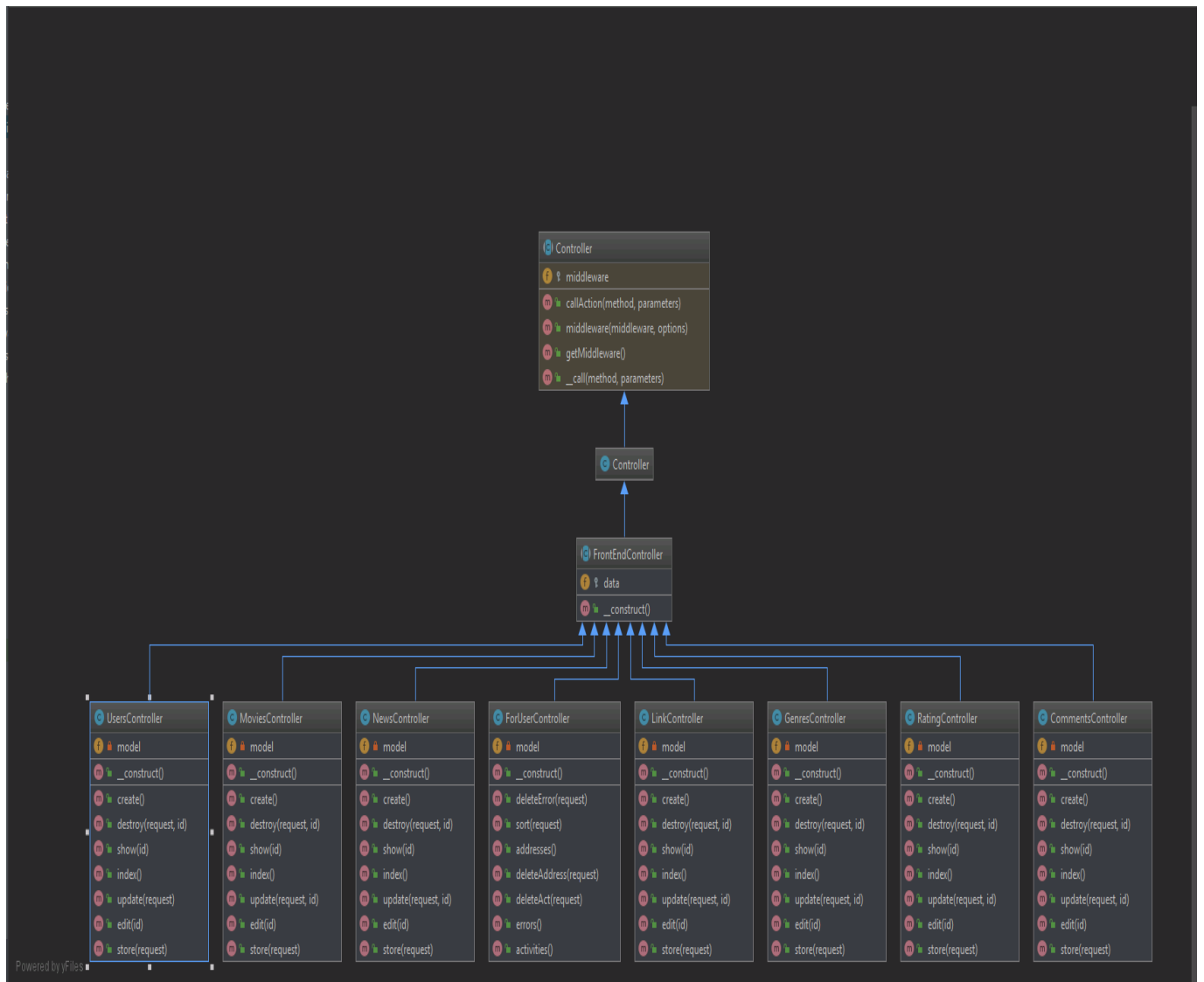


4.MVC Organizacija

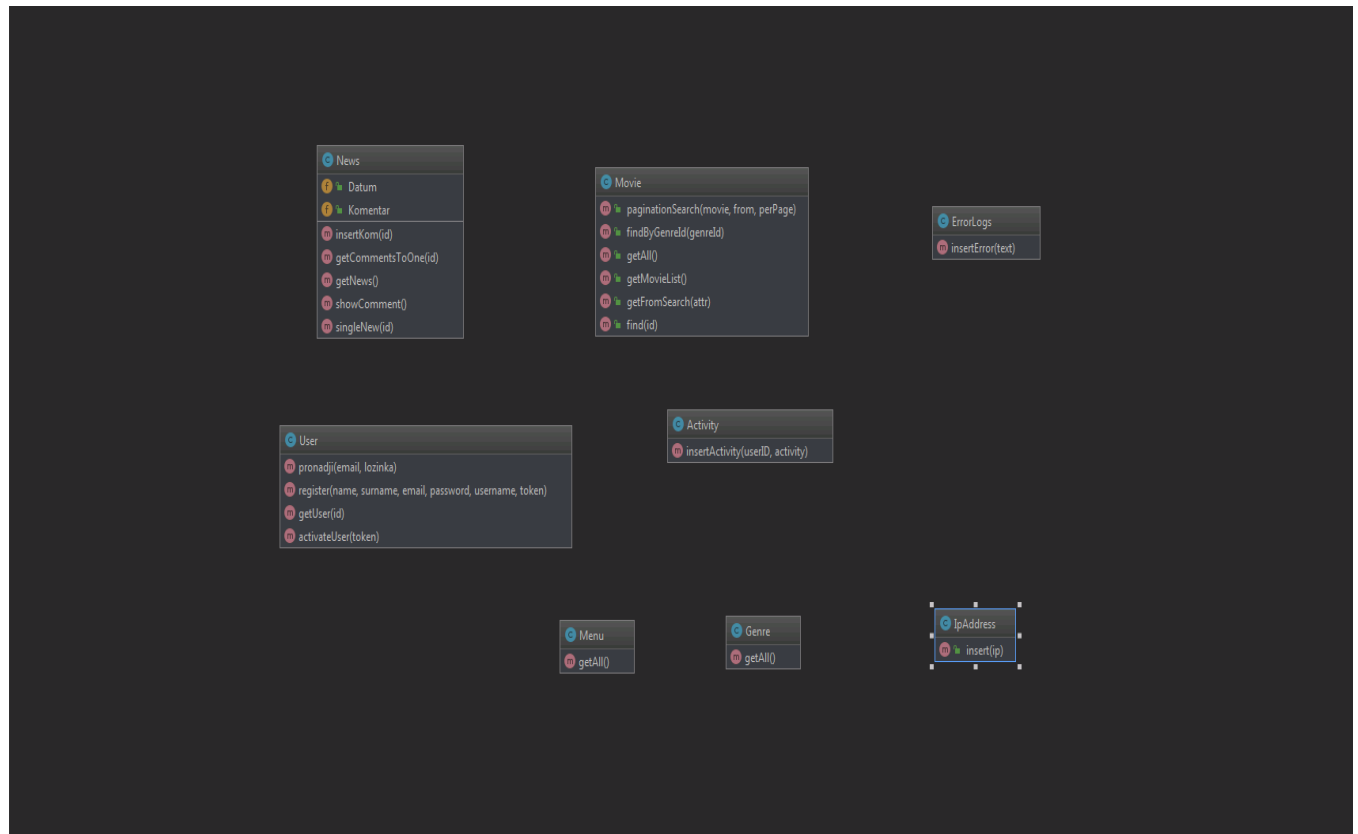
4.1 Kontroleri



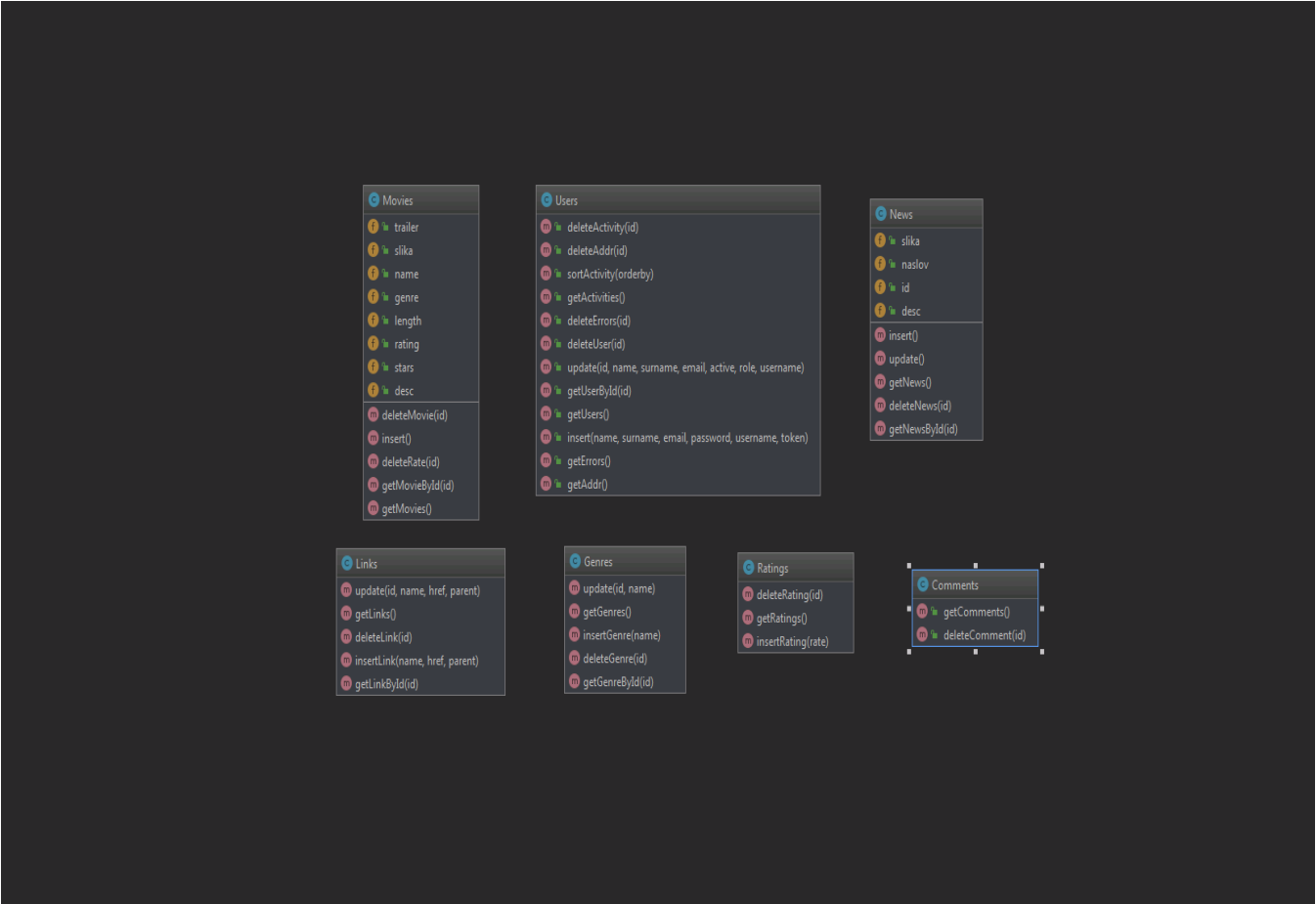
4.2 Admin strana-Kontroleri



4.3 Modeli



4.4 Admin strana- Modeli



4.5 View

Views
allnews.blade.php

author.blade.php
contact.blade.php
layout.blade.php
login.blade.php
movie-list.blade.php
movies.blade.php
register.blade.php
single.blade.php
single-new.blade.php

Views/common
head.blade.php
nav.blade.php
footer.blade.php

Views/components
login.blade.php
register.blade.php
movie.blade.php
single-movie.blade.php
single-news.blade.php

Views/admin/comments
index.blade.php

Views/admin/common
head.blade.php
nav.blade.php
footer.blade.php

Views/admin/genres
Index.blade.php

Edit.blade.php

Views/admin/links

index.blade.php

edit.blade.php

Views/admin/movies

Index.blade.php

Views/admin/news
index.blade.php
edit.blade.php

Views/admin/ratings
index.blade.php

Views/admin/users
index.blade.php

edit.blade.php

Views/admin

addresses.blade.php

errors.blade.php

activities.blade.php

admin.blade.php

5. JavaScript kodovi

5.1 script.js

```
//alert();
```

```
function contact() {
```

```
    var subject = document.getElementById("subject");
```

```
    var comment = document.getElementById("comment");
```

```
    var errors = [];
```

```
    if(subject.value == ""){
```

```
        subject.style.borderRadius = "5px";
```

```
        document.getElementById('errorMail').innerHTML = "You  
must type a subject!";
```

```
        errors.push("Please type a subject");
```

```

        document.getElementById("subject").style.border = "2px
solid red";

    }

    else {

        document.getElementById('errorMail').innerHTML = "";

        comment.style.borderRadius = "5px";

        document.getElementById("subject").style.border = "2px
solid #00ff00";

    }

    if(comment.value == ""){

        comment.style.borderRadius = "5px";

        document.getElementById('errorC').innerHTML = "You must
type a comment!";

        errors.push("Please type a comment");

        document.getElementById("comment").style.border = "2px
solid red";

    }

    else {

```

```

document.getElementById('errorC').innerHTML = "";

comment.style.borderRadius = "5px";

document.getElementById("comment").style.border = "2px
solid #00ff00";

}

if (errors.length > 0) {

    var ispis = "<ol class='lista'>";

    ispis += "<h3 class='pomeri'>Errors:</h3>";

    for (var i = 0; i < errors.length; i++) {

        ispis += "<li>" + errors[i] + "</li>";

    }

    ispis += "</ol>";

    document.querySelector("#poruka2").innerHTML = ispis;

}

else{

```

```
$.ajaxSetup({  
    headers: {  
        'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')  
    }  
});
```

```
$.ajax({  
    url: "/contact",  
    type: "post",  
    //dataType: "json",  
    data: {  
        subject: $('#subject').val(),  
        email: $('#email').val(),  
        ime: $('#ime').val(),  
        comment: $('#comment').val(),  
        userID: $('#userID').val()  
    },
```

```

success:function (data) {

    //window.location.href=podaci['odg'];

    //$("#poruka2").html("<h3 class='nekijos'>You sent
an email to the admin!</h3>");

    $('#pop').html("You sent an email to the admin!");
    $("#pop").css('display','block');
},

error:function (xhr,status,errMsg) {

    console.log(xhr.responseText);

    //$("#poruka2").html("<h3>Some errors
occured!</h3>");

    //alert('Greska');

}

});

```

```
}  
}
```

```
//klijentska validacija  
function provera()  
{
```

```
var ime,relme,prezime,rePrezime,usrname,reUser,email,
reEmail,sifra,reSira,nizGreske;
```

```
ime= document.getElementById("ime");
```

```
usrname= document.getElementById("username");
```

```
prezime= document.getElementById("prezime");
```

```
email = document.getElementById("email");
```

```
relme= /^[A-Z][a-z]{2,9}$/;
```

```
rePrezime= /^[A-Z][a-z]{2,14}$/;
```

```
reEmail = /^[a-z][A-z\.\-0-9]{4,35}\@[a-z]{2,5}(\.[a-
z]{2,5}){1,3}$/;
```

```
sifra=document.getElementById("password");
```

```
reSifra=/^([A-Za-z\d]){8,}$/;
```

```
nizGreske=[];
```

```
reUser=/^[a-z0-9\_]{4,15}$/;
```

```
if(!relme.test(ime.value))
```

```
{
```

```
    ime.style.border="2px solid red";
```



```

ime.style.borderRadius = "5px";
nizGreske.push("Name is not ok");

document.getElementById('error1').innerHTML = "Name is
not correct! Try again!";
}
else
{
ime.style.border = "2px solid #00ff00";
ime.style.borderRadius = "5px";
document.getElementById('error1').innerHTML = "";
}

if(!rePrezime.test(prezime.value))
{
prezime.style.border= "2px solid red";
prezime.style.borderRadius = "5px";
nizGreske.push("Surname is not ok");

document.getElementById('error2').innerHTML = "Surname
is not correct! Try again!!";

```

```
}  
else  
{  
    prezime.style.border = "2px solid #00ff00";  
    prezime.style.borderRadius = "5px";  
    document.getElementById('error2').innerHTML = "";  
}
```

```
if(!reUser.test(usrname.value))  
{  
    usrname.style.border= "2px solid red";  
    usrname.style.borderRadius = "5px";  
    nizGreske.push("Username is not ok");  
    document.getElementById('errorUser').innerHTML =  
"Username is not correct! Try again!";  
}  
else
```

```
{  
    username.style.border = "2px solid #00ff00";  
    username.style.borderRadius = "5px";  
    document.getElementById('errorUser').innerHTML = "";  
}
```

```
if(!reEmail.test(email.value))  
{  
    email.style.border = "2px solid red";  
    email.style.borderRadius = "5px";  
    nizGreske.push("Email is not ok");  
    document.getElementById('errorMail').innerHTML = "Email  
is not correct! Try again!";  
}  
else
```

```
{  
    email.style.border = "2px solid #00ff00";  
    email.style.borderRadius = "5px";  
    document.getElementById('erorMail').innerHTML = "";  
}
```

```
if(!reSifra.test(sifra.value))  
{  
    sifra.style.border= "2px solid red";  
    sifra.style.borderRadius = "5px";  
    nizGreske.push("Password is not ok");  
    document.getElementById('erorS').innerHTML = "Password  
is not correct! Try again!";  
}  
else  
{  
    sifra.style.border = "2px solid #00ff00";  
    sifra.style.borderRadius = "5px";
```

```
document.getElementById('errorS').innerHTML = "";  
}
```

//slanje podataka serveru, nakon sto su dobro popunjena polja
forme

```
if(nizGreske.length>0){  
  
    var ispis="<ol class='lista'>";  
    ispis += "<h3 class='pomeri'>Errors:</h3>";  
    for(var i=0;i<nizGreske.length;i++){  
        ispis += "<li>" + nizGreske[i] + "</li>";  
    }  
  
    ispis += "</ol>";
```

```
document.querySelector("#poruka1").innerHTML=ispis;
```

```

}

else{

    $.ajaxSetup({
        headers: {
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        }
    });

    $.ajax({
        url : "/register",
        type : "post",
        data : {
            firstName : $("#ime").val(),
            lastName : $("#prezime").val(),
            email : $("#email").val(),
            username : $("#username").val(),
            sifra: $("#password").val(),

```

```

    },
    success : function(data, xhr) {
        //alert('You have successfully registered');

        $("#pop").html("Successfull registration!");
        $("#pop").css('display','block');

        //$("#poruka1").html("<h1 class='naslov'>
Successfull Registration!</h1>");
    },
    error : function(xhr, status, error) {

        //console.log(xhr.responseText);
        var poruka = "Some errors occured";
        switch(xhr.code) {

            case 250:
                poruka = "Register not success!";

```

```
        break;
    }
    $("#poruka1").html(poruka);

}

});
//}
//var formData = getFormData();
//callAjax(formData);
}

}
```



```
//provera podataka za login
```

```
function loginProvera() {
```

```
    var email, sifra, regMail, regSifra, nizGreske;
```

```
    email = document.getElementById("email");
```

```
    sifra = document.getElementById("password");
```

```
    regMail = /^[a-z][A-z\.\-0-9]{4,35}\@[a-z]{2,5}(\.[a-z]{2,5}){1,3}$/;
```

```
    regSifra = /^[A-Za-z\d]{8,}$/;
```

```
    nizGreske = [];
```

```
if (!regMail.test(email.value)) {  
    email.style.border = "2px solid red";  
    email.style.borderRadius = "5px";  
    nizGreske.push("Email is not ok");  
    document.getElementById('errorMail').innerHTML = "Email  
is not correct! Try again!";  
} else {  
    email.style.border = "2px solid #00ff00";  
    email.style.borderRadius = "5px";  
    document.getElementById('errorMail').innerHTML = "";  
}
```

```
if (!regSifra.test(sifra.value)) {  
    sifra.style.border = "2px solid red";  
    sifra.style.borderRadius = "5px";  
    nizGreske.push("Password is not ok");
```

```
document.getElementById('errorS').innerHTML = "Password  
is not correct! Try again!";
```

```
} else {  
    sifra.style.border = "2px solid #00ff00";  
    sifra.style.borderRadius = "5px";  
    document.getElementById('errorS').innerHTML = "";  
}
```

```
if (nizGreske.length > 0) {
```

```
    var ispis = "<ol class='lista'>";  
    ispis += "<h3 class='pomeri'>Errors:</h3>";  
    for (var i = 0; i < nizGreske.length; i++) {  
        ispis += "<li>" + nizGreske[i] + "</li>";  
    }
```

```
    ispis += "</ol>";
```

```
document.querySelector("#poruka123").innerHTML = ispis;
```

```
} else {  
  
    $.ajaxSetup({  
        headers: {  
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')  
        }  
    });  
}
```

```
// function callAjax(obj) {
```

```
$.ajax({  
    url: "/login",  
    type: "post",  
    dataType: "json",  
    data: {  
        email: $("#email").val(),  
        password: $("#password").val(),  
        //send: true  
    }  
});  
}
```

```

    },

    success: function (data) {

        if(data.code === 200)

            window.location.href =
'http://localhost:8000/movies/';

            if(data.code === 422) {

                $("#pop").html("Sorry, you must first register!");

                $("#pop").css('display','block');

            }

        },

        error: function (xhr) {

            if(xhr.status === 422){

                $("#poruka123").html("<h3 class='naslov'>Please fill
all data!</h3>");

            }

```

```
}
```

```
});
```

```
}
```

```
}
```

```
//dodavanje komentara
```

```
$('#dugme225').click(function (e) {
```

```
    e.preventDefault();
```

```
    var datum=document.getElementById("jojdat");
```

```
var comment = document.getElementById("comment1");  
var errors=[];  
  
if(comment.value == ""){  
    comment.style.borderRadius = "5px";  
    document.getElementById('erorC1').innerHTML = "You  
must type a comment!";  
    errors.push("Please type a comment");  
    document.getElementById("comment1").style.border =  
"2px solid red";  
}  
else {  
    document.getElementById('erorC1').innerHTML = "";  
    comment.style.borderRadius = "5px";  
    document.getElementById("comment1").style.border =  
"2px solid #00ff00";  
}
```

```
if(datum.value == ""){  
    datum.style.border = "2px solid red";  
    datum.style.borderRadius = "5px";  
    errors.push("Choose a date");  
    document.getElementById('erordatum1').innerHTML =  
"You must choose a date!";  
}  
else{  
  
    datum.style.border = "2px solid #00ff00";  
    datum.style.borderRadius = "5px";  
    document.getElementById('erordatum1').innerHTML = "";  
  
}
```



```

if(errors.length>0){

    var ispis="<ol class='lista'>";
    ispis += "<h3 class='pomeri'>Errors:</h3>";
    for(var i=0;i<errors.length;i++){
        ispis += "<li>" + errors[i] + "</li>";
    }

    ispis += "</ol>";

    document.querySelector("#poruka5").innerHTML=ispis;
}

else{

    var id = $(this).attr("data-id");

```

```
$.ajaxSetup({  
    headers: {  
        'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')  
    }  
});
```

```
$.ajax({  
    url:"/comment",  
    method:"post",  
    dataType:"json",  
    data:{  
        //send:true,  
        datum:$("#jojdat").val(),  
        comment:$("#comment1").val(),  
        id:id,  
    },  
    success:function (data) {  
        //console.log(data);  
    }  
});
```

```
//alert(data.ime);
```

```
var ispis = "<div class='container'><h4 class='pozadi'><i  
class='glyphicon glyphicon-user'></i> " + data.ime + " | " +  
data.datum + " | " + "<i class='glyphicon glyphicon-  
comment'></i></h4>"  
  
+ "<div class='komentic'>" +  
  
"<textarea id='oblast' rows='3' cols='76'  
disabled='true'>" + data.komentar + "</textarea>" +  
  
"</div></div>";
```

```
$("#poruka5").html("<h2 id='naslov'>You commented  
on this news!</h2>");
```

```
$("#prikazi").append(ispis);  
$("#jojdat").attr("disabled",true);  
$("#comment1").attr("disabled",true);  
$("#dugme225").attr("disabled",true);  
$("#jojdat").css("border","1px solid black");
```

```
$("#comment1").css("border","1px solid black");

},
error:function (xhr,status,errMes) {
    //alert("Nije stigao");
    console.log(xhr.responseText);
}

});

}
```

```
});
```

```
//pretraga preko ajaxa
```

```
$("#dugmesa").click(function () {  
    var polje=$("#search").val();
```

```
    $.ajaxSetup({  
        headers: {  
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')  
        }  
    });
```

```
    $.ajax({  
        url:"/movie-list/search=" + polje,  
        method:"get",
```

```

    dataType:"json",
    data:{
        polje:polje
    },
    success:function (data) {

        //Pagination

        let rows = data.length;

        let numPerPage = 3;

        let numOfPages =Math.ceil( rows / numPerPage);

        //console.log(rows);

        let pagination = ' <ul class="pagination pagination-md">';

        for(let i = 0 ; i < numOfPages ; i++) {

            pagination += ` <li class="paginationLink" data-
page="${i+1}"

            data-movie="${polje}"

```

```

        data-perpage="{numPerPage}">
        <a class="page-link" >${i+1}</a></li>`
    }

    pagination += '</ul>';

    $('.paginationContainer').html(pagination);

```

```

    var ispis='<table id="table-
breakpoint"><thead><tr><th>No.</th><th>Movie
Name</th><th>Picture</th><th>Length</th><th>Genre</th><t
h>Stars</th><th>Rating</th></tr></thead><tbody>';

    for(let i = 0 ; i < data.length ; i++) {

        if(i > numPerPage)

            break;

        // for(var i=0;i<data.length;i++){

            ispis += ` <tr>

                <td>${data[i].idFilm}</td>

```

```

        <td>${data[i].naziv}</td>

        <td class="w3-list-img"></td>

        <td>${data[i].trajanjeMin} Minutes</td>

        <td>${data[i].nazivZ}</td>

        <td>${data[i].glavneUloge}</td>

        <td>${data[i].ocena}</td>

    </tr>

    ,

}

ispis += '</tbody></table>';

$('.agile-news-table').html(ispis);

},

error:function (xhr,status,errMes) {

```



```
        //alert("Nije stigao");  
        console.log(xhr.responseText);  
    }  
});
```

```
});
```

```
//paginacija linkovi
```

```
$(document).on('click','<div>.paginationLink', function() {  
    let page = $(this).data('page');
```

```

let movie = $(this).data('movie');

let numPerPage = $(this).data('perpage');

$.ajaxSetup({
    headers: {
        'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
    }
});

$.ajax({
    url : '/paginationSearch',
    method : 'get',
    dataType : 'json',
    data : {
        page:page,
        movie:movie,
        numPerPage:numPerPage
    },
    success:function(data){
        //render pagination
    }
});

```

```

    console.log(data);

    var ispis='<table id="table-
breakpoint"><thead><tr><th>No.</th><th>Movie
Name</th><th>Picture</th><th>Length</th><th>Genre</th><t
h>Stars</th><th>Rating</th></tr></thead><tbody>';

    for(var i=0;i<data.length;i++){

        ispis += ` <tr>

                <td>${data[i].idFilm}</td>

                <td>${data[i].naziv}</td>

                <td class="w3-list-img"></td>

                <td>${data[i].trajanjeMin} Minutes</td>

                <td>${data[i].nazivZ}</td>

                <td>${data[i].glavneUloge}</td>

                <td>${data[i].ocena}</td>

            </tr>

            `

    }

```

```
ispis += '</tbody></table>';

$('.agile-news-table').html(ispis);

},
error(err){
    console.log(err.responseText);
}
})
});
```

5.2 adminPanel.js

```
//alert();
```

```
$.ajaxSetup({
    headers: {
```

```
    'X-CSRF-TOKEN': $('meta[name="csrf-  
token"]').attr('content')  
  }  
});
```

//brisanje ip adresa

```
$('.adrBrisi').click(function(){  
  var id = $(this).attr('data-id');  
  $.ajax({  
    url:'/admin/deleteIpAddress',  
    type:'POST',  
    data:{  
      id:id  
    },  
    success:function(data){  
      //console.log(data);  
      //alert(id);  
      alert("Ip address deleted succesfully!");  
    }  
  });  
});
```

```

    },
    error:function(status,xhr,error){
        console.log(xhr.responseText);
    }

});

});

//brisanje gresaka

$('.errBrisi').click(function(){
    var id = $(this).attr('data-id');
    $.ajax({
        url:'/admin/deleteErrors',

```

```
type:'POST',
data:{

    id:id
},
success:function(data){
    //console.log(data);
    //alert(id);
    alert("Log error deleted succesfully!");
},
error:function(status,xhr,error){
    console.log(xhr.responseText);
}

});
```

```
});
```

```
//brisanje aktivnosti korisnika
```

```
$('#actBrisi').click(function(){  
    var id = $(this).attr('data-id');  
    $.ajax({  
        url:'/admin/deleteActivity',  
        type:'POST',  
        data:{  
            id:id  
        },  
        success:function(data){  
            //console.log(data);  
            //alert(id);  
            alert("User activity deleted succesfully!");  
        },
```



```
error:function(status,xhr,error){
    console.log(xhr.responseText);
}

});

});

//brisanje zanrova

$('.gnrBrisi').click(function(){
    var id = $(this).attr('data-id');
    $.ajax({
        url:'genres/'+id,
        type:'delete',
        data:{
```

```
        id:id
    },
    success:function(data){
        //console.log(data);
        //alert(id);
        alert("Genre deleted succesfully!");
    },
    error:function(status,xhr,error){
        console.log(xhr.responseText);
    }

});

});
```

```
//brisanje ocene za film
```

```
$('#.ratBrisi').click(function(){  
    var id = $(this).attr('data-id');  
    $.ajax({  
        url:'ratings/'+id,  
        type:'delete',  
        data:{  
            id:id  
        },  
        success:function(data){  
            //console.log(data);  
            //alert(id);  
            alert("Movie rating deleted succesfully!");  
        },  
        error:function(status,xhr,error){
```

```
        console.log(xhr.responseText);
    }

});

});

//brisanje komentara

$('.commBrisi').click(function(){
    var id = $(this).attr('data-id');
    $.ajax({
        url:'comments/'+id,
        type:'delete',
        data:{
```

```

        id:id
    },
    success:function(data){
        //console.log(data);
        //alert(id);
        alert("Comment deleted succesfully!");
    },
    error:function(status,xhr,error){
        console.log(xhr.responseText);
    }

});

});

```

```
//brisanje linkova
```

```
$('#linkBrisi').click(function(){  
    var id = $(this).attr('data-id');  
    $.ajax({  
        url:'links/'+id,  
        type:'delete',  
        data:{  
            id:id  
        },  
        success:function(data){  
            //console.log(data);  
            //alert(id);  
            alert("Navigation link deleted succesfully!");  
        },  
        error:function(status,xhr,error){
```

```
        console.log(xhr.responseText);
    }

});

});

//brisanje korisnika

$('.userBrisi').click(function(){
    var id = $(this).attr('data-id');
    $.ajax({
        url:'users/'+id,
        type:'delete',
        data:{
```

```
        id:id
    },
    success:function(data){
        //console.log(data);
        //alert(id);
        alert("User deleted succesfully!");
    },
    error:function(status,xhr,error){
        console.log(xhr.responseText);
    }

});

});
```



```
//brisanje filma
```

```
$('.filmBrisi').click(function(){  
    var id = $(this).attr('data-id');  
    $.ajax({  
        url:'movies/'+id,  
        type:'delete',  
        data:{  
            id:id  
        },  
        success:function(data){  
            //console.log(data);  
            //alert(id);  
            alert("Movie deleted succesfully!");  
        },  
        error:function(status,xhr,error){
```

```
        console.log(xhr.responseText);
    }

});

});

//brisanje vesti

$('.vestBrisi').click(function(){
    var id = $(this).attr('data-id');
    $.ajax({
        url:'news/'+id,
        type:'delete',
        data:{
```

```

        id:id
    },
    success:function(data){
        //console.log(data);
        //alert(id);
        alert("News deleted succesfully!");
    },
    error:function(status,xhr,error){
        console.log(xhr.responseText);
    }

});

});

```

```
//sortiranje aktivnosti po datumu
```

```
$('#order').click(function(){  
    var  
    izabrano=document.getElementById("orderby").selectedIndex;  
  
    var  
    orderby=document.getElementById("orderby").options[izabran  
o].value;  
  
    $.ajax({  
        method:'POST',  
        url:'/admin/sortActivity',  
        dataType:'json',  
        data:{  
            orderby:orderby  
        },  
        success:function(data){
```

```

        var ispis="<thead><tr><th scope='col'>Activity
        Id</th><th scope='col'>User Id</th><th
        scope='col'>Activity</th><th scope='col'>Date</th><th
        scope='col'>Time</th></tr></thead>";

        for(var i=0;i<data.length;i++){

            ispis+="<tr><td>" +data[i]['idAktivnost']+"</td><td>" +data[i]['id
            Korisnik']+"</td><td>" +data[i]['aktivnost']+"</td><td>" +data[i][
            'datum']+"</td><td>" + data[i]['vreme']+ "</td></tr>";

        }

        document.getElementById("table-
        breakpoint").innerHTML=ispis;

    },

    error:function(xhr,statusTxt,error){

        console.log(xhr.responseText);

    }

})

});

```

6. PHP kodovi (Laravel)

6.1 routes/web.php

```
13
14 Route::get("/", function() {
15     return redirect(route( name: "register-form"));
16 });
17
18
19
20 Route::get("/author", "MoviesController@author")->name("author");
21 Route::get("/contact", "ContactController@contactForm")->name("contact-form");
22 Route::get("/register", "LoginController@showRegisterForm")->name("register-form");
23 Route::get("/login", "LoginController@showLoginForm")->name("login-form");
24 Route::post("/register", "LoginController@doRegister")->name("do-register");
25 Route::post("/login", "LoginController@doLogin")->name("do-login");
26 Route::post("/contact", "ContactController@support")->name("contact-send");
27 Route::post("/comment", "NewsController@insertKom");
28
29
30 Route::middleware(['guardMiddleware'])->group(function () {
31     Route::get("/news", "NewsController@news")->name("news");
32     Route::get("/news/{id}", "NewsController@single")->name("single-news");
33     Route::get("/movies/", "MoviesController@index")->name("movies");
34     Route::get("/movie-list/", "MoviesController@lista");
35     Route::get("/movie/{id}", "MoviesController@single")->name("single-movie");
36     Route::get("/movie/categories/{id}", "MoviesController@byCategory")->name("movies-category");
37     Route::get("/movie-list/search={attr}", "MoviesController@search");
38     Route::get('/paginationSearch', 'MoviesController@paginationSearch');
39     Route::get('/verification/{token}', 'LoginController@verification');
40     Route::get("/logout", "LoginController@logout");
41     Route::get("/back", function () {
42         return redirect() ->route( route: "movies");
43     }) ->name("back");
44 });
```

```

46 //admin panel routes
47 Route::middleware(['checkingIp', 'guardMiddleware'])->group(function () {
48     Route::get("/admin", "MoviesController@adminPage");
49     Route::get("/admin/userActivities", "Admin\ForUserController@activities");
50     Route::get("/admin/errorLogs", "Admin\ForUserController@errors");
51     Route::get("/admin/ipAddresses", "Admin\ForUserController@addresses");
52     Route::resource("admin/users", "Admin\UsersController");
53     Route::resource("admin/movies", "Admin\MoviesController");
54     Route::resource("admin/links", "Admin\LinkController");
55     Route::resource("admin/comments", "Admin\CommentsController");
56     Route::resource("admin/ratings", "Admin\RatingController");
57     Route::resource("admin/news", "Admin\NewsController");
58     Route::resource("admin/genres", "Admin\GenresController");
59 });
60
61
62
63 Route::post("/admin/deleteIpAddress", "Admin\ForUserController@deleteAddress");
64 Route::post("/admin/deleteErrors", "Admin\ForUserController@deleteError");
65 Route::post("/admin/deleteActivity", "Admin\ForUserController@deleteAct");
66 Route::post("/admin/sortActivity", "Admin\ForUserController@sort");

```

6.2 Kontroleri

FrontendController.php

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
```

```
use App\Models\Genre;
```

```
use App\Models\Menu;
```

```
abstract class FrontEndController extends Controller
```

```
{
```

```
    protected $data;
```

```
    public function __construct()
```

```
    {
```

```
        $genreModel = new Genre();
```

```
        $this->data["genres"] = $genreModel->getAll();
```

```
        $menuModel=new Menu();
```

```
        $this->data["menus"]=$menuModel->getAll();
```

```
    }
```

```
}
```

ContactController.php


```
<?php
```

```
namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
```

```
use App\Models\Movie;
```

```
use App\Models\Genre;
```

```
use App\Mail;
```

```
use App\Models\Activity;
```

```
use App\Models\ErrorLogs;
```

```
class ContactController extends FrontEndController
```

```
{
```

```
    public function __construct()
```

```
    {
```

```
parent::__construct();  
$this->model = new Mail();  
}
```

```
public function contactForm(){  
    return view("contact",$this->data);  
}
```

```
public function support(Request $request){  
    $email = $request->input('email');  
    $name= $request->input('ime');  
    $subject = $request->input('subject');  
    $komentar = $request->input('comment');  
    $userID = $request->input('userID');  
    try{  
        $res = $this->model->support($email,$name,$subject,$komentar);  
        if($res) {
```

```
        Activity::insertActivity($userID,' Contact message  
for admin was sent');
```

```
        return response()->json(['code'=>$res]);
```

```
    }
```

```
    else {
```

```
        return response()->json(['code' => $res]);
```

```
    }
```

```
    }catch (\Exception $e) {
```

```
        ErrorLogs::insertError($e);
```

```
    }
```

```
}
```

```
}
```

LoginController.php

<?php

```
namespace App\Http\Controllers;

use App\Models\Genre;
use App\Models\User;
use Illuminate\Http\Request;
use App\Models\Activity;
use App\Mail;
use App\Models\ErrorLogs;
use Illuminate\Database\QueryException;
use Illuminate\Support\Facades\Log;


use Psy\Util\Json;
class LoginController extends FrontEndController
{

    private $model;
```

```
public function __construct()
{
    parent::__construct();
    $this->model = new User();
}
```

```
public function showLoginForm() {
    return view("login",$this->data);
}
```

```
public function showRegisterForm(){
    return view("register",$this->data);
}
```

```
public function
doRegister(\App\Http\Requests\RegisterUserRequest
$request) {
```

```

$name = $request->input('firstName');
$lastName = $request->input('lastName');
$username = $request->input('username');
$password = $request->input('sifra');
$email = $request->input('email');
$token = sha1(md5(time() . $password));

try{
    $insert=$this->model-
>register($name,$lastName,$email,$password,$username,$tok
en);

    if($insert){
        Activity::insertActivity($insert,' The user has
registered');

        $mail = new Mail();

        $code = $mail->register($token,$email);

        return response()->json(['code'=> $code]);
    }
}

```

```

else{
    ErrorLogs::insertError('Register failed');
    return response()->json(['code'=> 250]);
}

}catch (\Exception $e) {
    ErrorLogs::insertError($e);
    return response()->json($e);
}

}

```

```

public function
doLogin(\App\Http\Requests\LoginUserRequest $request){
    // dd($request);
    //if($request->has('btnLogin')){
    $email= $request->email;

```

```

$lozinka = $request->password;

$korisnik = new User();

$user = $korisnik->pronadji($email, $lozinka);

if($user){
    // Kreiranje sesije
    $request->session()->put('korisnik', $user);
    //var_dump($request->session()->get('korisnik'));

    Activity::insertActivity($user->id, ' User logged in');
    return response()->json(['code'=> 200]);

}
else{
    ErrorLogs::insertError('Login failed - Unprocessable
Entity');
    return response()->json(['code'=>422]);
}

```



```
}
```

```
public function logout(Request $request){  
    if($request->session()->has('korisnik')){  
        $id=$request->session()->get('korisnik')->id;  
        $request->session()->forget('korisnik');  
        $request->session()->flush();  
        Activity::insertActivity($id,' User logged out');  
    }  
    return redirect(route("login-form"));  
}
```

```
public function verification($token) {  
    $result = $this->model->activateUser($token);  
    if($result){  
        return redirect('/movies/');
```

```
}  
    return redirect('/movies/');  
  
}  
  
}
```

MoviesController.php

```
<?php  
  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;  
use App\Models\Genre;
```

```
use App\Models\Movie;  
use App\Models\ErrorLogs;  
use App\Models\Activity;
```

```
class MoviesController extends FrontEndController  
{  
    private $model;  
  
    public function __construct()  
    {  
        parent::__construct();  
        $this->model = new Movie();  
    }  
  
    public function index() {  
        $this->data['movies'] = $this->model->getAll();  
        return view("movies", $this->data);  
    }  
}
```

```
}
```

```
public function adminPage(){  
    return view("admin.admin",$this->data);  
}
```

```
public function lista() {  
    $this->data['movies'] = $this->model->getMovieList();  
    return view("movie-list", $this->data);  
}
```

```
function search(Request $request,$attr){  
    if($request->ajax()){  
        $attr=$request->polje;  
        //dd($attr);  
    }  
  
    try {
```

```

    $data = $this->model->getFromSearch($attr);

    if ($data) {

        Activity::insertActivity(session()->get('korisnik')->id, '
The user searching some movies');

        return response()->json($data);

    } else {

        ErrorLogs::insertError('Searching failed');

        return 'No results found';

    }

}

catch (\Exception $e){

    ErrorLogs::insertError($e);

    return response()->json($e);

}

}

```

```

function paginationSearch(Request $req) {
    //return "dkpla";

    $page = $req->page;
    $movie = $req->movie;
    $perPage = $req->numPerPage;
    //dd($perPage);

    $offset = ($page - 1) * $perPage;

    $data = $this->model-
>paginationSearch($movie,$offset,$perPage);

    //dd($res);

    return response()->json($data);
}

```

```

public function single($id) {

    $this->data['movie'] = $this->model->find($id);

    return view("single", $this->data);
}

```

```
}
```

```
public function byCategory($id) {  
    $this->data['movies'] = $this->model->findByGenreId($id);  
    return view("movies", $this->data);  
}
```

```
public function author(){  
    return view("author",$this->data);  
}
```

```
}
```

NewsController.php

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Http\Requests\CommentRequest;

use Illuminate\Http\Request;

use App\Models\News;

use Psy\Util\Json;

use App\Models\ErrorLogs;

use App\Models\Activity;


class NewsController extends FrontEndController
{
    private $model;

    public function __construct()
    {
        parent::__construct();

        $this->model = new News();
    }
}
```



```
public function news(){  
    $this->data["news"]=$this->model->getNews();  
    return view("allnews",$this->data);  
  
}
```

```
public function single($id){  
    $this->data["new"]=$this->model->singleNew($id);  
    $this->data["comments"]=$this->model->  
>getCommentsToOne($id);  
    return view("single-new",$this->data);  
  
}
```

```
public function insertKom(CommentRequest $request){  
    $id=$_POST['id'];  
    $this->model->Datum=$request->datum;  
    $this->model->Komentar=$request->comment;  
    try{
```

```

$insert=$this->model->insertKom($id);

if($insert){

    $poslednji=$this->model->showComment();

    //var_dump($poslednji);

    Activity::insertActivity(session()->get('korisnik')->id,' The
user commented on news');

    $ime = $poslednji->imeKorisnika;

    $datumNeki = $poslednji->datumKom;

    $komentar1 = $poslednji->komentar;

    $data = array(

        "ime" => $ime,

        "datum" => $datumNeki,

        "komentar" => $komentar1

    );

```

```
        return Json::encode($data);

    }

    else{

        ErrorLogs::insertError('Commenting failed');

    }

}catch (\Exception $e) {

    ErrorLogs::insertError($e);

    return response()->json($e);

}

}
```

Admin\CommentsController.php

```
<?php
```

```
namespace App\Http\Controllers\Admin;
```

```
use App\Http\Controllers\FrontEndController;
```

```
use App\Models\Admin\Comments;
```

```
use Illuminate\Http\Request;
```

```
use App\Http\Controllers\Controller;
```

```
use App\Models\Activity;
```

```
use App\Models\ErrorLogs;
```

```
class CommentsController extends FrontEndController
```

```
{
```

```
    /**
```

```
     * Display a listing of the resource.
```

```

*

* @return \Illuminate\Http\Response
*/

private $model;

public function __construct()
{
    parent::__construct();
    $this->model = new Comments();
}

public function index()
{
    $this->data['comments']=$this->model->getComments();
    return view("admin.comments.index",$this->data);
}

/**

```

* Show the form for creating a new resource.

*

* @return \Illuminate\Http\Response

*/

public function create()

{

//

}

/**

* Store a newly created resource in storage.

*

* @param \Illuminate\Http\Request \$request

* @return \Illuminate\Http\Response

*/

public function store(Request \$request)

{

//

```
}
```

```
/**
```

```
 * Display the specified resource.
```

```
 *
```

```
 * @param int $id
```

```
 * @return \Illuminate\Http\Response
```

```
 */
```

```
public function show($id)
```

```
{
```

```
    //
```

```
}
```

```
/**
```

```
 * Show the form for editing the specified resource.
```

```
 *
```

```
 * @param int $id
```

```
 * @return \Illuminate\Http\Response
```

```

*/
public function edit($id)
{
    //
}

/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    //
}

```



```

/**
 * Remove the specified resource from storage.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function destroy(Request $request,$id)
{
    $id=$request->id;

    try {

        $delete = $this->model->deleteComment($id);
        if($delete) {
            $id=session()->get('korisnik')->id;
            Activity::insertActivity($id, 'Admin deleted a
comment');
            return response(null, 204);
        }
        else{

```

```

        ErrorLogs::insertError('Comment deleting failed');
        return response()->json(['code'=> 500]);
    }

}

catch (\Exception $e) {
    ErrorLogs::insertError($e);
    return response()->json($e);
}
}
}

```

Admin\ForUserController.php

```
<?php
```

```
namespace App\Http\Controllers\Admin;
```

```
use App\Http\Controllers\FrontEndController;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

use App\Models\Admin\Users;

use Psy\Util\Json;

use App\Models\Activity;

use App\Models\ErrorLogs;


class ForUserController extends FrontEndController
{
    private $model;

    public function __construct()
    {
        parent::__construct();

        $this->model = new Users();
    }
}
```

```
public function activities(){  
    $this->data['activities']=$this->model->getActivities();  
    return view("admin.activities",$this->data);  
  
}
```

```
public function errors(){  
    $this->data['errors']=$this->model->getErrors();  
    return view("admin.errors",$this->data);  
  
}
```

```
public function addresses(){  
    $this->data['addresses']=$this->model->getAddr();  
    return view("admin.addresses",$this->data);  
  
}
```

```

public function deleteAddress(Request $request){
    $id=$request->id;
    try {
        $obrisi = $this->model->deleteAddr($id);
        if ($obrisi) {
            $id = session()->get('korisnik')->id;
            Activity::insertActivity($id, 'Admin deleted an ip
address');
            return "Obrisano";
        } else {
            ErrorLogs::insertError('Address deleting failed');
            return response()->json(['code' => 500]);
        }
    }

    catch (\Exception $e) {
        ErrorLogs::insertError($e);
        return response()->json($e);
    }
}

```

```
}
```

```
}
```

```
public function deleteError(Request $request){  
    $id=$request->id;  
    try {  
        $obrisi = $this->model->deleteErrors($id);  
        if($obrisi){  
            $id = session()->get('korisnik')->id;  
            Activity::insertActivity($id, 'Admin deleted an error');  
            return "Obrisano";  
        }  
        else {  
            ErrorLogs::insertError('Error logs deleting failed');  
            return response()->json(['code' => 500]);  
        }  
    }  
}
```

```

    }

    catch (\Exception $e) {

        ErrorLogs::insertError($e);

        return response()->json($e);

    }

}

public function deleteAct(Request $request){

    $id=$request->id;

    try {

        $obrisi = $this->model->deleteActivity($id);

        if($obrisi) {

            $id = session()->get('korisnik')->id;

            Activity::insertActivity($id, 'Admin deleted some user
activity');

            return "Obrisano";

        }
    }

```

```

else {
    ErrorLogs::insertError('User activity deleting failed');
    return response()->json(['code' => 500]);
}
}

catch (\Exception $e) {
    ErrorLogs::insertError($e);
    return response()->json($e);
}

}

```

```

public function sort(Request $request){
    $sort=$request->orderby;

```



```
$data=$this->model->sortActivity($sort);  
return Json::encode($data);  
  
}  
}
```

Admin\GenresController.php

```
<?php  
  
namespace App\Http\Controllers\Admin;  
  
use App\Http\Controllers\FrontEndController;  
use App\Http\Requests\GenreRequest;  
use App\Models\Admin\Genres;  
use Illuminate\Http\Request;  
use App\Http\Controllers\Controller;  
use App\Models\Activity;
```

```
use App\Models\ErrorLogs;
```

```
class GenresController extends FrontEndController
```

```
{
```

```
    /**
```

```
     * Display a listing of the resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
    */
```

```
    private $model;
```

```
    public function __construct()
```

```
    {
```

```
        parent::__construct();
```

```
        $this->model = new Genres();
```

```
    }
```

```

public function index()
{
    $this->data['admingenres']=$this->model->getGenres();
    return view("admin.genres.index",$this->data);
}

```

```

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */

```

```

public function create()
{
    //
}

```

```

/**
 * Store a newly created resource in storage.

```

```

*

* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/

public function store(Request $request)
{
    $name=$request->input("genreName");
    $request->validate([
        'genreName'=>'required|regex:/^[A-Z][a-z]{2,30}$/'
    ]);

    try {
        $insert = $this->model->insertGenre($name);
        if($insert) {
            $id=session()->get('korisnik')->id;
            Activity::insertActivity($id, 'Admin added a new
genre');

```

```
        return back()->with('success', 'Genre Inserted  
Successfully!');
```

```
    }
```

```
    else{
```

```
        ErrorLogs::insertError('Adding genre failed');
```

```
        return response()->json(['code'=> 500]);
```

```
    }
```

```
}
```

```
catch (\Exception $e) {
```

```
    ErrorLogs::insertError($e);
```

```
    return response()->json($e);
```

```
}
```

```
}
```

```
/**
```

```
* Display the specified resource.
```

```
*
```

```

* @param int $id
* @return \Illuminate\Http\Response
*/

public function show($id)
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    $this->data['onegenre']=$this->model->getGenreById($id);
    return view("admin.genres.edit",$this->data);
}

```

```
}
```

```
/**
```

```
 * Update the specified resource in storage.
```

```
 *
```

```
 * @param \Illuminate\Http\Request $request
```

```
 * @param int $id
```

```
 * @return \Illuminate\Http\Response
```

```
 */
```

```
public function update(Request $request, $id)
```

```
{
```

```
    $id=$request->input("id");
```

```
    $name=$request->input("genreName");
```

```
    $request->validate([
```

```
        'genreName' => 'required|regex:/^[A-Z][a-z]{2,40}$/',
```

```
    ]);
```

```

try {
    $update = $this->model->update($id,$name);
    if($update) {
        $id=session()->get('korisnik')->id;
        Activity::insertActivity($id, 'Admin edited a genre');
        return redirect(route("genres.index"))->with('success',
'Genre Edited Successfully!');

    }
    else{
        ErrorLogs::insertError('Editing genre failed');
        return response()->json(['code'=> 500]);
    }
}

catch (\Exception $e) {
    ErrorLogs::insertError($e);
    return response()->json($e);
}

```



```
}
```

```
/**
```

```
* Remove the specified resource from storage.
```

```
*
```

```
* @param int $id
```

```
* @return \Illuminate\Http\Response
```

```
*/
```

```
public function destroy(Request $request,$id)
```

```
{
```

```
    $id=$request->id;
```

```
    try {
```

```
        $delete = $this->model->deleteGenre($id);
```

```
        if($delete) {
```

```
            $id=session()->get('korisnik')->id;
```

```
            Activity::insertActivity($id, 'Admin deleted a genre');
```

```
            return response(null, 204);
```

```

    }
    else{
        ErrorLogs::insertError('Genre deleting failed');
        return response()->json(['code'=> 500]);
    }

}

catch (\Exception $e) {
    ErrorLogs::insertError($e);
    return response()->json($e);
}

}

}

```

Admin\LinkController.php

```
<?php
```

```
namespace App\Http\Controllers\Admin;
```

```
use App\Http\Controllers\FrontEndController;
```

```
use Illuminate\Http\Request;
```

```
use App\Http\Controllers\Controller;
```

```
use App\Models\Admin\Links;
```

```
use App\Models\Activity;
```

```
use App\Models\ErrorLogs;
```

```
class LinkController extends FrontEndController
```

```
{
```

```
    /**
```

```
     * Display a listing of the resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
    */
```

```
private $model;
```

```
public function __construct()  
{  
    parent::__construct();  
    $this->model = new Links();  
}
```

```
public function index()  
{  
    $this->data['links']=$this->model->getLinks();  
    return view("admin.links.index",$this->data);  
}
```

```
/**
```

```
* Show the form for creating a new resource.
```

```
*
```

```

* @return \Illuminate\Http\Response
*/

public function create()

{

    //

}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */

public function store(Request $request)
{
    $name=$request->input('linkName');
    $href=$request->input('linkHref');
    $parent=$request->input('parentLink');

```

```

$request->validate([
    'linkName'=>'required|regex:/^[A-Z][a-z]{2,30}$/ ',
    'linkHref'=>'required|regex:/^[\\a-z\\-]+$/',
    'parentLink'=>'required|regex:/^[0-9]{1,4}$/ '
]);

try {
    $insert = $this->model-
>insertLink($name,$href,$parent);

    if($insert) {
        $id=session()->get('korisnik')->id;

        Activity::insertActivity($id, 'Admin added a new link to
navigation');

        return back()->with('success', 'Navigation Link Inserted
Successfully!');

    }

    else{

        ErrorLogs::insertError('Adding link failed');

        return response()->json(['code'=> 500]);
    }
}

```

```

    }

}

catch (\Exception $e) {

    ErrorLogs::insertError($e);

    return response()->json($e);

}

}

/**
 * Display the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    //

```

```
}
```

```
/**
```

```
 * Show the form for editing the specified resource.
```

```
 *
```

```
 * @param int $id
```

```
 * @return \Illuminate\Http\Response
```

```
 */
```

```
public function edit($id)
```

```
{
```

```
    $this->data['onelink']=$this->model->getLinkById($id);
```

```
    return view("admin.links.edit",$this->data);
```

```
}
```

```
/**
```

```
 * Update the specified resource in storage.
```

```
 *
```

```
 * @param \Illuminate\Http\Request $request
```



```

* @param int $id
* @return \Illuminate\Http\Response
*/
public function update(Request $request, $id)
{
    $id=$request->input("id");
    $name=$request->input('linkName');
    $href=$request->input('linkHref');
    $parent=$request->input('parentLink');

    $request->validate([
        'linkName'=>'required|regex:/^[A-Z][a-z]{2,30}$/',
        'linkHref'=>'required|regex:/^[\\a-z\\-]+$/',
        'parentLink'=>'required|regex:/^[0-9]{1,4}$/'
    ]);

    try {
        $update = $this->model-
>update($id,$name,$href,$parent);

```

```
if($update) {  
    $id=session()->get('korisnik')->id;  
    Activity::insertActivity($id, 'Admin edited a navigation  
link');  
    return redirect(route("links.index"))->with('success',  
'Link Edited Successfully!');  
  
}  
else{  
    ErrorLogs::insertError('Editing link failed');  
    return response()->json(['code'=> 500]);  
}  
}  
  
catch (\Exception $e) {  
    ErrorLogs::insertError($e);  
    return response()->json($e);  
}
```

```

}

/**
 * Remove the specified resource from storage.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function destroy(Request $request,$id)
{
    $id=$request->id;

    try {

        $delete = $this->model->deleteLink($id);
        if($delete) {
            $id=session()->get('korisnik')->id;
            Activity::insertActivity($id, 'Admin deleted a navigation
link');

            return response(null, 204);

```

```

    }
    else{
        ErrorLogs::insertError('Link deleting failed');
        return response()->json(['code'=> 500]);
    }

}

catch (\Exception $e) {
    ErrorLogs::insertError($e);
    return response()->json($e);
}
}
}

```

Admin\MoviesController.php

```
<?php
```

```
namespace App\Http\Controllers\Admin;

use App\Http\Controllers\FrontEndController;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Models\Admin\Movies;
use App\Models\Activity;
use App\Models\ErrorLogs;
use Illuminate\Support\Facades\DB;

class MoviesController extends FrontEndController
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    private $model;
```

```
public function __construct()
{
    parent::__construct();
    $this->model = new Movies();
}
```

```
public function index()
{
    $this->data['adminmovies']=$this->model->getMovies();
    return view("admin.movies.index",$this->data);
}
```

```
/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
```

```

public function create()
{
    //
}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{

    $request->validate([

        'name' => 'required|regex:/^[A-Z][a-z\s]+$/ ',
        'desc' => 'required|min:10',
    ])
}

```

```
"slika" =>  
"file|required|mimes:jpg,jpeg,gif,png|max:3000", // in KB  
  
"length" => "required|numeric|min:40",  
  
"trailer" => "required|url", //active_url  
  
"stars"=>"required|min:10",  
  
"genre"=>"required|not_in:0",  
  
"rating"=>"required|regex:/^[0-9](\.[0-9])?$/"
```

```
]);
```

```
$slika=$request->file("slika");  
  
$fileName = $slika->getClientOriginalName();  
  
$fileName = time() . "_" . $fileName;  
  
public_path("images");  
  
try {  
  
    $slika->move(public_path("images"), $fileName);  
  
    $this->model->name=$request->input("name");
```



```

$this->model->desc=$request->input("desc");
$this->model->slika=$fileName;
$this->model->length=$request->input("length");
$this->model->trailer=$request->input("trailer");
$this->model->stars=$request->input("stars");
$this->model->genre=$request->input("genre");
$this->model->rating=$request->input("rating");

$this->model->insert();

$id=session()->get('korisnik')->id;
Activity::insertActivity($id, 'Admin added a new movie');

return back()->with('success', 'Movie Inserted
Successfully!');

}

catch (\Exception $e) {

    ErrorLogs::insertError('Adding movie failed');

    return response()->json($e);
}

```

```
}
```

```
}
```

```
/**
```

```
 * Display the specified resource.
```

```
 *
```

```
 * @param int $id
```

```
 * @return \Illuminate\Http\Response
```

```
 */
```

```
public function show($id)
```

```
{
```

```
    //
```

```
}
```

```
/**
```

* Show the form for editing the specified resource.

*

* @param int \$id

* @return \Illuminate\Http\Response

*/

public function edit(\$id)

{

//

}

/**

* Update the specified resource in storage.

*

* @param \Illuminate\Http\Request \$request

* @param int \$id

* @return \Illuminate\Http\Response

*/

public function update(Request \$request, \$id)

```

{
    //
}

/**
 * Remove the specified resource from storage.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function destroy(Request $request,$id)
{
    $id=$request->id;

    try {
        $idfilma=$this->model->getMovieById($id);
        $filePath=public_path("images/" ).$idfilma->slika;
    }
}

```

```

$delete = $this->model->deleteMovie($id);

$delRat=$this->model->deleteRate($idfilma->idOcena);

if($delete){
    unlink($filePath);
    $id=session()->get('korisnik')->id;
    Activity::insertActivity($id, 'Admin deleted a movie');
    return response(null, 204);
}
else{
    ErrorLogs::insertError('Movie deleting failed');
    return response()->json(['code'=> 500]);
}

}

catch (\Exception $e) {
    ErrorLogs::insertError($e);

```

```
        return response()->json($e);
    }
}
}
```

Admin\NewsController.php

```
<?php
```

```
namespace App\Http\Controllers\Admin;
```

```
use App\Http\Controllers\FrontEndController;
```

```
use App\Models\Admin\News;
```

```
use Illuminate\Http\Request;
```

```
use App\Http\Controllers\Controller;
```

```
use App\Models\Activity;
```

```
use App\Models\ErrorLogs;
```

```

class NewsController extends FrontEndController
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */

    private $model;

    public function __construct()
    {
        parent::__construct();
        $this->model = new News();
    }

    public function index()
    {

```

```
$this->data['news']=$this->model->getNews();  
return view("admin.news.index",$this->data);  
}
```

```
/**
```

```
* Show the form for creating a new resource.
```

```
*
```

```
* @return \Illuminate\Http\Response
```

```
*/
```

```
public function create()
```

```
{
```

```
    //
```

```
}
```

```
/**
```

```
* Store a newly created resource in storage.
```

```
*
```

```
* @param \Illuminate\Http\Request $request
```



```

* @return \Illuminate\Http\Response
*/

public function store(Request $request)
{
    $request->validate([
        'slika' =>
"file|required|mimes:jpg,jpeg,gif,png|max:3000", // in KB
        'naslov' => 'required|regex:/^[A-Z][a-z\s]+$/ ',
        'desc' => 'required|min:10'
    ]);

    $slika=$request->file("slika");

    $fileName = $slika->getClientOriginalName();

    $fileName = time() . "_" . $fileName;

    public_path("images");

    try {

        $slika->move(public_path("images"), $fileName);

        $this->model->naslov=$request->input("naslov");
    }
}

```

```

$this->model->desc=$request->input("desc");

$this->model->slika=$fileName;


$this->model->insert();

$id=session()->get('korisnik')->id;

Activity::insertActivity($id, 'Admin added some news');


return back()->with('success', 'News Inserted
Successfully!');

}

catch (\Exception $e) {

    ErrorLogs::insertError('Adding news failed');

    return response()->json($e);

}

}

/**

```

```
* Display the specified resource.  
*  
* @param int $id  
* @return \Illuminate\Http\Response  
*/
```

```
public function show($id)  
{  
    //  
}
```

```
/**  
* Show the form for editing the specified resource.  
*  
* @param int $id  
* @return \Illuminate\Http\Response  
*/
```

```
public function edit($id)  
{
```

```

    $this->data['onenew']=$this->model->getNewsById($id);
    return view("admin.news.edit",$this->data);
}

```

```

/**

```

```

 * Update the specified resource in storage.

```

```

 *

```

```

 * @param \Illuminate\Http\Request $request

```

```

 * @param int $id

```

```

 * @return \Illuminate\Http\Response

```

```

 */

```

```

public function update(Request $request,$id)

```

```

{

```

```

    $request->validate([

```

```

        'slika' =>

```

```

        "file|required|mimes:jpg,jpeg,gif,png|max:3000", // in KB

```

```

        'naslov' => 'required|regex:/^[A-Z][a-z\s]+$/ ',

```

```

        'desc' => 'required|min:10'

```

```

]);

$slika=$request->file("slika");

$fileName = $slika->getClientOriginalName();

$fileName = time() . "_" . $fileName;

public_path("images");

try {

    $slika->move(public_path("images"), $fileName);

    $this->model->id=$request->input("id");

    $this->model->naslov=$request->input("naslov");

    $this->model->desc=$request->input("desc");

    $this->model->slika=$fileName;

    $this->model->update();

    $id=session()->get('korisnik')->id;

    Activity::insertActivity($id, 'Admin edited some news');

```

```
        return redirect(route("news.index"))->with('success',  
'News Edited Successfully!');
```

```
    }
```

```
    catch (\Exception $e) {
```

```
        ErrorLogs::insertError('Editing news failed');
```

```
        return response()->json($e);
```

```
    }
```

```
}
```

```
/**
```

```
* Remove the specified resource from storage.
```

```
*
```

```
* @param int $id
```

```
* @return \Illuminate\Http\Response
```

```

*/
public function destroy(Request $request,$id)
{
    $id=$request->id;
    try {
        $idvesti=$this->model->getNewsById($id);
        $filePath=public_path("images/" ).$idvesti->slikaVest;
        $delete = $this->model->deleteNews($id);
        if($delete) {
            unlink($filePath);
            $id=session()->get('korisnik')->id;
            Activity::insertActivity($id, 'Admin deleted some
news');
            return response(null, 204);
        }
        else{
            ErrorLogs::insertError('News deleting failed');
            return response()->json(['code'=> 500]);
        }
    }
}

```

```
    }  
    catch (\Exception $e) {  
        ErrorLogs::insertError($e);  
        return response()->json($e);  
    }  
}  
}
```

Admin\RatingController.php

```
<?php
```

```
namespace App\Http\Controllers\Admin;
```

```
use App\Http\Controllers\FrontEndController;
```

```
use App\Http\Requests\RatingRequest;
```



```

use App\Models\Admin\Ratings;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

use App\Models\Activity;

use App\Models\ErrorLogs;


class RatingController extends FrontEndController
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    private $model;


    public function __construct()
    {
        parent::__construct();
    }

```

```

        $this->model = new Ratings();
    }

    public function index()
    {
        $this->data['ratings']=$this->model->getRatings();
        return view("admin.ratings.index",$this->data);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

```

```

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $rate=$request->input("rateNumber");
    $request->validate([
        'rateNumber'=>'required| regex:/^[0-9](\.[0-9])?$/'
    ]);
    try {
        $insert = $this->model->insertRating($rate);
        if($insert) {
            $id=session()->get('korisnik')->id;
            Activity::insertActivity($id, 'Admin added a new movie
rating');

```

```
        return back()->with('success', 'Movie Rating Inserted  
Successfully!');
```

```
    }
```

```
    else{
```

```
        ErrorLogs::insertError('Adding rating failed');
```

```
        return response()->json(['code'=> 500]);
```

```
    }
```

```
}
```

```
catch (\Exception $e) {
```

```
    ErrorLogs::insertError($e);
```

```
    return response()->json($e);
```

```
}
```

```
}
```

```
/**
```

```
 * Display the specified resource.
```

```
 *
```

```
 * @param int $id
```

```
 * @return \Illuminate\Http\Response
```

```

*/
public function show($id)
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    //
}

/**

```

```

* Update the specified resource in storage.
*
* @param \Illuminate\Http\Request $request
* @param int $id
* @return \Illuminate\Http\Response
*/

```

```

public function update(Request $request, $id)
{
    //
}

```

```

/**
* Remove the specified resource from storage.
*
* @param int $id
* @return \Illuminate\Http\Response
*/

```

```

public function destroy(Request $request,$id)

```

```

{
    $id=$request->id;
    try {
        $delete = $this->model->deleteRating($id);
        if($delete) {
            $id=session()->get('korisnik')->id;
            Activity::insertActivity($id, 'Admin deleted a movie
rating');
            return response(null, 204);
        }
        else{
            ErrorLogs::insertError('Rating deleting failed');
            return response()->json(['code'=> 500]);
        }
    }

    catch (\Exception $e) {
        ErrorLogs::insertError($e);
    }
}

```

```
        return response()->json($e);  
    }  
  
    }  
}
```

Admin\UsersController.php

```
<?php
```

```
namespace App\Http\Controllers\Admin;  
  
use App\Http\Controllers\FrontEndController;  
use Illuminate\Http\Request;  
use App\Http\Controllers\Controller;  
use App\Models\Admin\Users;
```



```
use App\Models\Activity;
```

```
use App\Models\ErrorLogs;
```

```
class UsersController extends FrontEndController
```

```
{
```

```
    /**
```

```
     * Display a listing of the resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
    */
```

```
    private $model;
```

```
    public function __construct()
```

```
    {
```

```
        parent::__construct();
```

```
        $this->model = new Users();
```

```
    }
```

```

public function index()
{
    $this->data['users']=$this->model->getUsers();
    return view("admin.users.index",$this->data);
}

```

```

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */

```

```

public function create()
{
    //
}

```

```

/**
 * Store a newly created resource in storage.

```

```

*

* @param \Illuminate\Http\Request $request
* @return \Illuminate\Http\Response
*/

public function store(Request $request)
{
    $name=$request->input("firstName");
    $surname=$request->input("lastName");
    $email=$request->input("email");
    $username=$request->input("username");
    $password=$request->input("password");
    $token = sha1(md5(time() . $password));
    $request->validate([

        'firstName' => 'required|regex:/^[A-Z][a-z]{2,9}$/',
        'lastName' => 'required|regex:/^[A-Z][a-z]{2,14}$/',
        'email' => 'required|regex:/^[a-z][A-z\.\-0-9]{4,35}\@[a-z]{2,5}(\.[a-z]{2,5}){1,3}$|/| max:60',
        'username' => 'required|regex:/^[a-z0-9\_]{4,15}$/',
    ]);
}

```

```
'password' => 'required|regex:/^([A-Za-z\d]){8,}$/'
```

```
]);
```

```
try {
```

```
    $insert = $this->model->insert($name,$surname,$email,$password,$username,$token);
```

```
    if($insert) {
```

```
        $id=session()->get('korisnik')->id;
```

```
        Activity::insertActivity($id, 'Admin added a new user');
```

```
        return back()->with('success', 'User Inserted Successfully!');
```

```
    }
```

```
    else{
```

```

        ErrorLogs::insertError('Adding user failed');

        return response()->json(['code'=> 500]);

    }

}

catch (\Exception $e) {

    ErrorLogs::insertError($e);

    return response()->json($e);

}

}

/**
 * Display the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */

```

```

public function show($id)
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    $this->data['oneuser']=$this->model->getUserById($id);
    return view("admin.users.edit",$this->data);
}

/**

```

```

* Update the specified resource in storage.
*
* @param \Illuminate\Http\Request $request
* @param int $id
* @return \Illuminate\Http\Response
*/

```

```

public function update(Request $request)
{
    $id=$request->input("id");
    //dd($id);
    $name=$request->input("firstName");
    $surname=$request->input("lastName");
    $email=$request->input("email");
    $username=$request->input("username");
    $role=$request->input("role");
    $active=$request->input("active");

    $request->validate([

```

```

'firstName' => 'required|regex:/^[A-Z][a-z]{2,9}$/',
'lastName' => 'required|regex:/^[A-Z][a-z]{2,14}$/',
'email' => 'required|regex:/^[a-z][A-z\.\-0-9]{4,35}\@[a-
z]{2,5}(\.[a-z]{2,5}){1,3}$|/ max:60',
'username' => 'required|regex:/^[a-z0-9\_]{4,15}$/',
'role'=>'required|numeric|regex:/^[12]$/',
'active'=>'required|regex:/^[01]$/'

```

```

]);

```

```

try {
    $update = $this->model-
>update($id,$name,$surname,$email,$active,$role,$username)
;

    if($update) {
        $id=session()->get('korisnik')->id;

        Activity::insertActivity($id, 'Admin edited an user');
    }
}

```



```
        return redirect(route("users.index"))->with('success',  
'User Edited Successfully!');
```

```
    }
```

```
    else{
```

```
        ErrorLogs::insertError('Editing user failed');
```

```
        return response()->json(['code'=> 500]);
```

```
    }
```

```
}
```

```
catch (\Exception $e) {
```

```
    ErrorLogs::insertError($e);
```

```
    return response()->json($e);
```

```
}
```

```
}
```

```
/**
```

```
* Remove the specified resource from storage.
```

*

* @param int \$id

* @return \Illuminate\Http\Response

*/

public function destroy(Request \$request,\$id)

{

 \$id=\$request->id;

 try {

 \$delete = \$this->model->deleteUser(\$id);

 if(\$delete) {

 \$id=session()->get('korisnik')->id;

 Activity::insertActivity(\$id, 'Admin deleted an user');

 return response(null, 204);

 }

 else{

 ErrorLogs::insertError('User deleting failed');

 return response()->json(['code'=> 500]);

```
    }

}

catch (\Exception $e) {

    ErrorLogs::insertError($e);

    return response()->json($e);

}

}

}
```

6.3 Modeli

Activity.php

```
<?php

/**

 * Created by PhpStorm.

 * User: Kale

 * Date: 7.3.2019
```

* Time: 13:09

*/

```
namespace App\Models;
```

```
class Activity
```

```
{
```

```
    static function insertActivity($userID, $activity)
```

```
    {
```

```
        return \DB::table('aktivnost_korisnika')
```

```
            ->insert([
```

```
                "idKorisnik" => $userID,
```

```
                "aktivnost" => $activity,
```

```
                "datum" => date('Y-m-d'),
```

```
                "vreme" => date('H:i:s')
```

```
            ]);
```

```
}
```

```
}
```

ErrorLogs.php

```
<?php
```

```
/**
```

```
 * Created by PhpStorm.
```

```
 * User: Kale
```

```
 * Date: 7.3.2019
```

```
 * Time: 13:13
```

```
*/
```

```
namespace App\Models;
```

```
class ErrorLogs
```

```
{  
    public static function insertError($text){  
        return \DB::table('log_greske')  
            ->insert([  
                "tekst" => $text,  
                "datum" => date('Y-m-d'),  
                "vreme" => date('H:i:s')  
            ]);  
    }  
}
```

Genre.php

```
<?php
```

```
namespace App\Models;
```

```
class Genre
{
    public function getAll() {
        return \DB::table("zanr")->get();
    }
}
```

IpAddress.php

```
<?php

/**
 * Created by PhpStorm.
 * User: Kale
 * Date: 7.3.2019
 * Time: 18:57
 */
```

```
namespace App\Models;
use Illuminate\Support\Facades\DB;
```

```
class IpAddress
{

    public function insert($ip){
        return DB::table('ip_adresa')
            ->insert([
                "ipAdresa"=>$ip
            ]);
    }

}
```

Menu.php


```
<?php
```

```
/**
```

```
 * Created by PhpStorm.
```

```
 * User: Kale
```

```
 * Date: 4.3.2019
```

```
 * Time: 14:22
```

```
 */
```

```
namespace App\Models;
```

```
use Illuminate\Support\Facades\DB;
```

```
class Menu
```

```
{
```

```
    public function getAll() {
```

```
        $menus = DB::table('linkovi')->get();
```

```
        return $menus;
    }
}
```

```
}
```

Movie.php

```
<?php
```

```
namespace App\Models;
use Illuminate\Support\Collection;
use Illuminate\Support\Facades\DB;
```

```
class Movie
{
```

```

/*public function getAll() {
    return DB::table("film")
        ->join("korisnik", "film.kreirao", "=", "korisnik.id")
        ->join("ocenafilma", "film.idOcena", "=",
"ocenafilma.idOcena")
        ->join("zanr", "film.idZanr", "=", "zanr.idZanr")
        ->paginate(6);
}*/

```

```

public function getAll(){
    return DB::table("film")
        ->join("ocenafilma", "film.idOcena", "=",
"ocenafilma.idOcena")
        ->paginate(6);
}

```

```

/*public function getMovieList() {
    return DB::table("film")
        ->join("korisnik", "film.kreirao", "=", "korisnik.id")

```

```

        ->join("ocena_filma", "film.idOcena", "=",
"ocena_filma.idOcena")

        ->join("zanr", "film.idZanr", "=", "zanr.idZanr")

        ->get();

    }*/

```

```

public function getMovieList(){

    return DB::table("film")

        ->join("ocena_filma", "film.idOcena", "=",
"ocena_filma.idOcena")

        ->join("zanr", "film.idZanr", "=", "zanr.idZanr")

        ->get();

}

```

```

public function getFromSearch($attr){

    return DB::table("film")

        ->join("ocena_filma", "film.idOcena", "=",
"ocena_filma.idOcena")

        ->join("zanr", "film.idZanr", "=", "zanr.idZanr")

```

```

->where("film.naziv","LIKE","%".$attr."%")

->get();

}

function paginationSearch($movie,$from,$perPage){
    return DB::table('film as f')
        ->join('ocenafilma as oc','f.idOcena','=','oc.idOcena')
        ->join("zanr as za", "f.idZanr", "=", "za.idZanr")
        ->where('f.naziv','LIKE','%'.$movie.'%')
        ->offset($from)
        ->limit($perPage)
        ->select('*')
        ->get();
}

```

```

public function find($id) {
    return DB::table("film")
        ->join("ocenafilma", "film.idOcena", "=",
"ocenafilma.idOcena")
        ->join("zanr", "film.idZanr", "=", "zanr.idZanr")
        ->where("film.idFilm", "=", $id)->first();
}

```

```

public function findByGenreId($genreId) {
    return DB::table("film")
        ->join("ocenafilma", "film.idOcena", "=",
"ocenafilma.idOcena")
        ->join("zanr", "film.idZanr", "=", "zanr.idZanr")
        ->where("film.idZanr", "=", $genreId)
        ->paginate(5);
}
}

```

News.php

```
<?php
```

```
/**
```

```
 * Created by PhpStorm.
```

```
 * User: Kale
```

```
 * Date: 2.3.2019
```

```
 * Time: 14:15
```

```
 */
```

```
namespace App\Models;
```

```
class News
```

```
{
```

```
    public $Datum;
```

```
    public $Komentar;
```

```
public function getNews(){  
    return \DB::table("vesti")  
        ->paginate(3);  
}
```

```
public function singleNew($id){  
    return \DB::table("vesti")  
        ->where("vesti.idVest", "=", $id)->first();  
}
```

```
public function getCommentsToOne($id){  
    return \DB::table("komentari")  
        ->where("komentari.idVest", "=", $id)  
        ->orderBy('komentari.datumKom','desc')  
        ->get();  
}
```



```

public function insertKom($id){
    return \DB::table("komentari")
        ->insert([
            'imeKorisnika'=>session()->get('korisnik')->ime,
            'komentar'=>$this->Komentar,
            'datumKom'=>$this->Datum,
            'idVest'=>$id

        ]);
}

```

```

public function showComment(){
    return \DB::table('vesti AS v')
        ->join('komentari AS k','v.idVest','=','k.idVest')
        ->limit(1)
        ->orderBy('k.idKomentar','desc')
        ->first();
}

```

```
}
```

```
}
```

User.php

```
<?php
```

```
/**
```

```
 * Created by PhpStorm.
```

```
 * User: Kale
```

```
 * Date: 5.3.2019
```

```
 * Time: 11:44
```

```
*/
```

```
namespace App\Models;
```

```
use Illuminate\Support\Facades\DB;
```

```

class User

{

    function
    register($name,$surname,$email,$password,$username,$token) {

        return DB::table('korisnik')
            ->insertGetId([
                "ime" => $name,
                "prezime" => $surname,
                "korisnicko_ime"=>$username,
                "lozinka" => md5($password),
                "email" => $email,
                "aktivan" => 0,
                "token" => $token,
                "uloga_id" => 2
            ]);
    }
}

```

```
}
```

```
public function pronadji($email, $lozinka){  
    return DB::table('korisnik AS k')  
        ->join('uloga AS u', 'k.uloga_id', '=', 'u.id')  
        ->where([  
            ["email", "=", $email],  
            ["lozinka", "=", md5($lozinka)]  
        ])  
        ->first();  
}
```

```
function activateUser($token){  
    return DB::table('korisnik')  
        ->where('token',$token)  
        ->update([  
            "aktivan" => 1  
        ]);  
}
```

```
}  
  
function getUser($id){  
    return DB::table('korisnik')  
        ->where('id','=',$id)  
        ->select('korisnik.id')  
        ->first();  
}
```

```
}
```

Admin\Comments.php

<?php

```
/**
```

```
 * Created by PhpStorm.
```

```
 * User: Kale
```

```
 * Date: 9.3.2019
```

```
 * Time: 21:12
```

```
 */
```

```
namespace App\Models\Admin;
```

```
use Illuminate\Support\Facades\DB;
```

```
class Comments
```

```
{
```

```
    public function getComments(){
```

```
        return DB::table("komentari")
```

```
            ->paginate(7);
```

```
    }
```

```
    public function deleteComment($id){
```

```
        return DB::table("komentar")
            ->where('idKomentar','=', $id)
            ->delete();
    }

}
```

Admin\Genres.php

```
<?php

/**
 * Created by PhpStorm.
 * User: Kale
 * Date: 9.3.2019
 * Time: 22:03
 */

namespace App\Models\Admin;
```

```
use Illuminate\Support\Facades\DB;
```

```
class Genres
```

```
{
```

```
    public function getGenres(){
```

```
        return DB::table("zanr")
```

```
            ->get();
```

```
    }
```

```
    public function insertGenre($name){
```

```
        return DB::table("zanr")
```

```
            ->insert([
```

```
                'nazivZ'=>$name
```

```
            ]);
```

```
    }
```

```
    public function getGenreById($id){
```

```
        return DB::table("zanr")
```



```
->where('idZanr','=',$id)
->first();
}
```

```
public function update($id,$name){
    return DB::table("zanr")
        ->where('idZanr','=',$id)
        ->update([
            "nazivZ"=>$name
        ]);
}
```

```
public function deleteGenre($id){
    return DB::table("zanr")
        ->where('idZanr','=',$id)
        ->delete();
}
```

```
}
```

Admin\Links.php

```
<?php
```

```
/**
```

```
 * Created by PhpStorm.
```

```
 * User: Kale
```

```
 * Date: 9.3.2019
```

```
 * Time: 20:43
```

```
 */
```

```
namespace App\Models\Admin;
```

```
use Illuminate\Support\Facades\DB;
```

```
class Links
```

```
{
```

```
public function getLinks(){  
    return DB::table("linkovi")  
        ->get();  
}
```

```
public function insertLink($name,$href,$parent){  
    return DB::table("linkovi")  
        ->insert([  
            'imeLinka'=>$name,  
            'putanja'=>$href,  
            'roditelj'=>$parent  
        ]);  
}
```

```
public function update($id,$name,$href,$parent){  
    return DB::table("linkovi")  
        ->where('idLink','=', $id)
```

```

->update([
    'imeLinka'=>$name,
    'putanja'=>$href,
    'roditelj'=>$parent
]);

}

public function getLinkById($id){
    return DB::table("linkovi")
        ->where('idLink','=',$id)
        ->first();
}

```

```

public function deleteLink($id){
    return DB::table("linkovi")
        ->where('idLink','=',$id)

```

```
        ->delete();  
    }  
}
```

Admin\Movies.php

```
<?php  
  
/**  
 * Created by PhpStorm.  
 * User: Kale  
 * Date: 7.3.2019  
 * Time: 23:02  
 */  
  
namespace App\Models\Admin;  
use Illuminate\Support\Facades\DB;  
use App\Models>ErrorLogs;
```

```

class Movies
{
    public $name;
    public $desc;
    public $genre;
    public $slika;
    public $length;
    public $trailer;
    public $rating;
    public $stars;

    public function getMovies(){
        return DB::table("film")
            -
            >join("ocenafilma","film.idOcena","=", "ocenafilma.idOcena")
            ->join("zanr","film.idZanr","=", "zanr.idZanr")
            ->paginate(8);
    }

```

```
public function deleteMovie($id){  
    return DB::table("film")  
        ->where("idFilm","=", $id)  
        ->delete();  
  
}
```

```
public function deleteRate($id){  
    return DB::table("ocenafilma")  
        ->where('idOcena','=', $id)  
        ->delete();  
  
}
```

```
public function getMovieById($id){  
    return DB::table("film")
```

```
->where('idFilm','=',$id)
->first();
}
```

```
public function insert()
{
```

```
    try {
```

```
        return DB::transaction(function () {
```

```
            $id = DB::table("ocenafilma")->insertGetId([
```

```
                "ocena" => $this->rating
```

```
            ]);
```

```
            DB::table("film")->insert([
```

```
                "naziv" => $this->name,
```

```
                "opis" => $this->desc,
```

```
                "slika" => $this->slika,
```



```
"trajanjeMin"=>$this->length,  
"trailer" => $this->trailer,  
"glavneUloge"=>$this->stars,  
"idZanr" => $this->genre,  
"idOcena" => $id,  
]);
```

```
});
```

```
} catch (\Exception $e) {  
    ErrorLogs::insertError($e);  
    return response()->json($e);  
}
```

```
}
```

```
}
```

Admin\News.php

```
<?php
```

```
/**
```

```
 * Created by PhpStorm.
```

```
 * User: Kale
```

```
 * Date: 9.3.2019
```

```
 * Time: 21:37
```

```
 */
```

```
namespace App\Models\Admin;
```

```
use Illuminate\Support\Facades\DB;
```

```
use Carbon\Carbon;
```

```
class News
{
    public $naslov;
    public $desc;
    public $slika;
    public $id;

    public function getNews()
    {
        return DB::table("vesti")
            ->get();
    }

    public function getNewsById($id)
    {
        return DB::table("vesti")
            ->where('idVest','=',$id)
            ->first();
    }
}
```

```
}
```

```
public function deleteNews($id){  
    return DB::table("vesti")  
        ->where("idVest","=", $id)  
        ->delete();  
}
```

```
public function insert(){  
    DB::table("vesti")  
        ->insert([  
            "slikaVest"=>$this->slika,  
            "naslovVest"=>$this->naslov,  
            "tekst"=>$this->desc
```

```
]);  
}
```

```
public function update() {  
    return DB::table('vesti')  
        ->where('idVest','=',$this->id)  
        ->update([  
            "slikaVest" => $this->slika,  
            "datum"=> Carbon::now()->toDateTimeString(),  
            "naslovVest" => $this->naslov,  
            "tekst" => $this->desc  
        ]);  
}  
  
}
```

Admin\Ratings.php

```
<?php

/**
 * Created by PhpStorm.
 * User: Kale
 * Date: 9.3.2019
 * Time: 21:26
 */

namespace App\Models\Admin;
use Illuminate\Support\Facades\DB;

class Ratings
{
    public function getRatings(){
        return DB::table("ocena_filma")
            ->get();
    }
}
```

```
}
```

```
public function insertRating($rate){  
    return DB::table("ocenafilma")  
        ->insert([  
            'ocena'=>$rate  
        ]);  
}
```

```
public function deleteRating($id){  
    return DB::table("ocenafilma")  
        ->where('idOcena','=', $id)  
        ->delete();  
}  
}
```

Admin\Users.php

```
<?php

/**
 * Created by PhpStorm.
 * User: Kale
 * Date: 7.3.2019
 * Time: 21:26
 */

namespace App\Models\Admin;
use Illuminate\Support\Facades\DB;

class Users
{

    public function getUsers(){
        return DB::table("korisnik")
            ->get();
    }
}
```



```
}
```

```
public function getUserById($id){
```

```
    return DB::table("korisnik")
```

```
        ->where('id','=',$id)
```

```
        ->first();
```

```
}
```

```
public function getActivities(){
```

```
    return DB::table("aktivnost_korisnika")
```

```
        ->get();
```

```
}
```

```
public function getErrors(){
```

```
    return DB::table("log_greske")
```

```
        ->paginate(7);
```

```
}
```

```
public function getAddr(){  
    return DB::table("ip_adresa")  
        ->get();  
}
```

```
public function deleteAddr($id){  
    return DB::table("ip_adresa")  
        ->where("idIp","=", $id)  
        ->delete();  
}
```

```
public function deleteErrors($id){  
    return DB::table("log_greske")  
        ->where("idGreska","=", $id)  
        ->delete();  
}
```

```
public function deleteActivity($id){  
    return DB::table("aktivnost_korisnika")  
        ->where("idAktivnost","=", $id)  
        ->delete();  
}
```

```
public function deleteUser($id){  
    return DB::table("korisnik")  
        ->where("id","=", $id)  
        ->delete();  
}
```

```
public function sortActivity($orderby){  
    return DB::table("aktivnost_korisnika")
```

```
->orderBy('datum',$orderby)

->get();

}
```

```
public function
insert($name,$surname,$email,$password,$username,$token)
{

    return DB::table('korisnik')

        ->insertGetId([

            "ime" => $name,

            "prezime" => $surname,

            "korisnicko_ime"=>$username,

            "lozinka" => md5($password),

            "email" => $email,

            "aktivan" => 0,

            "token" => $token,

            "uloga_id" => 2

        ]);

}
```

```
}
```

```
public function  
update($id,$name,$surname,$email,$active,$role,$username)  
{  
    return DB::table('korisnik')  
        ->where('id','=',$id)  
        ->update([  
            "ime" => $name,  
            "prezime" => $surname,  
            "email" => $email,  
            "aktivan" => $active,  
            "uloga_id" => $role,  
            "korisnicko_ime"=>$username,  
        ]);  
}
```

```
}
```

6.4 Klasa za slanje mail-a

App\Mail.php

```
<?php
```

```
/**
```

```
 * Created by PhpStorm.
```

```
 * User: Kale
```

```
 * Date: 7.3.2019
```

```
 * Time: 13:25
```

```
 */
```

```

namespace App;

use App\Models\ErrorLogs;

use PHPMailer\PHPMailer\PHPMailer;

use Prophecy\Exception\Exception;

use App\Models\User;


class Mail
{

    private $mail;

    private $code;

    private $user;


    public function __construct() {

        $this->mail = new PHPMailer();

        try {

            $this->mail->SMTPOptions = array(

                'ssl' => array(

```

```

        'verify_peer' => false,
        'verify_peer_name' => false,
        'allow_self_signed' => true
    )
);

$this->mail->isSMTP();

$this->mail->Host = 'smtp.gmail.com';

$this->mail->SMTPAuth = true;

$this->mail->Username = 'kalincevicnikola8@gmail.com';

$this->mail->Password = 'nikola97';

$this->mail->SMTPSecure = 'tls';

$this->mail->Port = 587;
} catch (Exception $e) {
    echo http_response_code(500);
}
}

```



```

function register($token,$email){
    try{
        $this->mail->setFrom('kalincevicnikola8@gmail.com',
        'Nikola Kalincevic');

        $this->mail->addAddress($email);

        $this->mail->isHTML(true);

        $this->mail->Subject = 'Registration';

        $this->mail->Body = 'Click on the following link: <a
href="http://127.0.0.1:8000/verification/'.$token.'">LINK</a>
to activate your account';

        $this->mail->send();

        $this->code = 200;

        return $this->code;

    }catch (\PHPMailer\PHPMailer\Exception $e){

        ErrorLogs::insertError($this->mail->ErrorInfo);

        return $e;

    }

}

```

```

public function support($email,$username,$subject,$text){

    try{

        $this->mail->setFrom('kalincevicnikola8@gmail.com',
$username);

        $this->mail-
>addAddress('kalincevicnikola8@gmail.com');

        $this->mail->addReplyTo($email, $username);

        $this->mail->isHTML(true);

        $this->mail->Subject = $subject;

        $this->mail->Body = $text;

        if($this->mail->send()){

            $this->code = 200;

            return $this->code;

        }

        else{

            $this->code = 500;

            return $this->code;

        }

    }catch (\PHPMailer\PHPMailer\Exception $e){

```

```
ErrorLogs::insertError($this->mail->ErrorInfo);  
  
return $e;  
  
}  
  
}
```

```
function confirmationMail($userID){  
    $this->user = new User();  
    $res = $this->user->getUser($userID);  
    $email = $res->email;  
    try{  
        $this->mail->setFrom('kalincevicnikola8@gmail.com',  
        'Movies Pro');  
        $this->mail->addAddress($email);  
        $this->mail->isHTML(true);  
        $this->mail->Subject = 'Confirmation mail';  
        $this->mail->Body = 'Your order has been accepted';  
        if($this->mail->send()){  
            $this->code = 200;
```

```
        return $this->code;

    }else{

        $this->code = 400;

        return $this->code;

    }

}catch (\PHPMailer\PHPMailer\Exception $e){

    ErrorLogs::insertError($this->mail->ErrorInfo);

    return $e;

}

}
```

6.5 Middleware

App\Http\Middleware\CheckIp.php

```
<?php
```

```
namespace App\Http\Middleware;
```

```
use App\Models\ErrorLogs;
```

```
use Closure;
```

```
use App\Models\IpAddress;
```

```
use App\Models\Activity;
```

```
class CheckIp
```

```
{
```

```
    /**
```

```
     * Handle an incoming request.
```

```
     *
```

```
     * @param \Illuminate\Http\Request $request
```

```
     * @param \Closure $next
```

```
     * @return mixed
```

```
    */
```

```

public function handle($request, Closure $next)
{

    // upis u bazu

    // upis u log

    if(!$request->session()->has('korisnik')) {

        $ipadresa = new IpAddress();

        $rezultat = $ipadresa->insert($request->ip());

        ErrorLogs::insertError('The user with ip_address ' .
        $request->ip() . ' trying unauthorized access to admin page!');

        return redirect("/register")-
        >with('poruka','Unauthorized Access!');

    }

    return $next($request);

```

```
}  
}
```

App\Http\Middleware\CheckIsAuth.php

```
<?php
```

```
namespace App\Http\Middleware;
```

```
use Closure;
```

```
class CheckIsAuth
```

```
{
```

```
    /**
```

```
     * Handle an incoming request.
```

```
     *
```

```
     * @param \Illuminate\Http\Request $request
```

```
     * @param \Closure $next
```

```

* @return mixed
*/
public function handle($request, Closure $next)
{
    if(!$request->session()->has('korisnik')){
        return redirect("/register")->with('poruka','Unauthorized
Access!');
    } else {
        return $next($request);
    }
}
}

```

6.6 Request-ovi

App\Http\Requests\CommentRequest.php


```
<?php
```

```
namespace App\Http\Requests;
```

```
use Illuminate\Foundation\Http\FormRequest;
```

```
class CommentRequest extends FormRequest
```

```
{
```

```
    /**
```

```
     * Determine if the user is authorized to make this request.
```

```
     *
```

```
     * @return bool
```

```
    */
```

```
    public function authorize()
```

```
    {
```

```
        return true;
```

```
    }
```

```

/**
 * Get the validation rules that apply to the request.
 *
 * @return array
 */
public function rules()
{
    return [
        'datum'=>'required',
        'comment'=>'required|min:5'
    ];
}
}

```

App\Http\Requests\LoginUserRequest.php

```
<?php
```

```
namespace App\Http\Requests;
```

```
use Illuminate\Foundation\Http\FormRequest;
```

```
class LoginUserRequest extends FormRequest
```

```
{
```

```
    /**
```

```
     * Determine if the user is authorized to make this request.
```

```
     *
```

```
     * @return bool
```

```
    */
```

```
    public function authorize()
```

```
    {
```

```
        return true;
```

```

}

/**
 * Get the validation rules that apply to the request.
 *
 * @return array
 */
public function rules()
{
    return [
        'email' => 'required|regex:/^[a-z][A-z\.\-0-9]{4,35}\@[a-
z]{2,5}(\.[a-z]{2,5}){1,3}$|/ max:60',
        'password' => 'required|regex:/^([A-Za-z\d]){8,}$/'
    ];
}
}

```

App\Http\Requests\RegisterUserRequest.php

```
<?php
```

```
namespace App\Http\Requests;
```

```
use Illuminate\Foundation\Http\FormRequest;
```

```
class RegisterUserRequest extends FormRequest
```

```
{
```

```
    /**
```

```
     * Determine if the user is authorized to make this request.
```

```
     *
```

```
     * @return bool
```

```
     */
```

```
    public function authorize()
```

```
    {
```

```
        return true;
```

```

}

/**
 * Get the validation rules that apply to the request.
 *
 * @return array
 */
public function rules()
{
    return [
        'email' => 'required|regex:/^[a-z][A-z\.\-0-9]{4,35}\@[a-z]{2,5}(\.[a-z]{2,5}){1,3}$|/ max:60',
        'firstName' => 'required|regex:/^[A-Z][a-z]{2,9}$/',
        'lastName' => 'required|regex:/^[A-Z][a-z]{2,14}$/',
        'sifra' => 'required|regex:/^([A-Za-z\d]){8,}$/',
        'username' => 'required|regex:/^[a-z0-9\_]{4,15}$/'

    ];
}

```

}