VISOKA TEHNIČKA ŠKOLA
STRUKOVNIH STUDIJA

Blog

Predmet: Klijent Server Sistemi

Profesor:

dr Mirko Kosanović

Miloš Kosanović

Student:

Marijana Stanisavljević

REr 10/17

Marijana Stanisavljević

23.1.2020.

SADRŽAJ

1. Uvod

U ovom projektu je urađena aplikacija koja za svrhu ima objavljivanje članaka od strane korisnika koji imaju nalog, kao i izlistavanje članaka od strane bilo prijavljenih, bilo neprijavljenih korisnika. Korisnik takođe može da briše ili menja članke koje je on kreirao. Od tehnologija su na klijentskoj strani korišćeni: HTML5, CSS3, Bootstrap 4, jQuery, EJS templejtski jezik. Na serverskoj strani su korišćeni NodeJS i Express framework. U aplikaciji je primenjena MVC arhitektura. Za skladištenje podataka je korišćena nerelaciona MongoDB baza podataka. Aplikacija ima sistem registracije i autentifikacije korisnika. Autentifikacija se vrši preko Passportjs modula Express framework-a, a u tu svrhu je korišćen i express-session modul za dobijanje informacija o korisniku koji je trenutno prijavljen. Za lakše upravljanje bazom podataka je korišćen i mongoose modul.

Kada se uđe na početnu stranu aplikacije, izlistani su svi članci iz baze sa svojim naslovima. Postoji navigacioni meni sa dva dugmeta. Jedno nudi korisniku da se registruje a drugo da se prijavi. Oba dugmeta okidaju modalne forme urađene uz pomoć Bootstrap 4.

Ako korisnik već ima nalog, kada se prijavi biće redirektivan na stranicu za pravljenje novog članka. Toj stranici pristup imaju samo prijavljeni korisnici. Već prijavljeni korisnik u navigacionom meniju ima ikonicu sa padajućim menijem iz koga ima opciju da se odjavi, izlista sve svoje članke, kao i da se vrati na stranicu za pravljenje novog članka. Na stranici gde izlistava svoje članke, ispod svakog članka ima dugme koje ga redirektuje na drugu rutu. Na toj ruti se prikazuje samo taj članak i korisnik ima opciju da ga izmeni ili obriše. Ako ga obriše vraća se na rutu gde se izlistavaju svi članci. Ako odabere opciju da ga izmeni, redirektovan je na rutu za izmenu članka. Nakon izmene se vraća na rutu koja prikazuje sve njegove članke.Taj isti navigacioni meni se nalazi na bilo kojoj stranici.

Ako korisnik nema nalog i napravi ga, redirektuje se na početnu stranu da bi se prijavio.

Ukoliko dođe do neke greške u aplikaciji, korisnik se obaveštava tako što se greške prikazuju na posebnoj stranici. U pitanju je error.ejs stranica, i korisnik i tada ima ponuđen navigacioni meni iz koga može da bude prosleđen ruti za prijavljivanje ili registrovanje.

Validacija formi u aplikaciji je urađena sa korisničke strane uz pomoć Bootstrap 4 klasa. 2. Instalacija i podešavanje projekta

Projekat je rađen u IDEA IDE, a za grafičko prikazivanje baze podataka je korišćen MongoDB Compass.

2. Instalacija i podešavanje projekta

Da bi projekat bio pokrenut potrebno je imati instalirane NodeJS i MongoDB na računaru.

2.1. Instaliranje modula

U ovom projektu nalazi se **package.json** fajl. U tom fajlu se nalazi lista svih modula i njihovih verzija koji su potrebni za pokretanje projekta. Pokretanjem naredbe **npm install** iz terminala, instaliraju se svi moduli navedeni u ovom fajlu.

3. Arhitektura aplikacije

U aplikaciji je primenjena MVC arhitektura, tako da se u direktorijumu models nalaze mongoose modeli baze, u views EJS fajlovi za prikaz korisničkog dela, a u controllers sva logika aplikacije. U routes direktorijumu se nalaze rute aplikacije, dok se u config direktorijumu nalazi kod vezan za primenu passport modula i autentifikaciju.

```
router.get('/', ctrlGet.home);
router.get('/new_article',ensureAuthenticated, ctrlGet.article);
router.get('/sign_out', ctrlGet.sign_out);
router.get('/user',ensureAuthenticated, ctrlGet.user);
router.post('/register', ctrlPost.register);
router.post('/sign_in', ctrlPost.sign_in);
router.post('/new_article', ensureAuthenticated,ctrlPost.new_article);
router.get('/all_articles',ensureAuthenticated, ctrlGet.all_articles);
router.get('/actions/:id', ensureAuthenticated, ctrlGet.single_article);
router.get('/delete/:id', ensureAuthenticated, ctrlGet.del);
router.get('/edit/:id', ensureAuthenticated, ctrlGet.edit);
router.post('/edit/:id', ensureAuthenticated,ctrlPost.edit);
```

3.1 Serverski deo

Node.JS- Osnova za sve ostale tehnologije korišćene u projektu. U pitanju je radno okruženje za izvršavanje Javascript koda na serveru

ExpressJS- je framework za aplikacije koje se rade u NodeJS-u

Mongoose- Object Data Modeling(ODM) biblioteka. Ona spaja NodeJS sa MongoDB

passport- Middleware za autentifikaciju za ExpressJS

passport-local- passport middlevare strategija za autentifikaciju uz pomoć korisničkog imena i lozinke

espress-session je bibilioteka koja služi za pravljenje sesija

bcryptjs-modul za hešovanje lozinke

3.2 Klijentski deo

HTML- HyperText Markup Languge -jezik za opisivanje elemenata na stranici

CSS- Cascading Style Sheets za stilizovanje elemenata

JavaScript-Interpretiran programski jezik višeg nivoa

JQuery- Javascript biblioteka za korisnički interfejs

Bootstrap 4- Framework za korisnički interfejs zasnovan na JQuery, HTML i CSS, sa gotovim elementima

3.3 Baza podataka

MongoDB je nerelaciona baza podataka. Sama baza se sastoji od kolekcija, a one se sastoj eod dokumenata u BSON formatu koji je sličan JSON. Kolekcije su ekvivalentne tabelama u relacionim bazama, dok su dokumenti ekvivalentni redovima u tabeli.

3.4 Komunikacija

"/" predstavlja početnu stranu aplikacije

"/new_article" je ruta koja pri get zahtevu generiše view za pravljenje novog članka a pri post zahtevu poziva kontroler za upisivanje članka u bazu

"/sign_out" poziva kontroler za odjavljivanje korisnika

"/user" početna strana za prijavljenog korisnika

"/register" pri post zahtevu poziva kontroler za registraciju korisnika

"/sign_in" poziva kontroler za autentifikaciju korisnika

"/all_articles" prijavljenom korisniku ispisuje sve njegove članke

"/actions/:id' ruta koja poziva kontroler koji prikazuje samo taj članak

"/delete/:id' ruta koja prilikom get zahteva priše članak i redirektuje na "/all_articles"

"/edit/:id" ruta koja prilikom get zahteva poziva kontroler koji renderuje stranicu za izmenu članka a prilikom post zahteva čuva izmene članka, redirektuje na "all_articles"

ensureAuthentificated je metoda koja ispituje da li je korisnik prijavljen ili ne.

4. Rad aplikacije

4.1 Opis implementacije

Rute objašnjene ranije u izveštaju.

```
router.get( path: '/', ctrlGet.home);
router.get( path: '/new_article',ensureAuthenticated, ctrlGet.article);
router.get( path: '/sign_out', ctrlGet.sign_out);
router.get( path: '/user',ensureAuthenticated, ctrlGet.user);
router.post( path: '/register', ctrlPost.register);
router.post( path: '/sign_in', ctrlPost.sign_in);
router.post( path: '/new_article', ensureAuthenticated,ctrlPost.new_article);
router.get( path: '/all_articles',ensureAuthenticated, ctrlGet.all_articles);
router.get( path: '/actions/:id', ensureAuthenticated, ctrlGet.single_article);
router.get( path: '/delete/:id', ensureAuthenticated, ctrlGet.del);
router.get( path: '/edit/:id', ensureAuthenticated, ctrlGet.edit);
router.post( path: '/edit/:id', ensureAuthenticated,ctrlPost.edit);
module.exports = router;
```

Kod za primenu passport modula

```
    const LocalStrategy = require('passport-local').Strategy;
const bcrypt = require('bcryptjs');

// Load User model
const db_models = require('../models/schemas');

module.exports = function(passport) {
    passport.use(
        new LocalStrategy({ usernameField: 'email' }, (email, password, done) => {
            // Match user
            db_models.User.findOne({
                email: email
            }).then(user => {
                if (!user) {
```

```
                return done(null, false,{ message: 'Ovaj mejl ne postoji' } );
            }

            // Match password
            bcrypt.compare(password, user.password, (err, isMatch) => {
                if (err) throw err;
                if (isMatch) {
                    return done(null, user);
                } else {
                    return done(null, false, { message: 'Neispravna lozinka' });
                }
            });
        });
    })
);

passport.serializeUser(function(user, done) {
    done(null, user._id);
});

passport.deserializeUser(function(_id, done) {
    db_models.User.findById(_id, function(err, user) {
        done(err, user);
    });
});
};
```

Kod za proveru da li je korsnik u sesiji.

```
module.exports.ensureAuthenticated= function (req, res, next) {
    if (req.isAuthenticated()) {
        return next();
    }
    res.render('home', {message: "Prijavite se"})
}
```

Kontroler za novi članak

```
module.exports.new_article=(req,res,next)=>{
    let article = new db_models.Article();
    article.title = req.body.title;
    article.author = req.user._id;
    article.body = req.body.body;
    console.log(req.body.toString());
    article.save(function(err){
        if(err){
            console.log(err);
            return;
        } else {
            res.redirect('/user');
        }
    });
};
```

Kontroler za kreiranje novog korisnika

```
module.exports.register=(req,res,next) =>{
    var username=req.body.username;
    var email=req.body.email;
    var password=bcrypt.hashSync(req.body.password,salt);

    new db_models.User( doc: {
        username: username,
        email: email,
        password: password
    }).save().then(
        res.render('home', {message: "Napravili ste nalog. Prijavite se."}))
        .catch(next)
};
```

Kontroler za prijavu korisnika:

```
module.exports.sign_in=(req, res, next) => {
    passport.authenticate( strategy: 'local', options: {
        successRedirect: '/new_article',
        failureRedirect: '/',
        failureFlash: true
    })(req, res, next);
};
```

Kontroler za brisanje članka

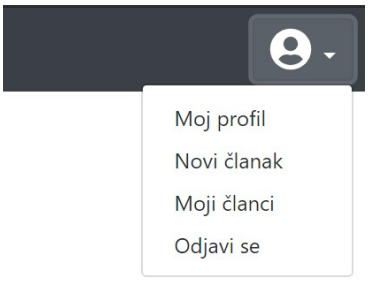
```
module.exports.del=function(req, res) {
    db_models.Article.findOne( conditions: {_id: req.params.id}).exec(function(err,article){
        if(article){
            article.remove();
            res.redirect('/all_articles');
        }
        else{
            res.redirect('/user');
        }
    })
}
```

Kontroler za izmenu članka:

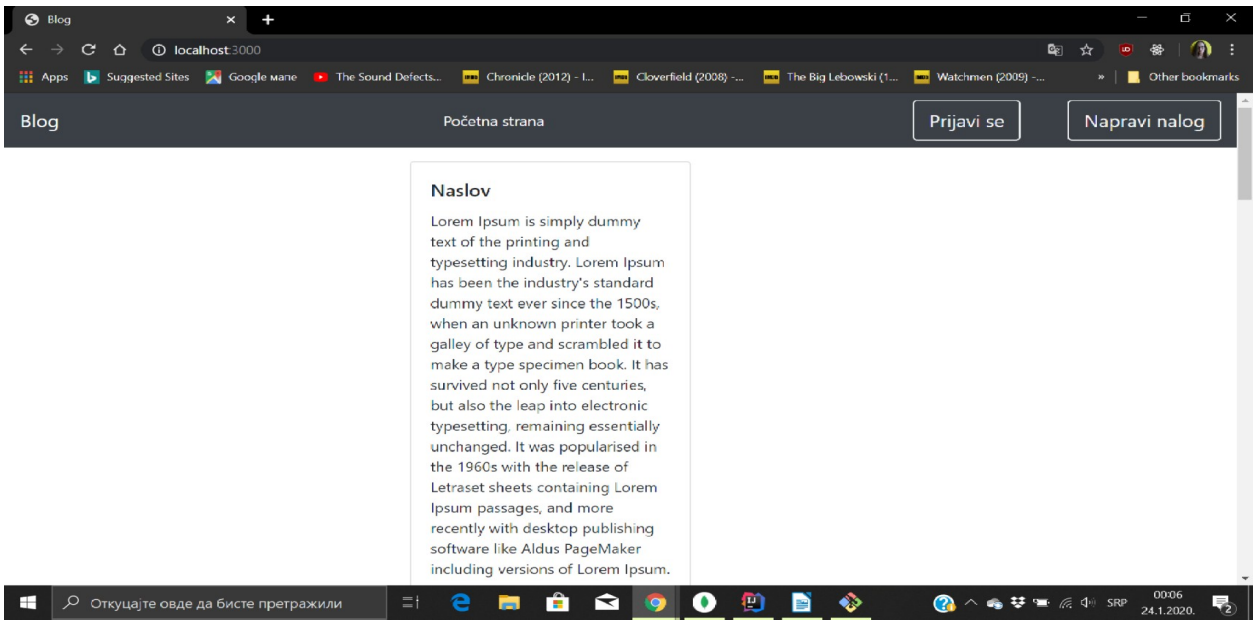
```
module.exports.edit=function (req,res) {
    db_models.Article.findById(req.params.id)
        .then(result => {
            if(result){
                res.render('edit',{
                    article:result
                });
            }
            else{
                res.redirect('/user');
            }
        })
        .catch(err => {
            res.redirect('/user');
        });
}
```

4.2 Opis funkcionalnosti – korisničko uputstvo

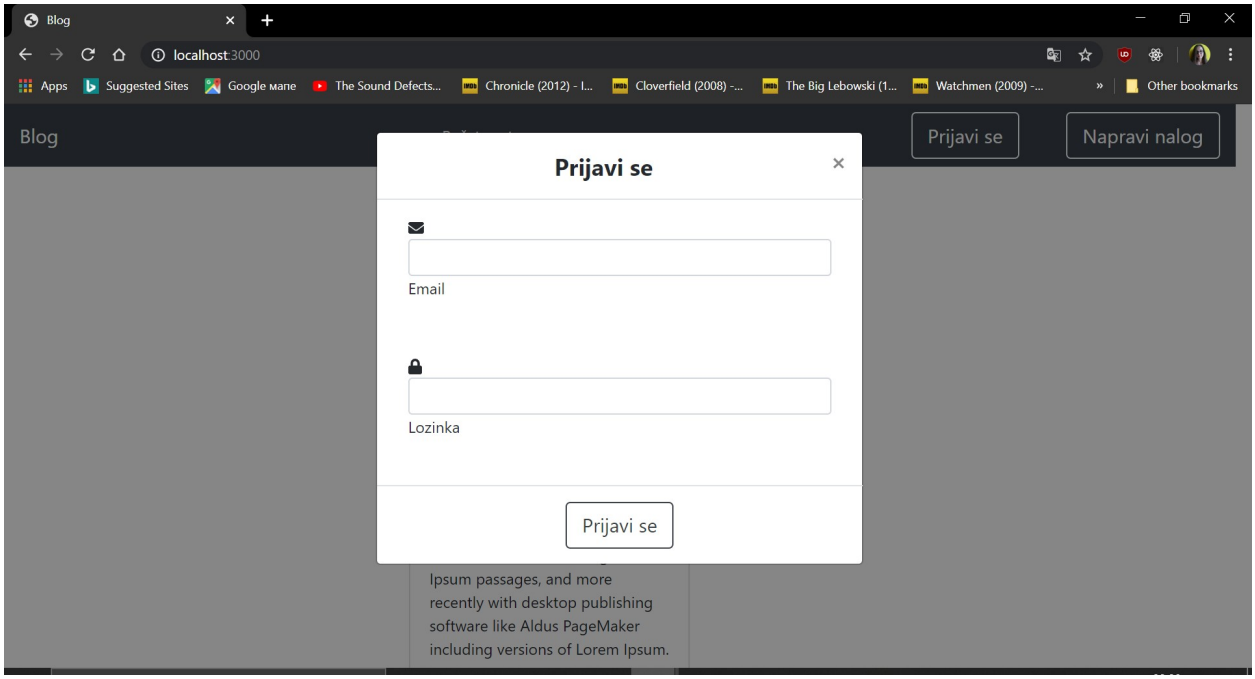
Padajući meni prijavljenog korisnika:



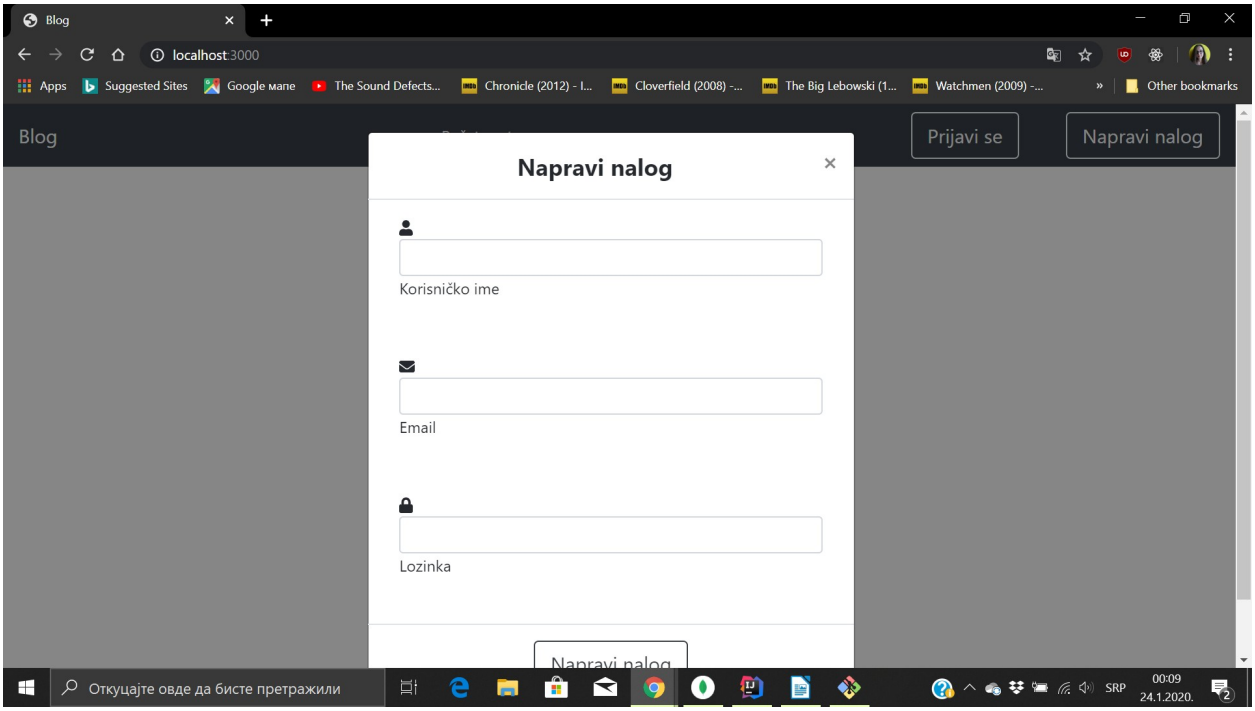
Početna strana



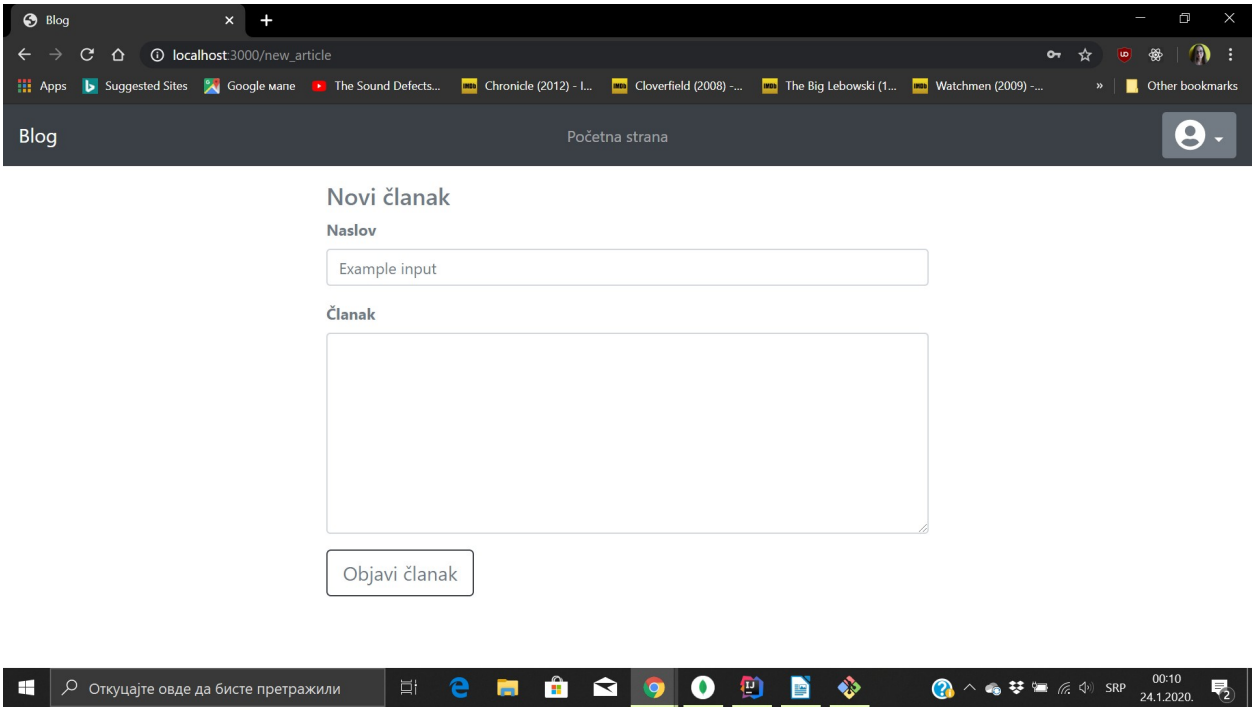
Modalnka forma za prijavu



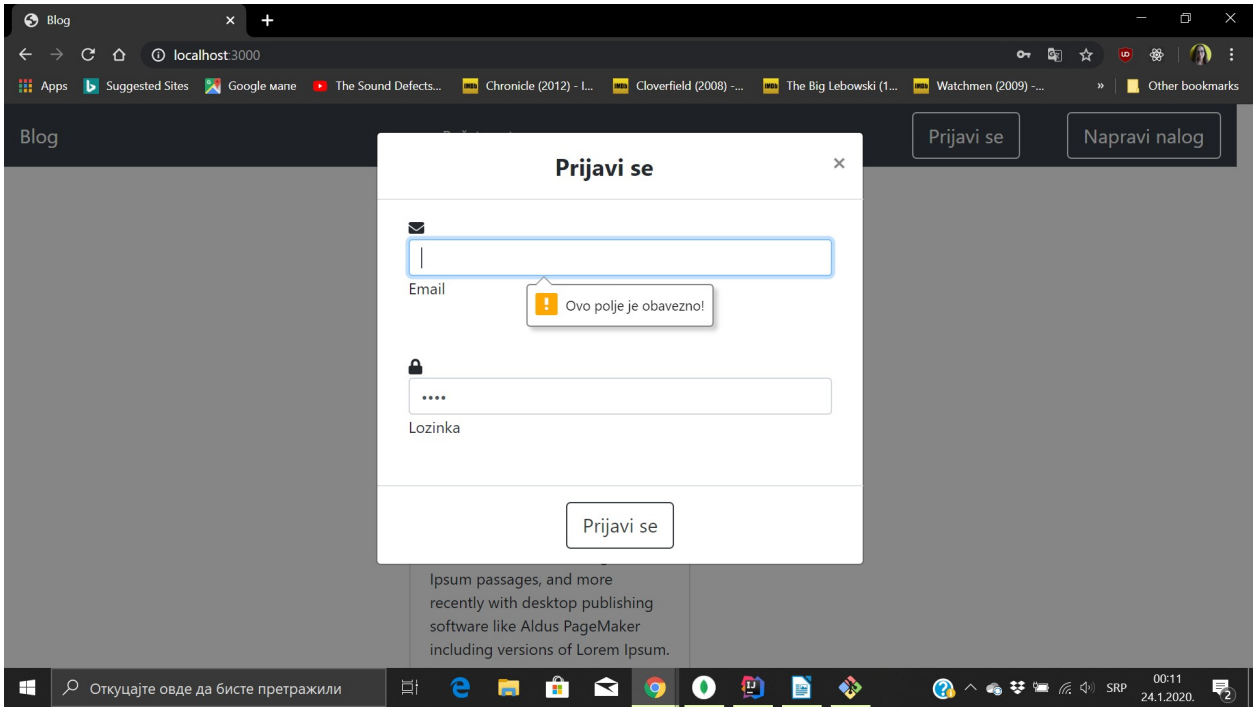
Modalna forma za registraciju



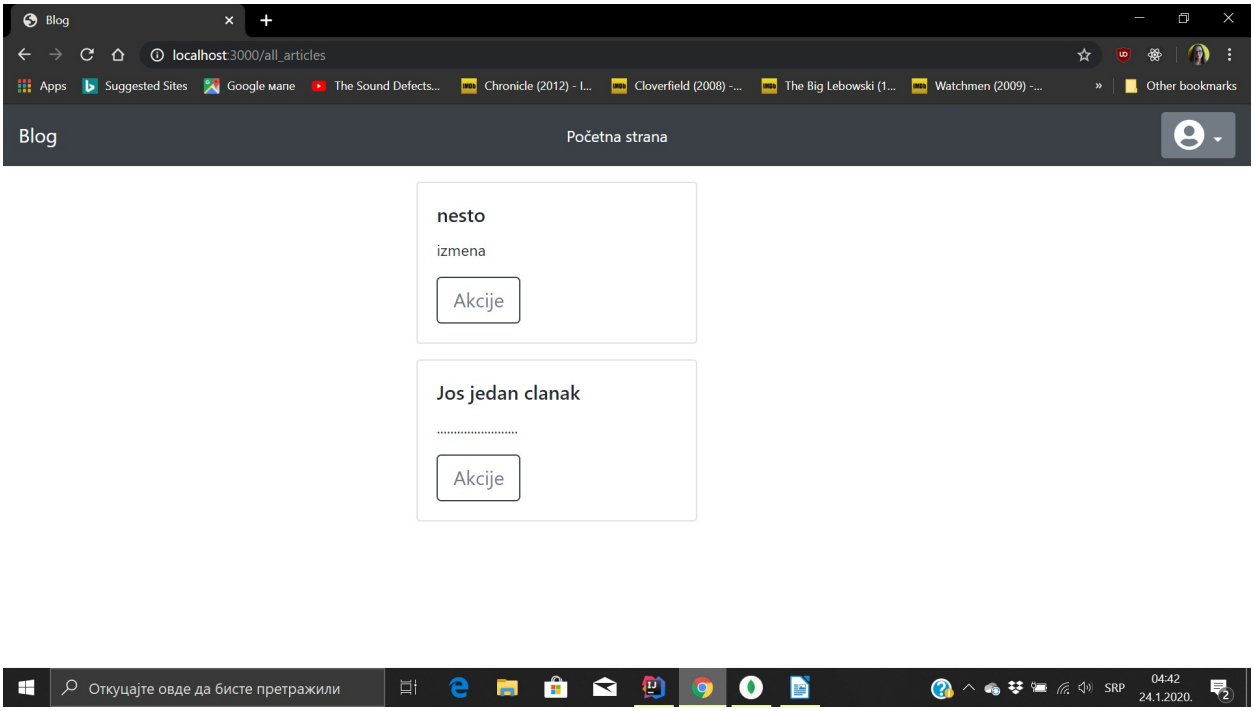
Stranica za preiranje novog članka



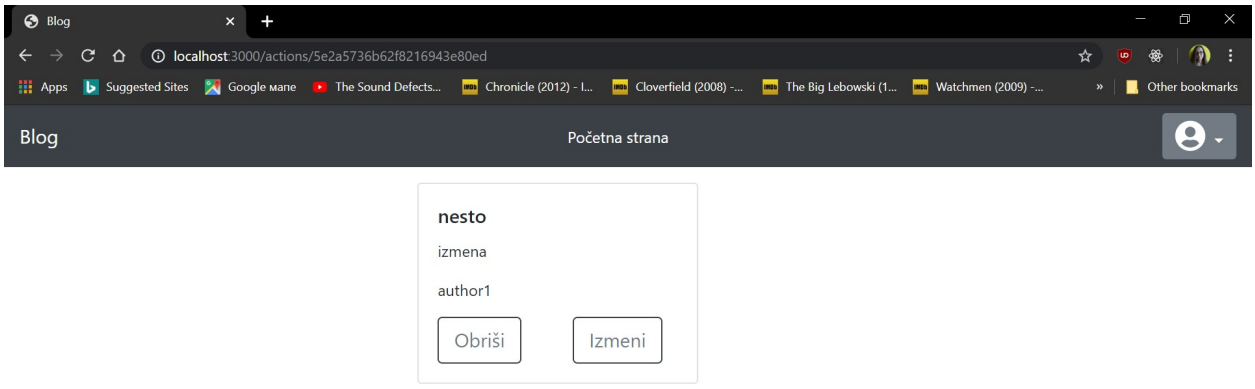
Validacija forme

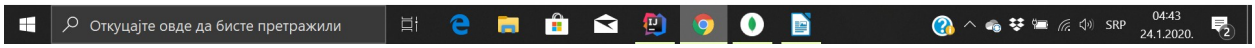


Svi članci korisnika:

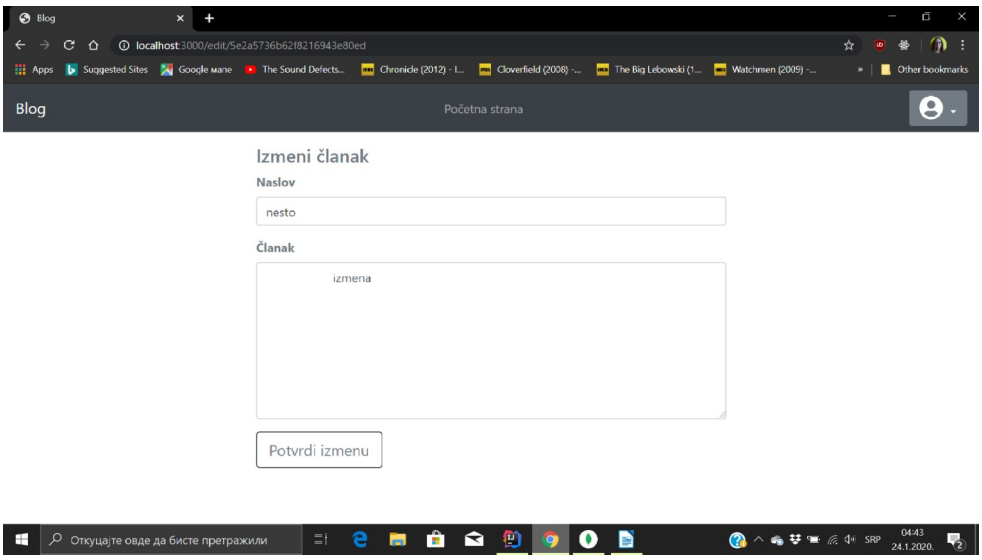


Izabrani članak:





Izmena članka:



5. Literatura

-NodeJS zvanična dokumentacija https://nodejs.org/en/
-MongoDB zvanična dokumentacija https://www.mongodb.com/
-ExpressJS zvanična dokumentacija https://expressjs.com/
-Passport.js dokumentacija https://www.npmjs.com/package/passport