Internet aplikacija za računalnu podršku igre Dungeons and Dragons

Vlajčić, Armando

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:631503

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-20**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

INTERNET APLIKACIJA ZA RAČUNALNU PODRŠKU IGRE DUNGEONS AND DRAGONS

Diplomski rad

Armando Vlajčić

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 10.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Armando Vlajčić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 840 R, 22.09.2017.
OIB studenta:	30682204118
Mentor:	Doc.dr.sc. Ivan Aleksi
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Tomislav Matić
Član Povjerenstva:	Dr.sc. Ivan Vidović
Naslov diplomskog rada:	Internet aplikacija za računalnu podršku igre Dungeons and Dragons
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	(rezervirao Armando Vlajčić) U ovom radu podrebno je izraditi web aplikaciju koja će služiti kao podrška za igranje Dungeons and Dragons. Web aplikacija će se izdraditi u modernim web tehnologijama. U njoj će korisnicima biti omogućeno stvaranje i vođenje svojih likova, kao i predmeta koji su sami osmislili koji su dio igre. Također, te stvari će biti moguće podijeliti s ostalim korisnicima.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	10.09.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTIRADA

Osijek, 25.09.2018.

Ime i prezime studenta:	Armando Vlajčić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 840 R, 22.09.2017.
Ephorus podudaranje [%]:	1

Ovom izjavom izjavljujem da je rad pod nazivom**: Internet aplikacija za računalnu podršku igre Dungeons and Dragons**

izrađen pod vodstvom mentora Doc.dr.sc. Ivan Aleksi

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1
1.1. Zadatak diplomskog rada 1
2. KORIŠTENE TEHNOLOGIJE I ALATI 2
2.1. NodeJS 2
2.1.1. Express 2
2.2. MongoDB 2
2.2.1. Mongoose 3
2.3. Twig 3
2.4. jQuery 3
2.5. Bootstrap 3
3. IZRADA APLIKACIJE 5
3.1. Baza podataka 5
3.2. Autentifikacija korisnika 7
3.2.1. Registracija 7
3.2.2. Prijava i odjava 9
3.3. Mogućnosti aplikacije 11
3.3.1. Likovi 11
3.3.2. Predmeti 14
3.3.3. Grupe 16
3.3.4. Prijatelji 20
4. ZAKLJUČAK 21
LITERATURA .....
SAŽETAK .....
ABSTRACT .....
ŽIVOTOPIS .....

1. UVOD

Dungeons and Dragons je društvena igra uloga prvi puta izdana 1974. godine. Od tada je izdano 5 različitih izdanja koji su doživjeli veliku popularnost u svijetu. Za igru je potrebna mašta, olovka, papir i set kockica, a sve osim mašte može se zamijeniti s jednom web aplikacijom. Upravo je i cilj ove aplikacije olakšati igranje igračima kao i modernizirati način igranja. Osim lakšeg igranja, igračima će biti olakšano dijeljenje i čuvanje informacija potrebnih za igru pa tako više neće biti potrebno nošenje velike količine papira i drugih stvari potrebnih za igru, nego će biti dosta imati računalo ili mobilni uređaj.

U aplikaciji će biti omogućena izrada lika, predmeta unutar igre, kao i grupe za igranje. Unutar grupe igrači će moći razgovarati s drugim igračima i dijeliti predmete koje posjeduju u igri, sve na lak i efikasan način. Svaki od tih dijelova aplikacije će biti predstavljen u ovom radu.

1.1. Zadatak diplomskog rada

U ovom radu podrebno je izraditi web aplikaciju koja će služiti kao podrška za igranje Dungeons and Dragons. Web aplikacija će se izdraditi u modernim web tehnologijama. U njoj će korisnicima biti omogućeno stvaranje i vođenje svojih likova, kao i predmeta koji su sami osmislili koji su dio igre. Također, te stvari će biti moguće podijeliti s ostalim korisnicima.

2. KORIŠTENE TEHNOLOGIJE I ALATI

2.1. NodeJS

NodeJS je razvijen 2009. godine te je dosad izašlo 10 velikih verzija. To je platforma izrađena u JavaScript programskom jeziku i služi za izradu web aplikacija. Za razliku od JavaScript-a koji se izvršava na klijentskoj strani, NodeJS se izvršava na serverskoj. Zbog toga, JavaScript možemo kombinirati i na klijentskoj strani i na serverskoj što nam uvelike olakšava posao jer za cjelokupnu aplikaciju koristimo samo jedan programski jezik, a ne kombinaciju nekoliko njih. Najčešće, pomoću NodeJS-a se izrađuju aplikacije koje se izvode u stvarnom vremenu, no može se koristiti za izradu bilo kakve web aplikacije. NodeJS je asinkron i pokretan događajima što je drukčiji pristup naspram PHP-a. To dovodi do toga da server ne blokira nove zahtjeve dok čeka odgovor nego nastavlja s izvođenjem. Pomoću mehanizma događaja omogućuje server da dobije odgovor od prošlih zahtjeva. Prednosti NodeJS, osim već navedenih, su njegova brzina, mogućnost korištenja na različitim platformama (engl. *cross platform*), otvoren kod (engl. *open source*) te njegova skalabilnost [1]. Također, NodeJS je modularan što omogućava korištenje samo onih modula koji su potrebni za izradu naše aplikacije.

2.1.1. Express

Express je razvojno okruženje (engl. framework) za NodeJS koji služi za lakšu izradu web aplikacija. On olakšava izradu aplikacija zbog toga što je puno stvari u njemu već napravljeno pa se može posvetiti posvetiti više vremena izradi aplikacije. Neke od najkorisnijih stvari su korištenje međuprograma (engl. *middleware*) s kojima se može obaviti neka akciju prije obrade zahtjeva, tablica rutiranja i dinamičko renderiranje HTML stranica zbog slanja argumenata pogledima. Zbog svoje jednostavnosti i mogućnosti, trenutno je najpopularnije razvojno okruženje za NodeJS.

2.2. MongoDB

MongoDB je *NoSQL* baza podataka, tj. nerelacijska baza podataka. To je baza koja je orijentirana prema dokumentima, a ne relacijama [2]. Budući da nemamo relacije, u jednom koraku se dohvaća cjelokupan dokument s pripadajućim vrijednostima. Također, ne mora se nužno odmah definirati točan nacrt modela, nego se može promijeniti kada nam to odgovara. MongoDB sprema dokumente u kolekcije, a svaki dokument unutar kolekcije može imati različitu shemu. Velika prednost takve

baze je njena skalabilnost i fleksibilnost. MongoDB pohranjuje podatke u binarnom obliku, takozvanom *BSON* formatu.

2.2.1. Mongoose

Mongoose je NodeJS modul koji se koristi u aplikaciji zbog toga što je to modul koji pruža sučelje prema MongoDB kolekcijama. On olakšava korištenje MongoDB-a zbog toga što pruža neke dodatne mogućnosti te su neke stvari poput validacije trivijalne za korištenje. Također, s Mongoose-om se može definirati modele i njihove veze s drugim modelima što dodatno olakšava korištenje baze podataka.

2.3. Twig

Twig je *template engine* napisan u PHP-u te se pomoću njega gradi HTML kod u pogledima (engl. *views*) . U njemu je već napravljeno mnoštvo stvari koje olakšavaju posao pa se pomoću njega može izraditi parcijalne poglede koji će se uključivati na više mjesta te se pomoću toga izbjegava pisanje duplog koda. Twig je brz, siguran i odlično dokumentiran što ga čini sjajnim izborom za tu funkcionalnost u ovoj aplikaciji. Također, Twig je napravljen od tvorca Symfony razvojnog okruženja koji je jedan od najvećih i najpopularnijih razvojnih okruženja za PHP što mu daje dodatan kredibilitet [3].

2.4. jQuery

jQuery je JavaScript biblioteka koja omogućava lakše manipuliranje HTML dokumentom [4]. Nastala je 2006. godine i od tada je doživjela 3 izdanja. Pomoću nje se s lakoćom izmjenjuje sadržaj dokumenta bez potrebe za osvježavanjem stranice što je jako korisno jer aplikacija mora biti responzivna na korisnički unos. Također, velika je prednost to što je napisana u JavaScriptu zbog toga što, kao što je ranije izrečeno, se može koristiti isti programski jezik na poslužiteljskoj i serverskoj strani.

2.5. Bootstrap

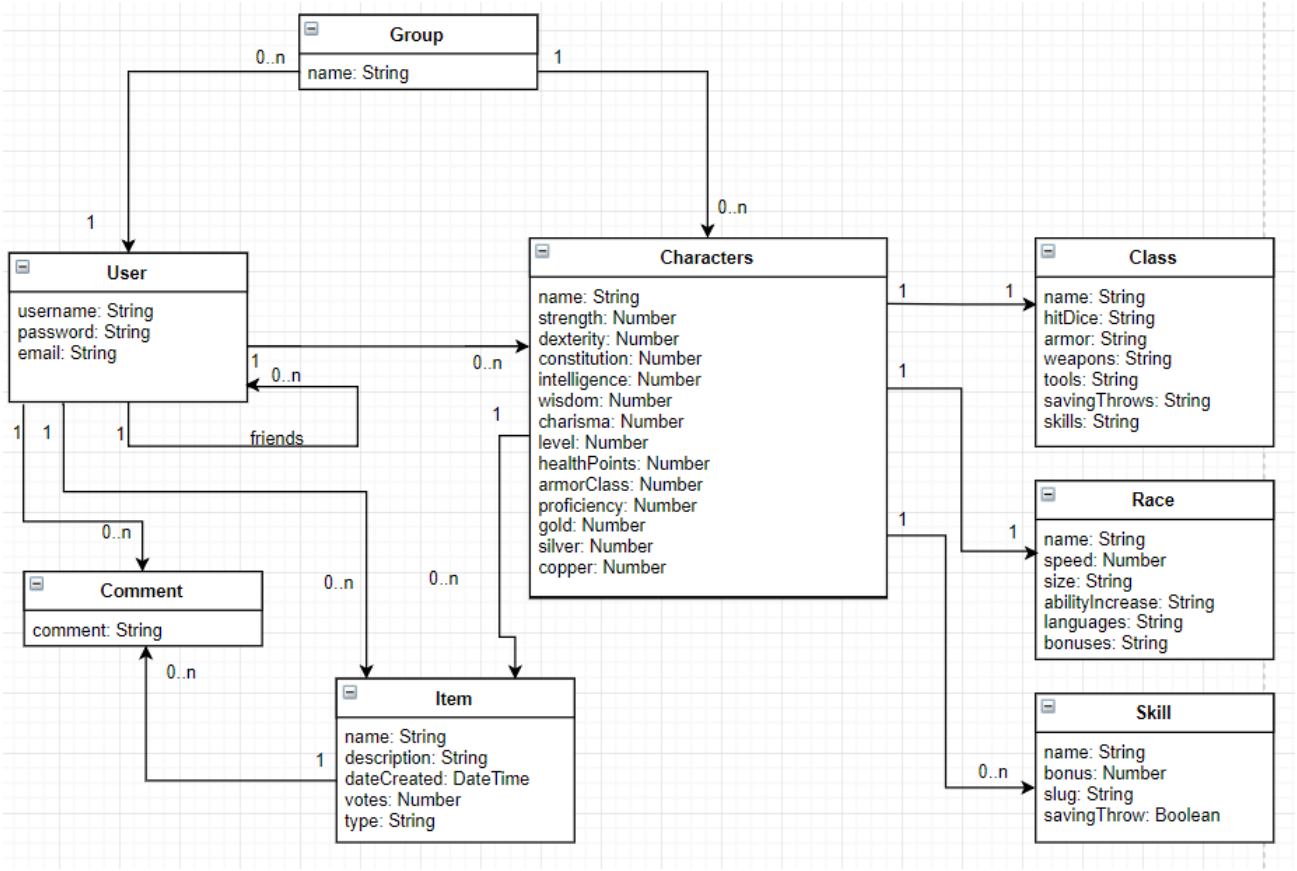
Bootstrap je biblioteka za HTML, CSS i JavaScript. Prvo izdanje Bootstrapa je bilo 2006. godine, a do sada je izašlo 4 verzije. U toj biblioteci se nalazi mnoštvo gotovog CSS stila i JavaScript funkcija spremnih za korištenje na najjednostavniji mogući način, jednostavnim uključivanjem potrebnih klasa na naše HTML elemente. Najveća prednost Bootstrapa je njegov *Grid system* pomoću kojeg se stranica može podijeliti na nekoliko dijelova te tako napraviti stranicu vizualno privlačnijom, ali i

responzivnom što je vrlo bitno u današnje vrijeme. Osim toga, Bootstrap ima podršku za sve bitnije stvari koji novi web preglednici podržavaju poput *flexboxa* koji je raspored izgleda (engl. *layout model*) koji dozvoljava responzivnim elementima da budu prilagođeni ovisno o veličini ekrana [5].

3. IZRADA APLIKACIJE

3.1. Baza podataka

Prije izrade ostatka aplikacije potrebno je definirati sve modele i njihove relacije. Najlakši način za prikaz toga je pomoći ER dijagrama koji je vidljiv na slici 3.1.



Slika 3.1. Prikaz ER dijagrama za aplikaciju

Na ER dijagramu je prikazano 8 modela koji su međusobno povezani na različite načine. Iz njega je primjetno kako će korisnik aplikacije imati poveznicu sa svojim komentarima, grupama, likovima i predmetima, kao i relaciju na model korisnika koja će označavati prijatelje tog korisnika. Ostali modeli (klasa, rasa i vještina) su povezani s modelom lika. Na slici 3.2. se može vidjeti implementacija modela korisnika, a na slici 3.3. modela lika (engl. *character*).

Model korisnika se sastoji od 3 polja: korisničkog imena, emaila i lozinke te relacije na model korisnika koji služi kao lista prijatelja. S obzirom na to da se ta relacija definira kao niz, korisnik može imati 0 ili više prijatelja. Ostale relacije modela su definirane na drugim modelima. Također,

polja za korisničko ime i email adresu su definirana da budu jedinstvena kako bi se mogla osigurati jednoznačnost korisnika.

```
let schema = new Schema({
    username: {type: String, required: true, unique: true},
    email: {type: String, required: true, unique: true},
    password: {type: String, required: true},
    friends: [{ type: Schema.Types.ObjectId, ref: 'User' }],
});
```

Slika 3.2. Shema modela korisnika

```
let schema = new Schema({
    name: {type: String, required: true},
    class: { type: Schema.Types.ObjectId, ref: 'Class' },
    race: { type: Schema.Types.ObjectId, ref: 'Race' },
    skills: [{ type: Schema.Types.ObjectId, ref: 'Skill' }
    items: [{ type: Schema.Types.ObjectId, ref: 'Item' }]
    strength: {type: Number, required: true},
    dexterity: {type: Number, required: true},
    constitution: {type: Number, required: true},
    intelligence: {type: Number, required: true},
    wisdom: {type: Number, required: true},
    charisma: {type: Number, required: true},
    level: {type: Number, required: true},
    healthPoints: {type: Number, required: true},
    armorClass: {type: Number, required: true},
    proficiency: {type: Number, required: true},
    gold: {type: Number, required: true},
    silver: {type: Number, required: true},
    copper: {type: Number, required: true},
    user: { type: Schema.Types.ObjectId, ref: 'User' },
});
```

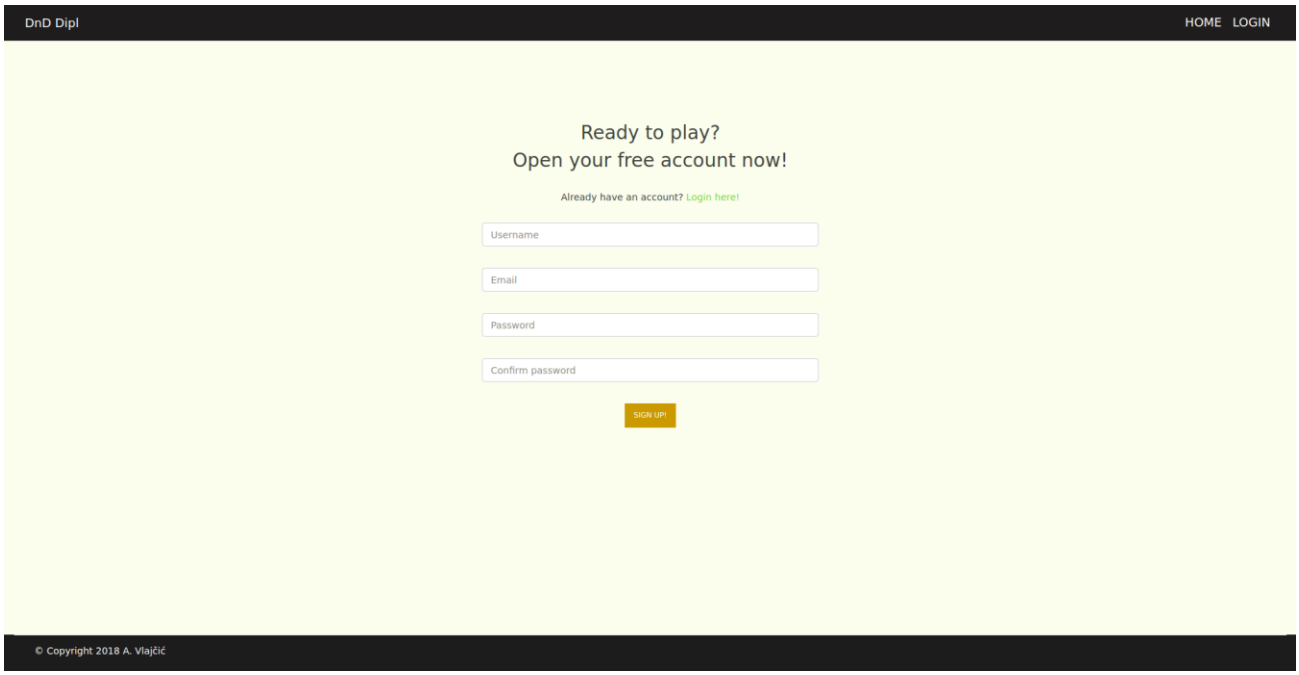
Slika 3.3. Prikaz sheme modela lika

Kao što se može vidjeti, model lika ima više polja i relacija na druge objekte, ali je sama srž opisa modela potpuno jednaka. Tako se može primijetiti da model lika ima relaciju na 0 ili više predmeta (engl. *item*), a relacija prema korisniku je definirana tako da se može referencirati samo na jednog korisnika. Sve relacije koje su definirane na takav način su neophodne.

3.2. Autentifikacija korisnika

3.2.1. Registracija

Za registraciju korisnik mora upisati željeno korisničko ime, vlastitu email adresu te svoju lozinku koju mora potvrditi u drugom polju. Na slici 3.4. se može vidjeti izgled forme za registraciju novog korisnika.



Slika 3.4. Izgled forme za registraciju korisnika

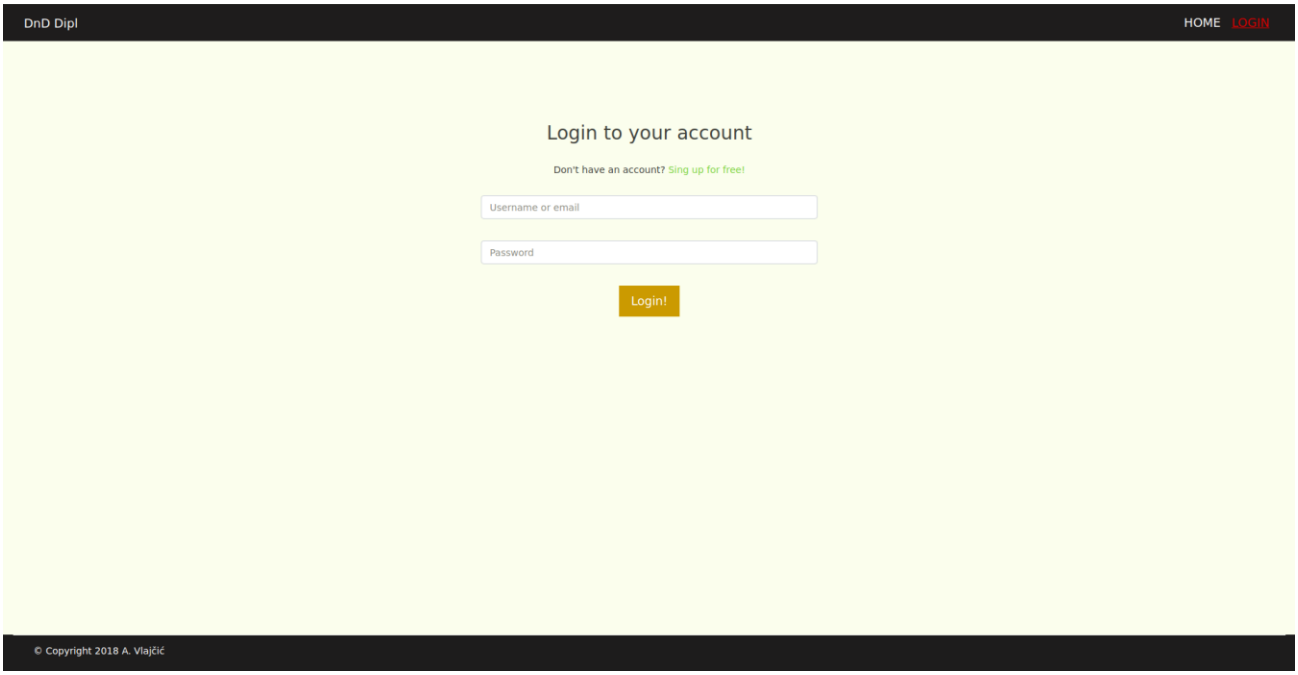
Podaci koje je korisnik upisao šalju se na rutu */users/signup* gdje se provjeravaju da li su svi podaci ispravno uneseni te ako je sve u redu novi korisnik se registrira. Kao što vidimo iz slike 3.5. prvo se u ruti provjerava da li su svi podaci poslani te se onda provjerava ispravnost podataka poput toga da li podatak za email adresu uistinu odgovara zadanom formatu email adrese. Nakon što je sigurno da su svi podaci ispravni, mora se provjeriti da li u aplikaciji već postoji korisnik s jednakim korisničkim imenom ili email adresom. Ako postoji bilo kakva greška ili je pronađen korisnik s istim podacima, korisnik će biti preusmjeren nazad na formu za registraciju te će mu se prikazati

odgovarajuća pogreška. U slučaju da je sve u redu pravi se novi korisnik nakon što se lozinka enkriptira kako bi zaštitili podatke u slučaju proboja baze podataka. Nakon što je novi korisnik stvoren i sačuvan u bazi podataka, korisnik se prijavljuje tako da se sprema u trenutnu sesiju te se preusmjerava na početnu stranicu.

```
router.post('/signup', function (req, res, next) {
    /* Check if not empty fileds */
    if (!validator.isEmpty(req.body.username) && !validator.isEmpty(req.body.email) &&
        !validator.isEmpty(req.body.password) && !validator.isEmpty(req.body.password_conf)) {
        /* Check if email and same password */
        if (validator.isEmail(req.body.email) && req.body.password === req.body.password_conf) {
            /* If we don't curretly have user with that email or username we create a new one*/
            Promise.all([findUserWithEmail(req.body.email, false), findUserWithUsername(req.body.username, false)])
            .then(() => bcrypt.hash(req.body.password, 5)).then(function (hash) {
                let user = new User({
                    username: req.body.username,
                    email: req.body.email,
                    password: hash,
                });
                return user.save();
            }).then(function (doc) {
                req.session.user = doc;
                res.redirect('/');
            }).catch(function (err) {
                req.session.registrationError = err.message;
                res.redirect('/signup');
            });
        } else {
            req.session.registrationError = "Wrong email or passwords don't match!";
            res.redirect('/signup');
        }
    } else {
        req.session.registrationError = "Missing data!";
        res.redirect('/signup');
    }
});
```

Slika 3.5. Ruta za registraciju korisnika

3.2.2. Prijava i odjava



Slika 3.6. Izgled forme za prijavu korisnika

Za prijavu korisnik može birati između korisničkog imena ili email adrese s kojom se registrirao. Nakon provjere podataka te ako je pronađen korisnik s odgovarajućim korisničkim imenom ili email adresom vrši se provjera kriptirane lozinke. Ako je sve u redu korisnik se sprema u sesiju te preusmjerava na početnu stranicu, a ukoliko je došlo do problema korisnik će biti preusmjeren na stranicu za prijavu te će mu se prikazati odgovarajuća poruka greške.

Odjava korisnika se odrađuje tako da se uništi trenutna sesija korisnika te tako njegovi podaci postanu nedostupni aplikaciji.

```
router.get('/logout', function (req, res, next) {
    req.session.destroy(function (err) {
        if (err) throw err;
        else{
            res.redirect('/');
        }
    })
});
```

Slika 3.7. Ruta za odjavu korisnika

```
router.post('/login', function (req, res, next) {
    if (!validator.isEmpty(req.body.usernameOrEmail) && !validator.isEmpty(req.body.password)) {
        Promise.all([findUserWithEmail(req.body.usernameOrEmail, true), findUserWithUsername(req.body.usernameOrEmail, true)])
       .then(function (doc) {
            if ((doc[0] !== undefined && doc[0].length === 1) || (doc[1] !== undefined && doc[1].length === 1)) {
                doc = doc.filter(val => val !== undefined);
                let foundUser = doc[0][0];

                bcrypt.compare(req.body.password, foundUser.password).then(function (res) {
                    if (res === false){
                        return Promise.reject(new Error('Wrong credentials!'));
                    }
                }).then(function () {
                    req.session.user = foundUser;
                    res.redirect('/');
                }).catch(function (err) {
                    req.session.loginError = err.message;
                    res.redirect('/login')
                });
            }else{
                return Promise.reject(new Error('Wrong credentials!'));
            }
        }).catch(function (err) {
            req.session.loginError = err.message;
            res.redirect('/login');
        });
    } else {
        req.session.loginError = "Missing data!";
        res.redirect('/login');
    }
});
```

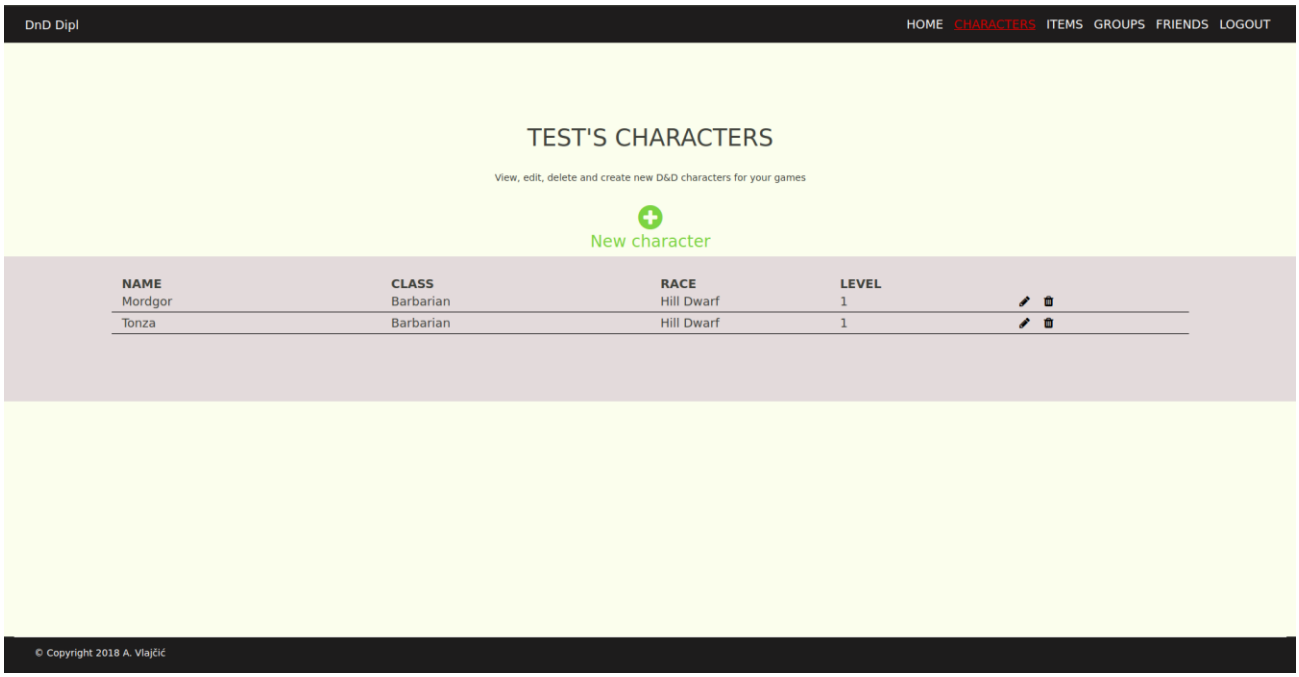
Slika 3.8. Ruta za prijavu korisnika

3.3. Mogućnosti aplikacije

Kada se korisnik prijavi ponuđene su mu različite mogućnosti na korištenje. Omogućeni su mu pregled postojećih i izrada novih likova, predmeta i grupa, kao i upravljanje svojom listom prijatelja. Također, postoji i mogućnost pregledavanja predmeta koje su napravili drugi korisnici te njihovo komentiranje i glasanje. Na kraju, korisniku može koristiti i pregled grupa u kojima je član te tamo obavljati radnje koje su vezane za grupu.

3.3.1. Likovi

Na stranici za pregled likova korisnik vidi sve likove koje je kreirao te ih tamo može urediti ili pak obrisati. Prilikom pristupanja toj stranici potrebno je dohvatiti sve likove korisnika kao i popuniti podatke o klasi i rasi lika, a kod za to može se vidjeti na slici 3.10.

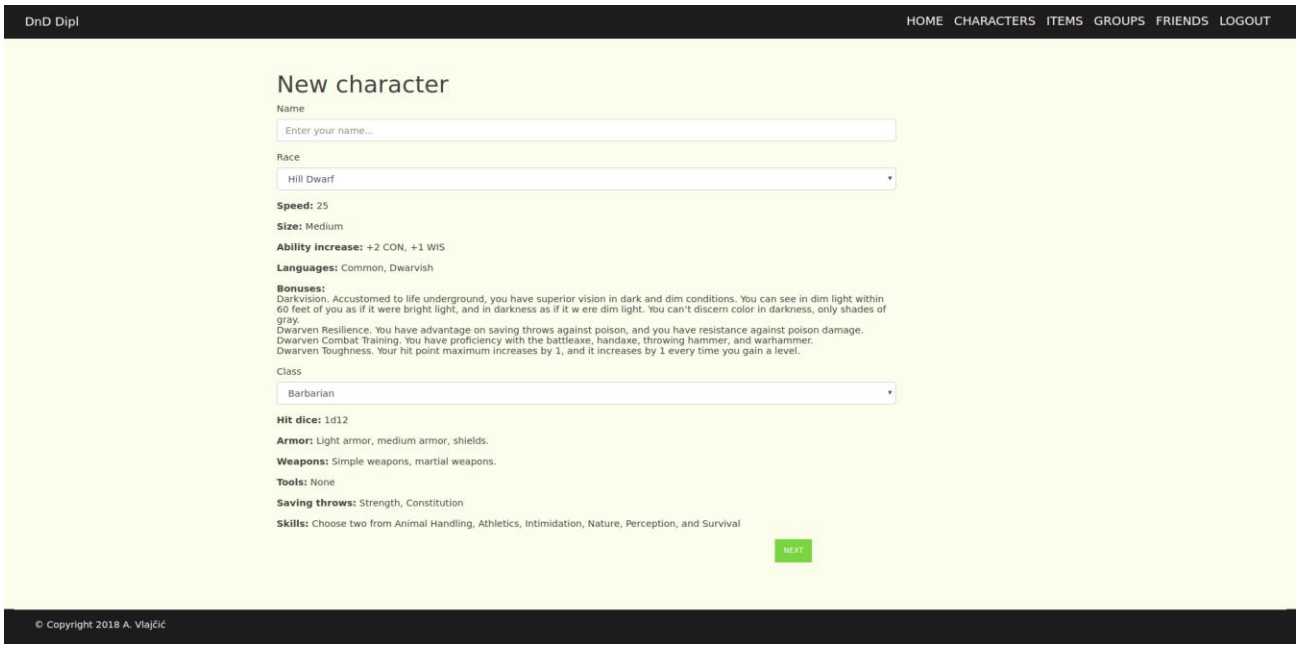


Slika 3.9. Izgled stranice za pregled likova

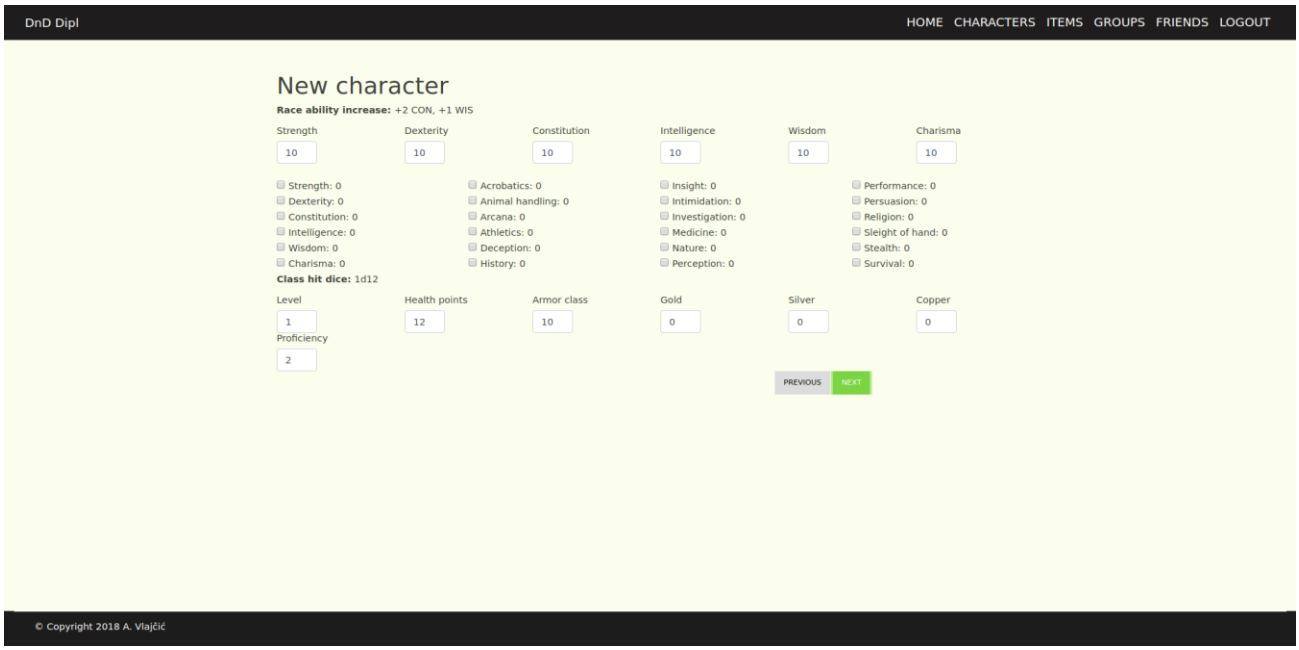
```
router.get('/mycharacters', isAuthenticated, function (req, res, next) {
    Character.find({user: req.session.user}).populate('class', 'name').populate('race', 'name').exec().then(function (characters) {
        res.render('users/mycharacters', {user: req.session.user, characters: characters, title: 'Characters'});
    }).catch(function (err) {
        res.render('users/mycharacters', {user: req.session.user, error: 'Problem with retrieving characters.', title: 'Characters'});
    });
});
```

Slika 3.10. Ruta za pregled svih likova korisnika

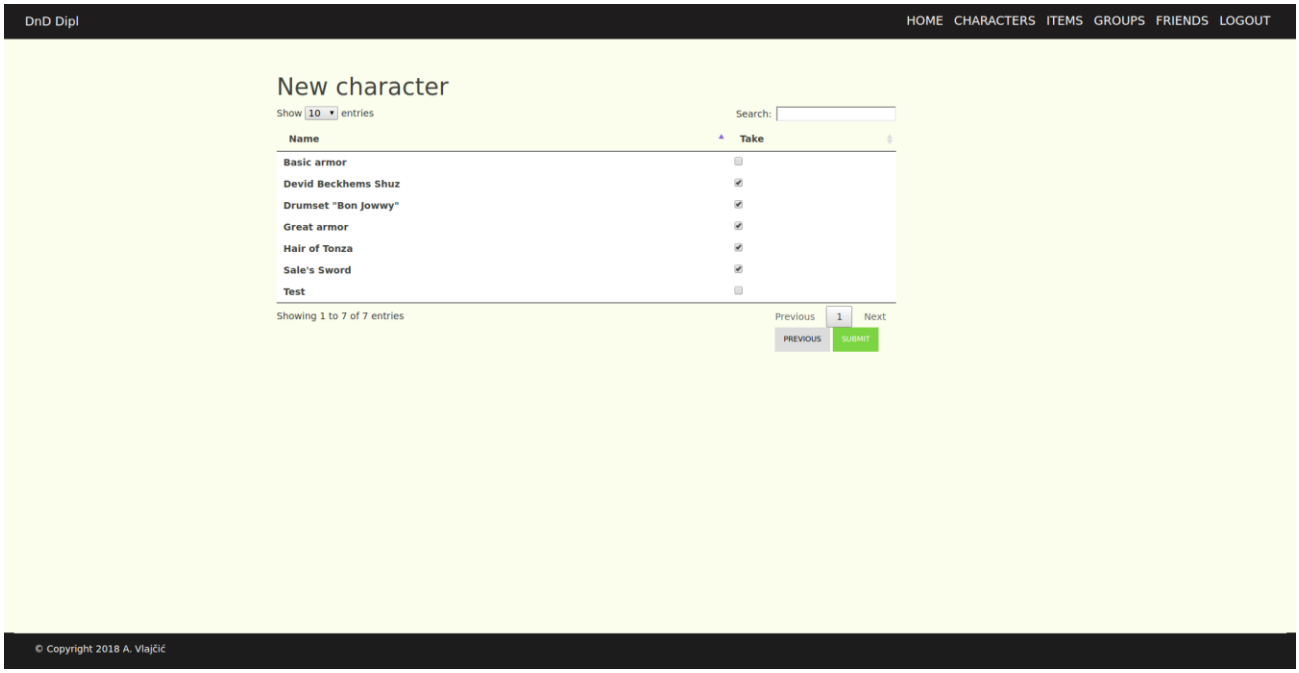
Kod kreiranja novog lika korisnik će morati ispuniti formu u 3 koraka. U prvom daje osnovne podatke o liku poput imena, klase i rase, u drugom koraku daje podatke o vještinama lika kao i podatke o trenutnoj razini lika i slično, te na kraju u zadnjem koraku korisnik odabire predmete koje njegov lik posjeduje. Nakon što je ispunio sve potrebne podatke novi lik se kreira te ga sada korisnik može koristiti kao lika u svojim grupama.



Slika 3.11. Prvi korak forme za kreiranje lika



Slika 3.12. Drugi korak forme za kreiranje lika



Slika 3.13. Treći korak forme za kreiranje lika

```
router.get('/characters/:characterId/edit', isAuthenticated, function (req, res, next) {
    Promise.all([
        Race.find(),
        Class.find(),
        Skill.find({savingThrow: false}),
        Item.find().sort({name: 1}),
        Character.findOne({_id: req.params.characterId}).populate('race', 'name').populate('class', 'name').populate('skills', 'name').exec()
    ]).then(function (doc) {
        // sort skills by name
        doc[2].sort(compareNames);
        let hasSkills =[];
        // get all skills from character
        doc[4].skills.forEach(function (skill) {
            hasSkills.push(skill.name);
        });
        res.render('users/newcharacter', {
            races: doc[0],
            classes: doc[1],
            skills: doc[2],
            items: doc[3],
            user: req.session.user,
            character: doc[4],
            hasSkills: hasSkills,
            title: doc[4].name + ' edit'
        });
    }).catch(function (err) {
        console.log(err);
    });
});
```

Slika 3.14. Ruta za uređivanje lika

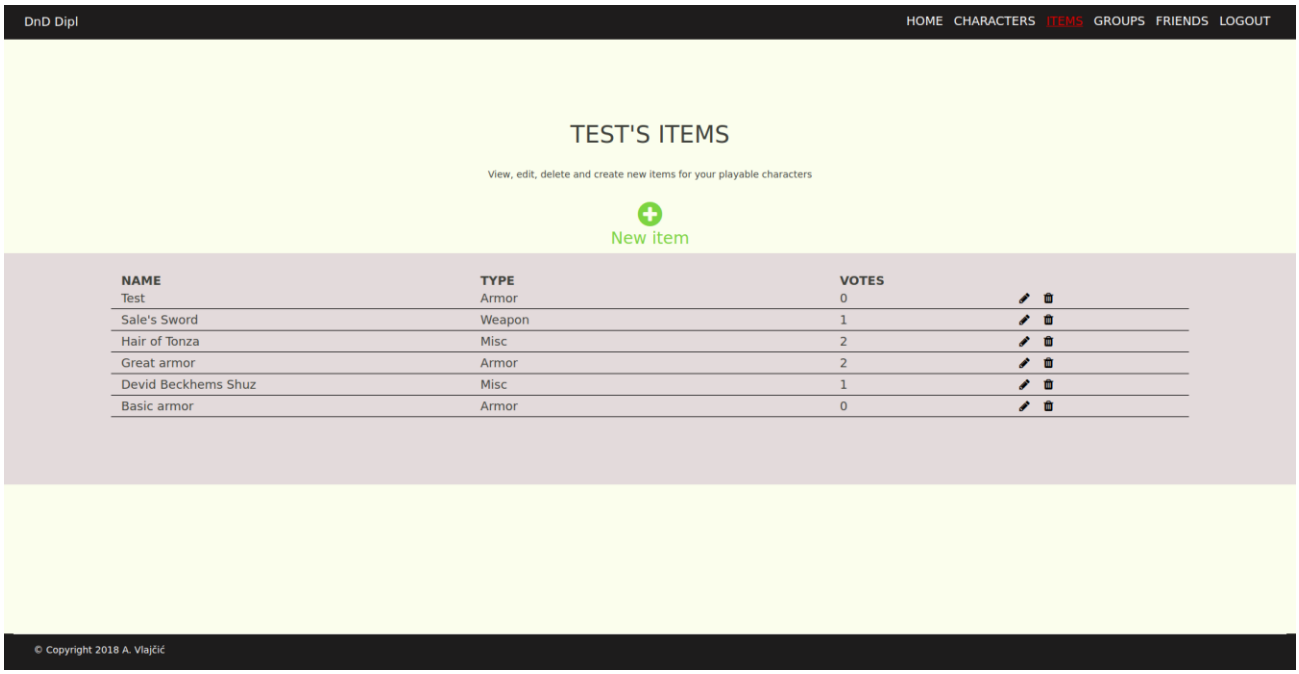
Prilikom kreiranja i uređivanja lika, koristi se isti pogled tako da se ta radnja obavlja na isti način. Jedina razlika u pogledu je u tome što će kod uređivanja lika polja za unos imati početnu vrijednost. Razlika prilikom pristupa ruti je u tome što se u bazi podataka mora pronaći lik kojeg korisnik želi

urediti te popuniti potrebne podatke kako bi korisniku bile odabrane stavke koje se odnose na tog lika.

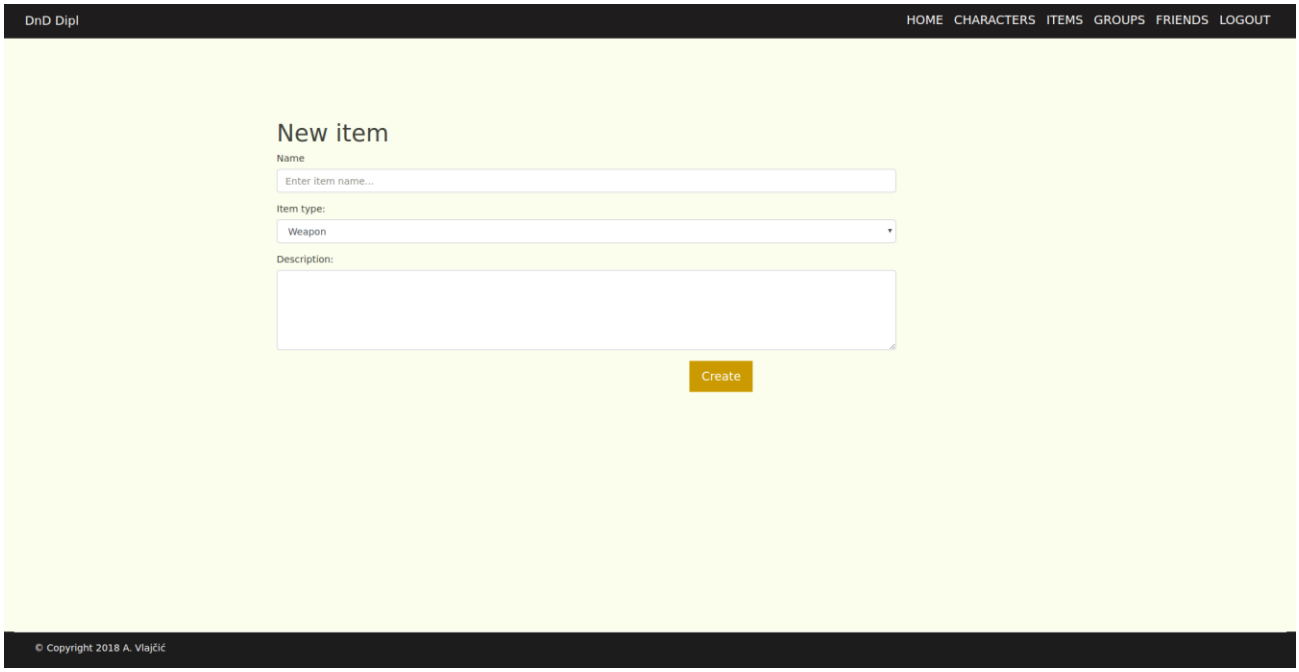
3.3.2. Predmeti

Kao i za lika, korisniku je omogućen pregled, kreiranje, uređivanje i brisanje svojih postojećih predmeta. Na stranici za pregled predmeta korisnik vidi listu svojih predmeta kao i njihov tip te broj glasova koji su mu dali drugi korisnici.

Kao što je vidljivo na slici 3.16. forma za kreiranje predmeta je jednostavnija jer se popunjavaju samo 3 polja: ime, tip i opis predmeta gdje korisnik daje više informacija o predmetu i njegovim sposobnostima.

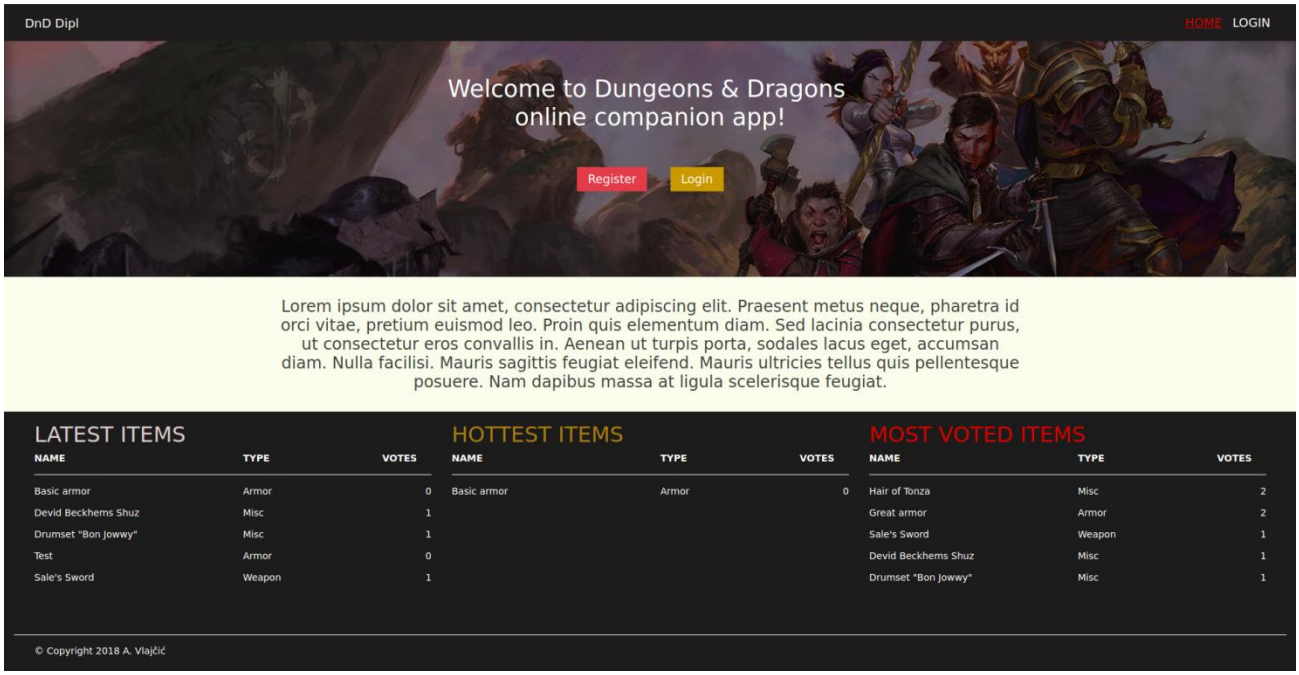


Slika 3.15. Stranica za pregled predmeta korisnika



Slika 3.16. Izgled forme za kreiranje predmeta

Svi predmeti koje korisnik kreira su javni što znači da ga svi ostali korisnici mogu vidjeti i koristiti za svoje likove. Također, na naslovnoj stranici postoji top lista svih predmeta po broju glasova koje je taj predmet dobio.

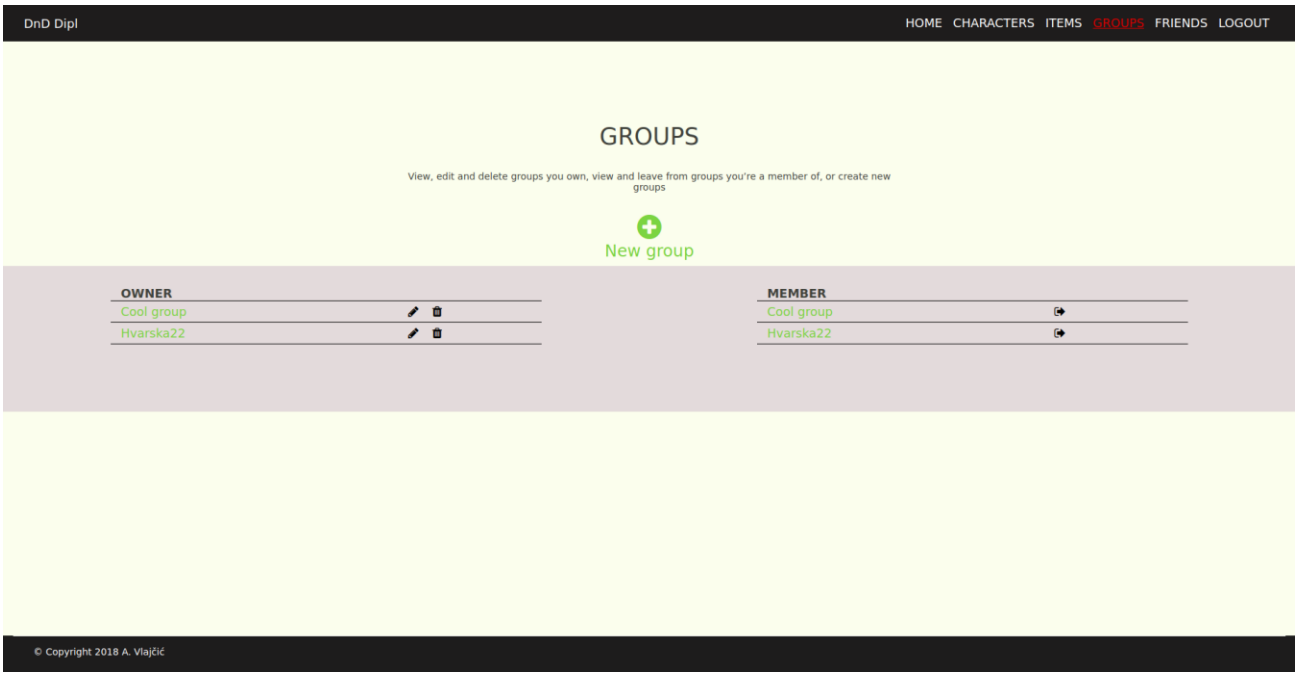


Slika 3.17. Naslovna stranica s top listama predmeta

Osim glasanja za predmete, na pogledu koji prikazuje jedan predmet i njegove informacije, moguće je i ostaviti komentar za taj predmet kako bi ga drugi korisnici mogli vidjeti. Za glasanje i komentiranje je potrebno biti prijavljen u aplikaciju, ali svatko može vidjeti informacije o predmetu poput imena i opisa, kao i autora tog predmeta.

3.3.3. Grupe

Grupe su jedna od najbitnijih i najvećih mogućnosti u aplikaciji. Sama bit igre *Dungeons and Dragons* je u tome da se igra u društvu i zbog toga postoje grupe. Korisnik može napraviti vlastitu grupu u koju će dodati odabrane prijatelje te preko nje ima pristup svim potrebnim informacijama. Kao i za likove i predmete, za grupe su također omogućene CRUD (*create, read, update, delete*) funkcije. No, osim što korisnik može napraviti i upravljati vlastitim grupama, može i biti član grupe nekog drugog korisnika.



Slika 3.18. Stranica za prikaz grupa korisnika

Unutar pregleda za pojedinačnu grupu, korisniku je omogućeno nekoliko stvari. Jedna od tih je čavrljanje (engl. *chat*). Tako korisnik može u stvarnom vremenu komunicirati s drugim članovima grupe, kao i imati povijest poslanih poruka. Čavrljanje je izrađeno pomoću Firebase alata [6] gdje se koristi njegova baza podataka u stvarnom vremenu kako bi korisnici odmah mogli vidjeti svoje poslane i primljene poruke.

Druga bitna stvar koju korisnik može napraviti unutar grupe je razmjena predmeta i zlatnika s drugim članovima grupe. Razmjena je izrađena na jednostavan i intuitivan način tako da korisnik u svega nekoliko sekundi može razmijeniti predmete njegovog lika s drugim članom u grupi. S tim je olakšan jedan od problema kada se igra igra na papiru. Implementacija te mogućnosti se može vidjeti na slici ispod.

```
router.post('/groups/:groupId/trade', isAuthenticated, function (req, res, next) {
    Promise.all([
        Group.findOne({_id: req.params.groupId}).populate({
            path: 'characters',
            populate: { path: 'user'}
        }).exec(),
        User.findOne({_id: req.body.user}),
        Item.find({_id: req.body.items})
    ]).then(function (doc) {
        let group = doc[0];
        let tradingUser = doc[1];
        let items = doc[2];
        let userCharacter;
        let tradingCharacter;

        for (var i = 0; i < group.characters.length; i++){
            if (group.characters[i]) {
                if (group.characters[i].user.username == req.session.user.username) {
                    userCharacter = group.characters[i];
                }else if (group.characters[i].user.username == tradingUser.username) {
                    tradingCharacter = group.characters[i];
                }
            }
        }

        userCharacter.gold -= req.body.gold;
        userCharacter.silver -= req.body.silver;
        userCharacter.copper -= req.body.copper;
        tradingCharacter.gold += req.body.gold;
        tradingCharacter.silver += req.body.silver;
        tradingCharacter.copper += req.body.copper;
```

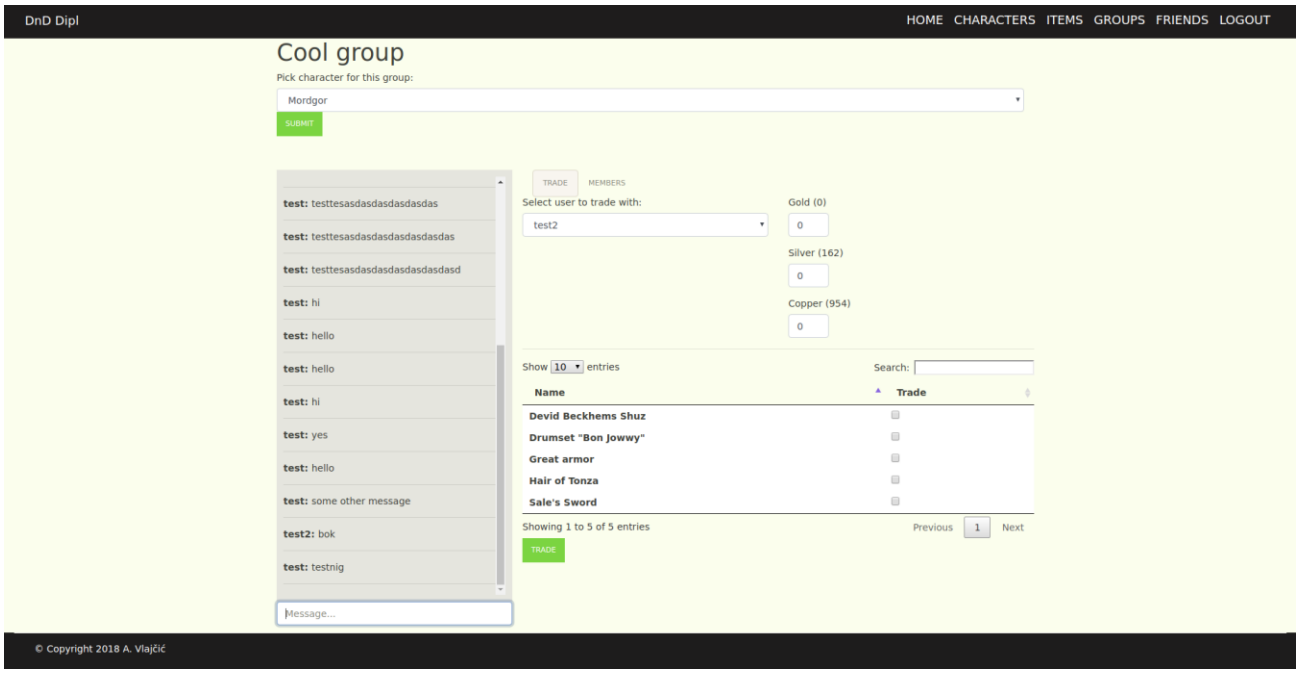
```
        for (let i = 0; i < items.length; i++){
            userCharacter.items.pull(items[i]);
            tradingCharacter.items.push(items[i]);
        }
        Promise.all([
            userCharacter.save(),
            tradingCharacter.save()
        ]).then(function () {
            res.redirect('/users/groups/' + req.params.groupId);
        })
    }).catch(function (err) {

    });

});
```

Slika 3.19. Ruta za razmjenu predmeta

Osim tih mogućnosti, korisnik na pogledu za grupu može vidjeti sve članove grupe, te ako je on vlasnik grupe može izbaciti bilo kojeg člana iz grupe.



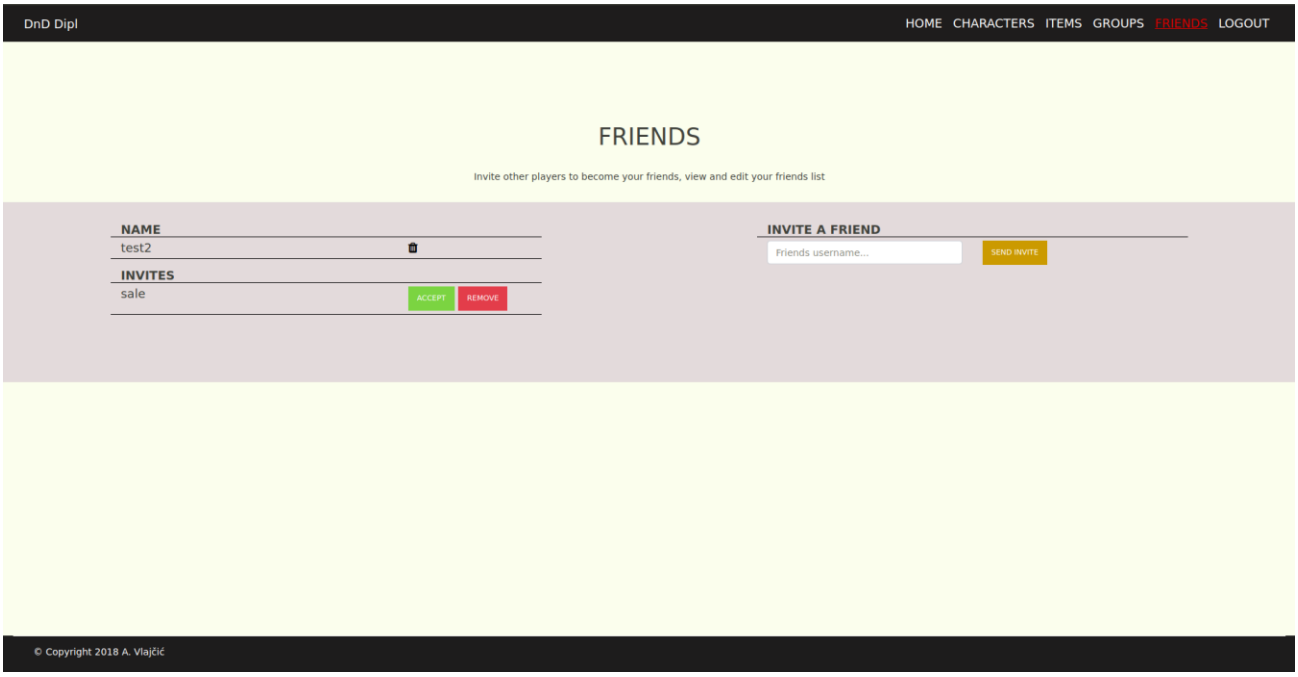
Slika 3.20. Stranica za prikaz pojedinačne grupe

Korisnik u bilo kojem trenutku može izaći iz svojih grupa te u tom slučaju moramo se pobriniti da grupa i dalje postoji ako ima drugi članova u njoj, a korisnik je bio vlasnik grupe. Zbog toga u tom slučaju odabire se neki drugi član grupe da postane novi vlasnik grupe i samim time dobije prava za upravljanja grupom. U slučaju da je korisnik koji je izašao iz grupe jedini aktivni korisnik grupe, tu grupu brišemo.

```
router.get('/groups/:groupId/leave', isAuthenticated, function (req, res, next) {
    Group.findOne({_id: req.params.groupId}).populate('users').exec().then(function (group) {
        var otherUser;
        for (var i= 0; i < group.users.length; i++){
            if (group.users[i]._id !== req.session.user._id){
                otherUser = group.users[i];
            }
        }
        if (otherUser){
            if (group.dm == req.session.user._id){
                group.dm = otherUser;
            }
            if (group.owner == req.session.user._id){
                group.owner = otherUser;
            }
            group.users.pull(req.session.user);
            group.save().then(function () {
                res.redirect('/users/mygroups');
            });
        }else{
            group.remove().then(function () {
                res.redirect('/users/mygroups');
            });
        }
    }).catch(function (err) {
        console.log(err);
    });
});
```

Slika 3.21. Ruta za izlazak iz grupe

3.3.4. Prijatelji



Slika 3.22. Stranica za prikaz liste prijatelja

U sklopu aplikacije korisniku je dostupno dodavanje novih prijatelja kao i upravljanje trenutnom listom prijatelja. Na istom pogledu gdje se nalazi upravljanje trenutnom listom i dodavanje novih prijatelja, nalazi je i pregled svih poziva na prijateljstvo. Tako ako neki drugi korisnik pošalje zahtjev za prijateljstvo, korisnik ima mogućnost potvrditi ili odbiti taj zahtjev.

4. ZAKLJUČAK

U ovom radu je izrađena web aplikacija koja služi kao pomoćni alat prilikom igranja društvene igre *Dungeons and Dragons*. Ona omogućava lakše vođenje likova, njihovih predmeta i bogatstva unutar igre. Dosadašnje igranje i vođenje podataka na papiru se može u potpunosti zamijeniti pomoću ove aplikacije. Osim toga, aplikacija daje mogućnost pravljenja vlastitih predmeta za igru, kao i pregledavanje predmeta drugih korisnika koji mogu poslužiti u vlastitoj igri. Za korištenje mogućnosti aplikacije potrebna je registracija korisnika ili prijava u sustav ako je korisnik već registriran.

Aplikacija je izrađena pomoću NodeJS tehnologije, a za bazu podataka se koristio MongoDB. Ti alati olakšavaju razboj web aplikacija zbog svoje brzine, mogućnosti, fleksibilnosti kao i skalabilnosti. Od alata treće strane (engl. *3rd party tools*) korišten je Firebase alat zbog jednostavnosti primjene i izrade čavrljanja unutar grupe.

Aplikacija se može nadograditi tako da bi se omogućilo kreiranje i upravljanje i nekim drugim stvarima koje se koriste unutar igre poput neprijatelja, klasa ili rasa. Također, moguće je razviti API (aplikacijsko programsko sučelje, engl. *application programming interface*) kako bi se olakšala izrada mobilne aplikacije koja bi imala iste mogućnosti kao i ova web aplikacija.

LITERATURA

[1] About NodeJS, https://nodejs.org/en/about/, 29.6.2018.

[2] What is MongoDb, https://www.mongodb.com/what-is-mongodb, 29.6.2018.

[3] Twig, https://twig.symfony.com/, 29.6.2018.

[4] jQuery, https://jquery.com/, 29.6.2018.

[5] Bootstrap, https://getbootstrap.com/, 29.6.2018.

[6] Firebase, https://firebase.google.com/, 24.8.2018.

SAŽETAK

Web aplikacija izrađena u sklopu ovog rada služi kao pomoć pri igranju društvene igre *Dungeons and Dragons*. Pomoću nje, zamijenjeno je korištenje olovke i papira pri igranju te se sve informacije bitne za igru mogu pronaći unutar aplikacije. Aplikacija je izrađena pomoću alata NodeJS,a za spremanje podataka se koristi MongoDB. Korisnik nakon registracije ima mogućnost kreiranja i upravljanja likovima, predmetima, grupama i prijateljima. Unutar grupe, korisnik ima mogućnosti čavrljanja u stvarnom vremenu s drugim članovima, kao i razmjene predmeta s drugim članovima grupe što uvelike olakšava i ubrzava igranje. Svim korisnicima aplikacije su također vidljivi svi predmeti koje su kreirali drugi korisnici te je omogućeno korištenje, glasovanje i komentiranje istih.

Ključne riječi: web aplikacija, NodeJS, MongoDB, Mongoose, Firebase, Dungeons and Dragons, čavrljanje

ABSTRACT

Web application for Dungeons and Dragons web support

Web application is developed as support for role playing game Dungeons and Dragons. With it, we can replace pen and paper that are usually used for playing the game and we have all important information about our game in the application. Application is developen in NodeJS and MongoDB is used for storing data. After registration, user can create and manage his characters, items, groups and friends. Within groups, user can chat with other members in real time and also can trade items with other members in fast and easy way. All items that are created by users are accessible to all users to use in their game, comment and vote for them.

Key words: web application , NodeJS, MongoDB, Mongoose, Firebase, Dungeons and Dragons, chat

ŽIVOTOPIS

Armando Vlajčić rođen je 8. veljače 1995. godine u Osijeku. 2009. godine upisuje III. Gimnaziju u Osijeku. Tamo sudjeluje na dva županijska natjecanja iz programiranje u kategoriji Algoritmi. Osim toga, bio je član tima koji je dva puta izborio pravo nastupa na međunarodnom natjecanju ACSL. 2013. godine završava srednju školu te upisuje preddiplomski studiji računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku kojeg završava 2016. godine. Iste godine upisuje diplomski studiji računarstva na istom fakultetu.

Kao student prve godine diplomskog studija, u svibnju 2017. godine, počinje raditi u tvrtki UHP Digital u Osijeku gdje je i obavio stručnu praksu u sklopu diplomskog studija.