

**SVEUČILIŠTE U ZAGREBU**  
**FAKULTET ORGANIZACIJE I INFORMATIKE**  
**VARAŽDIN**

**Marijan Kovač**

# **PLATFORMA ZA DISTRIBUCIJU VIDEOIGARA**

**PROJEKT IZ KOLEGIJA BAZE PODATAKA 2**

**Varaždin, 2022.**

**SVEUČILIŠTE U ZAGREBU**  
**FAKULTET ORGANIZACIJE I INFORMATIKE**  
**V A R A Ž D I N**

**Marijan Kovač**

**JMBAG: 0016142261**

**Studij: Informacijski sustavi**

# **PLATFORMA ZA DISTRIBUCIJU VIDEOIGARA**

**PROJEKT IZ KOLEGIJA BAZE PODATAKA 2**

**Mentor:**

Prof. dr. sc. Kornelije Rabuzin

**Varaždin, siječanj 2022.**

# Sadržaj

1. Uvod.....	1
2. Opis aplikacijske domene .....	2
3. Prikaz i razrada ERA modela .....	3
3.1. Okidači.....	5
3.2. Upiti.....	6
4. Opis korištenih alata i tehnologija.....	8
4.1. MongoDB .....	8
4.1.1. MongoDB Atlas.....	9
4.1.2. Povezivanje MongoDB Atlasa s aplikacijom.....	9
4.2. NodeJS.....	10
4.2.1. Priprema NodeJS za rad.....	10
4.2.2. Povezivanje NodeJS aplikacije s MongoDB Atlasom .....	11
4.2.3. Pokretanje aplikacije.....	11
5. Prikaz funkcionalnosti razvijenog sučelja .....	12
5.1. Dobrodošli.....	12
5.2. Registracija korisnika.....	14
5.3. Prijava korisnika .....	15
6. Izvorni kodovi cjelokupnog projekta.....	16
6.1. Aplikacija server.js .....	16
6.2. Modeli .....	22
6.3. Web-stranice.....	26
6.3.1. Dobrodošli.....	26
6.3.2. Prijava .....	28
6.3.3. Registracija.....	29
7. Zaključak .....	30
8. Popis literature .....	31

# 1. Uvod

U ovom će se seminarskom radu prikazati rad s MongoDB bazom podataka, jednom od vodećih NoSQL baza podataka, na primjeru izrade modela platforme za distribuciju videoigara.

Kroz rad će se opisati aplikacijska domena i izrađeni ERA model, a potom će se opisati izrada same baze podataka, odnosno sustava za upravljanje bazom podataka te izrada sučelja u obliku web-formi koje će upravljati određenim segmentima baze podataka.

S obzirom da je MongoDB nerelacijski tip baze podataka, da bi se postigle relacijske sheme, za sam razvoj baze podataka, sustava za upravljanje bazom podataka i razvoj sučelja, potrebno je koristiti mnogobrojne dodatne programske jezike i alate, kao što su Mongoose, JavaScript, Node.js i EJS.

Motivacija za odabir MongoDB baze podataka za izradu ovoga seminarskog rada proizašla je iz činjenice da zbog svoje nerelacijske prirode nudi veću fleksibilnost. Osim toga, zbog srodnosti sa JavaScript Object Notation (JSON) formatom, a samim time i programskim jezikom JavaScript, omogućava lako upravljanje i ugradnju, kao i izradu različitih vrsta aplikacija, od kojih prvenstveno web-aplikacija.

## 2. Opis aplikacijske domene

Aplikacijska domena koja će se u ovom projektu raditi odnosi se na platformu za distribuciju videoigara. Jedna od najpopularnijih takvih platformi svakako je Steam, koji je prvi put izdan 2003. godine. Za potrebe ovog seminarskog rada, prikazat će se simulacija njegove baze podataka, u obliku *third-party* web stranice pod nazivom SteamDB, koja bi služila informiranju drugih korisnika o postojećim korisnicima, njihovim postignućima, videoigrama te kolekcijama videoigara koje posjeduju.

U nastavku slijedi popis entiteta aplikacijske domene:

- Korisnik
- Recenzija
- Videoigra
- Žanr
- Izdavač
- Platforma

Osim popisanih entiteta, ključnu ulogu ima entitet **korisnik\_videoigra** koji uz entitet **Videoigra** predstavlja središte aplikacijske domene. Taj je entitet tipa veze više na više između entiteta Korisnik i Videoigra, a predstavlja poveznicu između dviju logičkih cjelina: dio kojim upravlja korisnik (registracija, prijava, recenzija i slično), te dio kojim upravlja administrator baze (dodavanje nove videoigre, žanrova, izdavača, platformi i slično). Dakle, on omogućuje korisniku da doda novu videoigru u svoju kolekciju, bilo kupnjom ili posuđivanjem.

Nadalje, od dodatnih entiteta, pojavljuju se **korisnik\_status** koji služi za prikupljanje dodatnih podataka o korisniku, točnije o njegovom korištenju same platforme, zatim **je\_igrao** koji služi za izdvajanje onih igara iz kolekcije (entiteta **korisnik\_videoigra**) koje je korisnik igrao barem jednom, te **videoigra\_platforma** koji predstavlja tip veze više na više između entiteta Videoigra i Platforma.

### 3. Prikaz i razrada ERA modela

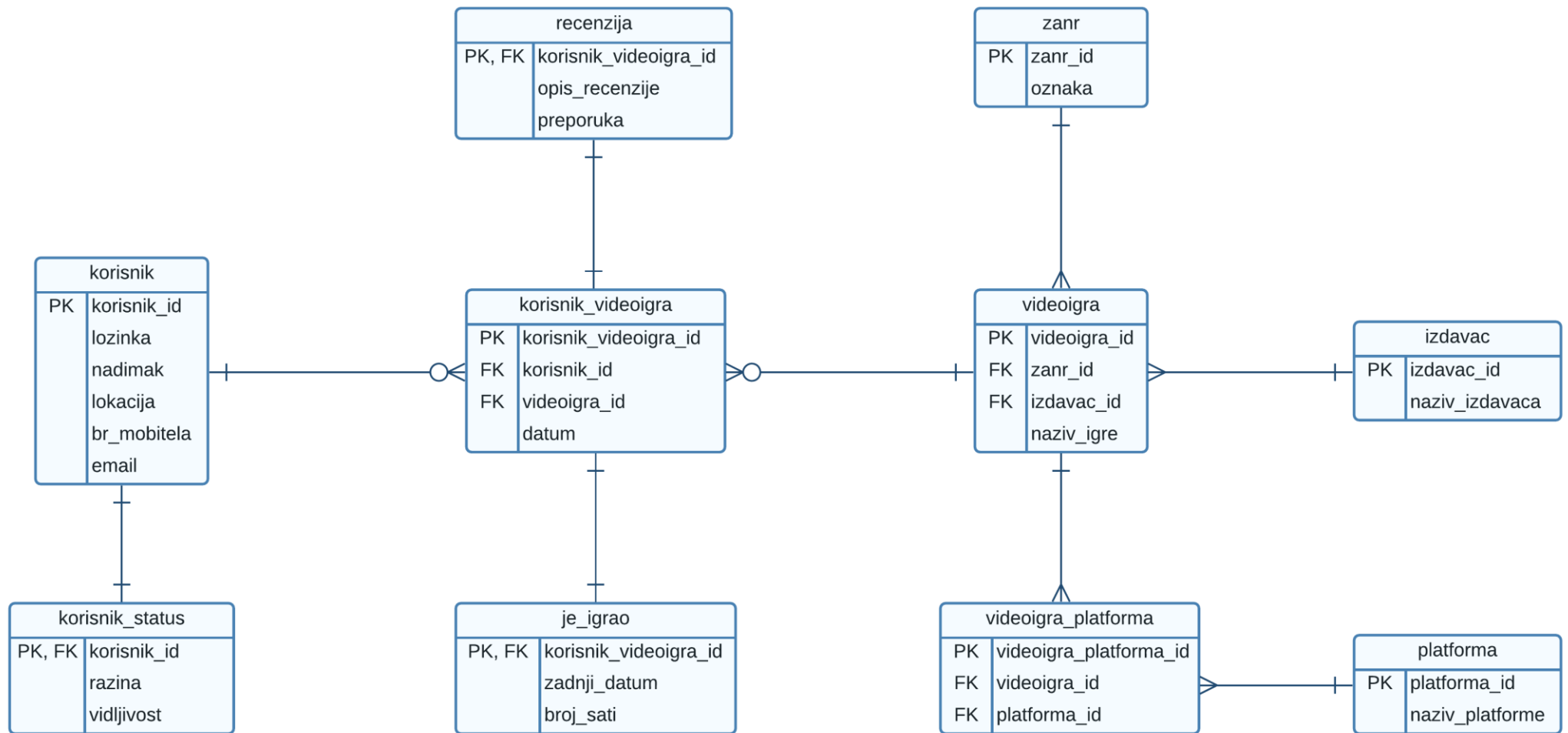
U ovom ćemo dijelu seminarskog rada opisati ERA model platforme za distribuciju videoigara SteamDB (stranica 4). On se sastoji od ukupno 10 entiteta, od kojih njih šest predstavljaju samu okosnicu aplikacije, dok ostali služe kao pomoćni entiteti.

Odnosi među entitetima postignuti su različitim tipovima veza, pa tako imamo tri binarne veze jedan naprema jedan, dvije veze jedan naprema više, te tri veze više naprema više. Dva su entiteta povezana s ukupno četiri tablice tako da oni pripadaju u veze višeg stupnja i oni predstavljaju svojevrsno čvorište aplikacijske domene, dok su ostale veze binarne.

Što se tiče opcionalnosti veza, uglavnom se koriste obavezne veze, osim kod veze više na više između Korisnika i Videoigre koja je opcionalna. Naime, korisnik se može registrirati na platformu, ali nikada ne mora ništa kupiti niti dodati u svoju kolekciju videoigara.

U nastavku ćemo prikazati relacijski model baze podataka za Platformu za distribuciju videoigara „SteamDB“, koji se izrađuje na temelju ERA modela:

- Korisnik (korisnik\_id, lozinka, nadimak, lokacija, br\_mobitela, email)
- Korisnik\_Status (korisnik\_id, razina, vidljivost)
- Korisnik\_Videoigra (korisnik\_videoigra\_id, korisnik\_id, videoigra\_id, datum\_dodavanja)
- Videoigra (videoigra\_id, zanr\_id, izdavac\_id, naziv\_igre)
- Recenzija (korisnik\_videoigra\_id, opis\_recenzije, preporuka)
- Je\_Igrao (korisnik\_videoigra\_id, zadnji\_datum, broj\_sati)
- Žanr (zanr\_id, oznaka)
- Izdavač (izdavac\_id, naziv\_izdavaca)
- Videoigra\_Platforma (videoigra\_platforma\_id, videoigra\_id, platforma\_id)
- Platforma (platforma\_id, naziv\_platforme)



Slika 1: ERA model (vlastita izrada, 2022.)

### 3.1. Okidači

Okidači su, za potrebe ovog projekta, kreirani na razini same aplikacije. Naime, MongoDB sam po sebi nema mogućnost stvaranja okidača, jer nema sustav za upravljanje bazom podataka, već njime upravlja aplikacija (više o tome će biti u nastavku ovoga seminara). Ovdje ćemo sada samo pokazati kako izgledaju primjeri okidača za dani ERA model.

Prvi okidač služi tome da se automatski prilikom registracije novog korisnika postave zadane vrijednosti u pomoćnom entitetu **korisnik\_status**.

```
korisnik.forEach(element => {  
  Korisnik_Status.create({korisnik_id: element.korisnik_id, razina: 0,  
vidljivost: "PRIVATNO"})  
  .then(korisnik_status => {console.log("> Dodana je nova veza korisnik-  
status korisnika!\n", korisnik_status)})  
  .catch(err => {console.log("Greška u dodavanju veze korisnik-status  
korisnika: ", err)})  
});
```

Drugi je okidač nešto složeniji. On osigurava da kada se korisnik pokuša registrirati ne unese već postojeće korisničko ime ili email u bazi.

```
Korisnik.findOne({$or: [{korisnik_id: korisnicko_ime}, {email: email}]})  
  .then(korisnik => {  
    if(korisnik){  
      req.flash("message", "Već je registriran korisnik s istim  
korisničkim imenom ili e-mail računom!");  
      res.redirect("/registracija");  
    }  
    else{  
      Korisnik.create({korisnik_id: korisnicko_ime, lozinka: lozinka,  
nadimak: nadimak, lokacija: lokacija, br_mobitela: br_mobitela, email:  
email})  
      .then(korisnik => { console.log("> Podaci o korisniku spremljeni!  
\n", korisnik)  
        Korisnik_Status.create({korisnik_id: korisnicko_ime, razina: 0,  
vidljivost: "privatno"})  
        .then(korisnik_status => { console.log("> Podaci o statusu  
korisnika spremljeni! \n", korisnik_status)  
          res.redirect("/prijava") })  
        .catch(err => {  
          console.error("> Greska u spremanju podataka: ", err)  
          req.flash("message", "Greška u spremanju podataka u  
bazu!");  
          res.redirect("/registracija")  
        })  
      })  
    }  
    .catch(err => {  
      console.error("> Greska u spremanju podataka: ", err)  
      req.flash("message", "Greška u spremanju podataka u bazu!");  
      res.redirect("/registracija")  
    })  
  });  
})
```



## 3.2. Upiti

Što se tiče upita, ovdje ćemo pokazati primjere jednog jednostavnog i jednog složenog upita, koji se koriste u samoj web-aplikaciji korištenjem alata Mongoose. Za razliku od SQL baze podataka, u NoSQL bazama koriste se mehanizmi agregacije za dohvaćanje rezultata upita.

Prvi upit odnosi se na dio s prijavom korisnika, gdje se mora provjeriti poklapaju li se korisničko ime i lozinka s podacima u bazi.

```
const korisnicko_ime = req.body.korisnicko_ime;
const lozinka = req.body.lozinka;

Korisnik.findOne({korisnik_id: korisnicko_ime})
  .then(korisnik => {
    if(korisnik){
      if(korisnik.lozinka === lozinka){
        res.redirect("/portfolio")
      }
      else{
        req.flash("message", "Netočna lozinka!");
        res.redirect("/prijava");
      }
    }
    else{
      req.flash("message", "Korisnik ne postoji!");
      res.redirect("/prijava");
    }
  })
  .catch(err => {
    req.flash("message", err);
    res.redirect("/prijava")
  })
```

Drugi upit je složeniji, i on se odnosi na ispis podataka o korisnikovim recenzijama, za čije je dohvaćanje potrebno koristiti ukupno četiri entiteta.

```
Korisnik_Videoigra.aggregate([
  {
    $lookup:{
      from: "korisnik",
      localField: "korisnik_id",
      foreignField: "korisnik_id",
      as: "podaci_korisnik"
    }
  },
  {
    $unwind: "$podaci_korisnik"
  },
  {
    $lookup:{
      from: "videoigra",
      localField: "videoigra_id",
      foreignField: "videoigra_id",
      as: "podaci_videoigra"
    }
  },
  {
    $unwind: "$podaci_videoigra"
  },
  {
    $lookup:{
      from: "recenzija",
      localField: "korisnik_videoigra_id",
      foreignField: "korisnik_videoigra_id",
      as: "podaci_recenzija"
    }
  },
  {
    $unwind: "$podaci_recenzija"
  },
  {
    $project:{
      nadimak: "$podaci_korisnik.nadimak",
      naziv_igre: "$podaci_videoigra.naziv_igre",
      opis_recenzije: "$podaci_recenzija.opis_recenzije",
      preporuka: "$podaci_recenzija.preporuka"
    }
  }
],
function(err, rezultat){
  res.render("pages/dobrodosli",{
    lista_rezultat: rezultat
  })
})
```

## 4. Opis korištenih alata i tehnologija

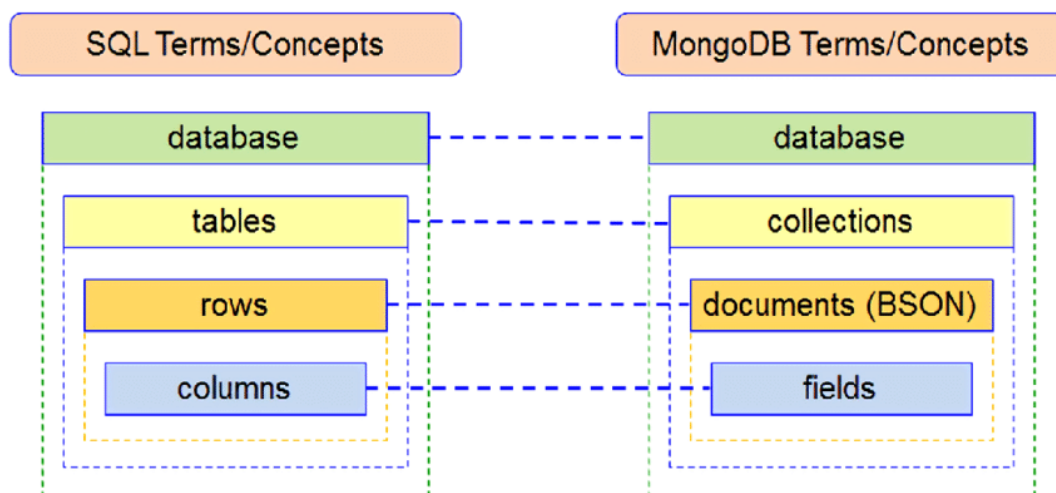
Kao što je već ranije spomenuto, za rad s MongoDB bazom podataka potrebno je koristiti više različitih alata. U ovom ćemo dijelu najprije reći nešto o samoj MongoDB bazi, a zatim i o svim korištenim alatima koji služe za modeliranje i administraciju baze podataka.

### 4.1. MongoDB

MongoDB je program otvorenog koda orijentiran na dokumente koje nalikuju na JSON format i klasificiran je kao NoSQL jezik koji nužno ne koristi sheme. Razvijen je od strane tvrtke MongoDB Inc. 11. veljače 2009. godine. [1]

Iako relativno mlad program, koristi se na mnogobrojnim popularnim web stranicama, zbog toga što nije strog kod strukture podataka. Podaci se u MongoDB uglavnom grupiraju zajedno, što znači da se ne raščlanjuju na povezane dokumente i tablice kao što je slučaj kod SQL baza podataka. [2]

Osim toga, MongoDB koristi i različite termine i koncepte za razliku od SQL baza podataka. Naime, s obzirom da je riječ o nerelacijskoj bazi podataka, u radu s MongoDB bazom podataka ne postoji pojam „tablica“, već se koristi pojam „kolekcija“, a isto tako ono što bi u SQL bazi bio redak, to je u MongoDB bazi „dokument“. Dodatna je razlika što MongoDB za svaki dokument automatski dodjeljuje ID, koji se može, ali i ne mora koristiti kao primarni ključ. Nadalje, „stupci“ u SQL bazi podataka u MongoDB bazi podataka mogli bi se poistovjetiti sa „poljima“. [3]



Slika 2: Razlika između SQL i MongoDB baze podataka

### 4.1.1. MongoDB Atlas

Kako bismo ostvarili vezu s aplikacijom, koristit ćemo MongoDB Atlas. MongoDB Atlas koristi oblak za povezivanje i upravljanje bazom podataka putem popularnih davatelja usluga kao što su Amazon Web Services (AWS), Azure i Google Computing Services (GCP).

Postupak prijave na MongoDB Atlas odvija se u četiri koraka:

- Izrada MongoDB Cloud računa
- Izrada MongoDB Atlas klastera
- Konfiguracija pristupa mreže i korisnika klastera
- Povezivanje s klasterom

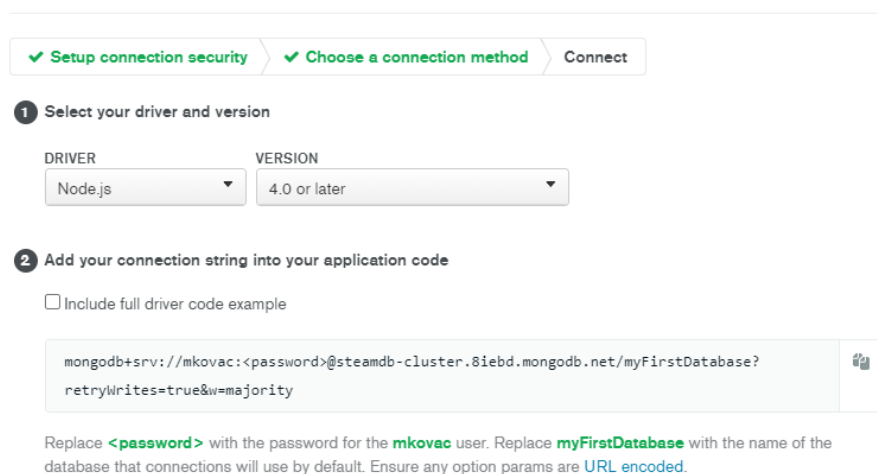
### 4.1.2. Povezivanje MongoDB Atlasa s aplikacijom

Kako bismo povezali aplikaciju s MongoDB atlasom, potrebno je odabrati opciju „Poveži“ nakon čega nam se izlistavaju tri opcije:

- Povezivanje s MongoDB Shellom
- Povezivanje s aplikacijom
- Povezivanje s MongoDB Compassom

U ovom slučaju odabire se opcija povezivanja s aplikacijom. Nakon toga, bira se upravljački program koji će se koristiti. Za potrebe ovog projekta koristit će se Node.js, koji će detaljnije biti opisan u nastavku. Konačno, generira se „string“ kojim se upravljački program povezuje na MongoDB Atlas (Slika 3).

Connect to steamdb-cluster



Connect to steamdb-cluster

✓ Setup connection security ✓ Choose a connection method Connect

1 Select your driver and version

DRIVER VERSION

Node.js 4.0 or later

2 Add your connection string into your application code

☐ Include full driver code example

mongodb+srv://mkovac:<password>@steamdb-cluster.8iebd.mongodb.net/myFirstDatabase?retryWrites=true&w=majority

Replace <password> with the password for the mkovac user. Replace myFirstDatabase with the name of the database that connections will use by default. Ensure any option params are URL encoded.

Slika 3: Povezivanje upravljačkog programa s MongoDB Atlasom

## 4.2. NodeJS

NodeJS je „back-end“ JavaScript okruženje razvijeno za više platformi kako bi se omogućilo izvršavanje koda izvan web-preglednika, a namijenjeno je prvenstveno za izradu aplikacija koje koriste web-sučelje. Razvijeno je nad V8 upraviteljem visokih performansi koje koriste brojni web-preglednici, između kojih i Google Chrome. [4] [5]

### 4.2.1. Priprema NodeJS za rad

Da bismo mogli u potpunosti koristiti sve mogućnosti NodeJS okruženja, potrebno je instalirati sve potrebne programske pakete. Oni se instaliraju pomoću npm-a (akronim za „Node Package Manager“, upravitelja paketa za programski jezik JavaScript, koji se u pravilu instalira zajedno s istim. [6])

Za instaliranje npm upravitelja paketa, potrebno se pozicionirati u direktorij projekta. U nastavku slijedi popis paketa koji se u projektu koristi:

- nodemon (za pokretanje lokalnog servera)
- express (okvir za web-aplikaciju)
- mongoose (alat za upravljanje MongoDB bazom podataka)
- body-parser (okruženje za dohvaćanje podataka iz web-stranice)
- ejs (alat za generiranje HTML predložaka)
- morgan (alat za ispis izvješća web-stranice)
- flash (alat za slanje poruka prema HTML-u)

Nakon uspješne instalacije, svi programski paketi nalaze se u direktoriju „node\_modules“.

## 4.2.2. Povezivanje NodeJS aplikacije s MongoDB Atlasom

Da bismo uspostavili poveznicu između aplikacije i MongoDB baze podataka, koristi se „string“ koji kopiramo iz MongoDB Atlasa (slika 3).

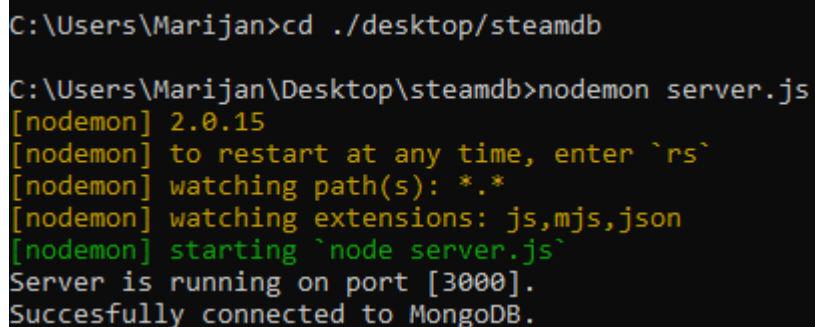
```
mongoose.connect("mongodb+srv://mkovac:nsLbzzPjJgBM9Sx@steamdb-  
cluster.8iebd.mongodb.net/steamdb",  
  {useNewUrlParser: true, useUnifiedTopology: true})  
.then(() => console.log("Succesfully connected to MongoDB."))  
.catch(err => console.error("Connection error: ", err));
```

## 4.2.3. Pokretanje aplikacije

Nakon izrade kompletnog koda aplikacije, najprije je potrebno je navesti port na koji želimo povezati aplikaciju. To vršimo na sljedeći način:

```
const port = 3000;  
app.listen(port, function(){  
  console.log("Server is running on port [%d].", port);  
})
```

Nakon toga, da bismo pokrenuli aplikaciju, u naredbenom retku se pozicioniramo u direktorij projekta te koristimo programski paket *nodemon* za pokretanje aplikacije:



```
C:\Users\Marijan>cd ./desktop/steamdb  
C:\Users\Marijan\Desktop\steamdb>nodemon server.js  
[nodemon] 2.0.15  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,json  
[nodemon] starting `node server.js`  
Server is running on port [3000].  
Succesfully connected to MongoDB.
```

Slika 4: Pokretanje aplikacije

## 5. Prikaz funkcionalnosti razvijenog sučelja

### 5.1. Dobrodošli



Slika 5: Funkcionalnost početne web-stranice „Dobrodošli“

```
app.get("/", function(req, res){
  Korisnik_Videoigra.aggregate([
    {
      $lookup:{
        from: "korisnik",
        localField: "korisnik_id",
        foreignField: "korisnik_id",
        as: "podaci_korisnik"
      }
    },
    {
      $unwind: "$podaci_korisnik"
    },
    {
      $lookup:{
        from: "videoigra",
        localField: "videoigra_id",
        foreignField: "videoigra_id",
        as: "podaci_videoigra"
      }
    },
    {
      $unwind: "$podaci_videoigra"
    },
    {
      $lookup:{
        from: "recenzija",
        localField: "korisnik_videoigra_id",
        foreignField: "korisnik_videoigra_id",
        as: "podaci_recenzija"
      }
    }
  ])
```

```

    },
    {
      $unwind: "$podaci_recenzija"
    },

    {
      $project:{
        nadimak: "$podaci_korisnik.nadimak",
        naziv_igre: "$podaci_videoigra.naziv_igre",
        opis_recenzije: "$podaci_recenzija.opis_recenzije",
        preporuka: "$podaci_recenzija.preporuka"
      }
    }
  ],
  function(err, rezultat){
    res.render("pages/dobrodosli",{
      lista_rezultat: rezultat
    })
  })
})

```



## 5.2. Registracija korisnika

Registriraj se na SteamDB

Korisničko ime:  
pero\_djetlic

Lozinka:  
pero123

Nadimak:  
Djetlic

Lokacija:  
Velika Gorica, Hrvatska

Broj mobitela:  
099123456789

E-mail:  
pero\_djetlic@gajba.com

REGISTRACIJA Greška: Već je registriran korisnik s istim korisničkim imenom ili e-mail računom!

Slika 6: Funkcionalnost stranice za registraciju

```
app.get("/registracija", function(req, res){
  const message = req.flash("message");
  res.render("pages/registracija", {message})
})

app.post("/registracija", async (req, res) => {
  const korisnicko_ime = req.body.korisnicko_ime
  const lozinka = req.body.lozinka
  const nadimak = req.body.nadimak
  const lokacija = req.body.lokacija
  const br_mobitela = req.body.br_mobitela
  const email = req.body.email

  Korisnik.findOne({$or: [{korisnik_id: korisnicko_ime}, {email: email}]})
  .then(korisnik => {
    if(korisnik){
      req.flash("message", "Već je registriran korisnik s istim
korisničkim imenom ili e-mail računom!");
      res.redirect("/registracija");
    }
    else{
      Korisnik.create({korisnik_id: korisnicko_ime, lozinka: lozinka,
nadimak: nadimak, lokacija: lokacija, br_mobitela: br_mobitela, email:
email})
      .then(korisnik => {
        console.log("> Podaci o korisniku spremljeni! \n", korisnik)

        Korisnik_Status.create({korisnik_id: korisnicko_ime, razina: 0,
vidljivost: "privatno"})
        .then(korisnik_status => {
          console.log("> Podaci o statusu korisnika spremljeni! \n",
korisnik_status)
          res.redirect("/prijava")
        })
      })
      .catch(err => {
```

```

        console.error("> Greska u spremanju podataka: ", err)
        req.flash("message", "Greška u spremanju podataka u
bazu!");
        res.redirect("/registracija")
    })
})
.catch(err => {
    console.error("> Greska u spremanju podataka: ", err)
    req.flash("message", "Greška u spremanju podataka u bazu!");
    res.redirect("/registracija")
});
}
})

```

### 5.3. Prijava korisnika

Prijava se na SteamDB

Korisničko ime:  
pero\_djetlic

Lozinka:  
987654321

PRIJAVA Greška: Netočna lozinka!

Slika 7: Funkcionalnost stranice za prijavu

```

app.get("/prijava", function(req, res){
    const message = req.flash("message");
    res.render("pages/prijava", {message})
})

app.post("/prijava", async (req, res) => {
    const korisnicko_ime = req.body.korisnicko_ime;
    const lozinka = req.body.lozinka;

    Korisnik.findOne({korisnik_id: korisnicko_ime})
    .then(korisnik => {
        if(korisnik){
            if(korisnik.lozinka === lozinka){
                res.redirect("/portfolio")
            }
            else{
                req.flash("message", "Netočna lozinka!");
                res.redirect("/prijava");
            }
        }
        else{
            req.flash("message", "Korisnik ne postoji!");
            res.redirect("/prijava");
        }
    })
    .catch(err => {
        req.flash("message", err);
        res.redirect("/prijava")
    })
})

```

## 6. Izvorni kodovi cjelokupnog projekta

### 6.1. Aplikacija server.js

```
const mongoose = require("mongoose");
const express = require("express");
const session = require("express-session");
const flash = require("connect-flash");
const morgan = require("morgan");
const app = express();
const bodyParser = require("body-parser");

app.use(session({
  secret: "steamdb",
  saveUninitialized: true,
  resave: true
}));

app.use(bodyParser.urlencoded({extended: true}));
app.use(bodyParser.json());
app.use(express.json());
app.use(morgan("combined"));
app.use(flash());

app.set("view engine", "ejs");

mongoose.connect("mongodb+srv://mkovac:nsLbzzPjJgBM9Sx@steamdb-
cluster.8iebd.mongodb.net/steamdb",
  {useNewUrlParser: true, useUnifiedTopology: true})
  .then(() => console.log("Succesfully connected to MongoDB."))
  .catch(err => console.error("Connection error: ", err));

const Korisnik = require("./models/korisnik")
const Korisnik_Status = require("./models/korisnik_status")
const Korisnik_Videoigra = require("./models/korisnik_videoigra")
const Recenzija = require("./models/recenzija")
const Je_Igrao = require("./models/je_igrao")
const Videoigra = require("./models/videoigra")
const Zanr = require("./models/zanr")
const Izdovac = require("./models/izdovac")
const Videoigra_Platforma = require("./models/videoigra_platforma")
const Platforma = require("./models/platforma");
const { findOne, insertMany } = require("./models/korisnik");

//popunjavanje tablica

Zanr.insertMany([
  {zanr_id: 1, oznaka: "Sandbox"},
  {zanr_id: 2, oznaka: "Shooter"},
  {zanr_id: 3, oznaka: "Action"},
  {zanr_id: 4, oznaka: "MOBA"},
  {zanr_id: 5, oznaka: "Simulation"}
])
.then(zanr => {
  console.log("> Dodani su novi žanrovi.\n", zanr)
```

```

Izdavac.insertMany([
  {izdavac_id: 1, naziv_izdavaca: "SCS Software"},
  {izdavac_id: 2, naziv_izdavaca: "Rockstar Games"},
  {izdavac_id: 3, naziv_izdavaca: "Activision"},
  {izdavac_id: 4, naziv_izdavaca: "Mojang"},
  {izdavac_id: 5, naziv_izdavaca: "Riot Games"}
])
.then(izdavac => {
  console.log("> Dodani su novi izdavači.\n", izdavac)

  Videoigra.insertMany([
    {videoigra_id: 1, zanr_id: 4, izdavac_id: 5, naziv_igre:
"League of Legends"},
    {videoigra_id: 2, zanr_id: 3, izdavac_id: 2, naziv_igre: "Grand
Theft Auto V"},
    {videoigra_id: 3, zanr_id: 5, izdavac_id: 1, naziv_igre: "Euro
Truck Simulator 2"},
    {videoigra_id: 4, zanr_id: 1, izdavac_id: 4, naziv_igre:
"Minecraft"},
    {videoigra_id: 5, zanr_id: 2, izdavac_id: 3, naziv_igre: "Call
of Duty: Black Ops 3"},
    {videoigra_id: 6, zanr_id: 4, izdavac_id: 5, naziv_igre: "Call
of Duty: World at War"}
  ])
  .then(videoigra => console.log("> Dodane su nove videoigre!\n",
videoigra))
  .catch(err => console.log("Greška u dodavanju videoigara: ", err))
})
.catch(err => console.log("Greška u dodavanju izdavaca: ", err))
})
.catch(err => console.log("Greška u dodavanju žanrova: ", err))

Platforma.insertMany([
  {platforma_id: 1, naziv_platforme: "PC"},
  {platforma_id: 2, naziv_platforme: "Playstation 4"},
  {platforma_id: 3, naziv_platforme: "XBOX 360"}
])
.then(platforma => {
  console.log("> Dodane su nove platforme!\n", platforma)

  Videoigra_Platforma.insertMany([
    {videoigra_platforma_id: 1, videoigra_id: 1, platforma_id: 1},
    {videoigra_platforma_id: 2, videoigra_id: 2, platforma_id: 1},
    {videoigra_platforma_id: 3, videoigra_id: 2, platforma_id: 2},
    {videoigra_platforma_id: 4, videoigra_id: 2, platforma_id: 3},
    {videoigra_platforma_id: 5, videoigra_id: 3, platforma_id: 2},
    {videoigra_platforma_id: 6, videoigra_id: 4, platforma_id: 1},
    {videoigra_platforma_id: 7, videoigra_id: 5, platforma_id: 1},
    {videoigra_platforma_id: 8, videoigra_id: 5, platforma_id: 2},
    {videoigra_platforma_id: 9, videoigra_id: 5, platforma_id: 3},
    {videoigra_platforma_id: 10, videoigra_id: 6, platforma_id: 1}
  ])
  .then(videoigra_platforma => console.log("> Dodane su nove veze
videoigra-platforma!\n", videoigra_platforma))
  .catch(err => console.log("Greška u dodavanju veza videoigra-platforma:
", err))
})
.catch(err => console.log("Greška u dodavanju platformi: ", err))

Korisnik.insertMany([

```

```

        {korisnik_id: "pero_djetlic", lozinka: "pero123", nadimak: "Djetlic",
lokacija: "Velika Gorica, Hrvatska", br_mobitela: null, email:
"pero_djetlic@gajba.com"},
        {korisnik_id: "njiha_njiha", lozinka: "aloba123", nadimak: "Njiha",
lokacija: "Široki brijeg, Bosna i Hercegovina", br_mobitela:
"+38599847271452", email: "njiha@bhtel.com"},
        {korisnik_id: "znjkf", lozinka: "lovehell321", nadimak: "Znjkf",
lokacija: null, br_mobitela: null, email: "znjkf@hotmail.com"}
    ])
    .then(korisnik => {
        console.log("> Dodani su novi korisnici!\n", korisnik)

        korisnik.forEach(element => {
            Korisnik_Status.create({korisnik_id: element.korisnik_id, razina: 0,
vidljivost: "PRIVATNO"})
            .then(korisnik_status => {
                console.log("> Dodana je nova veza korisnik-status korisnika!\n",
korisnik_status)
            })
            .catch(err => {console.log("Greška u dodavanju veze korisnik-status
korisnika: ", err)})
        });
    })
    .catch(err => {console.log("Greška u dodavanju novih korisnika: ", err)})

    Korisnik_Videoigra.insertMany([
        {korisnik_videoigra_id: 1, korisnik_id: "pero_djetlic", videoigra_id:
1, datum_dodavanja: Date.now()},
        {korisnik_videoigra_id: 2, korisnik_id: "pero_djetlic", videoigra_id:
4, datum_dodavanja: Date.now()},
        {korisnik_videoigra_id: 3, korisnik_id: "njiha_njiha", videoigra_id: 2,
datum_dodavanja: Date.now()},
        {korisnik_videoigra_id: 4, korisnik_id: "znjkf", videoigra_id: 1,
datum_dodavanja: Date.now()},
        {korisnik_videoigra_id: 5, korisnik_id: "znjkf", videoigra_id: 2,
datum_dodavanja: Date.now()},
        {korisnik_videoigra_id: 6, korisnik_id: "znjkf", videoigra_id: 3,
datum_dodavanja: Date.now()},
        {korisnik_videoigra_id: 7, korisnik_id: "znjkf", videoigra_id: 4,
datum_dodavanja: Date.now()},
        {korisnik_videoigra_id: 8, korisnik_id: "znjkf", videoigra_id: 5,
datum_dodavanja: Date.now()},
        {korisnik_videoigra_id: 9, korisnik_id: "znjkf", videoigra_id: 6,
datum_dodavanja: Date.now()}
    ])
    .then(korisnik_videoigra => {
        console.log("> Dodane su nove veze korisnik-videoigra! \n",
korisnik_videoigra)

        Recenzija.insertMany([
            {korisnik_videoigra_id: 1, opis_recenzije: "Nemojte igrat ovu
igricu.", preporuka: NE},
            {korisnik_videoigra_id: 2, opis_recenzije: "Najbolja igrica ikad
napravljen.", preporuka: DA},
            {korisnik_videoigra_id: 3, opis_recenzije: "FPS dropovi, ne valja
ništa.", preporuka: NE},
            {korisnik_videoigra_id: 5, opis_recenzije: "Money money money",
preporuka: DA},
            {korisnik_videoigra_id: 8, opis_recenzije: "Cheaters.", preporuka:
NE}
        ])
    })

```

```

.then(recenzija => {
  console.log("> Dodane su nove recenzije! ", recenzija)

  Je_Igrao.insertMany([
    {korisnik_videoigra_id: 1, zadnji_datum: Date.now(), broj_sati:
1954},
    {korisnik_videoigra_id: 2, zadnji_datum: Date.now(), broj_sati:
2068},
    {korisnik_videoigra_id: 3, zadnji_datum: Date.now(), broj_sati:
14},
    {korisnik_videoigra_id: 5, zadnji_datum: Date.now(), broj_sati:
558},
    {korisnik_videoigra_id: 8, zadnji_datum: Date.now(), broj_sati:
3},
  ])
  .then(je_igrao => {
    console.log("> Dodane su nove veze videoigre korisnika-je
igrao! ", je_igrao)
  })
  .catch(err => {
    console.log("Greška u dodavanju nove veze videoigre korisnika-
je igrao: ", err)
  })
  })
  .catch(err => {
    console.log("Greška u dodavanju novih recenzija: ", err)
  })
})
.catch(err => {
  console.log("Greška u dodavanju novih veza korisnik-videoigra: ", err)
})

//web-stranica

app.get("/", function(req, res){
  Korisnik_Videoigra.aggregate([
    {
      $lookup:{
        from: "korisnik",
        localField: "korisnik_id",
        foreignField: "korisnik_id",
        as: "podaci_korisnik"
      }
    },
    {
      $unwind: "$podaci_korisnik"
    },
    {
      $lookup:{
        from: "videoigra",
        localField: "videoigra_id",
        foreignField: "videoigra_id",
        as: "podaci_videoigra"
      }
    },
    {
      $unwind: "$podaci_videoigra"
    },
    {
      $lookup:{
        from: "recenzija",

```

```

        localField: "korisnik_videoigra_id",
        foreignField: "korisnik_videoigra_id",
        as: "podaci_recenzija"
    }
},
{
    $unwind: "$podaci_recenzija"
},
{
    $project:{
        nadimak: "$podaci_korisnik.nadimak",
        naziv_igre: "$podaci_videoigra.naziv_igre",
        opis_recenzije: "$podaci_recenzija.opis_recenzije",
        preporuka: "$podaci_recenzija.preporuka"
    }
}
],
function(err, rezultat){
    res.render("pages/dobrodosli",{
        lista_rezultat: rezultat
    })
})
})

app.get("/registracija", function(req, res){
    const message = req.flash("message");
    res.render("pages/registracija", {message})
})

app.post("/registracija", async (req, res) => {
    const korisnicko_ime = req.body.korisnicko_ime
    const lozinka = req.body.lozinka
    const nadimak = req.body.nadimak
    const lokacija = req.body.lokacija
    const br_mobitela = req.body.br_mobitela
    const email = req.body.email

    Korisnik.findOne({$or: [{korisnik_id: korisnicko_ime}, {email: email}]})
    .then(korisnik => {
        if(korisnik){
            req.flash("message", "Već je registriran korisnik s istim
korisničkim imenom ili e-mail računom!");
            res.redirect("/registracija");
        }
        else{
            Korisnik.create({korisnik_id: korisnicko_ime, lozinka: lozinka,
nadimak: nadimak, lokacija: lokacija, br_mobitela: br_mobitela, email:
email})
            .then(korisnik => {
                console.log("> Podaci o korisniku spremljeni! \n", korisnik)

                Korisnik_Status.create({korisnik_id: korisnicko_ime, razina: 0,
vidljivost: "privatno"})
                .then(korisnik_status => {
                    console.log("> Podaci o statusu korisnika spremljeni! \n",
korisnik_status)
                    res.redirect("/prijava")
                })
            })
            .catch(err => {
                console.error("> Greska u spremanju podataka: ", err)
            })
        }
    })
})

```

```

        req.flash("message", "Greška u spremanju podataka u
bazu!");
        res.redirect("/registracija")
    })
    })
    .catch(err => {
        console.error("> Greska u spremanju podataka: ", err)
        req.flash("message", "Greška u spremanju podataka u bazu!");
        res.redirect("/registracija")
    });
}
})
})

app.get("/prijava", function(req, res){
    const message = req.flash("message");
    res.render("pages/prijava", {message})
})

app.post("/prijava", async (req, res) => {
    const korisnicko_ime = req.body.korisnicko_ime;
    const lozinka = req.body.lozinka;

    Korisnik.findOne({korisnik_id: korisnicko_ime})
    .then(korisnik => {
        if(korisnik){
            if(korisnik.lozinka === lozinka){
                res.redirect("/portfolio")
            }
            else{
                req.flash("message", "Netočna lozinka!");
                res.redirect("/prijava");
            }
        }
        else{
            req.flash("message", "Korisnik ne postoji!");
            res.redirect("/prijava");
        }
    })
    .catch(err => {
        req.flash("message", err);
        res.redirect("/prijava")
    })
})

app.get("/portfolio", function(req, res){
    res.render("pages/portfolio")
})

const port = 3000;
app.listen(port, function(){
    console.log("Server is running on port [%d].", port);
})

```



## 6.2. Modeli

```
const Izdavac_Schema = new mongoose.Schema({
  izdavac_id: {
    type: Number,
    required: true,
    unique: true
  },
  naziv_izdavaca: {
    type: String,
    required: true,
    unique: true
  }
},{versionKey: false}
)

const Izdavac = mongoose.model(
  "Izdavac",
  Izdavac_Schema,
  "izdavac"
)

const Je_Igrao_Schema = new mongoose.Schema({
  korisnik_videoigra_id: {
    type: Number,
    ref: "Korisnik_Videoigra.korisnik_videoigra_id",
    required: true,
    unique: true
  },
  zadnji_datum: {
    type: Date
  },
  broj_sati: {
    type: Number
  }
},{versionKey: false}
)

const Je_Igrao = mongoose.model(
  "Je_Igrao",
  Je_Igrao_Schema,
  "je_igrado"
)

const Korisnik_Status_Schema = new mongoose.Schema({
  korisnik_id: {
    type: String,
    ref: "Korisnik.korisnik_id",
    unique: true,
    required: true
  },
  razina: {
    type: Number,
    required: true
  },
  vidljivost: {
    type: String,
    enum: ["PRIVATNO", "SAMO-PRIJATELJI", "JAVNO"],
    required: true, uppercase: true, default: "PRIVATNO"
  }
})
```

```

    }
  }, {versionKey: false}
)

const Korisnik_Status = mongoose.model(
  "Korisnik_Status",
  Korisnik_Status_Schema,
  "korisnik_status"
)

const Korisnik_Videoigra_Schema = new mongoose.Schema({
  korisnik_videoigra_id: {
    type: Number,
    required: true,
    unique: true
  },
  korisnik_id: {
    type: String,
    ref: "Korisnik.korisnik_id",
    required: true
  },
  videoigra_id: {
    type: Number,
    ref: "Videoigra.videoigra_id",
    required: true
  },
  datum_dodavanja: {
    type: Date,
    required: true
  }
}, {versionKey: false},
)

Korisnik_Videoigra_Schema.index({korisnik_id: 1, videoigra_id: 1},
  {unique: true})

const Korisnik_Videoigra = mongoose.model(
  "Korisnik_Videoigra",
  Korisnik_Videoigra_Schema,
  "korisnik_videoigra"
)

const Korisnik_Schema = new mongoose.Schema({
  korisnik_id: {
    type: String,
    unique: true,
    required: true
  },
  lozinka: {
    type: String,
    required: true
  },
  nadimak: {
    type: String,
    required: true
  },
  lokacija: {
    type: String,
    required: false
  },
  br_mobitela: {

```

```

        type: String,
        unique: true,
        required: false
    },
    email: {
        type: String,
        unique: true,
        required: true
    },
}, {versionKey: false}
)

const Korisnik = mongoose.model(
    "Korisnik",
    Korisnik_Schema,
    "korisnik"
)

const Platforma_Schema = new mongoose.Schema({
    platforma_id: {
        type: Number,
        required: true,
        unique: true
    },
    naziv_platforme: {
        type: String,
        required: true,
        unique: true
    }
}, {versionKey: false}
)

const Platforma = mongoose.model(
    "Platforma",
    Platforma_Schema,
    "platforma"
)

const Recenzija_Schema = new mongoose.Schema({
    korisnik_videoigra_id: {
        type: Number,
        ref: "Korisnik_Videoigra.korisnik_videoigra_id",
        required: true,
        unique: true
    },
    opis_recenzije: {
        type: String,
        required: true
    },
    preporuka: {
        type: String,
        enum: ["DA", "NE"],
        required: true,
        uppercase: true
    }
}, {versionKey: false}
)

const Recenzija = mongoose.model(
    "Recenzija",
    Recenzija_Schema,

```

```

    "recenzija"
  )

const Videoigra_Platforma_Schema = new mongoose.Schema({
  videoigra_platforma_id: {
    type: Number,
    required: true,
    unique: true
  },
  videoigra_id: {
    type: Number,
    ref: "Videoigra.videoigra_id",
    required: true
  },
  platforma_id: {
    type: Number,
    ref: "Platforma.platforma_id",
    required: true
  }
}, {versionKey: false})

Videoigra_Platforma_Schema.index({videoigra_id: 1, platforma_id: 1},
  {unique: false})

const Videoigra_Platforma = mongoose.model(
  "Videoigra_Platforma",
  Videoigra_Platforma_Schema,
  "videoigra_platforma"
)

const Videoigra_Schema = new mongoose.Schema({
  videoigra_id: {
    type: Number,
    required: true,
    unique: true
  },
  zanr_id: {
    type: Number,
    ref: "Zanr.zanr_id",
    required: true
  },
  izdovac_id: {
    type: Number,
    ref: "Izdavac.izdovac_id",
    required: true
  },
  naziv_igre: {
    type: String,
    required: true,
    unique: true
  }
}, {versionKey: false})

Videoigra_Schema.index({zanr_id: 1, izdovac_id: 1},
  {unique: false})

const Videoigra = mongoose.model(
  "Videoigra",
  Videoigra_Schema,

```

```

    "videoigra"
  )

const Zanr_Schema = new mongoose.Schema({
  zanr_id: {
    type: Number,
    required: true,
    unique: true
  },
  oznaka: {
    type: String,
    required: true,
    unique: true
  }
}, {versionKey: false})

const Zanr = mongoose.model(
  "Zanr",
  Zanr_Schema,
  "zanr"
)

```

## 6.3. Web-stranice

### 6.3.1. Dobrodošli

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dobrodošli - SteamDB</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css
" integrity="sha384-
BVYiISIFeKldGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
  <style>
    .buttons{
      position: relative;
      text-align: center;
      margin-top: 20px;
      padding-top: 20px;
    }
  </style>
</head>
<body style="background-image: url('https://wallpaperbat.com/img/184256-
forest-desktop-wallpaper.jpg');">
  <h1 style="color: white ; font-family: 'Segoe UI', Tahoma, Geneva,
Verdana, sans-serif; text-align: center; margin-top: 25; margin-bottom:
0px; padding-top: 0px; padding-bottom: 0px;">Dobrodošli na SteamDB</h1>
  <h1 style="color: white ; font-family: 'Segoe UI', Tahoma, Geneva,
Verdana, sans-serif; text-align: center; margin-top: -25px; margin-bottom:
0px; padding-top: 0px; padding-bottom: 5px;">_____</h1>

```

```

<h3 style="color: white ; font-family: 'Segoe UI', Tahoma, Geneva,
Verdana, sans-serif; font-size: 20px ;text-align: center; margin-top: 0;
margin-bottom: 5px; padding-top: 5px; padding-bottom: 25px;">Što kažu
korisnici o našoj platformi?</h3>
<div>
  <table class="table">
    <thead>
      <tr>
        <th scope="col"> </th>
        <th scope="col"> </th>
        <th scope="col"> </th>
        <th scope="col">Korisnik</th>
        <th scope="col">Videoigra</th>
        <th scope="col">Komentar</th>
        <th scope="col">Preporuka</th>
        <th scope="col"> </th>
        <th scope="col"> </th>
        <th scope="col"> </th>
      </tr>
    </thead>
    <tbody>
      <%lista_rezultat.forEach(rezultat => {%)>
        <tr>
          <td> </td>
          <td> </td>
          <td> </td>
          <td><%= rezultat.nadimak %></td>
          <td><%= rezultat.naziv_igre %></td>
          <td><%= rezultat.opis_recenzije %></td>
          <td><%= rezultat.preporuka %></td>
          <td> </td>
          <td> </td>
          <td> </td>
        </tr>
      <%)>
    </tbody>
  </table>
</div>

<div class="inline">
  <form action="/prijava" class="buttons">
    <button class="submit-button" ><span style="font-family: 'Segoe
UI', Tahoma, Geneva, Verdana, sans-serif; ">Prijava</span></button>
  </form>
  <form action="/registracija" class="buttons">
    <button class="submit-button" ><span style="font-family: 'Segoe
UI', Tahoma, Geneva, Verdana, sans-serif; ">Registracija</span></button>
  </form>
</div>

</body>
</html>

```

## 6.3.2. Prijava

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Prijava - SteamDB</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css
" integrity="sha384-
BVYiISIFeKldGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
</head>
<body style="background-image: url('https://wallpaperbat.com/img/184256-
forest-desktop-wallpaper.jpg');">
  <h1 style="color: white ; font-family: 'Segoe UI', Tahoma, Geneva,
Verdana, sans-serif; text-align: center; margin-top: 0; margin-bottom:
15px; padding-top: 25px; padding-bottom: 25px;">Prijava se na SteamDB</h1>
  <form class="container" method="post" action="/prijava" id="prijava">
    <div class="form-group">
      <text style="color:white; font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif; font-size: 20px;">Korisničko ime: </text>
<input class="form-control" name="korisnicko_ime" required >
      <text style="color: white; font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif; font-size: 20px;">Lozinka: </text> <input
class="form-control" name="lozinka" required>

    </div>

    <div class="inline">
      <button> <span style="font-family: 'Segoe UI', Tahoma, Geneva,
Verdana, sans-serif; font-size: 20px;">PRIJAVA</span></button>
      <% if(message != '') { %>
        <text style="margin-left: 10px; color:indianred; font-
family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; font-size: 20px;">
Greška: <%= message %> </text>
      <% } %>
    </div>

  </form>

</body>
</html>
```

### 6.3.3. Registracija

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registracija - SteamDB</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css
" integrity="sha384-
BVYiISIFeKldGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
</head>
<body style="background-image: url('https://wallpaperbat.com/img/184256-
forest-desktop-wallpaper.jpg');">
  <h1 style="color: white ; font-family: 'Segoe UI', Tahoma, Geneva,
Verdana, sans-serif; text-align: center; margin-top: 0; margin-bottom:
15px; padding-top: 25px; padding-bottom: 25px;">Registriraj se na
SteamDB</h1>
  <form class="container" method="post" action="/registracija">
    <div class="form-group">
      <text style="color:white; font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif; font-size: 20px;">Korisničko ime: </text>
<input class="form-control" name="korisnicko_ime" required>
      <text style="color:white; font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif; font-size: 20px;">Lozinka: </text> <input
class="form-control" name="lozinka" required>
      <text style="color:white; font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif; font-size: 20px;">Nadimak: </text> <input
class="form-control" name="nadimak" required>
      <text style="color:white; font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif; font-size: 20px;">Lokacija: </text> <input
class="form-control" name="lokacija" required>
      <text style="color:white; font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif; font-size: 20px;">Broj mobitela: </text>
<input class="form-control" name="br_mobitela" required>
      <text style="color:white; font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif; font-size: 20px;">E-mail: </text> <input
class="form-control" name="email" required>
    </div>

    <div class="inline">
      <button> <span style="font-family: 'Segoe UI', Tahoma, Geneva,
Verdana, sans-serif; font-size: 20px;">REGISTRACIJA</span></button>
      <% if(message != '') { %>
        <text style="margin-left: 10px; color:indianred; font-
family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; font-size: 20px;">
Greška: <%= message %> </text>
      <% } %>
    </div>
  </form>
</body>
</html>
```



## 7. Zaključak

U izradi ovoga projekta pokazali smo kako možemo koristiti MongoDB kao NoSQL (nerelacijsku) bazu podataka te unatoč tome stvoriti relacije i formate (modele) za naše potrebe. Možemo zaključiti kako se MongoDB zbog svoje nestrukturiranosti lako integrira u programske jezike, a najbolje se slaže s JavaScriptom, što je pokazano i kroz ovaj projekt.

Za izradu samog projekta bilo je potrebno izdvojiti puno vremena, što je ponajviše rezultat mogega slabog do nikakvog znanja programskog jezika JavaScript. Ipak, zadovoljan sam što sam odabrao baš ove alate i bazu podataka za izradu ovoga projekta, jer su mi poslužili kao vrlo dobra priprema za nadolazeće kolegije.

## 8. Popis literature

- [1] <https://en.wikipedia.org/wiki/MongoDB>, pristupano 14.1.2022.
- [2] <https://hr.unedose.fr/article/how-a-mongodb-database-can-better-organize-your-data>, pristupano 14.1.2022.
- [3] <https://www.mongodb.com/nosql-explained/nosql-vs-sql>, pristupano 14.1.2022.
- [4] <https://en.wikipedia.org/wiki/Node.js>, pristupano 14.1.2022.
- [5] <https://v8.dev/>, pristupano 14.1.2022.
- [6] [https://en.wikipedia.org/wiki/Npm\\_\(software\)](https://en.wikipedia.org/wiki/Npm_(software)), pristupano 14.1.2022.