- Izraditi RESTful API koristeći Node.js i Express
- Postaviti Node i Express okruženje
- REST glagoli
- Representaional State Transfer (REST)

- node –v (postoji li node.js na računalu)
- npm init –y (kreirati prazan folder te u njemu kreirati package.json)
- npm install express (skinuti express-kreira se package.lock.json)
- kreirati praznu datoteku app.js (kod je na idućem slide-u)
- node app.js (sa ovom naredbom pokrenuti prvi node projekt, koji se poslužuje na portu 3000)

```
import express from ‘express’

const app = express()
const port = process.env.PORT || 3000;

app.get('/', (req, res) =>{
    res.send("Welcome to my API!!!");
 })

app.listen(port, ()=>{
console.log("Running on port" + port);
})
```

- npm install eslint –D (za formatiranje koda)
- flag –D označava development dependency-a
- Ubačena je skripta lint (package.json)
- npm run lint -- --init (odgovoriti na par pitanja vezana uz konfiguriranje config file-a od eslint-a)

- nodemon (npm install nodemon)
- nodemon-alat koji restart-a aplikaciju prilikom spremanja izmjenjenog koda (točnije nodemon izmjenjenu aplikaciju iznova build-a i iznova je posluži)
- više pročitati na https://nodemon.io/
- u package.json je ubačen NodemonConfig
- npm install (za provjeru)
- u rijetkim situacijama, kada mislimo da smo sve uredno postavili i da nam u kodu nema grešaka, a linter ili node i dalje prijavljuju greške, može se izbrisati node_modules direktorij i sa naredbom „npm install” sve iznova postaviti
- u package.json je dodana skripta start
- npm start (aplikacija se sada poslužuje na portu 5000)

- **Skinut je Express** -light node.js web framework http://expressjs.com/

- **Potrebno je skinuti MongoDB**
- MongoDB je NoSQL baza podataka koja služi za spremanje podataka u obliku dokumenata. Najčešće dokumenti imaju JSON strukturu.
- mongoDb Compass – je GUI klijent koji se koristi za MongoDB
- link za dowmnoad https://www.mongodb.com/try/download/community

- **Potrebno je skinuti Mongoose -** sa komandom „npm install mongoose”
- Mongoose je ODM (Object Document Mapper)
- Koristeći JS biblioteku mongoose, podaci se modeliraju na jednostavniji način. Definira se schem-om koja dolazi iz paketa mongoose (u primjeru sa predavanja vidjeti models/MovieModel.js). Zbog jednostavnijeg shvaćanja-možete ODM zamisliti kao ORM za NoSQL (nerelacijske baze podataka).
- (https://mongoosejs.com/)

- Povezati se na bazu podataka sa mongoose-om (niže navedena linija u datoteci app.js)
- const db = mongoose.connect('mongodb://localhost/bookAPI')
- Ubaciti prve podatke u bazu podataka (u primjeru sa predavanja pogledati datoteku DataImportInstructons)

- 1. komandna linija: Podizanje MongoDB na svom računalu ("C:\Program Files\MongoDB\Server\5.0\bin\mongod.exe" --dbpath="c:\data\db")
- 2. komandna linija: Pokrenuti projekt sa skriptom start (npm start)

Koristiti jedan terminal za pokretanje servera, a drugi sa pokretanje baze. Oba se izvršavaju istovremeno.

Primjer sa GET glagolom.

- Dohvatiti sve iteme iz baze- Movie.find():

  url se upisuje http://localhost:4000/api/movies

- Dohvatiti (pretražiti) bazu po kriteriju - Movie.find(query, funkcija):

  u url se upisuje http://localhost:4000/api/movies?genre=action

- Dohvatiti samo jedan item po id-u - Movie.findById():
- u url se upisuje: http://localhost:4000/api/movies/_2918379127

- U drugom primjeru je pokazana izvedba POST glagola
- Potrebno je koristiti neki klijent za slanje post, put, patch, delete requestova npr. Postman, Curl, …
- U primjeru 2 kod je isti kao u primjeru 1, samo je dodan POST glagol i kod je podijeljen po node modulima

- Koristi se middleware bodyParser za čitanje podataka iz tijela requesta
- import bodyParser from "body-parser"

  app.use(bodyParser.urlencoded({extended:true}));

  app.use(bodyParser.json());

- Isprobati:
- prvo napraviti GET request sa alatom Postman, pa onda POST
- prilikom POST requesta ubaciti podatke o filmu u body requesta

- Treći primjer sa predavanja je proširen sa glagolima PUT, PATCH i DELETE
- Put-mijenja cijeli item
- Patch – mijenja samo dio item-a (dio koji se promijenio)
- Delete – briše se item
- Middleware functions

- U Primjeru 3, u movieRuter.js datoteci korišten je middleware. Brine se za to, da ukoliko dodje do greške, vrati error response i izvršavanje ne ide dalje u obradu glagola. Primjer je na sljedećem slide-u.

```
movieRouter.use('/movies/:movieId', (req, res, next)=>{
    Movie.findById(req.params.movieId,(err, movie)=>{
        if(err){
            return res.send(err);
        }
        if(movie){
           req.movie = movie;
            return next();
       }
        return res.sendStatus(404);
    });
});
```