STUDENT GRADE TRACKER

Martin Stefanovic REr 8/18

O APLIKACIJI

Projekat Student Grade Tracker omogućava korisniku da doda predmet koji ima na fakultetu /u školi kao i ciljanu ocenu koju očekuje da ima iz tog predmeta. Zatim da za svaki predmet doda obavezu koju treba da izvrši kako bi položio taj predmet, svaka obaveza ima svoj level važnosti (normal, important, critical) kao i naslov i opis obaveze.
Korisnik zatim može da pregleda sve obaveze grupisane po predmetima, da obriše, izmeni i čekira završenu obavezu kao i da doda konačnu ocenu za predmet koji je položio.
Takođe korisniku je omogućeno da prati statistiku trenutnog i ciljanog proseka, statistiku položenih predmeta kao i preostalih obaveza.

Tehnologije

Za kreiranje ove aplikacije korišćene su sledeće tehnologije:

- NodeJS (Express framework)
- MongoDB
- Bootstrap

NodeJS

Node.js je radno okruženje JavaScript programskog jezika koje se koristi na strani web servera. JavaScript se tradicionalno koristi na strani klijenta – tzv. front-end razvoj, a Node je omogućio da se JS koristi i za back-end razvoj odnosno serversko izvršavanje koda.

Node.js ima veliki broj modula – JavaScript biblioteka ili funkcija koje obavljaju određene specifične zadatke i koje možete pridodati svojoj

aplikaciji. Postoje neki ugrađeni moduli, dok su drugi dostupni pre svega putem NPM repozitorijuma – Node Package Manager.

Express

Express, Expressjs ili Express.js je Node.js web framework – na srpskom bi se moglo reći da to znači Node.js okvir. Koristi se za programiranje web aplikacija i API (Aplikacioni programski interfejs). Express olakšava i ubrzava mnoge radnje i spasava nas pisanja suvišnih linija koda prilikom kreiranja Node.js aplikacija. Exspress.js je pozadinski deo MEAN steka, zajedno sa Mongo DB bazom podataka i celokupnim Angular.js okvirom. **MEAN - MongoDB, Express, Angular i Node.js.*

MongoDB

MongoDB pripada grupi tzv. nerelacionih (engl. NoSQL) SUBP, odnosno, same baze podataka koje se kreiraju su nerelacione. Postoji više vrsta nerelacionih baza podataka, a baze podataka u MongoDB SUBP pripadaju vrsti koja se naziva baza dokumenata (engl. document database). To znači da se svi podaci čuvaju u obliku dokumenata. Format u kojem su zapisani ovi dokumenti je sličan formatu JSON objekata. Vrednosti svojstava (nekada kažemo i polja (engl. field)) dokumenta mogu biti: niske, brojevi, drugi dokumenti, nizovi, nizovi drugih dokumenata, itd. Naredna slika ilustruje primer dokumenta koje je sačuvan u nekoj bazi podataka u MongoDB SUBP.

Bootstrap

Bootstrap je front-end biblioteka koja sadrži već kreirane komponente kao i utility klase kreirane uz pomoću HTML-a, CSS-a i JavaScript-a. Bootstrap su razvili developeri u Twitter-u kako bi dobili konzistentnost prilikom kreiranja novih web stranica. On sadrži stilove vezane za forme, modal-e, slajdere, naslove, i još puno toga.

jQuery

jQuery je jednostavna JavaScript biblioteka, čiji je slogan "write less, do more". Doprinosi bržem kreiranju stranice samim tim što sa manje koda možemo da uradimo više kompleksnih stvari nego u „čistom“ JavaScript-u.

Instalacija i podešavanje

Pre svega na računaru moramo imati instaliran NodeJS, kao i MongoDB. Ovo su linkovi za download:

https://nodejs.org/en/download/

https://www.mongodb.com/try/download/community

Zatim u folderu gde želimo da započnemo naš projekat treba da pokrenemo sledeće komande:

```
Npm init -y
//Kako bi inicijalizovali package.json fajl
```

```
Npm i express ejs mongoose
//Kako bi instalirali express i mongoose
```

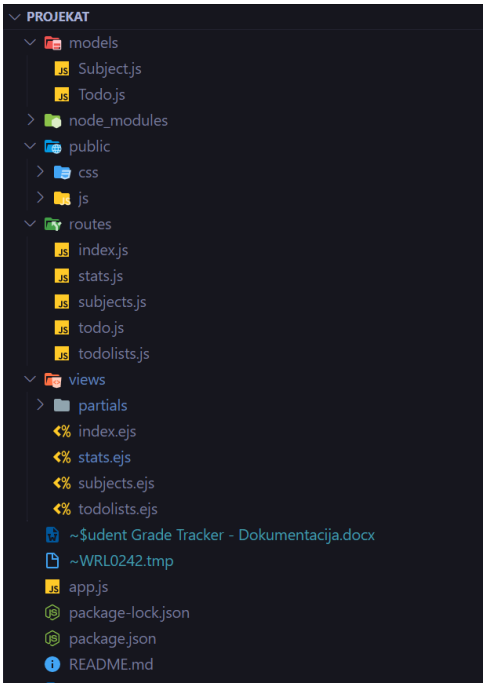
Zatim je potrebno pokrenudi MongoDB server

```
mongod
```

Struktura foldera i fajlova u projektu

Na slici se vidi struktura fajlova koje je potrebno da kreiramo za naš projekat.

- Models – Modeli kao što su predmeti i todo taskovi
- Node_modules – Ovo su paketi koji se dodaju automatski kada pokrenemo npm install
- Public – Ovde ubacujemo css i js fajlove za front-end deo
- Routes – Rute koje imamo u aplikaiciji
- Views – Ovde se nalaze naši .ejs templejti koji sadrže HTML, oni koji se često ponavljaju kao što su header i footer možemo da izdvojimo u poseban folder partials.
- App.js – Glavni fajl aplikacije



Instalacija Bootstrapa

Bootstrap možemo instalirati na više načina, na primer preuzimanjem kompresovanih css i js fajlova sa njihovog sajta ili preko cnd-a tako što u header.ejs templejtu dodamo cdn link:

```
<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">
```

Zatim u footer.ejs dodamo potreban javascipt cdn link:

```
<!-- Bootstrap -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.bundle.min.js" integrity="sha384-ygbV9kiqUc6oa4msXn9868pTtWMgiQaeYH7/t7LECLbyPA2x65Kgf80OJFdroafW" crossorigin="anonymous"></script>
```

NAPOMENA: Kako bi mogli da “pregazimo” stilove koje donosi Bootstrap, custom CSS I custom JS fajl moramo da uključimo posle ovih linkova.

Instalacija jQuery-a

Potrebno je uključiti sledeći cdn link u footer.ejs templejtu:

```
<!-- jQuery -->

<script src="https://code.jquery.com/jquery-3.5.1.min.js" integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0=" crossorigin="anonymous"></script>
```

Rute

Kako bi naša aplikacija radila moramo da definišemo rute za get all, update one, delete one, add one. To ću pokazati na primeru Subjects modela iz aplikacije iako isto ovo imam i za todo,stats,todolists...

```
// ? Default
router.get("/subjects", async (req, res)=>{
    const allSubjects = await Subject.find();
    const page_name = 'subjects';

    res.render("subjects", {subjects: allSubjects, page_name:page_name});
})
```

#Default ruta je inicjalna ruta za prikaz svih predmeta. Tu imamo dve konstante koje uz pomoću funkcije render() prosleđujemo u template. Prva promenjiva je allSubjects koja dobija sve predmete preko funkcije find() (*primenjene na kolekciju Subject)

```
// ? Add subject
router
.post("/subjects/add", (req, res)=>{
    const {goalGrade} = req.body;
    const {subject} = req.body;
    const {grade} = req.body;
    const newSubject = new Subject({goalGrade: goalGrade, subject: subject, grade:grade})

    // Save the subject
    newSubject.save().then(()=>{
        console.log('Uspesno dodavanje predmeta');
        res.redirect("/subjects")
    })
    .catch((err)=>{
        console.log(err);
    })
})
```

#Add subject ruta služi za dodavanje predmeta u bazu. Podatke o predmetu šaljemo kroz POST formu i zatim ih uzimamo preko req.body. Kada preuzmemo podatke uz pomoću new Subject(...) kreiramo nov upis u bazu.

```
// ? Delete subject
.get("/subjects/delete/:_id", (req, res)=>{
    const {_id} = req.params;
    Subject.deleteOne({_id})
    .then(()=>{
        console.log('Uspesno obrisan predmet');
        res.redirect("/subjects")
    })
    .catch(err => {
        console.log(err);
    })
})
```

#Delete subject ruti kao dodatni parametar prosleđujemo ID predmeta koji želimo da obrišemo a zatim taj ID ubacujemo u funkciju deleteOne() koja briše predmet po ID-u. Ova f-ja se takodje primenjuje na Subjects kolekcijom.

```
// ? Update subject
.post('/subjects/update', function(req, res) {
    const {_id} = req.body;
    const {passed} = req.body;
    const {grade} = req.body;
    const {goalGrade} = req.body;
    console.log(_id);
    Subject.updateOne({_id}, {grade: grade, passed: passed, goalGrade: goalGrade}, function(err, res) {
        if (err) throw err;
        console.log("1 document updated");
    });
    res.redirect("/todolists")
});
```

#Update subject ruta služi za update-ovanje jednog predmeta. Kao i kod add rute i ovde podatke uzimamo iz POST forme ali ih ovog puta prosleđujemo u funkciju updateOne(), prve parametar te f-je je ID predmeta koji update-ujemo a drugi su polja kao ključevi i nove vrednosti.

Modeli

Modeli odredjuju kako treba da izgleda Predmet koji upisujemo. Konkretno u ovom slučaju je korišćenja mongooseSchema.

```
const mongoose = require("mongoose");

const SubjectSchema = new mongoose.Schema({
  subjectKey: {
    type: String,
    required: false
  },
  subject: {
    type: String,
    required: false
  },
  grade: {
      type: Number,
      required: false
  },
  goalGrade: {
      type: Number,
      required: false
  },
  passed:{
      type: Boolean,
      required: false
  }
});

module.exports = new mongoose.model("Subject", SubjectSchema);
```

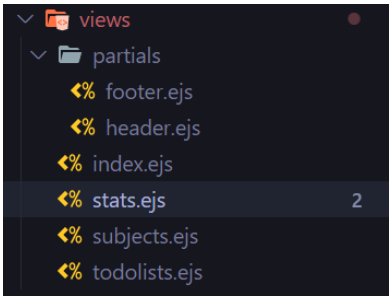
App.js

Sve rute koje dodamo u našu aplikaciju moramo da registrujemo u app.js fajlu koji je glavni fajl naše aplikacije.

```
const express = require("express");
const mongoose = require("mongoose");

const app = express();


// * Connection to mongodb
mongoose.connect("mongodb://localhost/todo_express",
{
    useNewUrlParser: true,
    useUnifiedTopology: true
})

// * Middlewares
app.use(express.urlencoded({extended: true}))
app.use(express.static("public"));
app.set("view engine", "ejs");


// * Routes
app.use(require("./routes/index"))
app.use(require("./routes/todo"))
app.use(require("./routes/subjects"))
app.use(require("./routes/todolists"))
app.use(require("./routes/stats"))

// * Server config
app.listen(3000, () => console.log("Server started listening on port 3000"))
```

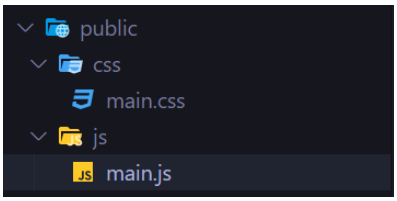
Views

U slučaju kada ne koristimo neki eksterni framework za front-end kao što je na primer Angular, React itd. možemo da koristimo .ejs template.
Evo strukture views fodlera. Partials pod folder sadrži elemente koji se ponavljaju na svakoj stranici a to su header i footer.



Public (front end CSS & JavaScript)

Što se tiče stilizovanja front-end dela sve to smeštamo u folder public u kome imamo podfoldere za CSS i JS.



Kako bih napravio samo jedan modal preko koga mogu da editujem bilo koji predmet, unutar svakog predmeta dodao sam edit dugme sa data-atributima preko kojih sam kada se klikne na dugme upisivao u modal trenutne podatke za predmet koji želim da izmenim. Kako bi koda bio čitljiviji i lakši za pisanje koristio sam jQuery umesto čistog JavaScript-a.

```
// * Update subject on todolists page
let chosenSubject = {
    _id: 0,
    grade: 0,
    goalGrade: 0,
    passed: false
}

let subjectIdField = $("#subjectIdField");
let subjectStatusField = $("#subjectStatus");
let goalGradeField = $("#goalGrade");
let gradeField = $("#grade");

$( ".btn-edit" ).on( "click", function() {
    chosenSubject._id = $(this).data('subject-id');
    chosenSubject.grade = $(this).data('subject-grade');
    chosenSubject.goalGrade = $(this).data('subject-goal-grade');
    chosenSubject.passed = $(this).data('status');

    subjectIdField.attr('value', `${chosenSubject._id}`)
    goalGradeField.attr('value', `${chosenSubject.goalGrade}`)
    gradeField.attr('value', `${chosenSubject.grade}`)
});
```

Sledeća stvar koju sam dodao u aplikaciju jeste dark mode. Imamo glavni objekat darkModeFunc koji ima metodu za enable/disable dark moda. U suštini sve što radi ova funkcija jeste dodavanje ili sklanjanje klase .darkmode sa body taga, kada je klasa dodata za to imam posebno napisana CSS pravila. Da bi se dark mode state sačuvao kada korisnik refrešuje stranicu ili potpuno isključi svoj pretraživač iskoristio sam localStorage kako bih sačuvao info o tome da li je darkMode uključen ili nije.

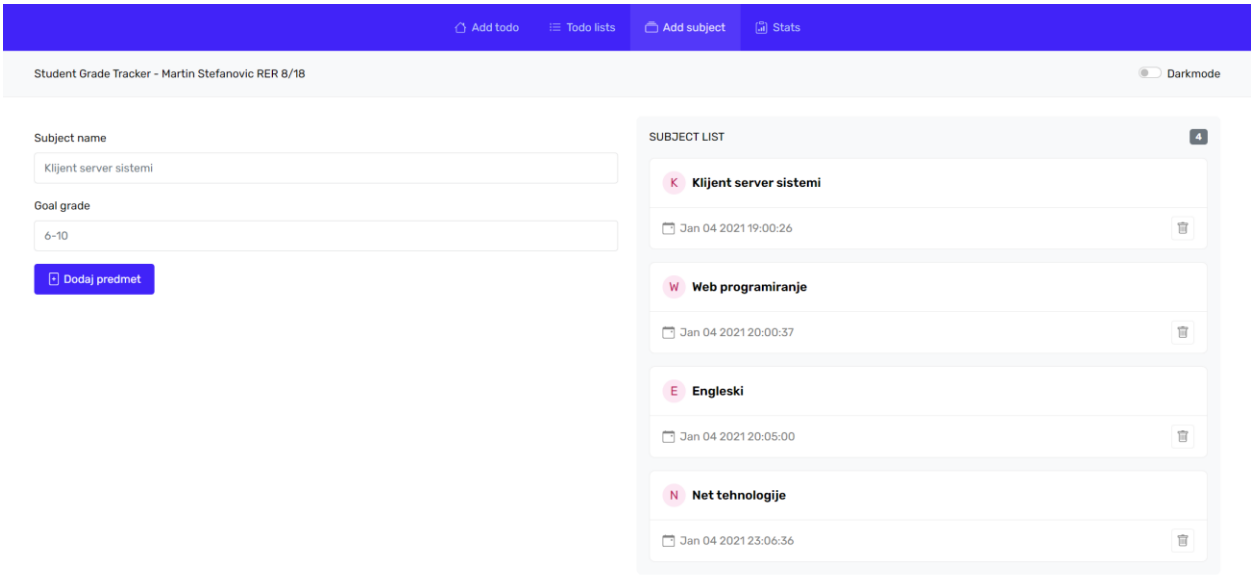
```
// * Darkmode Logic
let darkModeToggleButton = $('#darkmodeToggle');

let darkModeFunc = {
    enable: ()=>{
        darkModeToggleButton.prop('checked', 'checked')
        $('body').addClass('darkmode');
    },
    disable: ()=>{
        $('body').removeClass('darkmode');
    }
}

let darkMode = window.localStorage.getItem('darkmode');

if(darkMode == 'true'){
    darkModeFunc.enable();
}else{
    darkModeFunc.disable();
}

darkModeToggleButton.on('change', function () {
    console.log('klik');
    if($(this).prop('checked')){
        window.localStorage.setItem('darkmode', 'true');
        darkModeFunc.enable();
    }else{
        window.localStorage.setItem('darkmode', 'false');
        darkModeFunc.disable();
    }
})
```

Zaključak

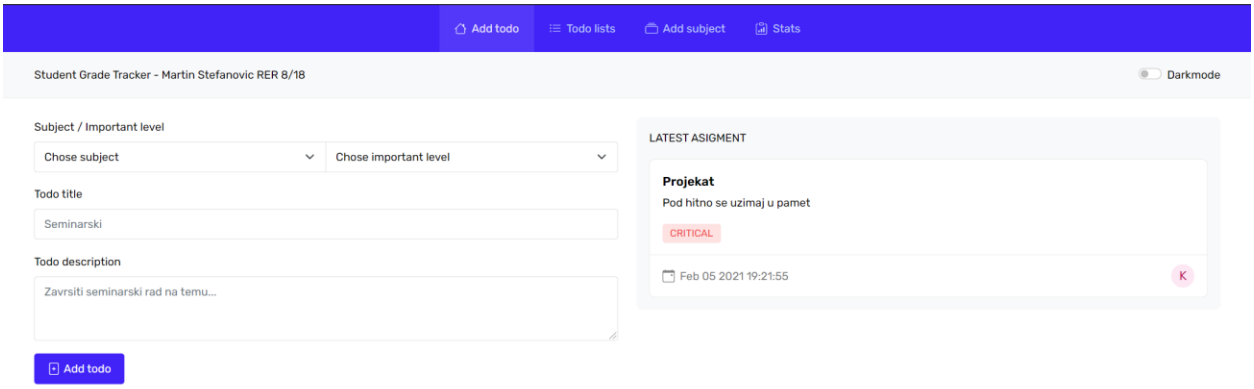
Uz pomoću NodeJS-a moguće je napraviti kako jednostavne tako i veoma složene web aplikacije. Kako bi aplikacija bila lakša za održavanje i skalabilna ja bih za front-end izabrao ipak neki javascipt framework uz pomoću kog bi naša aplikacija mogla da bude single-page aplikacija koja bi radila bez reloada stranice i time poboljšala korisničko iskustvo. Ipak za ovako male projekte nije greška koristiti i template pomoću .ejs.

Izgled aplikacije

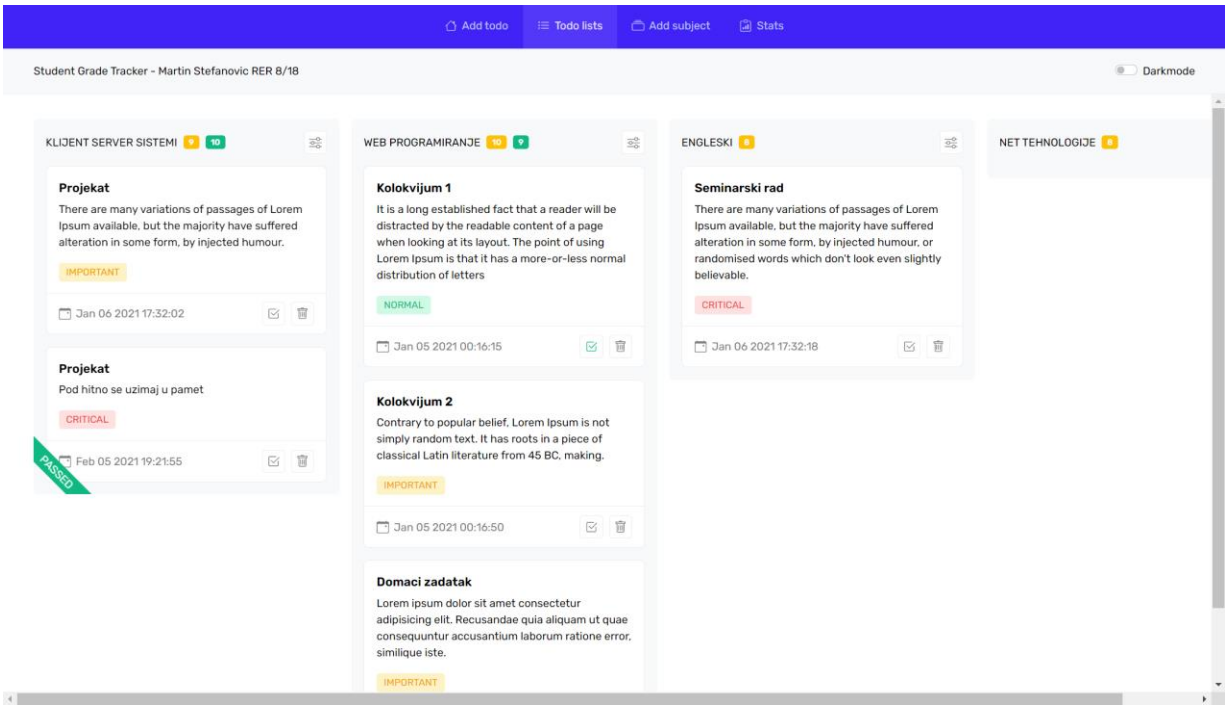
Add subject stranica preko koje dodajemo predemete ali takodje možemo da vidimo i sve dodate predmete:

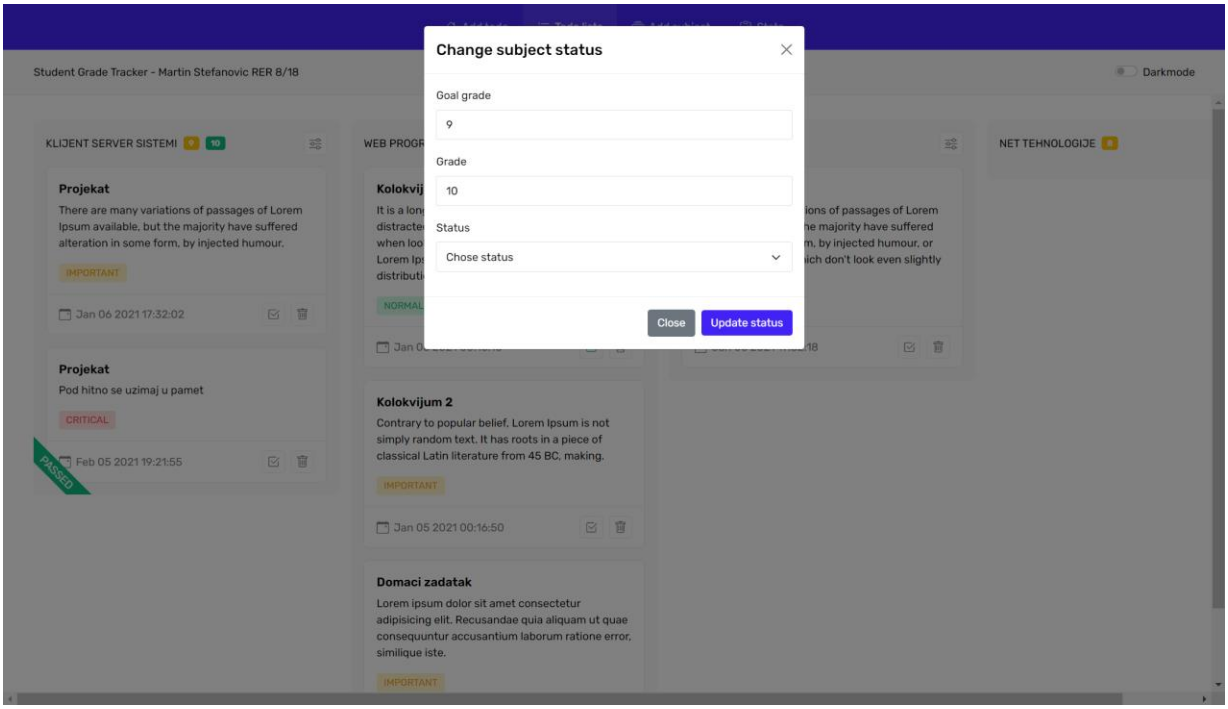


Add todo stranica gde dodajemo obavezu, predet biramo iz padajućeg menija (lista predmeta se update-uje svaki put kada dodamo novi predmet), biramo važnost obaveze I upisujemo naslov I opis. Sa desne strane vidimo poslednje dodatu obavezu.

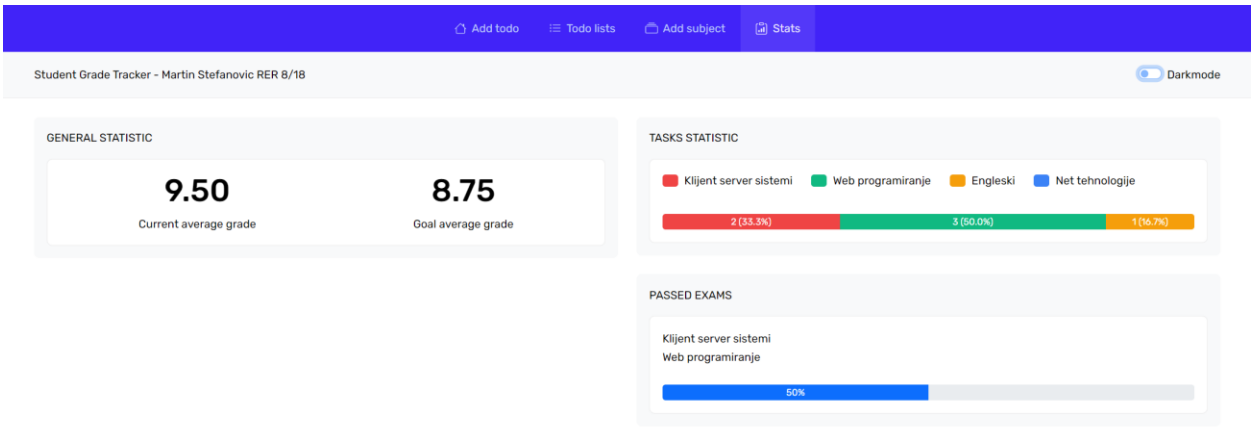


Todo lists stranica služi kako bi videli obaveze koje imamo razvrstane po predmetima, možemo da čekiramo obavezu ili je obrišemo. Imamo pregled željene (žuta boja) i realne (zelena) ocene, moguće je i ažuriranje predmeta što može da se vidi na ekranu ispod. Ako je predmet obeležen kao položen imamo Zeleni bedž sa natpisom PASSED.

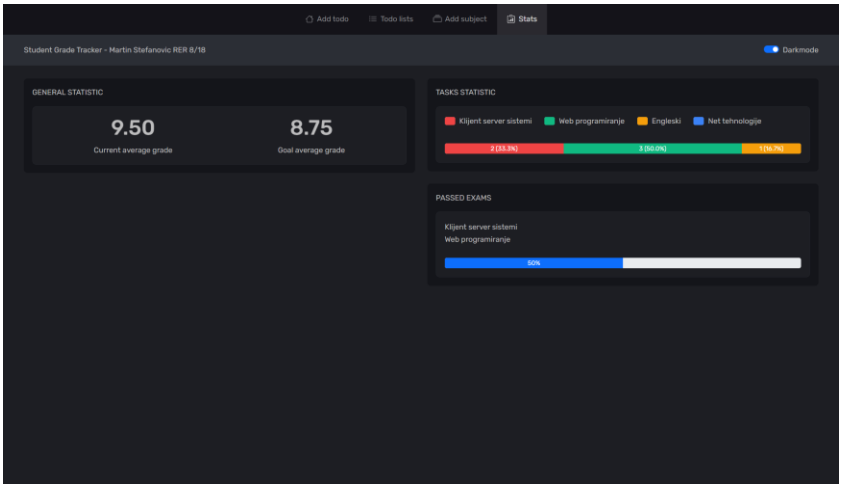


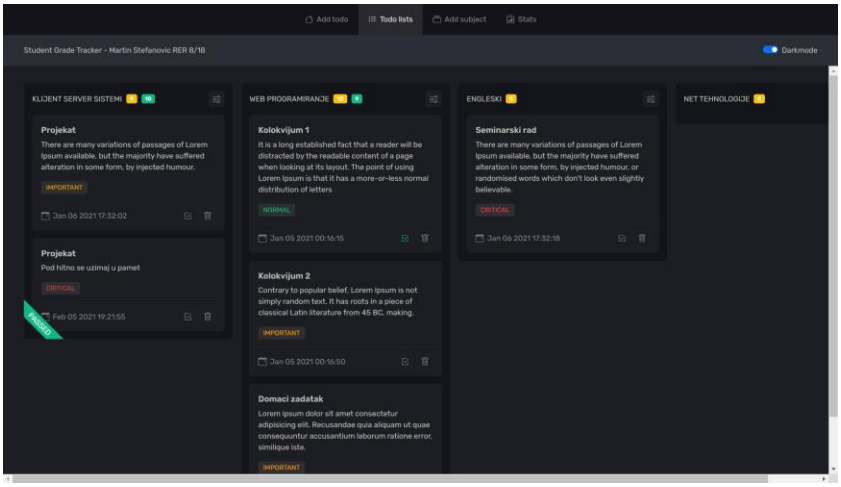


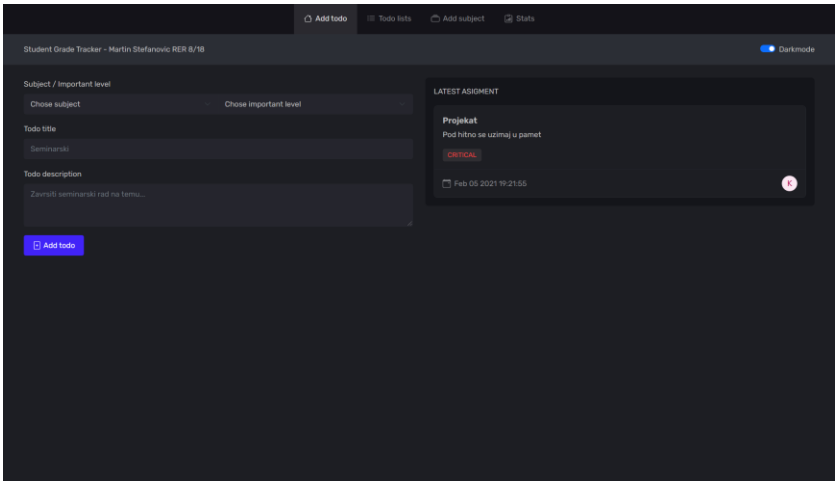
Stats stranica nam omogućava da vidimo statistiku. Trenutni prosek, ciljani prosek. Postotak obaveza po svakom nepoloženom predmetu kao i sve položene ispite.



Darkmode


Add todo
Todo lists
Add subject
Stats
Student Grade Tracker - Martin Stefanovic RER 8/18
Darkmode
GENERAL STATISTIC
9.50
Current average grade
8.75
Goal average grade
TASKS STATISTIC
Klijent server sistemi
Web programiranje
Engleski
Net tehnologije
PASSED EXAMS
Klijent server sistemi
Web programiranje
50%


Add todo
Todo lists
Add subject
Stats
Student Grade Tracker - Martin Stefanovic RER 8/18
Darkmode
KLIJENT SERVER SISTEMI
WEB PROGRAMIRANJE
ENGLESKI
NET TEHNOLOGIJE
Projekat
There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour.
IMPORTANT
Jan 06 2021 17:32:02
Projekat
Pod hitno se uzimaj u pamet
CRITICAL
Feb 05 2021 19:21:55
Kolokvijum 1
It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters
NORMAL
Jan 05 2021 00:16:15
Kolokvijum 2
Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making.
IMPORTANT
Jan 05 2021 00:16:50
Domaci zadatak
Lorem ipsum dolor sit amet consectetur adipisicing elit. Recusandae quia aliquam ut quae consequuntur accusantium laborum ratione error, similique iste.
IMPORTANT
Seminarski rad
There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable.
CRITICAL
Jan 06 2021 17:32:18


Add todo
Todo lists
Add subject
Stats
Student Grade Tracker - Martin Stefanovic RER 8/18
Darkmode
Subject / Important level
Chose subject
Chose important level
Todo title
Seminarski
Todo description
Završiti seminarski rad na temu...
Add todo
LATEST ASIGMENT
Projekat
Pod hitno se uzimaj u pamet
CRITICAL
Feb 05 2021 19:21:55