



Република Македонија



Универзитет „Св.Кирил и Методиј“ - Скопје  
Факултет за информатички науки и компјутерско инженерство

Тема:

## Веб апликација за резервација на автобуски билети

---

-Дипломска работа-

**Ментор:**

проф. д-р Милош Јовановиќ

**Кандидат:**

Борјан Коњановски - 163063

**Комисија:**

проф. д-р Иван Чорбев

проф. д-р Иван Китановски

Ноември 2020

## Содржина

Листа на слики .....	3
1. Апстракт .....	4
2. Вовед .....	5
3. Опис на користени технологии .....	6
3.1 HTML .....	6
3.2 CSS .....	7
3.3 JavaScript.....	7
3.4 Bootstrap.....	8
3.5 jQuery .....	8
3.6 Node.JS .....	9
3.7 Express.JS .....	10
3.8 MongoDB .....	11
3.9 Amazon AWS Cloud9 .....	12
4. Веб апликација .....	13
4.1 Почетна Страница .....	13-14
4.2 Регистрација .....	15-16
4.3 Најава .....	17-18
4.4 Корисничка страница .....	19-20
4.5 Превозници .....	21-22
4.6 Купување билети .....	23-24
4.7 Плаќање .....	25-26
4.8 Администратор .....	27-28
5. Заклучок.....	29
6. Референци.....	30

## Листа на слики

Слика 1: Три главни компоненти при развивање Веб Апликации .....	6
Слика 2: HTML5 и CSS3 .....	7
Слика 3: Bootstrap и jQuery .....	8
Слика 4: Официјалното лого на Node.js .....	9
Слика 5: Model View Controller .....	10
Слика 6: Приказ на податоците во MongoDB .....	11
Слика 7: Работна околина AWS Cloud9 .....	12
Слика 8: Преглед на главната страна .....	13
Слика 9: Рута “/” .....	14
Слика 10: index.ejs.....	14
Слика 11: Форма за регистрација .....	15
Слика 12: GET и POST рути за “/register”.....	16
Слика 13: register.ejs .....	16
Слика 14: Форма за најава .....	17
Слика 15: GET и POST рути за “/login” .....	18
Слика 16: login.ejs .....	18
Слика 17: Корисничка страница .....	19
Слика 18: Приказ на празна листа на билети .....	19
Слика 19: GET рута за “/user”.....	20
Слика 20: user.ejs .....	20
Слика 21: Превозници .....	21
Слика 22: Преглед на линии по превозник .....	21
Слика 23: Рути за превозниците .....	22
Слика 24: Google Maps API .....	22
Слика 25: Листа на купени билети .....	23
Слика 26: Откажана линија .....	23
Слика 27: Рути за купување и бришење билети .....	24
Слика 28: Прозорец за плаќање .....	25
Слика 29: Успешно плаќање .....	25
Слика 30: POST рута за успежно плаќање .....	26
Слика 31: Печатење билети .....	26
Слика 32: Преглед на сите линии од админ .....	27
Слика 33: Додавање нова линија .....	27
Слика 34: Постапување на глобална променлива currentUser .....	28
Слика 35: Користење на currentUser за проверка на авторизација.....	28
Слика 36: Овозможување опција за додавање на нова линија.....	28

## 1. Апстракт

*Во рамки на оваа дипломска работа е опишан процесот на изработка на целосна full-stack веб апликација, како и начинот на кој се користи истата.*

*Ќе бидат опишани технологиите кои се употребени при изработката на ова апликативно веб решение како и потребата од еден ваков систем на пазарот.*

*За крај ќе биде даден заклучок кој ќе ги опфати сите аспекти на една ваква апликација, како и можноста во иднина да се надогради, развива и прошири истата.*

## 2. Вовед

Во денешно време покрај добра реклама и маркетинг стратегија, од клучна важност за еден бизнис или организација е да се има добро познавање за пазарот кој е од интерес, како и за потенцијалните клиенти на тој пазар. Без добро познавање на пазарот или околината, нема начин да се направи правилна проценка во која насока да се движи развојот на бизнисот.

Под добро познавање на пазарот се подразбира и постојаното следење на најновите светски трендови и норми со цел да се подобри квалитетот на услугата и да се задржи местото пред конкуренцијата.

Еден ваков сектор на пазарот е комерцијалниот транспорт, поточно автобускиот превоз кој во нашата држава има огромна потреба од модернизирање.

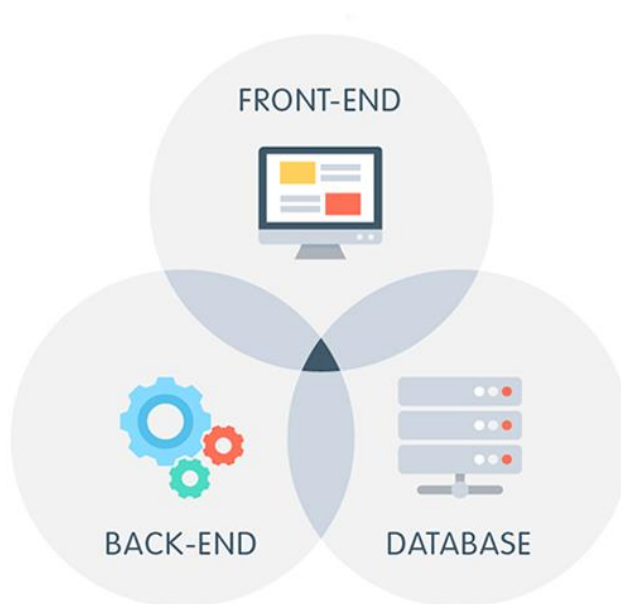
Ако се размисли, начинот на кој функционираат автобуските превозници во нашата држава нема претрпено поголеми промени во последните 3 децении. Традиционално, кога некој има потреба да патува со автобус, најпрво оди во билетарата на превозникот (најчесто се наоѓаат на самата автобуска станица), ги прегледува достапните линии и купува билет за посакуваната линија, односно резервира место. Ова често се прави дури и неколку денови пред самото тргнување, со цел патникот да го осигура неговото место за тоа патување.

Целиот овој процес на купување билет претставува дополнителна потешкотија за патникот, бидејќи тој мора физички да оди до билетарата на превозникот за што во некои градови може да изгуби време и до неколку часови. Поради ова, во најголем број од државите низ светот постои можност за електронско купување на билети преку интернет.

Целта на оваа дипломска работа е токму решавање на овој проблем и претставување на **Веб апликација за резервација на автобуски билети.**

### 3. Опис на користени технологии

За изработка на оваа веб апликација искористен е одреден *stack* на технологии. Во делот за *back-end* користено е *Node.JS* со *Express*, за чување на податоците користена е локална база на податоци *MongoDB*, додека во *front-end* делот се користени технологиите: *HTML*, *CSS*, *Javascript*, *jQuery* и *Bootstrap*. Целиот проект е достапен на [GitHub](#).



Слика 1: Три главни компоненти при развивање Веб Апликации

#### 3.1 HTML

*Hypertext Markup Language (HTML)* е основниот јазик на вебот уште од неговото претставување во 1993 година па се до денес. *HTML* не е програмски јазик. Тој претставува таканаречен описен или “*markup*” јазик, преку кој се дефинира структурата на страната и се опишуваат елементите на таа страна и им се дава семантичко значење. Јазикот е составен од тагови преку кои се креираат различни типови на елементи како листи, табели, наслови, параграфи, форми за внесување, слики, линкови и др.

*HTML* јазикот има еволуирано низ неколку верзии што содржат измени низ текот на годините: *HTML 2.0* претставен во 1995 година, *HTML 3.0* во 1997, *HTML 4.0* во 1998.

Во овој проект е искористен *HTML 5*, поточно најновата ревизија *HTML 5.2*.

## 3.2 CSS

*Cascading Style Sheets (CSS)* или јазик на таканеречени каскадни стилови претставува јазик кој се користи за стилизирање на *HTML* елементите на една веб страна. *CSS* опишува како треба да изгледа дизајнот на страната која се прикажува. Тоа се прави на тој начин што прво се селектираат посакуваните елементи, и потоа им се нанесуваат различни *CSS* вредности. Најчесто овие стилови се чуваат во екстерни *CSS* датотеки, но може да се вметнатаат и во самиот *HTML* документ.

Во овој проект се користи најновата верзија *CSS 3* претставена во 2011 година.



Слика 2: *HTML5* и *CSS3*

## 3.3 JavaScript

Стандардниот *JavaScript* програмски јазик е претставен уште во 1995 година и примарно е наменет за извршување во рамки на еден веб прелистувач. Тој се извршува на клиентската страна (*client-side*). Низ годините, *JavaScript* станува многу популарен и масовно се развива такашто во денешно време ретко може да се најде веб апликација која не користи *JavaScript* на еден или друг начин. Заедно со *HTML* и *CSS*, *JavaScript* е еден од трите основни јазици за развивање на *front-end* делот од една веб апликација. Со користење на *JavaScript* се овозможуваат многу функционалности преку манипулација на статичните *HTML* елементи, промена на *CSS* вредностите за стилизирање и главно се користи за додавање интерактивност и динамичност на веб апликацијата. *JavaScript* датотеките имаат наставка *.JS* која е дел од имињата на многу модерни библиотеки (*libraries*) и работни рамки за развој (*frameworks*).

Во овој проект се користи *JavaScript* верзија 1.7.

### 3.4 Bootstrap

*Bootstrap* претставува рамка за развој заснована на *CSS* која се користи за побрзо и поефикасно креирање на *front-end* делот на една веб страна. Првично е претставена во 2011 година од страна на *Twitter*. Денеска е бесплатен и отворен проект хостиран на *GitHub* поддржан од голема заедница на корисници што придонесуваат за неговата популарност. *Bootstrap* вклучува готови класи, претходно стилизирани компоненти и поддржува *responsive* и *mobile-first* дизајн. Ова значи дека фокусот е на градење дизајни што изгледаат уредно на сите големини на екрани преку скалирање на содржината, примарно на се позастапените преносни мобилни уреди (смартфони и таблети).

Во овој проект се користи *Bootstrap 4*, поточно верзијата *4.5.2*



Слика 3: *Bootstrap* и *jQuery*

### 3.5 jQuery

*jQuery* претставува популарна *JavaScript* библиотека која значително го олеснува пишувањето *JavaScript* и заштедува многу време при развој на една веб апликацијата. Слично како и *Bootstrap*, *jQuery* овозможува користење на готови функции и методи. Со користење на *jQuery* многу вообичаени функционалности за кои се потребни многу редови на *JavaScript* код, се групирани во различни методи кои потоа можат да се повикаат преку една линија код. Дел од функционалностите на *jQuery* библиотеката се манипулација на *HTML* и *CSS*, настани (*events*), ефекти, анимации, додатоци и многу др.

Иако во поново време се почесто се користат други *JavaScript* работни рамки како *React*, *Angular* и *Vue*, *jQuery* останува водечка *JavaScript* библиотека.

Во овој проект се користи *jQuery* верзијата *3.5.1*.



### 3.6 Node.JS

*Node.JS* претставува околина за извршување на *JavaScript* код надвор од веб прелистувачот. *Node* е целосно бесплатен, *open-source* и *cross-platform*. Направен е во 2009 година врз основа на тогашниот *V8 JavaScript engine* наменет за *Google Chrome*.



Слика 4: Официјалното лого на Node.js

Со користење на *Node*, се овозможува *JavaScript* да се употреби и на серверската страна (*server-side*). Ова е многу корисно бидејќи овозможува низ целиот процес на развивање *full-stack* веб апликација да се користи еден конзистентен програмски јазик.

*Node* има свој *package-manager* нареачен *Node Package Manager (npm)* кој се користи за автоматско пребарување и инсталација на илјадници различни достапни модули изработени од програмери низ целиот свет што всушност претставуваат готови пакети со код за додавање на одредена функционалност. Сите овие модули се бесплатни.

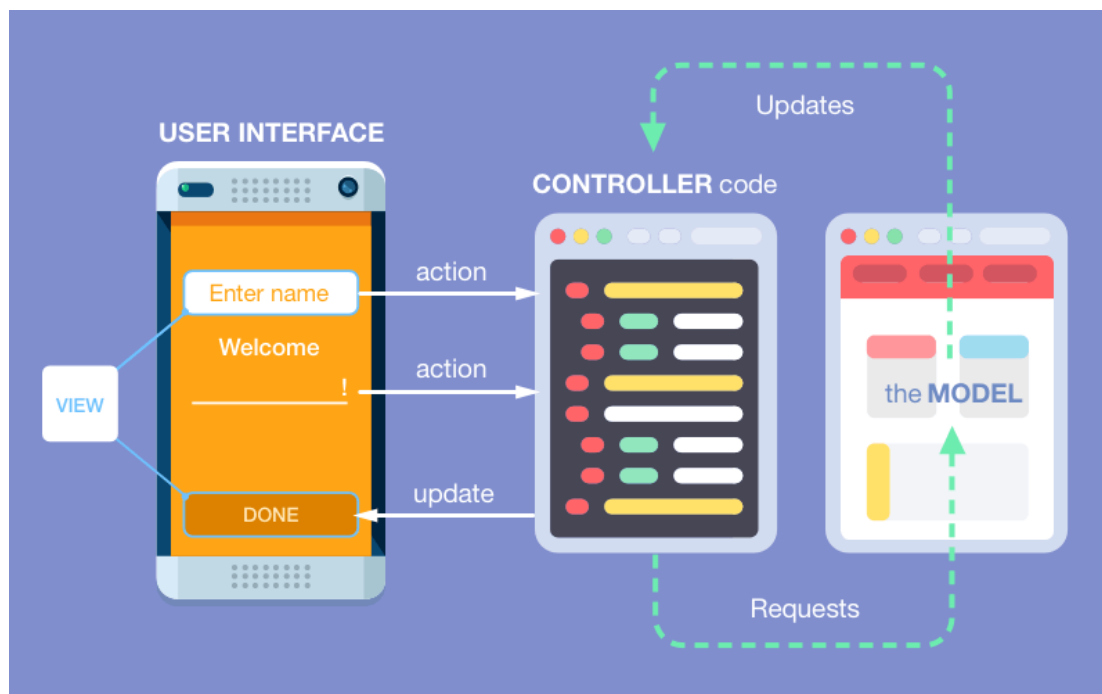
*Node* модули што се користат во рамки на оваа веб апликација се:

- *ejs v3.1.5* – *view engine* за рендерирање на код вметнат во темплејти
- *body-parser v1.19.0* – за парсирање на телото на испратените *HTTP* барања
- *express.JS v4.17.1* – *framework* за градење серверски веб апликации (стр. 10)
- *express-session v1.17.1* – додавка на *express* за работа со кориснички сесии
- *mongoose v5.10.10* – за работа со *MongoDB* база на податоци (стр. 11)
- *passport.JS v0.4.1* – популарен модул за управување со автентикација
- *passport-local v1.0.0* – додаток стратегија за *passport.JS*
- *passport-local-mongoose v6.0.1* – додаток за *passport.JS*

Во овој проект се искористени *Node.JS* верзија 10.22.1 и *npm* верзија 6.14.6.

### 3.7 Express.JS

Многу често *Node* се користи во комбинација со *Express.JS* работната рамка за развивање на *back-end* веб апликации со користење на *Model View Controller (MVC)* шаблон.



Слика 5: Model View Controller

*MVC* претставува архитектура во која кодот е поделен на три главни компоненти:

- *Controller* е одговорен за примање и процесирање на барање од корисникот
- Податочниот *model* претставува ентиет кој постои во апликацијата
- *View* го рендерира *Model*-от во соодветен излезен формат

*Express.JS* главно е одговорен за справување со кориснички барања (*HTTP request handling*).

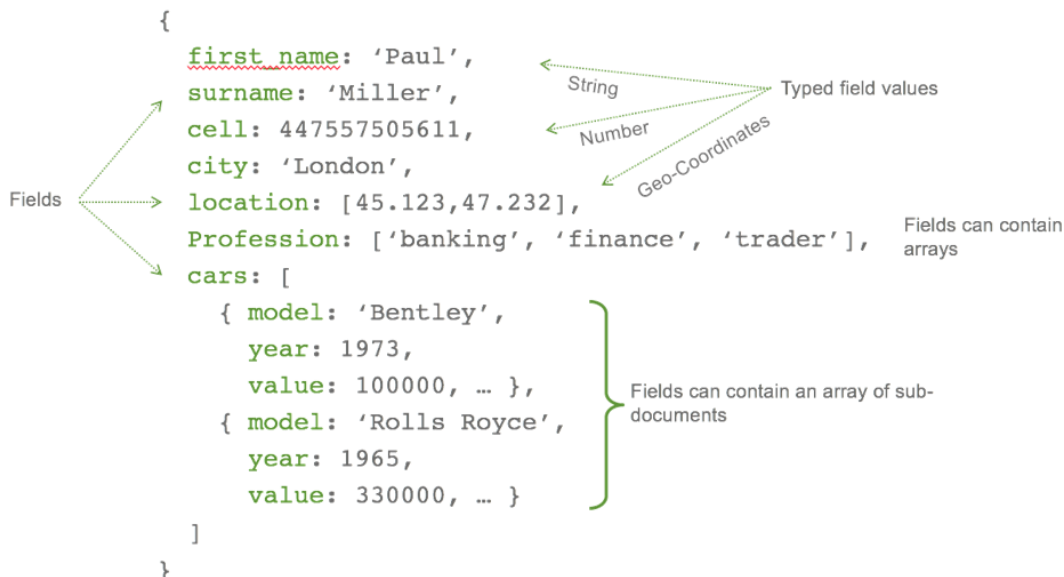
**HyperText Transfer Protocol (HTTP)** е главниот протокол за управување со комуникацијата помеѓу клиентот и серверот. Главно се состои од барања (*requests*) и одговори (*responses*) испратени преку веб. Протоколот има свои статусни кодови кои ги враќа во зависност од барањето на корисникот (најчести се *200-OK*, *404- Not Found* и *500 – Internal server error*).

Бидејќи *HTTP* претставува *Stateless* протокол кој не памти информации за тековниот корисник, потребни се алатки како *Node* модулот *express-session* за управување со кориснички сесии.

Во овој проект се искористени *Express.js* верзија 4.17.1 и *express-session* верзија 1.17.1.

### 3.8 MongoDB

*MongoDB* претставува популарен систем за управување со бази на податоци.



Слика 6: Приказ на податоците во *MongoDB*

За разлика од традиционалните релациони бази на податоци кој користат табели со меѓусебни релации и *SQL (Structured Query Language)* за манипулација со податоците, *MongoDB* е *NoSQL* односно користи документ-базиран модел каде што податоците се чуваат во формат кој наликува на *JSON (JavaScript Object Notation)*. Наместо во табели податоците се чуваат во колекции.

Целта на *MongoDB* е да обезбеди безбедно место за складирање и организација на податоците на апликацијата и овозможување на основите *CRUD* операции:

- **Create** – Креирање на нов запис
- **Read** – Читање на постоечки запис
- **Update** – Промена на постоечки запис
- **Delete** – Бришење на запис

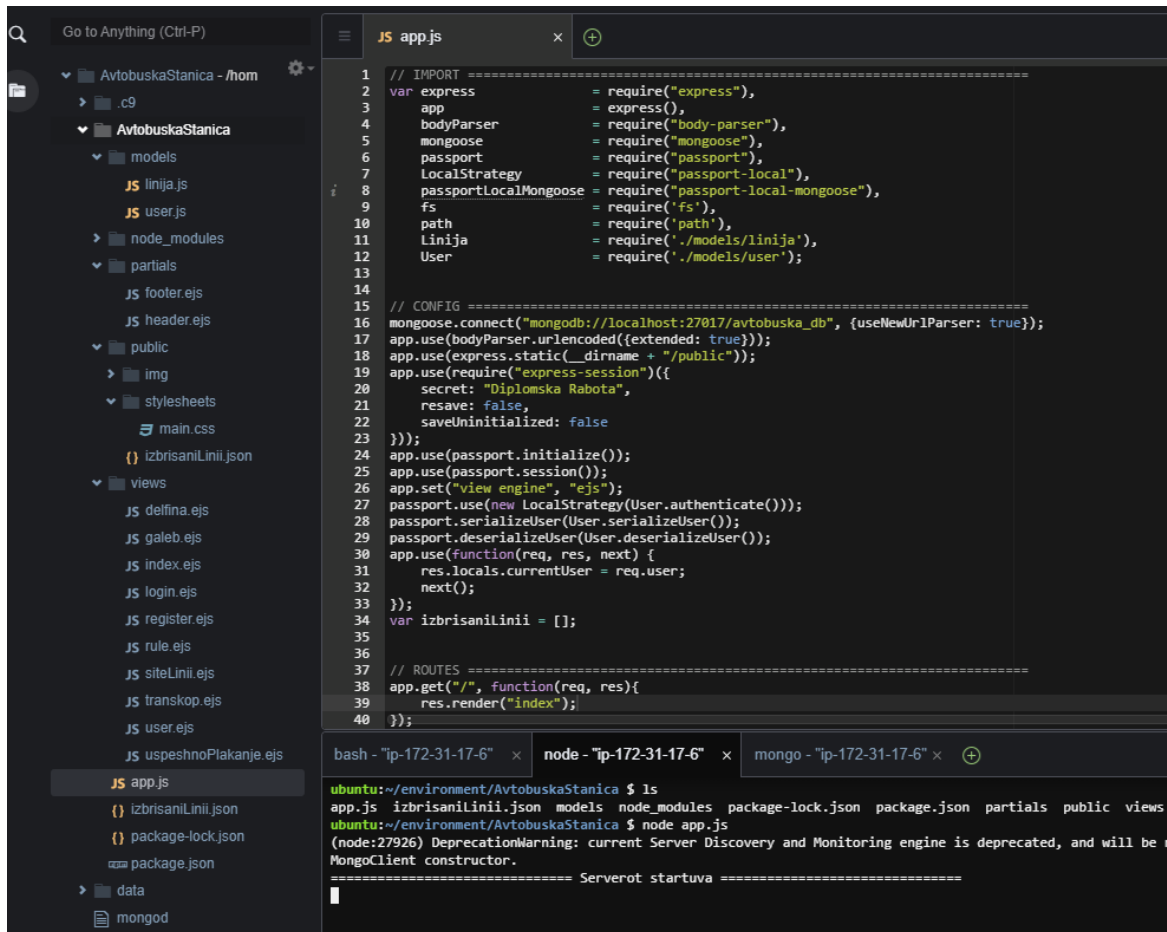
Во нашата веб апликација за резервација на автобуски билети, податоците за превозниците, линиите, билетите и резервациите се складираат во колекции на локална *MongoDB* база на податоци преку која се овозможени сите овие *CRUD* функционалности.

За полесно манипулирање со податоците кога се работи во *Node* се користи модулот *mongoose*, додека за автентикација на корисниците се користи додатокот *passport-local-mongoose*.

Во овој проект се искористени *MongoDB* верзија 3.6.3 и *mongoose* верзија 5.10.10.

### 3.9 Amazon AWS Cloud9

Оваа веб апликација е хостирана на платформата *Cloud9* на *Amazon Web Services (AWS)*.



Слика 7: Работна околина AWS Cloud9

Проектот е организиран во следната хиерархиска структура на датотеки:

- Во *root* директориумот се наоѓаат сите други под-директориуми и датотеки и тука е инсталиран и конфигуриран *MongoDB* серверот (*mongod*)
- Во директориумот *AvtobuskaStanica* се наоѓаат сите останати датотеки
- Во *models* се наоѓаат моделите за корисник (*User*) и линија (*Linija*)
- Во *node\_modules* се наоѓаат сите потребни пакети и модули користени од *Node*
- Во *partials* се наоѓаат *header* и *footer* фајловите
- Во *public* се наоѓаат сликите (*img*) и *CSS* фајлот (*stylesheet*)
- Во *views* се чуваат сите погледи (*templates*) што се рендерираат за корисникот
- Самата *Node* апликација заедно со рутите е содржана во датотеката *app.js*
- *izbrishaniLinii.json* е датотека што ги памти избришаните линии (стр. 23)
- *package.json* и *package-lock.json* се две датотеки потребни за да работи *npm*

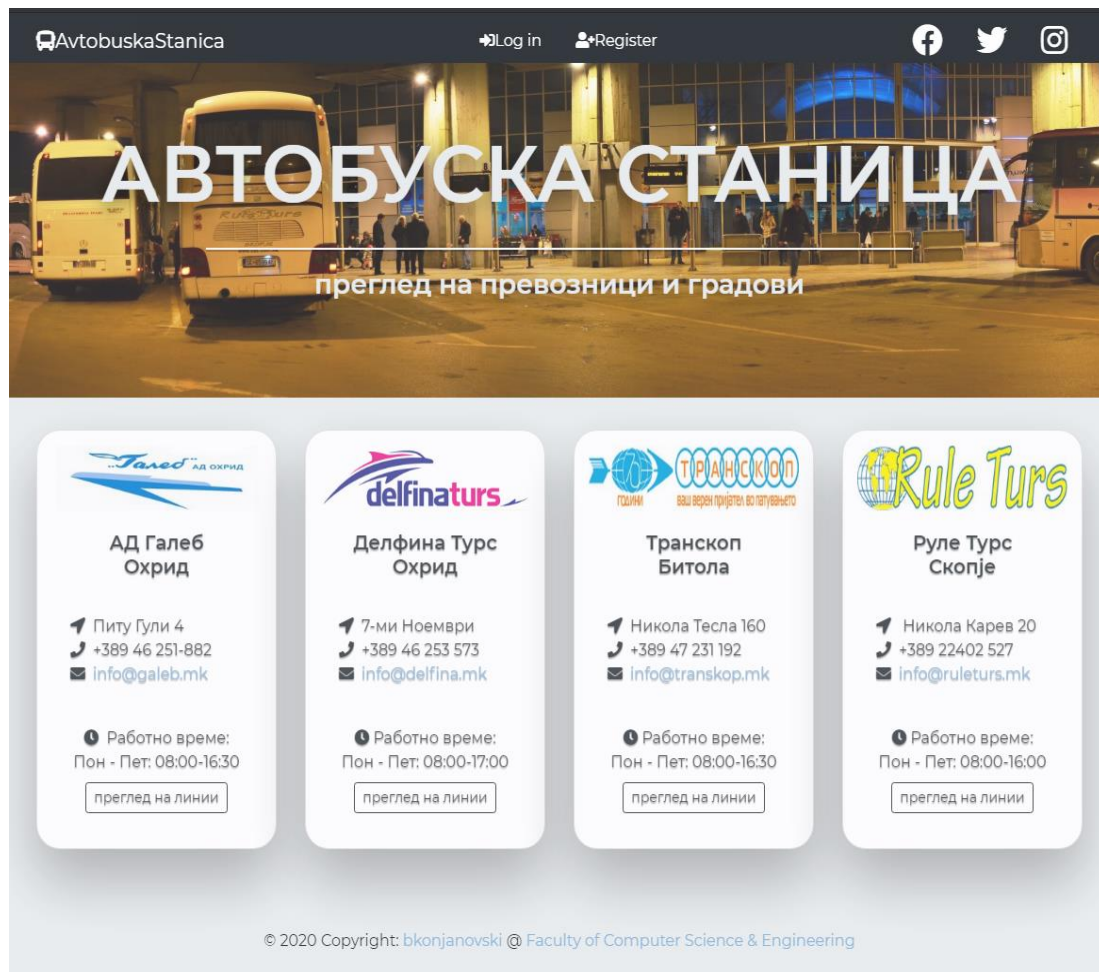
## 4. Веб Апликација

Со користење на досега наведените технологии е изработена оваа веб апликација.

Апликацијата овозможува резервација на автобуски билети од различни превозници и до различни дестинации, пресметува цена на билетите и има опција за нивно плаќање. Дополнително, содржи опции за додавање нови линии и бришење на постоечките од страна на администраторот на системот.

Во следните страници е опишан начинот на кој се користи апликацијата, како и преглед на соодветните *front-end* и *back-end* делови.

### 4.1 Почетна страница



Слика 8: Преглед на главната страна

По пристапувањето на страната на нејзината *root* патека (“/”) се појавува насловната страница на апликацијата. Се состои од 3 дела: навигациската лента (*nav-bar*) која се состои од опции за најавување и регистрирање, како и линкови до социјални мрежи. Потоа се наоѓа главниот дел на слики со *slideshow* кои всушност претставуваат *carousel* во *jumbotron* елемент и сликите се движат со помош на дефинирана *jQuery* функција. Најдолу се информациите за четирите превозници кои ги користи системот. На самиот крај на страната има и краток *footer* кои се состои од линк до мојот *GitHub* профил, како и линк до страната на ФИНКИ. Овој *footer*, заедно со навигациската лента се наоѓаат на сите останати страници од апликацијата и нивниот изглед не се менува. Ова е имплементирано со тоа што овие два сегменти (*header* и *footer*) се сместени во две посебни датотеки под директориумот *partials*, и сите останати страници линкуваат до нив две, со цел да се добие конзистентен изглед низ целата апликација.

До почетната страница се пристапува преку рутата (“/”).

```
// ROUTES =====
app.get("/", function(req, res){
  res.render("index");
});
```

Слика 9: Рута “/”

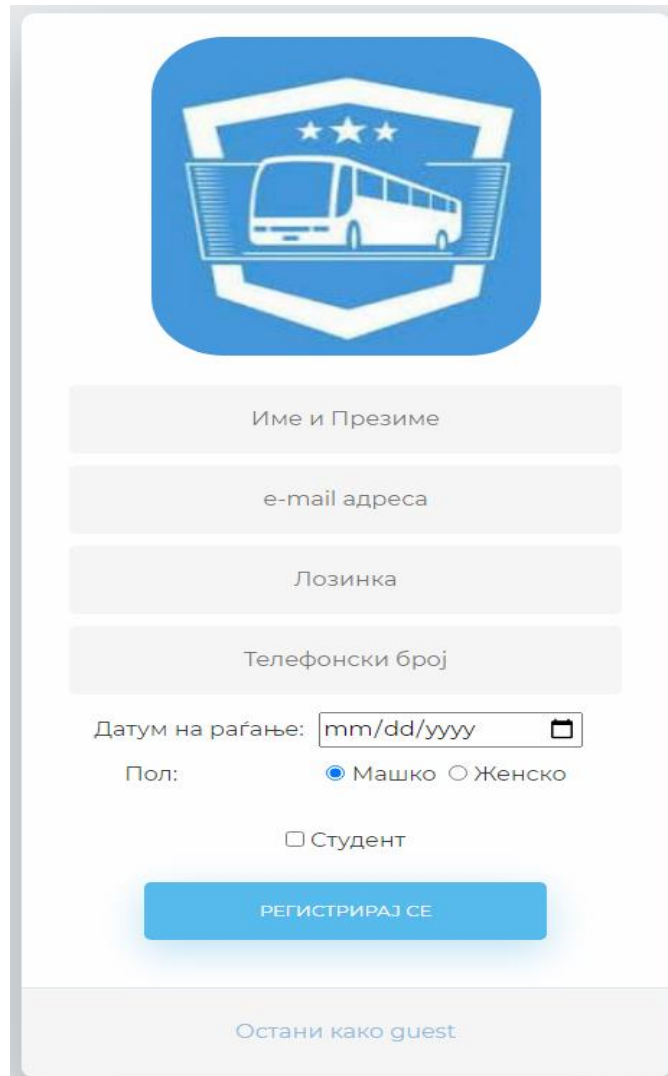
Погледот (*view*) за оваа страница се рендерира во *index.ejs*

```
JS index.ejs
1 <%- include ('../partials/header') %>
2
3 <!-- Slideshow -->
4 <section id="header" class="jumbotron text-center fadeIn second">
5   <div class="carousel slide" data-ride="carousel">
6     <div class="carousel-inner">
7       <div class="carousel-item active">
8         
9       </div>
10      <div class="carousel-item">
11        
12      </div>
13      <div class="carousel-item">
14        
15      </div>
16      <div class="main-text hidden-xs" id="mainText">
17        <div class="col-md-12">
18          <h2>Автобуска Станица</h2> <hr>
19          <h3>Преглед на превозници и градови</h3>
20        </div>
21      </div>
22    </div>
23  </div>
24 </section>
25
26 <!-- Prevoznici -->
27 <section id="prevoznici" class="fadeIn third">
28   <div class="container">
29     <div class="row">
30       <div class="col-lg-3 mb-3">
31         <div class="card">
32           <a href="http://galeb.mk/?strana=13" target="_new"></a>
33           <div class="card-body">
34             <h5 class="card-title text-center"><strong>АД Галеб <br>Охрид</strong></h5>
35             <span class="card-text"><p>
36               <br>
37               <i class="fa mr-2 fa-location-arrow"></i>Питу Гули 4 <br>
38               <i class="fa mr-2 fa-phone"></i>+389 46 251-882 <br>
```

Слика 10: *index.ejs*

## 4.2 Регистрација

Со притискање на “*Register*” копчето на навигациската лента, корисникот е пренасочен на “*/register*” патеката каде што се појавува форма за негова регистрација доколку се работи за нов корисник.



Име и Презиме

е-mail адреса

Лозинка

Телефонски број

Датум на раѓање: mm/dd/yyyy

Пол: ☒ Машко ☐ Женско

☐ Студент

РЕГИСТРИРАЈ СЕ

Остани како guest

Слика 11: Форма за регистрација

Тука корисникот ги внесува неговите лични податоци за да креира негова сметка (*account*) на системот. Сите полиња се задолжителни за пополнување. Е-mail полето мора да е во формат на email (*abc@xyz.com*), додека пак лозинката мора да содржи најмалку 8 карактери и да е составена од најмалку една мала, една голема буква и една цифра. Доколку некој од овие услови не е исполнет, поставени се валидатори кои ќе му напоменат на корисникот да ги корегира внесените податоци. Најдолу на формата корисникот има опција и да не се најави, односно да остане како *guest* на системот, со што ќе може само да ги прегледува линиите, но не и да купува билети.



Соодветната *GET*-рута за регистрација е *"/register"*, а логиката се содржи во истоимената *POST* рута преку дефинираниот *User* модел:

```
app.get("/register", function(req, res){
  res.render("register");
});

app.post("/register", function(req, res){
  User.register(new User(
    {username: req.body.username, imePrezime: req.body.imePrezime, datumNaRaganje: req.body.datumNaRaganje,
    telefon: req.body.telefon, pol: req.body.pol, student: req.body.student}
  ), req.body.password, function(err, user){
    if(err){
      console.log(err);
      return res.render('register');
    }
    passport.authenticate("local")(req, res, function(){
      res.redirect("/user");
    });
  });
});
```

Слика 12: *GET* и *POST* руми за *"/register"*

Погледот (*view*) за оваа страница се рендерира во *register.ejs*

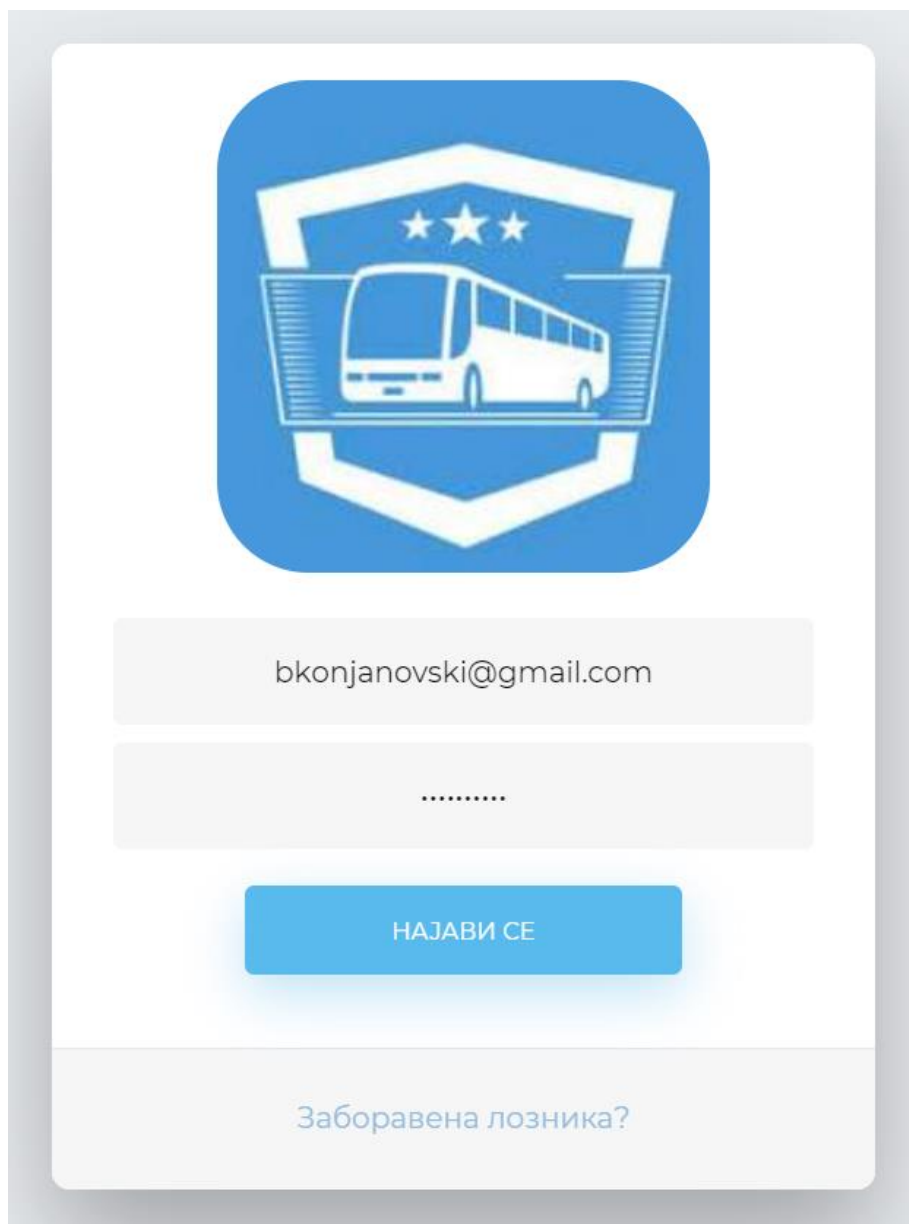
```
JS register.ejs x (+)
1 <%= include ('../partials/header') %>
2
3 <!-- Форма за регистрација -->
4 <div class="wrapper fadeInDown">
5 <div id="formContent">
6 <div class="fadeIn second">
7 
8 </div>
9 <form action="/register" method="POST" autocomplete="off">
10 <input type="text" class="fadeIn third" name="imePrezime" placeholder="Име и Презиме" required>
11 <input type="email" class="fadeIn third" name="username" placeholder="e-mail адреса" required>
12 <div class="validacija"> <input type="password" name="password" placeholder="Лозинка"
13 class="fadeIn third" pattern="(?!.*\d)(?!.*[a-z])(?!.*[A-Z]).{8,}" required>
14 <span class="tooltipText">Лозинката мора да содржи најмалку 8 карактери и да е составена од најмалку една мала, една голема буква и цифра</span>
15 </div>
16 <input type="text" class="fadeIn third" name="telefon" placeholder="Телефонски број" required> <br>
17 <label for="birthday" class="fadeIn fourth">Датум на раѓање: </label><input type="date" class="fadeIn fourth" name="datumNaRaganje" required> <br>
18 <label class="fadeIn fourth labela">Пол:</label>
19 <input type="radio" name="pol" value="машко" checked class="fadeIn fourth"><span class="fadeIn fourth">Машко</span>
20 <input type="radio" name="pol" value="женско" class="ml-1 fadeIn fourth"><span class="fadeIn fourth">Женско</span><br><br>
21 <input type="checkbox" name="student" value="true" class="fadeIn fourth">
22 <label class="fadeIn fourth"><span class="fadeIn fourth">Студент</span></label><br>
23 <input type="submit" class="fadeIn fourth" value="РЕГИСТРИРАЈ СЕ">
24 </form>
25 <div id="formFooter">
26 <a class="underlineHover" href="/">Остани како guest</a>
27 </div>
28
29 </div>
30 </div>
31
32 <%= include ('../partials/footer') %>
33
```

Слика 13: *register.ejs*



### 4.3 Најава

Доколку се работи за постоечки корисник кои има веќе сметка на системот, со притискање на “Log in” копчето на навигациската лента, корисникот е пренасочен на “/login” патеката каде што се појавува форма за негова најава.

The image shows a login form for a bus system. At the top is a blue shield-shaped logo with a white bus icon and three stars above it. Below the logo are two input fields: the first contains the email address 'bkonjanovski@gmail.com' and the second contains a masked password '.....'. A blue button with the text 'НАЈАВИ СЕ' is positioned below the password field. At the bottom of the form is a link that says 'Заборавена лозника?'.

Слика 14: Форма за најава

Тука корисникот ги внесува неговите постоечки e-mail адреса и лозинка за да се најави. Доколку најавувањето не е успешно, формата се појавува повторно. Во случај на заборавена лозника, со притискање на линкот се праќаат насоки за ресетирање на лозинката на неговата email адреса (фиктивно). Доколку најавувањето е успешно, тој се пренасочува на корисничката страница.

Слично како и претходно, за најава се потребни две рути: *GET* и *POST /login*

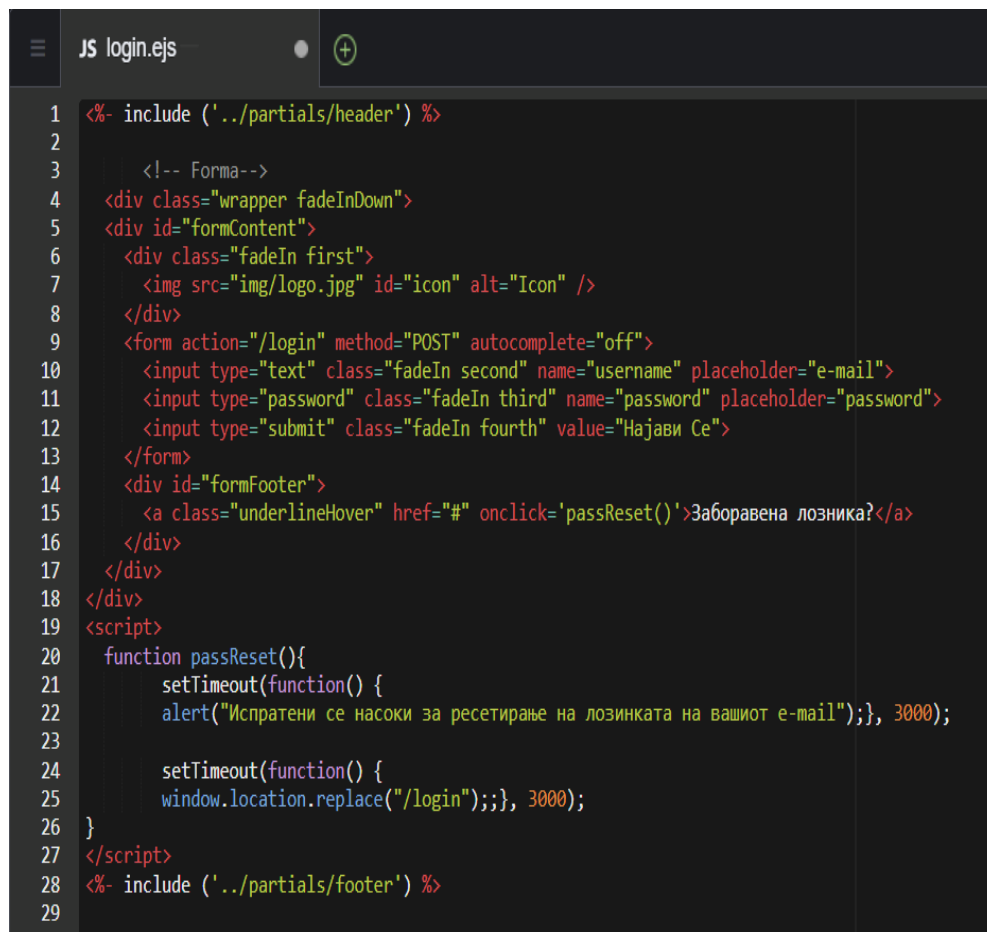
```
app.get("/login", function(req, res){
  res.render("login");
});

app.post("/login", passport.authenticate("local", {
  successRedirect: "/user",
  failureRedirect: "/login"
}) ,function(req, res){
});
```

Слика 15: GET и POST рути за “/login”

Овие функционалности за автентикација се овозможени преку *PassportJS* модулот заедно со додатоците *passport-local-mongoose* и *express-session*.

Функцијата *isLoggedIn* проверува дали моментално има најавен корисник, во спротивно го пренасочува корисникот до страницата за најавување.



```
JS login.ejs
1 <%- include ('../partials/header') %>
2
3 <!-- Forma-->
4 <div class="wrapper fadeInDown">
5 <div id="formContent">
6 <div class="fadeIn first">
7 
8 </div>
9 <form action="/login" method="POST" autocomplete="off">
10 <input type="text" class="fadeIn second" name="username" placeholder="e-mail">
11 <input type="password" class="fadeIn third" name="password" placeholder="password">
12 <input type="submit" class="fadeIn fourth" value="Најави Се">
13 </form>
14 <div id="formFooter">
15 <a class="underlineHover" href="#" onclick='passReset()'>Заборавена лозника?</a>
16 </div>
17 </div>
18 </div>
19 <script>
20 function passReset(){
21   setTimeout(function() {
22     alert("Испратени се насоки за ресетирање на лозинката на вашиот e-mail");}, 3000);
23
24   setTimeout(function() {
25     window.location.replace("/login");}, 3000);
26 }
27 </script>
28 <%- include ('../partials/footer') %>
29
30
```

Слика 16: login.ejs

## 4.4 Корисничка страница

По успешна регистрација или најава од страна на корисникот, тој е пренасочен на “/user” патеката, односно на неговата корисничка страница.

**Борјан Коњановски**

Студент  
bkonjanovski@gmail.com  
070232551  
1997-07-30

Мој билети:

Превозник	Тргува од	Тргува во	Дестинација	Цена	
galeb	Охрид	Пон. 09:30 ч.	Скопје	500 ден.	Избриши
transkop	Прилеп	Сре. 15:00 ч.	Битола	250 ден.	Избриши
rule	Струмица	Саб. 21:30 ч.	Охрид	650 ден.	Избриши

Слика 17: Корисничка страница

Мој билети:

Моментално немате неплатени билети!

преглед на линии

Слика 18: Приказ на празна листа на билети

Тука корисникот ги гледа неговите купени и сеуште неплатени билети и има опција истите да ги избрише. Доколку корисникот нема моментално неплатени билети, тоа му е прикажано. Со притискање на “Log out” копчето на навигациската лента, корисникот се одјавува и е пренасочен на почетната страница.

```

app.get("/user", isLoggedIn, function(req, res) {
  if (req.user.username=="admin") {
    res.redirect("/siteLinii");
  } else {
    izbrisanilinii = JSON.parse(fs.readFileSync('izbrisanilinii.json'));
    res.render("user", {izbrisanilinii: izbrisanilinii});
  }
});

```

Слика 19: GET рута за "/user"

```

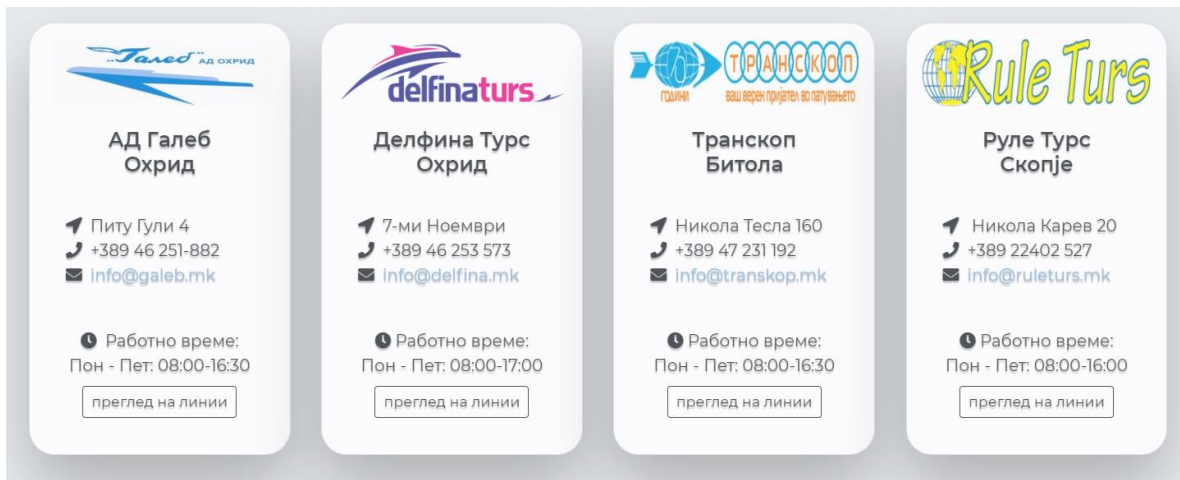
1 <%- include ('../partials/header') %>
2
3 <div class="container">
4   <div class="card mt-5 fadeIn second" id="user-card">
5     <div class="card-body">
6       <h5 class="card-title"><strong><%= currentUser.imePrezime %></strong></h5>
7       <span class="card-text"><p>
8         <% if(currentUser.pol=="женско"){ %>
9           </a>
10        <% } else { %>
11          </a>
12        <% } %>
13        <% if(currentUser.student){ %>
14          <br><br><i class="fas mr-1 fa-graduation-cap"></i>Студент
15        <% } %>
16        <br><i class="fa mr-2 fa-envelope"></i><a href="mailto: info@galeb.mk"><%= currentUser.usern
17        <i class="fa mr-2 fa-phone"></i><%= currentUser.telefon %><br>
18        <i class="fa mr-2 fa-table"></i><%= currentUser.datumNaRaganje%>
19        </p>
20      </span>
21    </div>
22  </div> <br><br>
23  <h5 class="fadeIn third" >Moj билети: </h5>
24
25  <% if (currentUser.linii.length == 0) { %>
26  <h3 class="fadeIn third">Моментално немате неплатени билети!</h3>
27  <% } else { %>

```

Слика 20: user.ejs

## 4.5 Превозници

Доколку корисникот е најавен, тој може да купува билети од четирите различни превозници на системот: *Галеб*, *Делфина Турс*, *Транскоп* и *Руле Турс*.



Слика 21: Превозници

За таа цел тој може да се врати на почетната страница со притискање на логото “*AvtobuskaStanica*” на навигациската лента за да ги прегледа различните линии по секој превозник посебно или да ги прегледа сите достапни линии наеднаш со притискање на копчето “*преглед на линии*”.

The image shows a map of Ohrid with a red pin marking 'Galeb Ohrid'. Below the map is a table of bus routes. The table has five columns: 'Превозник' (Carrier), 'Тргува од' (Origin), 'Тргува во' (Destination), 'Дестинација' (Destination), and 'Цена' (Price). Each row represents a different route, and each cell has a 'Купи билет' (Buy ticket) button.

Слика 22: Преглед на линии по превозник

```

app.get("/galeb", function(req, res){
  Linija.find({prevoznik: "galeb"}, function(err, siteLinii){
    if(err){
      console.log(err);
    } else {
      res.render("galeb", {linii: siteLinii});
    }
  });
});

app.get("/delfina", function(req, res){
  Linija.find({prevoznik: "delfina"}, function(err, siteLinii){
    if(err){
      console.log(err);
    } else {
      res.render("delfina", {linii: siteLinii});
    }
  });
});

app.get("/transkop", function(req, res){
  Linija.find({prevoznik: "transkop"}, function(err, siteLinii){
    if(err){
      console.log(err);
    } else {
      res.render("transkop", {linii: siteLinii});
    }
  });
});

app.get("/rule", function(req, res){
  Linija.find({prevoznik: "rule"}, function(err, siteLinii){
    if(err){
      console.log(err);
    } else {
      res.render("rule", {linii: siteLinii});
    }
  });
});

```

Слика 23: Руте за превозниците

При пристапување до рутите на некој од четирите превозници, се рендерира соодветниот поглед (*galeb.ejs*, *delfina.ejs*, *transkop.ejs*, *rule.ejs*) кој во него содржи *API* линк од *Google Maps* што ја прикажува локацијата на нивната билетара.

```





<div class="row">
  <iframe src=
    "https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d6011.060243374613!2d20.
    807783634887684!3d41.12295200000001!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x1350
    db639818e3cb%3A0xd23676e3a2e4bcf6!2sGaleb%200hrid!5e0!3m2!1sen!2smk!4v1603746656937!5m2!1sen!2smk"
    class="fadeIn second" width="100%" height="320" frameborder="0" style="border:0; margin-top: 60px" allowfullscreen></iframe>
</div>

```

Слика 24: *Google Maps API*


## 4.6 Купување билети

Со притискање на копчето “Купи Билет” за која било линија, линијата се додава во листата на купени билети на корисникот. Доколку корисникот е регистриран како студент, тој добива попуст од 20% на вкупната сума за плаќање. Пред да плати, корисникот на оваа страница има можност да ги прегледа досега купените билети и по потреба да избриши постоечки или да додаде нови билети.

Мој билети:					
Превозник	Тргува од	Тргува во	Дестинација	Цена	
galeb	Охрид	Пон. 09:30 ч.	Скопје	500 ден.	 Избриши
transkop	Прилеп	Сре. 15:00 ч.	Битола	250 ден.	 Избриши
rule	Струмица	Саб. 21:30 ч.	Охрид	650 ден.	 Избриши
rule	Штип	Сре. 12:00 ч.	Кратово	300 ден.	 Избриши
				Вкупно за плаќање: 1700 ден.	
				- 20% студентски попуст: 1360 ден.	
<a href="#">преглед на линии</a>					 Плати

Слика 25: Листа на купени билети

Доколку се случи некоја линија за која корисникот веќе има купено билет во меѓувреме да е откажана, таа му се прикажува со црвена и боја и соодветна порака за рефундирање на средствата.

откажани - за рефундирање обратете се на шалтер или на телефонскиот број на превозникот					
Превозник	Тргува од	Тргува во	Дестинација	Цена	
galeb	Охрид	Пон. 09:30 ч.	Скопје	500 ден.	 Избриши

Слика 26: Откажана линија

```

app.get("/user/add/:id", isLoggedIn, function(req, res) {
  Linija.findById(req.params.id, function(err, linija){
    if(err) {
      console.log(err);
    } else {
      User.findById(req.user.id, function(err, user){
        if(err) {
          console.log(err);
        } else {
          user.linii.push(linija);
          user.save(function(err, user){
            if(err){
              console.log(err);
            } else {
              res.redirect("/user");
            }
          });
        }
      });
    }
  });
});

app.get("/user/delete/:id", isLoggedIn, function(req, res) {
  User.findById(req.user.id, function(err, user){
    if(err) {
      console.log(err);
    } else {
      for(var i=0;i<user.linii.length;i++){
        if(user.linii[i]._id==req.params.id)
          user.linii.splice(i, 1);

        user.save(function(err, user){
          if(err){
            console.log(err);
          } else {
            res.redirect("/user");
          }
        });
      }
    }
  });
});

```

Слика 27: Руты за купување и бришење билети

Линкот на копчето “Купи Билет” води кон патеката “/user/add/:id” каде прво се пребарува линијата во базата на податоци според уникатното *ID* што е автоматски додадено за секој запис во базата од страна на *MongoDB*. Потоа таа линија се додава во листата на линии на тековно најавениот корисник (*user.linii.push(linija)*).

Слично е и за бришење на постоечка линија. Со притискање на копчето “Избриши” се пристапува до рутата “user/delete/:id” каде *ID* е автоматско генерираниот стринг идентификатор од базата. Прво се пребарува тековно најавениот корисник, а потоа се брише линијата што е зададена со тој *ID* (*user.linii.splice(i, 1)*). Во случајов корисничките линии се третираат како обична низа.

Избришаните линии се читаат од датотеката *izbrishaniLinii.json*.



## 4.7 Плаќање

Откако корисникот е завршен со купување на билети, тој треба да плати. Плаќањето е овозможено директно *online* (фиктивно). Со притискање на копчето “Плати” му се отвара модален прозорец каде што тој треба да ги внесе информациите за неговата кредитна или дебитна банкарска картичка. Сите полиња се задолжителни и се валидираат.

### Информации за плаќање

960 МКД



Целосно име и презиме



Борјан Коњановски

Број на картичка



XXXX-XXXX-XXXX-XXXX

Истекува

ММ



/

YY



CVV код?

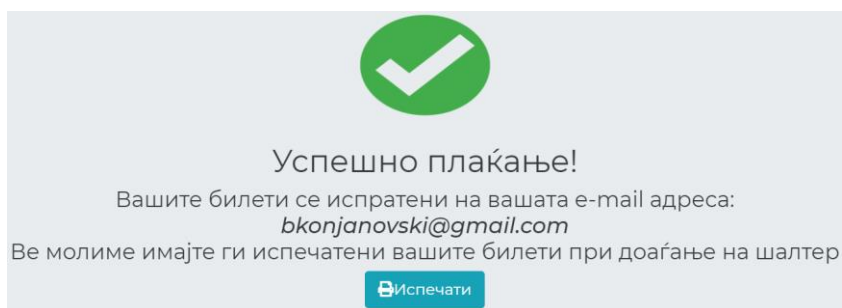
...

Плати

Откажи

Слика 28: Прозорец за плаќање

Доколку плаќањето е успешно, се појавува страна каде што корисникот треба да ги испечати платените билети за потврда. Платените билети ќе се тргнат од неговата листа на билети на корисничката страница.



Слика 29: Успешно плаќање


```

app.post("/user/uspeshnoPlakanje", isLoggedIn, function(req, res){
  setTimeout(function() {res.render("uspeshnoPlakanje", {izbrisanilinii: izbrisanilinii}), 5000);
  User.update({username : req.user.username}, { $set: { linii: [] }}, function(err){
    if(err) {
      console.log(err);
    } else {
      console.log("Uspeshno plakanje");
    }
  });
});
});

```

Слика 30: POST рута за успешно плаќање

Во рутата се бришат постоечките линии и корисникот се пренасочува на страницата за успешно плаќање. Тука се прикажани сите платени билети со можност за печатење.



### Успешно плаќање!

Вашите билети се испратени на вашата е-mail адреса:  
**bkonjanovski@gmail.com**

Ве молиме имајте ги испечатени вашите билети при доаѓање на шалтер

[Испечати](#)


Превозник	Тргнува од	Тргнува во	Дестинација	Цена
transkor	Прилеп	Сре. 15:00 ч.	Битола	250 ден.
rule	Струмица	Саб. 21:30 ч.	Охрид	650 ден.
rule	Штип	Сре. 12:00 ч.	Кратово	300 ден.

Вкупно за плаќање: ~~1200 ден.~~  
- 20% студентски попуст: **960 ден.**

[Врати се назад](#)

© 2020 Copyright: bkonjanovski @ Faculty of Computer Science & Engineering

Print 1 sheet of paper

Destination  HP LaserJet Professio

Pages All

Copies 1

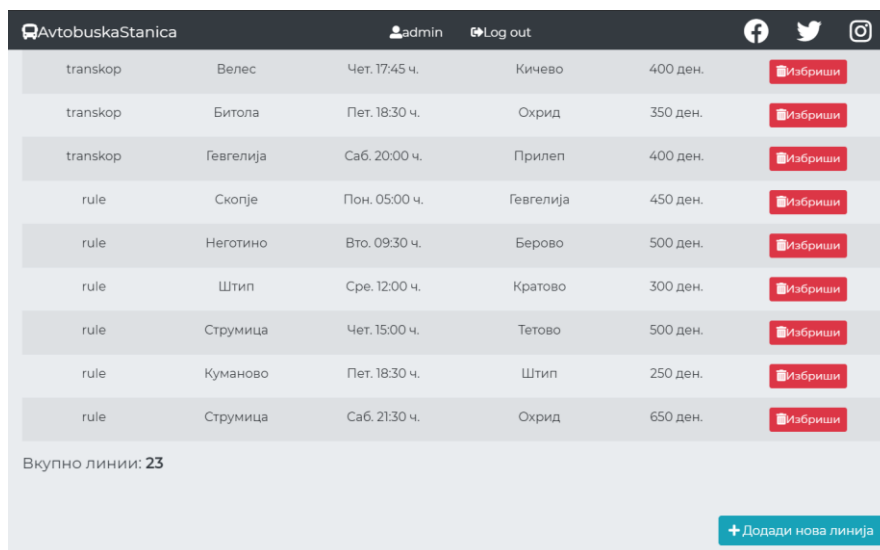
More settings

Print Cancel

Слика 31: Печатење билети

## 4.8 Администратор

На системот постои специјален корисник *Администратор* за додавање на нови линии и бришење на постоечките. До него се пристапува при најава со корисничко име *admin* и лозинка *RootAdministrator1*.



The screenshot shows the Admin interface for 'AvtobuskaStanica'. At the top, there's a header with the site name, user 'admin', a 'Log out' button, and social media icons. Below is a table of bus lines. Each row contains: company name, route, departure time, destination, price, and a 'Delete' button. At the bottom, it shows 'Вкупно линии: 23' and a '+ Додади нова линија' button.

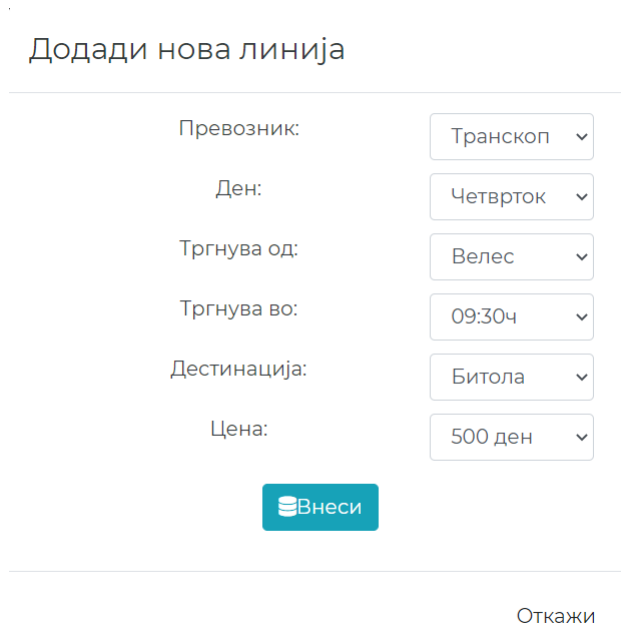
transkop	Велес	Чет. 17:45 ч.	Кичево	400 ден.	Избриши
transkop	Битола	Пет. 18:30 ч.	Охрид	350 ден.	Избриши
transkop	Гевгелија	Саб. 20:00 ч.	Прилеп	400 ден.	Избриши
rule	Скопје	Пон. 05:00 ч.	Гевгелија	450 ден.	Избриши
rule	Неготино	Вто. 09:30 ч.	Берово	500 ден.	Избриши
rule	Штип	Сре. 12:00 ч.	Кратово	300 ден.	Избриши
rule	Струмица	Чет. 15:00 ч.	Тетово	500 ден.	Избриши
rule	Куманово	Пет. 18:30 ч.	Штип	250 ден.	Избриши
rule	Струмица	Саб. 21:30 ч.	Охрид	650 ден.	Избриши

Вкупно линии: 23

+ Додади нова линија

Слика 32: Преглед на сите линии од админ

Со притискање на копчето “Додади нова линија”, се отвара модален прозорец во кој се внесуваат податоците за нова линија (превозник, ден, тргнување, дестинација и цена).



The screenshot shows a modal form titled 'Додади нова линија'. It contains several dropdown menus for selecting the transporter, day, departure time, destination, and price. A 'Внеси' button is at the bottom, and an 'Откажи' button is at the bottom right.

Додади нова линија

Превозник: Транскоп ▾

Ден: Четврток ▾

Тргува од: Велес ▾

Тргува во: 09:30ч ▾

Дестинација: Битола ▾

Цена: 500 ден ▾

Внеси

Откажи

Слика 33: Додавање нова линија

Кога некој корисник е најавен како администратор, тогаш неговиот username не е како на обичните корисници, нивната e-mail адреса, туку е едноставно *admin*.

Сите од страниците што содржат *CRUD* (*Create, Read, Update, Delete*) функционалности најпрво проверуваат за каков корисник се работи. За таа цел се поставува глобална променлива *currentUser* која го претставува тековно најавениот корисник на системот.

```
app.use(function(req, res, next) {
  res.locals.currentUser = req.user;
  next();
});
```

Слика 34: Поставување на глобална променлива *currentUser*

Оваа променлива понатаму се користи во сите страници каде што има потреба од проверка за администраторски привилегии. Доколку корисникот не е најавен, тој може само да ги прегледува линиите, без да купува билети. Ако корисникот е најавен како обичен корисник, тогаш може да купува билети и да ги брише од неговата листа на купени билети, како и да ги плаќа. Доколку корисникот е најавен како администратор, има можност да додава нови линии и да ги брише постоечките.

```
<% if(currentUser && currentUser.username == "admin"){ %>
  <a href="/delete/<%= linija._id %>" class="btn btn-danger btn-sm"
  onclick='return confirm("Дали сте сигурни дека сакате да ја избришете линијата?")'><i class="fa fa-trash-alt"></i>Избриши</a>
<% } %>
<% if(currentUser && currentUser.username != "admin"){ %>
  <a href="/user/add/<%= linija._id %>" class="btn btn-success btn-sm" ><i class="fa fa-cart-plus"></i>Купи билет</a>
<% } %>
</td>
<% }}; %>
```

Слика 35: Користење на *currentUser* за проверка на авторизација

```
<% if(currentUser && currentUser.username == "admin"){ %>
  <a data-toggle="modal" data-target="#myModal" class="btn btn-info float-right fadeIn third" >
    <i class="fa fa-plus mr-1"></i>Додади нова линија</a>
<% } %>
```

Слика 36: Овозможување опција за додавање на нова линија

## 5. Заклучок

Во оваа дипломска работа беше претставена веб апликација за резервација на автобуски билети.

Воведувањето на еден ваков модерен систем е неопходно во динамичното денешно време, имајќи во предвид дека многу голем дел на населението од сите старосни групи и социјални слоеви секојдневно користи автобуски превоз и едно вакво решение би им било од корист на сите. Ова апликативно веб решение не е иновација во самата смисла на зборот, бидејќи е достапно уште пред многу години во многу држави низ светот, но за жал сеуште не е достапно во нашата држава.

Секако оваа дипломска работа е само почетокот и може да служи за понатамошно развивање на еден проект од вакви размери. Следни чекори што би можеле да се превземеат во насока на комплетирање на овој систем се

- да се промовира и да се развие маркетинг стратегија за апликацијата
- да се земат во предвид најстарата категорија на патници
- да се прилагоди апликацијата за оние патници со посебни потреби
- да се додаде опција за најавување преку социјалните мрежи
- да се развијат соодветни мобилни апликации за паметните телефони
- да се овозможи системот за плаќање и преку телефонски броеви
- да се интегрира овој систем со некој постоечки студентски сервис
- да се интегрира овој систем со постоечки системи на јавен градски превоз
- да се надогради апликацијата со уште понапредни и понови технологии

и уште многу други можности.

За крај, кога една ваква услуга што е од јавен интерес би се појавила на пазарот, поддршката од различни страни, како и добрата заработка и профит од истата се загарантирани, како една дополнителна мотивација за реализирање на ваков проект.

## 6. Референци

- [1] <https://www.upwork.com/resources/what-is-a-software-stack>
- [2] [https://en.wikipedia.org/wiki/Cross-platform\\_software](https://en.wikipedia.org/wiki/Cross-platform_software)
- [3] <https://home.cern/science/computing/birth-web/short-history-web>
- [4] <https://getbootstrap.com/docs/4.3/about/overview/>
- [5] [https://en.wikipedia.org/wiki/Responsive\\_web\\_design](https://en.wikipedia.org/wiki/Responsive_web_design)
- [6] <https://api.jquery.com/>
- [7] <https://nodejs.org/en/about/>
- [8] <https://docs.npmjs.com/about-npm>
- [9] <https://expressjs.com/en/4x/api.html>
- [10] [https://en.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)
- [11] <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>
- [12] <http://www.passportjs.org/docs/>
- [13] <https://docs.mongodb.com/manual/>
- [14] <https://www.mongodb.com/nosql-explained>
- [15] <https://aws.amazon.com/cloud9/>