SUSTAV ZA ORGANIZIRANJE PRIJEVOZA OSOBA

Duboković, Rino

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Algebra University College / Visoko učilište Algebra**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:225:858194

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-19**

ALGEBRA VISOKO UČILIŠTE

Repository / Repozitorij:

Algebra Univerity College - Repository of Algebra Univerity College

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

SUSTAV ZA ORGANIZIRANJE PRIJEVOZA OSOBA

Rino Duboković

Zagreb, Siječanj 20.

Student vlastoručno potpisuje Završni rad na prvoj stranici ispred Predgovora s datumom i oznakom mjesta završetka rada te naznakom:

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, datum.

Ime Prezime

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

U radu je opisana web aplikacija „Drite“ koja pomaže spajanju putnika i vozača u gradskom prometu, tehnologija korištena u izradi web aplikacije, arhitektura i komponente sustava kao i nedostaci, prednosti i planirana proširenja. Kroz uvod je opisana motivacija za izradom cijelog projekta, dok je u zaključku stavljen naglasak na dobrim potvrđenim načinima izrade rada i prednostima koji nose moderne tehnologije.

The paper describes the web application "Drite" that helps connect passengers and drivers in urban traffic, the technology used to create the web application, the architectures and components of the system, as well as the disadvantages, advantages and planned extensions. The introduction describes the motivation behind the idea of the whole project, while the conclusion emphasizes the well-proven ways of designing work and the benefits of modern technologies.

Ključne riječi: dijeljenje prijevoza, smanjenje CO2, MVP, smanjenje gužva, promet, Drite, ušteda goriva, internetska tržišta

Sadržaj

1. Uvod 1

2. Pregled korištenih tehnologija 2

2.1. ASP.NET MVC 3

2.2. SQL Server 6

2.3. ORM 8

2.3.1. Entity Framework 9

2.4. Ostale tehnologije 11

2.4.1. Bootstrap 11

2.4.2. JQuery 12

2.4.3. JavaScript 14

3. Arhitektura sustava 15

3.1. Baza podataka 19

3.2. Web aplikacija 21

3.3. Komponente sustava 21

3.3.1. Autentikacija korisnika 23

3.3.2. Grafičko korisničko sučelje 24

3.3.3. Administratorsko sučelje 24

3.3.4. Sustav vrednovanja vozača i putnika 25

3.3.5. Obavještavanje korisnika 25

3.3.6. Komunikacijski sustav 25

3.3.7. Pregled rezervacija 26

4. Prednosti, nedostaci i planirana proširenja 27

5. Zaključak 29

Literatura 30

Popis slika..... 32

Popis kôdova 33

1. Uvod

„Drite“ ili sustav za organiziranje prijevoza osoba je jednostavna web aplikacija koja omogućava dijeljenje prijevoza / vozila između „vozača“ koji putuju na određenim relacijama i imaju viška prostora u autu i „putnika“ koji imaju potrebu stići do određene točke na brži, jeftiniji, mirniji i dostupniji način. „Vozači“ mogu jednostavno unijeti relaciju puta kroz web aplikaciju, a „putnici“ mogu jednostavno rezervirati prijevoz.

Ciljana skupina su ljudi koji mogu unijeti unaprijed na kojim će se relacijama voziti u budućnosti kao npr. ljudi koji svakodnevno idu na posao u isto vrijeme. Ova web aplikacija je napravljena s funkcijama koje imaju većina internetskih tržišta (engl. *online marketplace*)[1], a to su:

- Sustav pregleda ponude odnosno unesenih dostpunih prijevoza
- Sustav rezerviranja ponude odnosno prijevoza
- Sustav za komunikaciju između “vozača” i “putnika”
- Sustav recenzija koji “putnik” može staviti za određeni prijevoz koji je rezervirao
- Registracija i prijava korisnika na stranicu
- Forma za unos ponude odnosno prijevoza

„Drite“ je napravljen s ciljem uštede goriva, smanjenja ispuštanja stakleničkih plinova, smanjenja financijskih troškova za putnike kao i za vozače, smanjenja gradskog prometa odnosno gužva u prometu, povećanja socijalizacije između ljudi koji se koriste navedenom aplikacijom.

Izvori za izradu rada su knjige navedene pri kraju rada i internetski sadržaji. Stručno znanje za izradom aplikacije stečeno je na visokom učilištu Algebra tokom školovanja. Kroz rad opisane su tehnologije korištene u projektu, arhitektura baze podataka i svaka pojedina komponenta sustava. Za kraj rada opisani su nedostatci, proširenja i prednosti ove aplikacije.

[1] Online marketplace je web lokacija za e-trgovinu na kojoj sadržaj aplikacije stavljaju više strana, dok vlasnik marketplace-a je zadužen za plaćanje između dvije ili više strana aplikacije. Npr. Uber, Ebay, Booking.com, Kickstarter (Wikipedia, 2020).

2. Pregled korištenih tehnologija

„Drite“ je dinamička web aplikacija što znači da joj se može pristupiti samo preko interneta ili intraneta kroz web preglednik. Razlika dinamičke web aplikacije i statične web aplikacije je u funkcionalnosti i dinamičnosti.[2] Metaforički statičnu web aplikaciju možemo usporediti sa drvenom tablom (pano-om) gdje lijepimo razne post-it papiriće na kojima su razno razne informacije, ali u trenutku kada želimo promijeniti neku informaciju moramo izbrisati poruku i napisati novu ili staviti novi papirić. Taj dio je statičan jer mi moramo ići u kôd i mijenjati ručno stvari. Dok za primjer dinamičke stranice možemo uzeti kalkulator. U kalkulator se unesu željeni brojevi s funkcijama, ali konačni rezultat funkcije je dinamička radnja, to je onaj dio koji mi nismo ručno napravili nego je aplikacija za nas napravila.

Za izradu dinamičke web aplikacije korišteni su slijedeći programski jezici:

- C#
- JQuery
- JavaScript

Stilskim i markup jezicima:

- CSS – stilski
- HTML - markup

Također su korišteni i sljedeći okviri (engl. *Frameworks*)

- Bootstrap
- Entity Framework
- ASP.NET MVC

Za bazu podataka korišten je okvir Entity Framework koji generira SQL server bazu u pozadini.

[2] Wikipedia, 2020

2.1. ASP.NET MVC

.NET je programerska platforma koja sadrži razne alate, programske jezike, i biblioteke (engl. *Libraries*) koji služe za izradu različitih vrsta aplikacija. ASP.NET je produžetak .NET platformi koji sadrži alate i biblioteke specifične za izradu web aplikacija. Ovo su samo od nekih stvari kojima ASP pridonosi .NET platformi i koje su korištene u projektu:

- Sintaksa predložaka za web stranice zvana Razor
- Biblioteka za uobičajene web obrasce poput MVC-a (engl. *Model-View-Controller*)
- Sistem autentikacije

ASP.NET je otvorenog izvora (engl. *open source*) što znači da svaki programer može doprinijeti razvoju ASP.NET platforme. ASP.NET aplikacije se mogu razvijati i pokretati na ostalim operacijskim sustavima:

- Linux
- macOS
- Windows
- Docker

Razor je serverski markup jezik i koristi se za izradu dinamičkih stranica u C# programskom jeziku. Serverski jezik znači da se vrti na serveru i može manipulirati bazom podataka, dok markup jezici su programski jezici za označavanje podataka i njime se opisuju podatci i dokument. Razor se označava sa znakom „@“ prije upisivanja teksta.

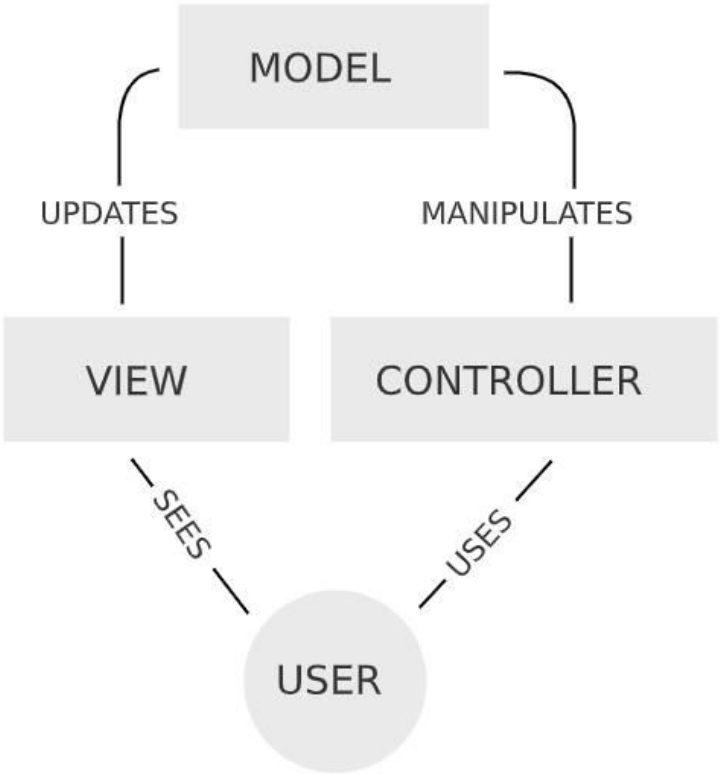
Primjer kôda koji ispisuje godinu koja je u trenutku izvođenja kôda:

```
@DateTime.Now.Year
```

MVC je obrazac softverske arhitekture koji se koristi da programski kôd razdvoji u 3 logičke cjeline. Osim razdvajanja logičkih cjelina koristi se još i za ponovno i paralelno korištenje programskog kôda.[3]

[3] Galloway et al., 2014; Chadwick et al., 2012; dotnet.microsoft.com, 2020; w3schools.com 2020; docs.microsoft.com, 2020

- Model - centralni dio obrasca. dinamička struktura podataka aplikacije koja izravno upravlja podatcima, logikom i pravilima aplikacije i neovisna je o korisničkom sučelju
- Pogled (engl. *View*) – zaslužan za prikaz svake informacije na web stranici
- Kontroler (engl. *Controller*) – element koji manipulira modelom kroz komandne korisnika koji dolaze iz View elementa



Slika 2.1 Grafički prikaz MVC-a

C# programski jezik se koristi u Razor-u prilikom izrade logike i pristupa podacima u ASP.NET-u. C# je objektno orijentirani programski jezik koji je napravljen da iskoristi mogućnosti .NET platforme.

Neke od prednosti ASP.NET MVC nad ASP.NET Web Form-e:[4]

- Potpuna kontrola nad prikazanim HTML-om
- Čisto razdvajanje briga (engl. *separation of concerns*)
- Omogućava razvoj temeljen na testovima (engl.*Test-driven development*) jer omogućava da možemo testirat u malim ciklusima, manje stvari[5]
- Laka integracija sa JavaScript okvirima

ASP.NET Web Form je najstariji način korištenja ASP.NET platforme, omogućava izradu dinamičkih web aplikacija kroz povuci-i-pusti (engl. *drag-and-drop*), event-driven model koji je jako sličan izradi aplikacija za desktop uređaje.[6]

[4] Stackoverflow, 2020
[5] Wikipedia, 2020
[6] dotnet.microsoft.com, 2020

2.2. SQL Server

SQL Server je relacijska baza podataka kojoj je primarni jezik za upite Transact SQL (T-SQL), što znači da osim osnovnih i klasičnih SQL upita (kao SELECT upit) dozvoljava i složenije stvari poput mijenjanja programskog toka (IF-ELSE naredba).[7] U priloženom kodu možemo vidjeti IF-ELSE naredbe pisane u Transact SQL-u. Ovisno o varijabli @MARK program ispisuje različite nazive ocjena, tako da npr. ako je @MARK varijabla jednaka broju 1 program će ispisati: „Disappointing“. Ocjene se koriste za ocjenjivanje „vozača“ od strane „putnika“.

```
IF (@MARK==1)
      PRINT 'Disappointing';
ELSE IF (@MARK==2)
      PRINT 'Bad';
ELSE IF (@MARK==3)
      PRINT 'Good';
ELSE IF (@MARK==4)
      PRINT 'Super';
ELSE IF (@MARK==5)
      PRINT 'Great';
```

Kôd 2.1 Primjer IF ELSE naredbe u SQL

SQL Server je korišten u web.config [8] datoteci gdje je preko Connection string-a definirano spajanje na bazu. Web.config datoteka služi za upravljanje connection string-om baze, upravljanje ponašanja grešaka i sigurnosti stranice ili specifičnog direktorija.[9] U connection string-u definiramo informacije o izvoru podataka i način spajanja na izvor podataka.

```
<connectionStrings>
    <add name= "WebShopContext"
    connectionString="Data Source=(localdb)\mssqllocaldb;
```

[7] Ben-Gan, 2016; Wikipedia, 2020
[8] Naziv datoteke
[9] uk.godaddy.com, 2020; Quora, 2020

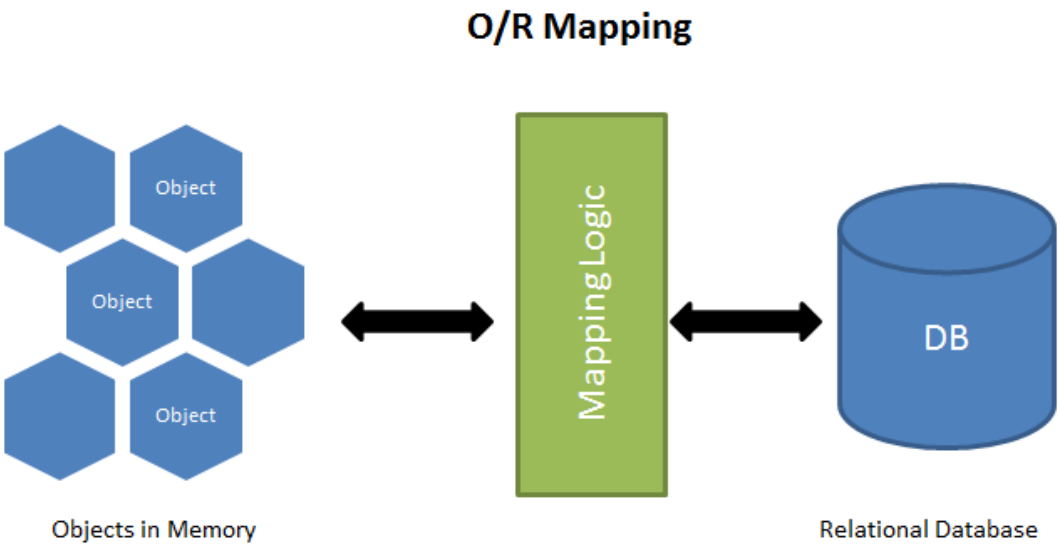
```
    Initial Catalog=TestingFinal;
    Integrated Security=SSPI;" providerName="System.Data.SqlClient" />
</connectionStrings>
```

Kôd 2.2 Primjer Connection string-a korišten u radu

Ostatak SQL Servera se samo-generirao u pozadini korištenjem Entity Framework – a i ORM-a.

2.3. ORM

ORM (engl. *Object-relational mapping)* je programska tehnika za konvertiranje tipnih sustava (engl. *Type systems)* koristeći se objektno-orijentiranim programskim jezikom. U programskim jezicima, tipni sustav je skup pravila koji različitim svojstvima računalnog programa dodjeljuje svojstvo kao što su varijable, izrazi, funkcije ili moduli.[10] Objektno-orijentirano programiranje se bazira na objektima koji mogu sadržavati podatke u obliku polja i programski kôd u obliku procedura. Svojstvo objekata su njegove procedure koje mogu dohvatiti i mijenjati podatke objekta s kojim su povezani.[11] Entity Framework se koristio kao ORM koji je generirao SQL Server bazu.



Slika 2.2 Grafički prikaz ORM-a

[10] Wikipedia, 2020
[11] Wikipedia, 2020

2.3.1. Entity Framework

Entity Framework je ORM koji programerima koji rade s .NET tehnologijom olakšava rad s bazom i manipuliranjem .NET objekata. Također, skraćuje pisanje programskog kôda koji je zaslužan za pristup podacima. Postoje tri različite vrste pristupa implementacije Entity Framework-a:[12]

- Prvo kôd (engl. *First code*)
- Prvo baza podataka (engl. *Database First*)
- Prvo model (engl. *Model First*)

Kod Code First-a se prvo kreiraju klase entiteta sa definiranim svojstvima i zatim Entity Framework prilikom pokretanja .NET kôda kreira bazu odnosno tablice i odnose temeljene na klasama entiteta. Nakon svake zadovoljavajuće promjene baze poželjno je dodati migraciju koja nam omogućava spremanje stanja baze podataka i povratak u bilo koju migraciju u željenom trenutku.

Pristup Code First-a je korišten na ovom projektu. Kod Database First-a prvo se naprave tablice i baza zatim se stvori Data Model od prethodno kreirane baze.

Kod Model First-a baza, tablice i odnosi se mogu kreirati preko korisničkog sučelja, klikom miša. Nakon što poklikamo željenu strukturu baze, Entity Framework za nas odradi sav posao kreiranja u pozadini.

Prilikom pokretanja Code First-a potrebno je napisati i izvršiti sljedeću komandu u konzoli: „Install-Package EntityFramework.“ koja omogućuje instalaciju EntityFramework-a za taj projekt.

Postoje dvije vrste definiranja odnosa između entitetskih klasa:

- Data Annotations – da li je varijabla primarni, strani ključ, obavezna, neobavezna, nulabilna…itd. definiramo tako da stavimo iznad varijable oznaku. Npr. ako želimo ograničiti unos broja mjesta u autu između 1 i 10 možemo iznad varijable BrojMjesta tipa integer staviti oznaku „Range“, također ako želimo ispisati poruku korisniku u slučaju da je stavio broj koji nije u tom opsegu brojeva možemo dodati oznaku „ErrorMessage“:

[12] Wikipedia, 2020

```
[Range(1, 10,ErrorMessage = "You cannot put more than 10 places in car")]
public int BrojMjesta { get; set; }
```

Kôd 2.3 Primjer korištenja Dana Annotations-a

- Fluent API – za korištenje fluent API-a trebamo premostiti OnModelCreating metodu u DbContext-u i u njoj definirati ponašanje određene entitetske klase. Tako npr. ako želimo definirati email kao obavezan unos za studenta možemo napisati:[13]

```
modelBuilder.Entity<Student>().Property(s ⇒ s.Email).IsRequired();
```

Prednosti Code First pristupa:

- Može se kreirati baza i tablice iz objekta
- Kontroliranje verzije baze kroz migracije – možemo vratiti bazu u određenu točku vremena odnosno migraciju
- Dobra za manje aplikacije

Nedostaci:

- Sve vezano za bazu podataka se mora pisati unutar Visual Studia
- Stored procedure se moraju pisati unutar programskog koda
- Za mijenjanje baze se mora promijeniti klasa entiteta i onda pokrenuti „update-database“ komandu
- Ne preferira se za aplikacije koje manipuliraju s puno podataka

Prednosti Database First pristupa:

- Jednostavan za kreiranje podatkovnog modela
- Grafičko korisničko sučelje koje omogućava lako kreiranje i povezivanje ključeva bez pisanja koda
- Preferira se za aplikacije koje manipuliraju s puno podataka

Nedostaci:

- Prilikom kreiranja podatkovnog modela se generira puno automatski generiranog koda koji kasnije otežava snalaženje u slučaju nekakvih grešaka.

[13] tutorialspoint.com, 2020.

- Prilikom dodavanja funkcionalnosti se mora produžiti (engl. *extend*) generirana klasa.[14]

2.4. Ostale tehnologije

U ostale tehnologije su stavljeni programski jezici koji se koriste na front-endu. Dijelovi su samo-generirani u pozadini poput Bootstrap-a, dok JQuery se koristi za brzi prikaz informacija na stranici kroz Ajax pozive. Dodatano, JavaScript se koristi za dizajn odabira datuma (engl. *date picker*), prikaz google mape s rutom početne i odredišne točke i prijedlog adrese prilikom unošenja lokacije.

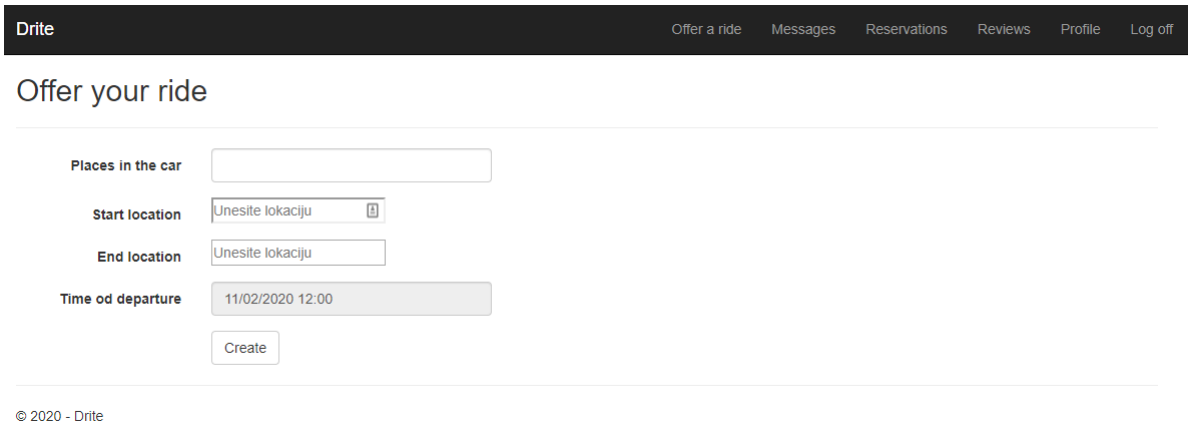
2.4.1. Bootstrap

Bootstrap je besplatni CSS (engl. *Cascading Style Sheets*) okvir usmjeren na responzivnost u front-end programiranju. Sadrži CSS i opcionalne predloške bazirane na Javasciptu za tipografiju, forme, gumbove, navigaciju i ostale komponente na sučelju. CSS je stilski jezik koji se koristi za dizajn i manipulaciju prezentacije dokumenta napisanog pomoću HTML jezika. HTML je markup jezik koji se koristi za izradu web stranica. HTML označava različite tipove sadržaja – tekst, slika, video, poveznice itd. koji omogućava web pregledniku da ih prepozna i prikaže na stranici.[15] Na slici (

Slika 2.3) možemo vidjet formu za stavljanje prijevoza koja uključuje varijable „Places in the car“ koja definira broj ljudi koje vozač želi primiti u auto prilikom vožnje, „Start location“ kao početna točka putovanja, „End location“ odredišna točka putovanja, „Time of departure“ datum i vrijeme kada kreće i gumb „Create“ koji klikom stvara i pohranjuje prijevoz u bazu podataka gdje je prijevoz nakon spremanja u bazu spreman za nadolazeće rezervacije.

Za navigacijsku traku i cijelo vidljivo sučelje je korišten samo generirajući Bootstrap.

[14] c-sharpcorner.com, 2020.
[15] Wikipedia, 2020



Slika 2.3 Grafičko korisničko sučelje u Bootstrap-u

2.4.2. JQuery

JQuery je JavaScript biblioteka koja je napravljena da pojednostavi JavaScript. JQuery bibiloteka je skraćeni oblik JavaScript-a, sve što može JavaScript, JQuery to može u manje linija koda. JQuery nam nudi niz gotovih funkcija za jednostavno implementiranje i manipuliranje html elemenata. Kroz web aplikaciju se koristi JQuery Ajax pozivi koji se koriste za izradu asinkronih web aplikacija koji se koriste za slanje i dohvaćanje podataka u bazu u pozadini (asinkrono). Na taj način korisnik dobiva osjećaj instantnosti da se radnja događa u stvarnom vremenu bez otvaranje nove stranice ili osvježavanja postojeće.[16] Na Slici (Slika 2.4 Prikaz ajax poziva u PogleduSlika 2.4) možemo vidjeti klijentsku stranu AJAX poziva gdje na početku ulovimo događaj kada je kliknut „Submit“ gumb i preveniramo uobičajeno radnju „Submit“ gumba koja automatski šalje podatke na server i otvara novu stranicu. Zatim pozivamo metodu „addMessage“ u kojoj se šalje kontroleru poruka i „id“ razgovora. Nakon što kontroler izvrši uspješno spremanje poruke u bazu podataka vraća objekt u JSON formatu kojeg stavlja u tablicu napisanu u „Pogledu“[17]

[16] Wikipedia, 2020; Duckett, 2014
[17] Wikipedia, 2020

```
@section scripts{

    <script>

    $("#submit").click(function (e) {

        e.preventDefault();
        addMessage();

    });

    function addMessage() {

        $.ajax({
            url: '@Url.Action("AddMessage","Home")',
            data: { note: $("#message").val(), conversationId: $("#conversationId").val()},
            success: function (messageJson) {

                $("#message").val("");



                var result = "";

                    result += "<tr>";
                result += "<td>" + messageJson.Sender+"</td>";
                result += "<td>" + messageJson.Note+"</td>";
                result += "<td>" + messageJson.DateTimeMessage+"</td>";
                result += "<td>" + messageJson.Recipient +"</td>";
                    result += "</tr>";



                $('#table').append(result);

            }
        })
    }
    </script>

}
```

Slika 2.4 Prikaz ajax poziva u Pogledu

Na slici (Slika 2.5) u kontroleru možemo vidjeti da metoda prima poruku i „id“ razgovora, preko id-a aplikacija nalazi razgovor s tim id-em“. Poruku, pošiljatelja i razgovor tog „id-a“ sprema u bazu podataka i stvara novi objekt MessageJson koji prima sve navedene varijable plus primatelja i trenutno vrijeme slanja. Zatim šalje nazad na klijentsku stranu gdje Pogled to stavlja u tablicu odnosno prikazuje korisniku.

```
public ActionResult AddMessage(string note, string conversationId)
{
    Message message = new Message();
    Conversation conversation = db.Conversations.Find(Guid.Parse(conversationId));

    ApplicationUser user = db.Users.FirstOrDefault(u => u.Email == User.Identity.Name);

    message.Note = note;
    message.Sender = user;
    message.Conversation = conversation;


    db.Entry(conversation).State = EntityState.Modified;
    db.Messages.Add(message);
    db.SaveChanges();



    MessageJson messageJson = new MessageJson();

    messageJson.Note = message.Note;
    messageJson.DateTimeMessage = message.DateTimeMessage.ToLocalTime().ToString();
    messageJson.Sender = user.FirstName;
    messageJson.Recipient = conversation.Sender.FirstName;


    return Json(messageJson, JsonRequestBehavior.AllowGet);
}
```

Slika 2.5 Prikaz ajax poziva u kontroleru

2.4.3. JavaScript

JavaScript je skriptni dinamički programski jezik koji omogućava web aplikaciji veću interaktivnost i lakše korištenje. Koristi se da korisničko sučelje napravi ljepšim (kao animacije), interakciju s korisnicima (kao validacija varijabli) i za kontrolu html objekata.[18]

Iz sljedeće usporedbe dvaju kôda možemo vidjeti generalnu razliku dva programska jezika. Oba koda rade istu funkciju – dohvaćaju (engl. *select*) element klase naziva „ride“ [19]

```
document.getElementsByClassName("ride");
```

Kôd 2.4 Primjer JavaScript koda

```
$("ride");
```

Kôd 2.5 Primjer JQuery koda

[18] Duckett, 2014
[19] c-sharpcorner.com, 2020

3. Arhitektura sustava

Arhitektura cijelog sustava je podijeljena na Model-Kontroler-Pogled način koji je opisan u 2.1 cjelini. Kontroleri su napravljeni prema Modelima, svaki Model ima svoj kontroler koji upravlja njime tako npr. za Model „Reservation“ postoji kontroler „ReservationController“. Osim navedenih kontrolera postoje još i AccountController i ManageController koji su samo-generirani od strane ASP.NET-a i oni su zaduženi za registraciju i prijavu korisnika na stranicu. Postoji još i HomeController u kojem se nalaze sve važnije funkcije – pretraživanje prijevoza, objava prijevoza, slanja poruka između korisnika, rezerviranja prijevoza. Pogledi (*engl. Views)* su raspoređeni u datotekama po nazivima kontrolera i u svakoj datoteci su različiti Pogledi ovisno o potrebama aplikacije. Tako npr. za „Review“ datoteku imamo „CreateReview.cshtml“, „Reviews“ i „ViewReview.cshtml“ gdje je za potrebe ove aplikacije bilo potrebno da korisnik može davati recenziju (CreateReview.cshtml) vozaču i da „vozač“ može pregledavati recenzije (Reviews.cshtml) koje je dobio i više pojedinoj recenziji (ViewReview.cshtml). Također osim navedenih datoteka postoji i „Shared“ datoteka u kojoj se nalaze dijeljenje stavke cijelog projekta poput zaglavlja, podnožja koji se nalaze na svakoj stranici web aplikacije. Osim navedenog MVC dijela kreirane su sljedeće datoteke:

- „Enums“ – za stavljanje enumeracija „Mark“ za potrebe ocjene recenzija i „Sex“ za odabir spola korisnika prilikom registriranja. Pošto takvi tipovi podataka ne postoje unutar standardnog tipnog sustava, napravljene su ručno

Kontroleri Drite aplikacije:

- AccountController – zadužen je za upravljanje korisničkih računa. Registracija/stvaranje i mijenjanje računa, prijava i odjava korisnika sa računa
- ConversationController – služi za dohvaćanje svih razgovora pojedinog korisnika i za dohvaćanje jednog određenog razgovora koje kontroler prosljeđuje Pogledu na ispis
- HomeController – glavni kontroler koji je zadužen za početnu stranicu i cijeli tijek od početne stranice prema pretraživanju prijevoza i rezervaciju prijevoza. Kontroler je zadužen za sve ekrane i funkcije koji nisu unutar vozačkog, putničkog i

administratorskog sučelja. Također, obavještavanje e-mailom korisnika koji traži prijevoz i slanje poruka između korisnika

- MessageController – dohvaća sve poruke za prijavljenog korisnika
- ProfileController – zaslužan za prikaz i mijenjanje profilnih podataka kao što su broj mobitela, opis osobe, adresa, opis auta, registracijski broj tablica, spol, da li je vozač
- ReservationController – dohvaća sve rezervacije za prijavljenog korisnika i prikazuje ih
- ReviewController – sprema svaku novu recenziju i omogućava pregled recenzija za prijavljenog korisnika
- RideController – dohvaća sve vožnje za prijavljenog korisnika i šalje ih Pogledu na prikaz
- UserController – omogućava prikaz svih korisnika koji su ikad registrirani na aplikaciji i mogućnost brisanja profila odnosno mijenjanja podataka. Kontroler koji je napravljen za administratorske potrebe kontrole svih korisnika na platformi.

Iz priloženog je vidljivo da svaki kontroler ima nastavak Controller, to je obavezni nastavak za imenovanje svakog kontrolera jer omogućava ASP.NET-u da preko imenovanje prepozna klase u kojem se nalaze kontroleri, pogledi i modeli. Pogledi koji se nalaze u „View“ datoteci su nazvani po metodama u kontrolerima. ASP.NET spoji naziv Pogleda sa nazivom metode u kontroleru. Dobra praksa imenovanja da su nazivi metoda, Pogleda, kontrolera samo objašnjivi kao što je primjer „Reservations“ gdje možemo očekivati prikaz svih rezervacija.

Datoteke i Pogledi Drite aplikacije su:

- Account:
  - CreateAccount – služi za dovršetak izrade računa. Kroz taj Pogledu korisnik može upisati sve opcionalne podatke profila adresa, mobitel…itd.
  - EditAccount – forma za mijenjanje prethodno navedenih podataka
  - Login – forma za prijavu korisnika unutar web aplikacije. Email i lozinka su potrebni za autentikaciju korisnika na Drite platformi
  - Register – forma za unos obaveznih početnih podataka potrebnih za kreiranje računa
- Conversation
  - Conversation – prikaz razgovora koji omogućava instant komunikaciju s drugim korisnikom
  - Conversations – prikaz svih razgovora za prijavljenog korisnika

- Home
  - EmailNotification – forma za unos email-a koja se pojavljuje ukoliko nema prijevoza sa traženim parametrima, a korisnik je prijavljen na aplikaciju
  - FinishReservation – kada je korisnik rezervirao prijevoz prikazuje mu informacijsku poruku da je rezervirao
  - Index – početna stranica na kojoj se nalazi forma za pretraživanje prijevoza i dodatne informacije o Drite aplikaciji
  - Message – forma za unos poruke koja se šalje prilikom traženja prijevoza, ukoliko putnik želi kontaktirati vozača prije rezerviranje prijevoza
  - NotAvailable – prikazuje poruku da nema dostupnih prijevoza za tražene parametre, prikazuje se ukoliko korisnik nije prijavljen
  - OffferRide – forma za unos podataka i kreiranje vožnje - broj ljudi, lokacija početnog odredišta, lokacija završnog odredišta i vrijeme polaska
  - Reserve – prilikom pretraživanja možemo pogledati listu dostupnih prijevoza, a klikom na jedan objekt liste nam otvara Reserve Pogled koji sadrži više informacija o odabranom prijevozu i prikaz vozačke rute na mapi
  - ShowAvailableRides – prikazuje sve dostupne vožnje po traženim parametrima
- Profile
  - EditProfile – forma koja omogućava korisniku promjenu osobnih podataka
  - ProfileByUser – prikazuje profil za prijavljenog korisnika
- Reservation
  - Reservations – pregled svih rezervacija za prijavljenog korisnika
- Review
  - CreateReview – forma za kreiranje recenzija
  - Reviews – pregled svih recenzija za prijavljenog korisnika
  - ViewReview – pregled odabrane recenzije
- Ride
  - Rides – pregled svih vožnji za prijavljenog vozača
- Shared – dokument koji nazivom daje informaciju ASP.NET-u da se unutar te datoteke nalaze Pogledi koji su dijeljeni s drugim Pogledima. Koristi se za zaglavlja i podnožja i omogućava da na jednom mjestu definiramo zaglavlje i onda ga koristimo kroz veći broj Pogleda
  - _Layout – Pogled kojeg implementiraju svi Pogledi u Drite aplikaciji, unutar _Layout-a je definirano podnožje i zaglavlje

- _LoginPartial – Pogled koji definira različito zaglavlje ovisno o vrsti prijavljenog korisnika, Putnik, Vozač ili Admin
- Error – u slučaju greške u aplikaciji se prikazuje Pogled Error koji služi kao obavijest da je korisnik naišao na grešku aplikacije
- Success – u slučaju uspješno odrađenog zadatka prikazuje potvrdnu poruku da je uspješno izvršena radnja

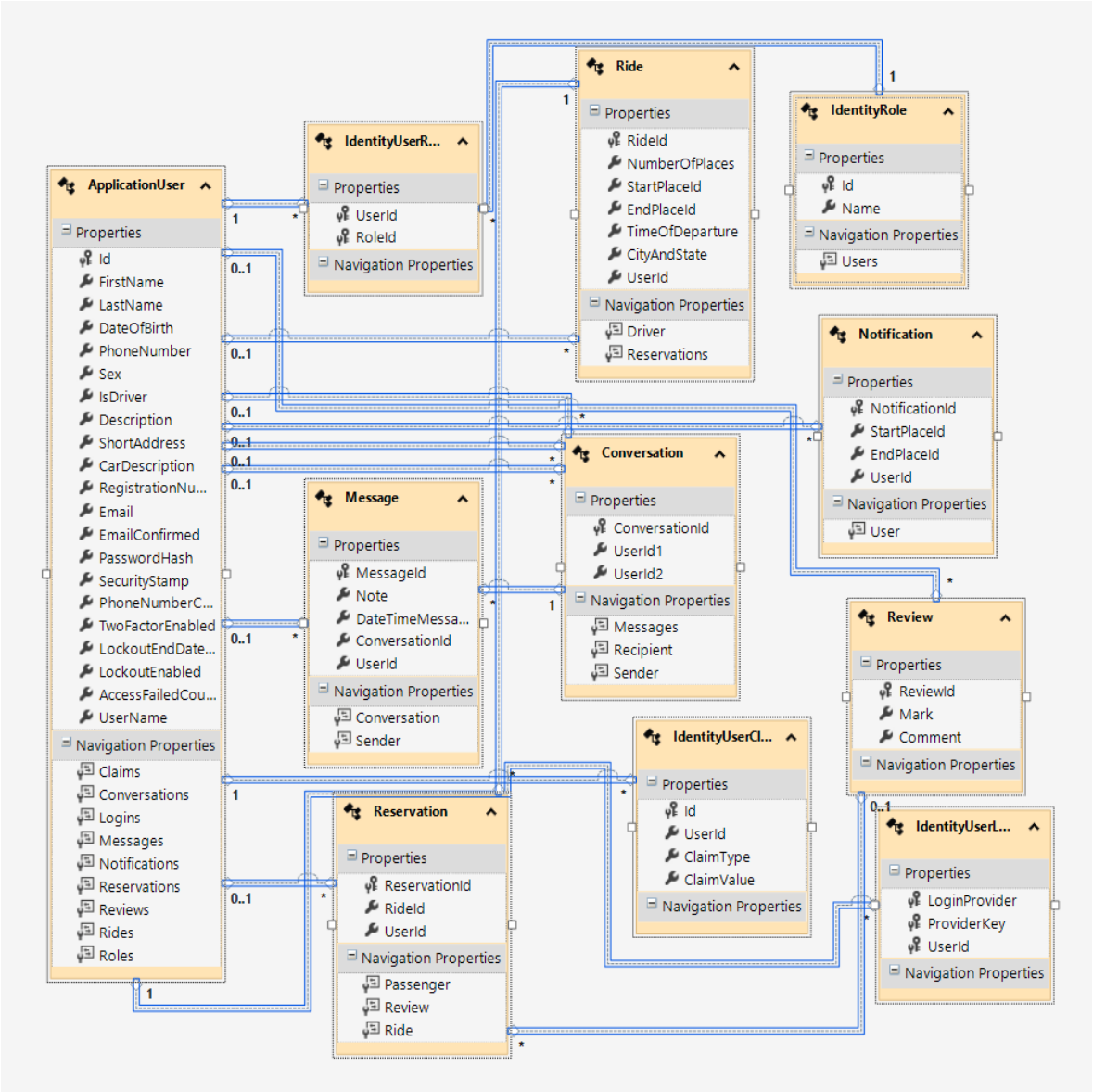
- User – pristup Pogledu je moguć samo preko administratorskog sučelja
  - Edit – omogućava adminu promjenu podataka o pojedinom korisniku
  - Users – pregled svih korisnika na jednom mjestu

3.1. Baza podataka

Baza podataka je konstruirana i samo-generirana kroz entitetske klase u EntityFramework first code pristupu. Prvi korak koji je poduzet u izradi baze podataka je bilo skiciranje tablica i odnosa između tablica. Nakon logičkog postavljanja baze, cijeli idejni dio je stavljen u kôd odnosno entitetske klase u ASP.NET-u. Baza podataka ima sljedeće entitete (tablice) odnosno Modele :

- ApplicationUser – najvažniji entitet koji predstavlja korisnika i pamti veliku količinu podataka o njemu kao što su ime i prezime, datum rođenja, broj mobitela, email itd.
- Ride – zaslužan za podatke o određenoj vožnji kao što su odredišna točka, polazišna točka, vrijeme itd.
- Notification – koristi podatke koji omogućavaju obavještavanje korisnika
- Reservation – podaci o rezervaciji odnosno tko je rezervirao određeni prijevoz
- Review – spojen s entitetom „Reservation“ jer samo ako postoji rezervacija može postojati i recenzija
- Message – tablica zaslužena za odgovore tko je kada, s kojom porukom kontaktirao kojeg korisnika
- Conversation – objedinjuje sve poruke između dvoje korisnika ako postoje
- IdentityUserLogin – zaslužan za odobravanje pristupanja pojedinog korisnika u sustav
- IdentityRole – entitet kojem se može definirati različite role
- IdentityUserRole – da li određeni korisnik spada u rolu „Driver“, „Passenger“ ili „Admin“

Svi navedeni entiteti su međusobno povezani odnosima odnosno ključevima koje možemo vidjeti na slici (Slika 3.1)



Slika 3.1 Dijagram baze podataka

3.2. Web aplikacija

Web aplikacija je klijentsko-serverski program koju klijent odnosno korisnik aplikacije pokreće u svojem web pregledniku. Ideja projekta više paše mobilnoj aplikaciji nego web aplikaciji, ali zbog većeg razumijevanja i poznavanja web tehnologija je izabrana web aplikacija. Pristup web aplikaciji je moguć samo preko localhost adrese koja predstavlja adresu računala na kojem se vrti aplikacija.[20]

3.3. Komponente sustava

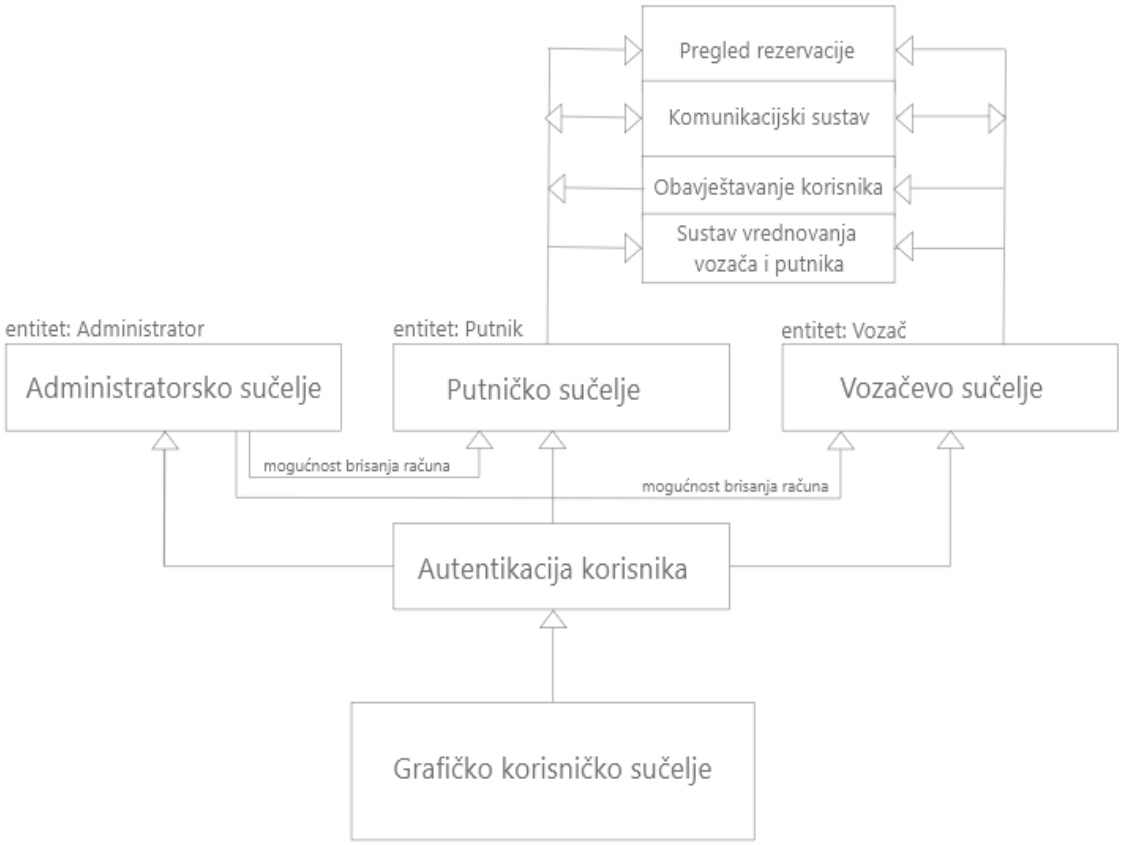
„Drite“ aplikacija je jednostavna aplikacija s osnovnim funkcionalnostima i komponentama. Komponente sustava su podijeljene na složenije dijelove cijelog sustava. Na slici (Slika 3.2) možemo vidjeti kako su sve komponente isprepletene / međuovisne u sustavu. Grafičko sučelje se proteže kroz cijeli vizualni dio aplikacije i uz pomoć njega možemo pristupiti autentikaciji korisnika odnosno klikom na gumb „Log in“ u desnom gornjem kutu aplikacije. Prilikom upisivanja pristupnih podataka sustav prepoznaje da li je riječ o korisniku, vozaču ili administratoru i sukladno tom svakog od navedenih kategorija korisnika šalje na svoje sučelje.

Administratorsko sučelje omogućava brisanje, mijenjanje i pregled računa svih korisnika.

Kroz putničko sučelje pristupamo pregledu prošlih i aktualnih rezervacija, komunikacijskom sustavu koji je dvosmjeran odnosno putnik može slati i primati poruke od vozača, obavještavanju korisnika o stavljenom prijevozu i pristup sustavu vrednovanja vozača.

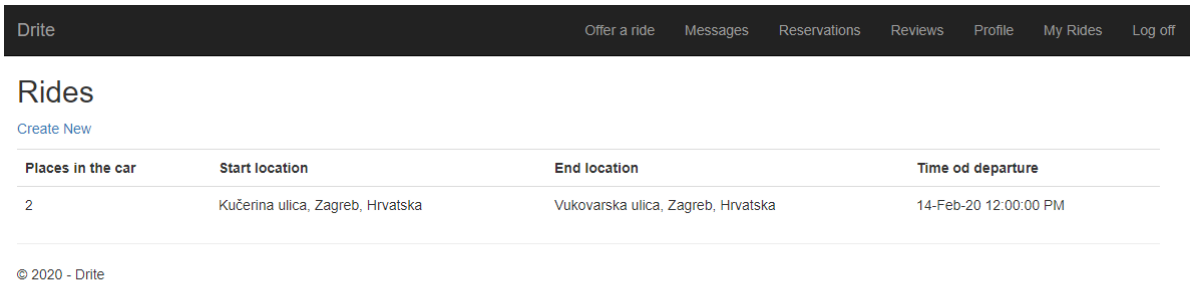
Vozač kroz svoje sučelje može pregledati svoje vožnje, komunicirat s putnikom preko komunikacijskog sustava, i prilikom stavljanja prijevoza sustav automatski obavijesti putnika ukoliko je na listi koje treba obavijestiti. Također, vozač ima pristup i putničkom sučelju, pa može vrednovati vozače i pregledavati rezervacije.

[20] whatismyipaddress.com, 2020



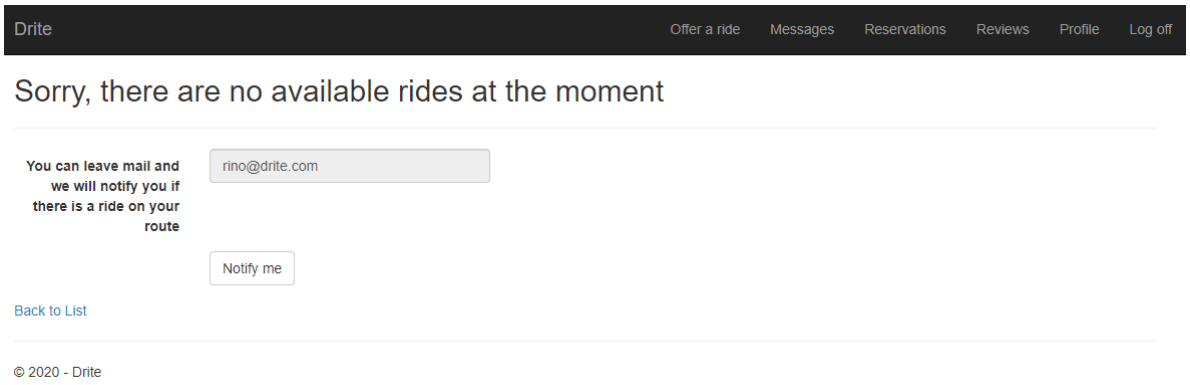
Slika 3.2 Dijagram međuovisnosti komponenata

Na slici (Slika 3.3) prikazan je izgled vozačevog sučelja i opcije koje ima u navigacijskoj traci. Vozač u pregledu prijevoza ima kreiran jedan prijevoz koji polazi iz Kučerine ulice u Zagrebu do Vukovarske ulice, 14.02.2020. u 12:00 i ima dva mjesta u autu.



Slika 3.3 Vozačevo sučelje

Na slici (Slika 3.3) prikazan je ekran koji se prikazuje u trenutku kada putnik ne može naći prijevoz za određenu rutu i vrijeme. U tom slučaju mu se prikazuje poruka da nema prijevoza, ali je moguće ostaviti mail na kojem dolazi obavijest u slučaju da se u međuvremenu taj prijevoz pojavio.



Slika 3.4 Obavještavanje korisnika

3.3.1. Autentikacija korisnika

Autentikacija korisnika se odvija preko samo generiranog kôda koji radi na način da prilikom registriranja sprema podatke o prijavi (email i lozinka) zatim prilikom prijave provjerava da li postoji taj email s upisanom istom lozinkom. Ako postoji prijavljuje ga u sustav i stavlja u kolačić koji omogućava korisniku dugoročnu prijavljenost u aplikaciji čak i u slučaju ako izađe iz web stranice. U bazu podataka se ne stavlja tekstualni „čisti“ oblik lozinke već lozinka sa hash-iranim kriptografskim ključem. Prilikom provjere lozinke postoji dekripcijski sustav koji dekriptira hash-iranu lozinku i onda je uspoređuje s unesenom. Na taj način lozinka ostaje zaštićena od pogleda ljudi koji imaju pristup bazi podataka [21]

[21] codeproject.com, 2020.

3.3.2. Grafičko korisničko sučelje

Grafičko korisničko sučelje (engl. *user interface*) je vizualni dio aplikacije, sve ono što je vidljivo golim okom korisniku te aplikacije. Od navigacijske trake, teksta, slika, linkova, do gumbova, polja za upis podataka, sve to spada u grafičko korisničko sučelje. Grafičko sučelje „Drite“ aplikacije je napravljeno od samo generičkog predloška koji se generirao od strane ASP.NET-a s bootstrap-om. Početna stranica web aplikacije je rađena personalizirano da ima izgled i funkcionalnost koju imaju i ostala „**internetska tržišta“** koji stavljaju tražilicu ponude na najvećem dijelu početnog ekrana web stranice. Web aplikacija je responzivna što znači da će biti ugodna oku na svim vrstama ekrana od mobilnih uređaja, tableta do velikih monitora. U bootstrapu možemo definirati različito ponašanje grafičkog sučelja na 4 različite veličine ekrana:[22]

- Mali ekrani (od 576px do 768px) – mobilni uređaji
- Srednji ekrani (od 768px do 992px) – tableti
- Veliki ekrani (od 992px do 1200px) – desktop uređaji
- Ekstra veliki (od 1200px na više) – veliki monitori, televizije

3.3.3. Administratorsko sučelje

Administratorsko sučelje se najčešće koristi za upravljanje i kontrolu platforme odnosno korisnika na platformi kroz specijalizirano sučelje koje se razlikuje funkcionalnostima od ostalih sučelja. Podatci za pristup (email i lozinka) administratorskom sučelju su hard kodirani u programskom kôdu što znači da su definirani u programskom kôdu i kada bi htjeli promijeniti pristupne podatke bi morali otići nazad u kôd i promijeniti ih. U web aplikaciji su definirane tri role odnosno vrste korisnika od kojih svaka ima svoje personalizirano sučelje:

- Korisnik (putnik) – svaka osoba koja se registrirala na platformu. Ima mogućnost stavljanja prijevoza ako želi postati vozač, pregleda razgovora, rezervacija, recenzija, profila

[22] getbootstrap.com, 2020

- Vozač – osoba koja je označila da je vozač prilikom registracije ili unutar forme za mijenjanje profila. Vozač može sve što i korisnik, ali može još i pregledavati prijevoze koje je stavio
- Admin – svaka osoba koja ima pristupne administratorske podatke

3.3.4. Sustav vrednovanja vozača i putnika

Sustav vrednovanja vozača i putnika kako se često naziva sustav recenzija se koristi da kroz međusobno ocjenjivanje korisnika na platformi u ovom slučaju putnika i vozača pruža sigurnost unutar platforme. Većina aplikacija koje spadaju u kategoriju „**internetskih tržišta“** ga koristi. Putnik može dati vozaču 5 vrsta ocjena za cjelokupno iskustvo tokom odrađene vožnje, uz ocjenu još može ostaviti i komentar, koji najčešće bude vezan za vožnju vozača. Moguće ocjene:

- Disappointing
- Bad
- Good
- Super
- Great

3.3.5. Obavještavanje korisnika

Ako korisnik prilikom pretraživanja prijevoza ne naiđe na niti jedan prijevoz ima opciju kliknuti gumb koji će ga automatski obavijestiti e-mailom u slučaju da netko stavi prijevoz na toj relaciji i u tom traženom periodu.

3.3.6. Komunikacijski sustav

Komunikacijski sustav služi za razgovor između korisnika unutar aplikacije koji je tematski vezan uz rezervaciju. U sličnim aplikacijama „Chat“ se najčešće koristi za neke stvari koje nisu rečene kroz aplikaciju npr. gdje točno će vozač pokupiti putnika, da li ima dodatnu prtljagu, da li smije kućna životinja u auto, da li je moguće krenuti 20 minuta ranije itd.. U takvim situacijama chat koji je vezan za rezervaciju i za prijevoz jako dobro dođe. Slanje poruke je također moguće prilikom rezervacije prijevoza za slučaj da putnik želi pitati vozača neko od prethodnih spomenutih pitanja prije rezerviranja prijevoza. Na navigacijskoj

traci prijavljenih korisnika postoji cijeli dio Messages-a gdje se mogu pregledati svi razgovori koje ste imali preko Drite web aplikacije. Razgovor između korisnika se prikazuje u realnom vremenu, što znači kada putnik pošalje poruku ona se automatski pokaže u razgovoru. Prikaz poruka u realnom vremenu se odvija preko Ajax poziva.

3.3.7. Pregled rezervacija

Pregled svih osobnih rezervacija je moguć samo za prijavljene korisnike koje su napravile rezervaciju u prošlosti. Preko rezervacija se može doći do podataka tko je vozio, kada se rezervacija odvila, broj telefona, recenzije vozača i moguće je poslati poruku vozaču. Pregled rezervacija je zamišljen da se koristi više u informativne svrhe da korisnik može vidjeti s kim se kada vozio i da može naknadno kontaktirati vozača.

4. Prednosti, nedostaci i planirana proširenja

Prednosti ovakve aplikacije na tržištu su brojne:

- Ekonomska ušteda - prilikom dijeljenja prijevoza, vozač dijeli/štedi troškove goriva i amortizacije vozila dok putnik može platiti po manjim cijenama od postojećih na tržištu kao što su npr. javni prijevoz ili taksi (Uber, Bolt i slične aplikacije)
- Brzina dolaska s jedne lokacije na drugu. Iako postoje već spomenute aplikacije koje vas jako brzo mogu prenijeti s jedne točke na drugu, postoji segment tržišta gdje te aplikacije nisu dostupne odnosno gdje bi „Drite“ bio brža opcija prijevoza
- Druženje i upoznavanje novih ljudi koristeći „Drite“ – upravo je BlaBlaCar – platforma za dijeljenje prijevoza na duže relacije po tome dobila i ime. Prilikom dijeljenja prijevoza u autu se stvara privatnija, ugodnija atmosfera koja potiče ljude na druženje
- Ekološkiji prijevoz – znamo kako auti stvaraju CO2 ugljični monoksid koji pridonosi globalnom zatopljenju. Korištenjem dijeljenog prijevoza bi se smanjio broj taksista, a dugoročnije i ljudi koji imaju auto odnosno koji svakodnevno koriste osobni auto
- Manje gužva u prometu odnosno „lakši promet“ – gužve u prometu stvaraju veliki gubitak vremena za sudionike u prometu. Statistike iz 2016. godine govore da 76.3 % svih američkih vozača voze sami dok voze auto odnosno voze bez suputnika[23]. Dijeljenje prijevoza bi iskoristilo volumenski kapacitet auta i onda bi npr. umjesto 4 auta mogao biti samo jedan auto s četvero ljudi što štedi oko 20m2 (metar kvadratni) na cesti

Prednosti razvijenih funkcionalnosti:

- Komunikacijski sustav – omogućava brz i lak dogovor između putnika i vozača
- Autentikacija korisnika – omogućava sigurno spremanje osobnih podataka poput email-a, lozinke, broj registracijske tablice itd. koji omogućavaju korisnicima lakše rezerviranje, stavljanje prijevoza i sigurnost

[23] brookings.edu, 2020

- Sustav recenzija – omogućava provjeru obostrane sigurnosti na način da putnici mogu stavljati vozačima komentar i ocjenu za određenu vožnju. Na taj način recenzije postaju vidljive i drugim korisnicima aplikacije te im pomažu pri odabiru najboljeg vozača
- Obavještavanje korisnika – omogućava automatsku obavijest putniku ako određeni vozač stavi prijevoz za određenu relaciju i vrijeme

Nedostaci realiziranih funkcija:

- Prilikom pretraživanja prijevoza putniku izbacuje samo one relacije koje su iste kao tražene relacije. Uzmimo za primjer da idemo iz Ilice 52 na Ilicu 100 i utipkamo u aplikaciju početnu točku (Ilica 52) i završnu točku (Ilica 100), sustav će izbaciti samo one prijevoze koji voze od Ilice 52 do Ilice 100. Ovo će jako otežati funkcionalnost aplikacije
- Prilikom stavljanja prijevoza, nije se osiguralo da vozač mora obavezno unijeti podatke o vozilu. Na taj način svatko tko je ulogiran može biti vozač
- Recenzije je moguće stavljati prije odrađene vožnje što omogućava lažno recenziranje vozača

Planirana proširenja aplikacije:

- Ispravak svih nedostatka navedenih u prijašnjem dijelu
- Omogućavanje slanje automatskih poruka preko telefona ili mail-a u trenucima kada se netko registrira, kada netko objavi prijevoz ili kada netko rezervira prijevoz
- Napraviti grafički profil korisnika da korisnici mogu međusobno pregledavati profile. Npr. mogućnost dodavanje slike korisnika, mogućnost verificiranja osobne iskaznice, mogućnost verificiranja vozila
- Mogućnost plaćanja karticom odnosno platni sustav
- Dodatne stranice poput „O nama“ gdje se opisuje tim, „Pomoć“ gdje ljudi mogu naći pomoć oko korištenja aplikacije, „FAQ“ gdje su odgovori na česta pitanja koja korisnici postavljaju
- Redizajn grafičkog sučelja aplikacije

5. Zaključak

Rad u navedenim tehnologijama modernog doba samo-generiranog kôda jako olakšava rad programeru, fokus se stavlja na logici kôda, a ne na suhoparnom ponavljanju pisanja jednog te istog kôda. Također odabrana struktura kôda MVC i Entity Framework jako olakšavaju programeru logičko razdijeljene kôda na cjeline, i s tim mu pomaže u lakšem snalaženju, navigiranju kroz kôd. Ponovno se pokazalo da planiranje, skiciranje, prototipiranje uvelike pomaže pri pokretanju bilo kakvog većeg projekta. Crtanje ekrana, dijagrama baze podataka, odnosa između entiteta u bazi, sve te radnje pridonose boljem postavljanju početka projekta. Da li će aplikacija/projekt uspjeti ostvariti svoje ciljeve navedene u uvodu? Tržište, marketing i angažman osnivača će odlučiti o tom. Aplikacija je još u početnom stadiju MVP-a (engl. *Minimal viabale product*), spremna za početno testiranje. Smjer daljnjeg razvijanja aplikacije je naveden u poglavlju Prednosti, nedostaci i planirana proširenja, navedeni smjer je temeljen na funkcionalnostima drugih sličnih aplikacija poput Blablacar-a, Uber-a…itd. Ipak, daljnji razvoj mora biti orijentiran prema korisnicima aplikacije i graditi platformu uz njihove povratne informacije.

Literatura

[1] JON GALLOWAY , BRAD WILSON, K. SCOTT ALLEN, DAVID MATSON; Professional asps.net mvc 5; 2014.;

[2] JESS CHADWICK ,TODD SNYDER,HRUSIKESH PANDA; Programming ASP.NET MVC 4: Developing Real-World Web Applications with ASP.NET MVC; 2012.;

[3] JON DUCKETT; WILEY; JavaScript and JQuery: Interactive Front-End Web Development; 2014.;

[4] ITZIK BEN-GAN; T-SQL Fundamentals; 2016.;

[5] https://en.wikipedia.org/wiki/Object-relational_mapping 02.01.2020.

[6] https://en.wikipedia.org/wiki/Object-oriented_programming 02.01.2020.

[7] https://docs.microsoft.com/en-us/ef/ 02.01.2020.

[8] https://en.wikipedia.org/wiki/Entity_Framework 02.01.2020.

[9] https://www.c-sharpcorner.com/blogs/entity-framework-code-first-vs-database-first-approach 02.01.2020.

[10] https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework) 02.01.2020.

[11] https://www.quora.com/What-is-web-config-file-What-is-the-use-of-the-config-file-in-web-applications 02.01.2020.

[12] https://en.wikipedia.org/wiki/Ajax_(programming) 02.01.2020.

[13] https://simple.wikipedia.org/wiki/HTML 02.01.2020.

[14] https://en.wikipedia.org/wiki/JQuery 02.01.2020.

[15] https://en.wikipedia.org/wiki/Web_application 02.01.2020.

[16] https://whatismyipaddress.com/localhost 02.01.2020.

[17] https://www.codeproject.com/Articles/654846/Security-In-ASP-NET-MVC 02.01.2020.

[18] https://getbootstrap.com/docs/4.1/layout/overview/ 02.01.2020.

[19] https://en.wikipedia.org/wiki/Online_marketplace 02.01.2020.

[20] https://www.brookings.edu/blog/the-avenue/2017/10/03/americans-commuting-choices-5-major-takeaways-from-2016-census-data/ 02.01.2020.

[21] https://uk.godaddy.com/help/what-is-a-webconfig-file-5445 02.01.2020.

[22] https://hr.wikipedia.org/wiki/Microsoft_SQL_Server 05.01.2020.

[23] https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet 05.01.2020.

[24] https://www.w3schools.com/asp/razor_syntax.asp 05.01.2020.

[25] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller 05.01.2020.

[26] https://en.wikipedia.org/wiki/C_Sharp_(programming_language) 05.01.2020.

[27] https://stackoverflow.com/questions/102558/biggest-advantage-to-using-asp-net-mvc-vs-web-forms 05.01.2020.

[28] https://dotnet.microsoft.com/apps/aspnet/web-forms 05.01.2020.

[29] https://www.tutorialspoint.com/entity_framework/entity_framework_first_example.htm 05.01.2020.

[30] https://www.c-sharpcorner.com/article/javascript-vs-jquery-difference-between-javascript-and-jquery/ 05.01.2020.

Popis slika

Slika 2.1 Grafički prikaz MVC-a 4

Slika 2.2 Grafički prikaz ORM-a 8

Slika 2.3 Grafičko korisničko sučelje u Bootstrap-u 12

Slika 2.4 Prikaz ajax poziva u Pogledu 13

Slika 2.5 Prikaz ajax poziva u kontroleru 14

Slika 3.1 Dijagram baze podataka 20

Slika 3.2 Dijagram međuovisnosti komponenata 22

Slika 3.3 Vozačevo sučelje 22

Slika 3.4 Obavještavanje korisnika 23

Popis kôdova

Kôd 2.1 Primjer IF ELSE naredbe u SQL 6

Kôd 2.2 Primjer Connection string-a korišten u radu 7

Kôd 2.3 Primjer korištenja Dana Annotations-a 10

Kôd 2.4 Primjer JavaScript koda 14

Kôd 2.5 Primjer JQuery koda 14

ALGEBRA
VISOKO
UČILIŠTE

SUSTAV ZA ORGANIZIRANJE PRIJEVOZA OSOBA

Pristupnik: Rino Duboković, 0023110075

Mentor: doc. dr. sc. Vedran Juričić