WEB APLIKACIJA ZA NARUDŽBU HRANE

Vuković, Luka

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

WEB APLIKACIJA ZA NARUDŽBU HRANE

Luka Vuković

Zagreb, siječanj 2020.

„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, 18.02.2020.

Luka Vukovic

Predgovor

Zahvaljujem mentoru Aleksandru Radovanu koji je pratio cijeli proces nastajanja završnog rada i svojim savjetima i entuzijazmom usmjeravao me kako da prevladam probleme koji bi se pojavili prilikom izrade završnog rada. Zahvaljujem svojoj obitelji i prijateljima koji su cijelo vrijeme bili uz mene i davali mi podršku.

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

Sažetak

Motivacija za izradu ovog rada je konstantna želja za razvojem znanja i vještina te napredovanje u struci, a kao glavni cilj izdvaja se izrada web aplikacije s potencijalnom primjenom u produkcijskoj okolini. Izgradit će se MVC aplikacija koja omogućuje narudžbu hrane, pregled recepata, nutritivnih vrijednosti i dostavu namirnica. Aplikacija će imati mogućnost rada za autorizirane i ne autorizirane korisnike. Ne autorizirani korisnici imati će samo jednu mogućnost, a to je pregled restorana i gotovih jela dostupnih za narudžbu. Autorizirani korisnici moći će pregledavati restorane registrirane u aplikaciji, naručivati gotova jela, pregledavati recepate za izradu istih te će imati mogućnost narudžbe sirovih namirnica za izradu konkretnog jela. Aplikacija će imati mogućnost registracije za privatne i poslovne korisnike. Poslovni korisnički računi tj. restorani, nakon provjere podataka, imati će mogućnost unašanja novih proizvoda u svoj „meni“, izmjenu postojećih podataka te pregled povijesti poslovanja izvršene putem aplikacije. Komunikacija će se odvijati uz pomoć vanjskog REST servisa koji će komunicirati s bazom podataka. Nakon što zaprimi narudžbu, restoran može prihvatiti ili odbiti zaprimljenu narudžbu te ukoliko je potrebno, upisati dodatne komentare vezane za narudžbu. Nakon što restorani obrade narudžbu, potvrda dolazi do krajnjeg korisnika gdje on može vidjeti stanje svoje narudžbe, predviđeno vrijeme dostave te dodatne komentare ukoliko ih ima. Plaćanje narudžbi će se vršiti online putem kartice ili gotovinom.

Ključne riječi: web aplikacija, MVC, baza podataka, REST servis.

Summary

Motivation for making this work is a constant desire for the development of knowledge and skills same as advancement in the professions. The main goal which stands out is the development of web application with a potential use in production environments. An MVC application will provide informations about nutritional values, food ordering, recipe reviewing and groceries delivery. The application will be able to work for authorized and non-authorized users. Not authorized users will have only one option which is an overview of the restaurants and their meals available to order. Authorized users will be able to browse the restaurants registered in the app, order meals, review the recipe for making them and also they will be able to order raw foods to make specific dishes. Web application will have the option of registering for private and business users. Business user accounts would be able to insert and modify products. Bussiness overview data would be available for bussiness accounts. Communication will be based on an external REST service that will communicate with the database. Once the order is sent, the restaurant can accept or reject received order. After the restaurants have processed the order, end-users will receive confirmation of their order, they would be able to track their order and to see the estimated time of delivery. Payment for the order will be possible online by card or cash.

Ključne riječi: Web application, MVC, Database, REST service.

Sadržaj

1. Uvod

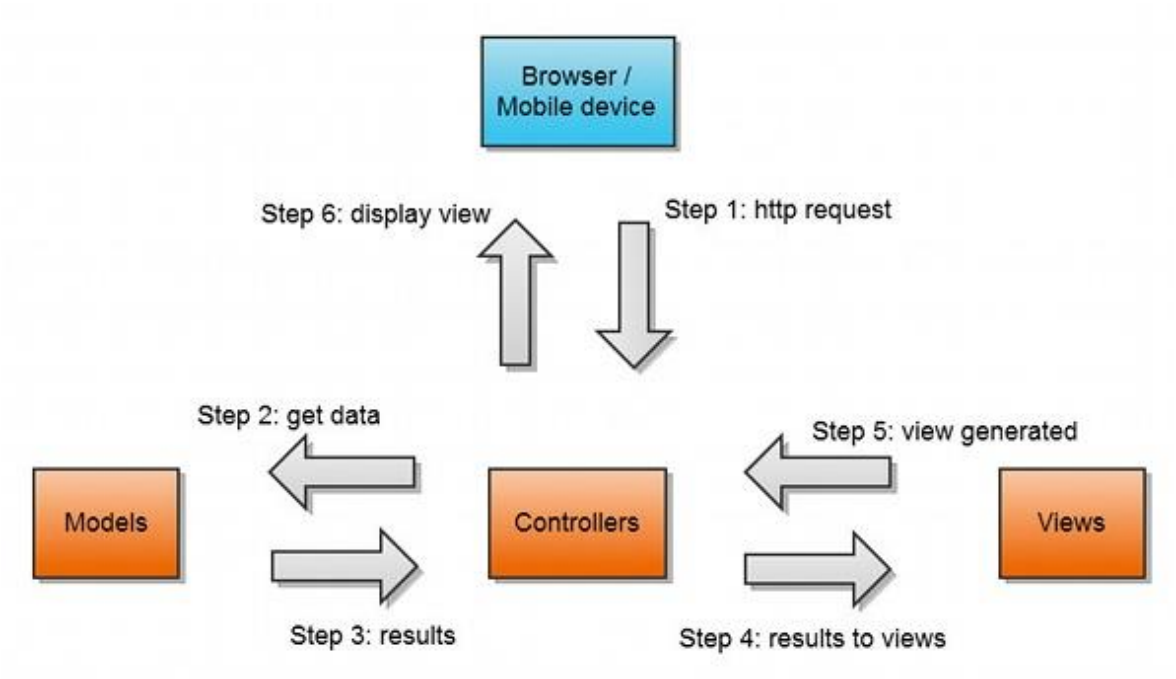
Glavna ideja ovog završnog rada je upoznavanje najnovijih tehnologija koje se koriste u razvoju web aplikacija te prikaz interoperabilnosti različitih sustava u stvarnom okruženju.

Krajnji proizvod rada bit će web aplikacija koja omogućuje narudžbu gotovih jela i namirnica iz registriranih restorana. Aplikacija je izrađena u C# programskom jeziku te je temeljena na MVC (*Model-View-Controller*) arhitekturi. Složenost aplikacije nalaže korištenje više različitih tehnologija, a to je moguće zbog interoperabilnosti informacijskih sustava. Cilj je pokazati međudjelovanje izabranih tehnologija.

Nakon predstavljanja arhitekture i modela baze podataka, rad će bit usmjeren na analizu postojećih rješenja te usporedbu s novom aplikacijom. U nastavku će biti opisane funkcionalnosti aplikacije kao što su registracija i prijava poslovnih i privatnih korisnika, kreiranje ponude poslovnih korisnika te proces narudžbe i obrade zahtjeva. Potom će biti opisana komunikacija s bazom koja će se u potpunosti odvijati koristeći REST web servis. Naposlijetku će se opisati dodatne mogućnosti aplikacije koje se odnose na računanje energetske vrijednosti odabranih jela, pregled recepata za odabrana jela te narudžbu namirnica potrebnih za izradu određenog jela. Na kraju rada biti će prikazane prednosti i nedostaci novonastale aplikacije u odnosu s postojećima.

2. Uvod u MVC arhitekturu

[1]*Model-View-Controller* označava uzorak arhitekture (engl. *architectural pattern*) koji služi za podjelu aplikacije u 3 logična dijela, model, pregled (engl. *view*) i upravljač (engl. *controller*). Svaki od tih sastavnih dijelova (engl. *component*) je izgrađen da obrađuje (engl. *handle*) određeni dio aplikacije. MVC je jedna od najkorištenijih okosnica (engl. *framework*) za stvaranje i razvoj skalabilnih i proširivih web aplikacija. Ovakav pristup razvoju aplikacije omogućuje bolju preglednost koda, lakše održavanje i izmjenu postojećeg koda te također pojednostavljuje rad više razvojnih inženjera na istom projektu. [1] Iz slike 1. može se vidjeti životni ciklus (engl. *Life cycle*) web zahtjeva na aplikaciju napravljenu po MVC arhitekturi. Dolazni zahtjev koji biva obrađen u aplikacijskom sloju označava početak ciklusa. Aplikacijski sloj vrši komunikaciju s poslužiteljem te razmjenjuje informacije koje je ulazni zahtjev zahtijevao. Nakon što podatkovni sloj vrati podatke aplikaciji koja obradi primljene podatke, prezentacijski sloj generira prikaz obrađenih podataka koje kao takve vraća u aplikacijski sloj koji ga prosljeđuje web poslužitelju s kojega je primio zahtjev.

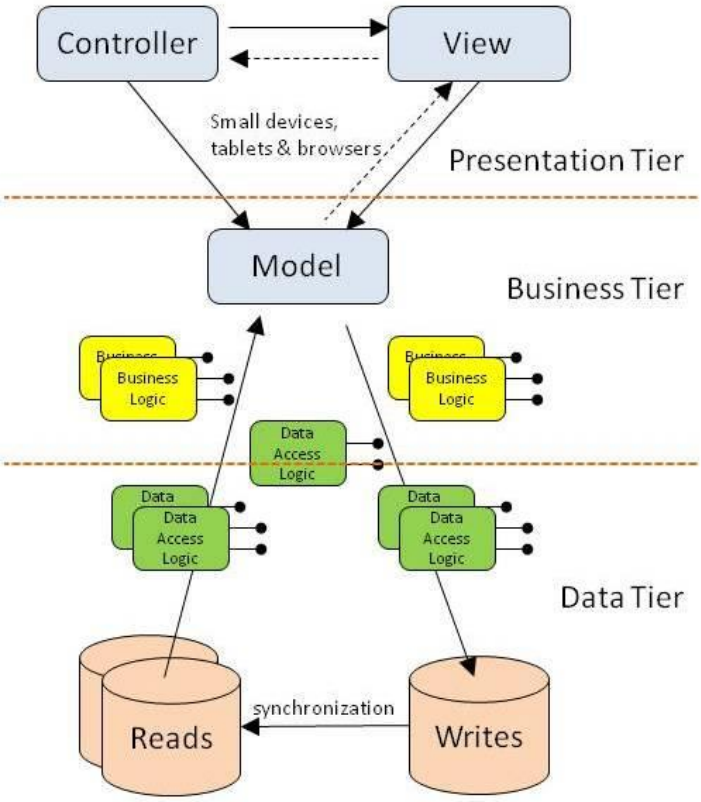


Slika 1. Prikaz MVC arhitekture[2]

[1] *https://dotnet.microsoft.com/apps/aspnet/mvc*
[2] *https://yii4beginners.files.wordpress.com/2014/05/mvc1.jpg*

2.1. Primjena MVC arhitekture u aplikaciji

Ovaj dio završnog rada prikazivati će konkretnu primjenu MVC pristupa u novo nastaloj aplikaciji te odvajanje sve tri komponente arhitekture (prezentacijski sloj, aplikacijski sloj i podatkovni sloj). [1] Na slici 2. je vidljivo kako je logička organizacijska struktura MVC implementirana u aplikaciji. Vidljivo je da korisnik aplikacije jedino komunicira s prezentacijskim slojem, koji je zadužen za prikazivanje potrebnih podataka primljenih od aplikacijskog sloja. Aplikacijski sloj obrađuje dobivene podatke s podatkovnog sloja koje potom šalje prezentacijskom sloju kako bi bili ispravno prikazani na sučelju. Podatkovni sloj aplikacije odnosi se na dio aplikacije koji je zadužen za pružanje i dostavu podataka aplikacijskom sloju kako bi aplikacija mogla uspješno raditi. Prezentacijski sloj nema komunikaciju s podatkovnim slojem niti obratno, ta dva sloja komuniciraju jedino preko aplikacijskog sloja aplikacije. [8]



Slika 2. Prikaz MVC arhitekture u stvarnom okruženju[3]

[3] *http://criticaltechnology.blogspot.com/2011/09/mvc-in-three-tier-architecture.html*

2.1.1. Prezentacijski sloj

Prezentacijski sloj aplikacije ima svrhu vizualnog sučelja aplikacije. Aplikacijski sloj jasno komunicira s prezentacijskim na način da međusobno razmjenjuju podatke koji moraju biti prikazani na vizualnom dijelu aplikacije tj. korisnik putem vizualnog sučelja dolazi do podataka koji su mu potrebni. Aplikacija je raspodijeljena na više kontrolera koji su zaduženi za određene *view*-e. *View*-i predstavljaju zasebne web stranice ovisno za koji dio aplikacije su namijenjeni. Kontroler prima podatke o korisnikovim akcijama na vizualnom sučelju i ovisno o tome izvršavaju se akcije koje omogućuju rad aplikacije. [8] Kontroleri će biti logički razdijeljeni na cjeline koje će obrađivati akcije vezane za određene dijelove modela. Akcije koje će obrađivati zahtjeve vezane uz korisnike, restorane i proizvode biti će implementirane u kontrolere. Prezentacijski sloj biti će izveden od HTML elemenata, CSS oblikovanja te JS skripti.

2.1.2. Aplikacijski sloj

Aplikacijski sloj aplikacije odnosi se na poslovnu logiku aplikacije. Dohvaća podatke s podatkovnog sloja koje potom obrađuje i prosljeđuje na vizualno sučelje. „*Model*“ aplikacije dio je aplikacijskog sloja i predstavlja podatke s kojima aplikacija radi. Sastoji se od C# klasa koje su referencirane i strukturirane po objektima baze podataka. Unutar aplikacije postoje par ključnih modela s kojima aplikacija radi. Korisnici, restorani i proizvodi su modeli s kojima aplikacija radi u prezentacijskom sloju, dok se u poslovnom sloju još radi s računima i narudžbama. Poslovna logika biti će napisana u programskom jeziku C#.

2.1.3. Podatkovni sloj

Podatkovni sloj aplikacije je se odnosi na dio aplikacije u kojem se nalazi izvor podataka. Isključivo je zadužen za dohvat podataka iz baze te unos i promjenu novih ili postojećih podataka. On ne smije sadržavati nikakvu poslovnu logiku, a u ovom primjeru biti će izveden pomoću Entity frameworka koji će komunicirati s bazom podataka. Aplikacija će koristiti Microsoft SQL za implementaciju baze podataka.

3. Baza podataka unutar aplikacije

[4]Baza podataka, kao temeljni i glavni izvor podataka u aplikaciji, odnosi se na poslužitelja iz kojeg aplikacija čita podatke potrebne za rad. Definirati će se što je baza podataka, koje su njene funkcije, što sve ona nudi te kako se koristi u aplikaciji. Podatak je jednostavna neobrađena izolirana misaona činjenica koja ima neko značenje. Baza podataka može se definirati kao organizirana kolekcija ili skup podataka i informacija. Najuobičajeniji način za pohranu i organizaciju baza podataka jest tablična organizacija podataka u kojoj su podaci raspodijeljeni u stupce i retke te su međusobno povezani. Baze podataka susreću se gotovo u svim poslovnim djelatnostima koje su danas poznate iz razloga što su jedan od naj preglednijih načina za organizaciju podataka. Rad s podacima je uvelike olakšan implementacijom jezika za upravljanje podacima u same baze podataka. Spremanje, čitanje, pisanje samo su neke od mogućnosti koje se na vrlo jednostavan i brz način mogu izvršavati uz pomoć SQL jezika. Različiti oblici SQL jezika mogući su ovisno o bazi podataka u kojoj se radi. Ovisno o aplikaciji, baza podatka može biti pohranjena lokalno na računalo na kojem je napravljena ili na serveru, gdje je dostupna širem broju korisnika. [2]

Aplikacija „Food4You“ koristi bazu podataka pohranjenu na internetu. Zbog konstantnog razvoja internetskih tehnologija, web aplikacija te generalnog sadržaja na internetu razvili su se mnogi poslužitelji koji nude uslugu posluživanje na internetu (engl. *hosting*). Baza će biti pohranjena na jednom od takvih servisa, ali za potrebe testiranja, inačica baze biti će pohranjena i lokalno. Komunikacija s bazom se odvija pomoću REST web API-ja o čemu će više biti rečeno u nastavku rada. Nad podacima se mogu izvršavati razne akcije, upiti, sortiranja i ostalo. Najčešće akcije korištene na podacima ujedno se nazivaju i CRUD operacije koje se odnose na stvaranje (engl. *create*), čitanje (engl. *read*), izmjenu (engl. *update*) i brisanje (engl. *delete*) podataka.

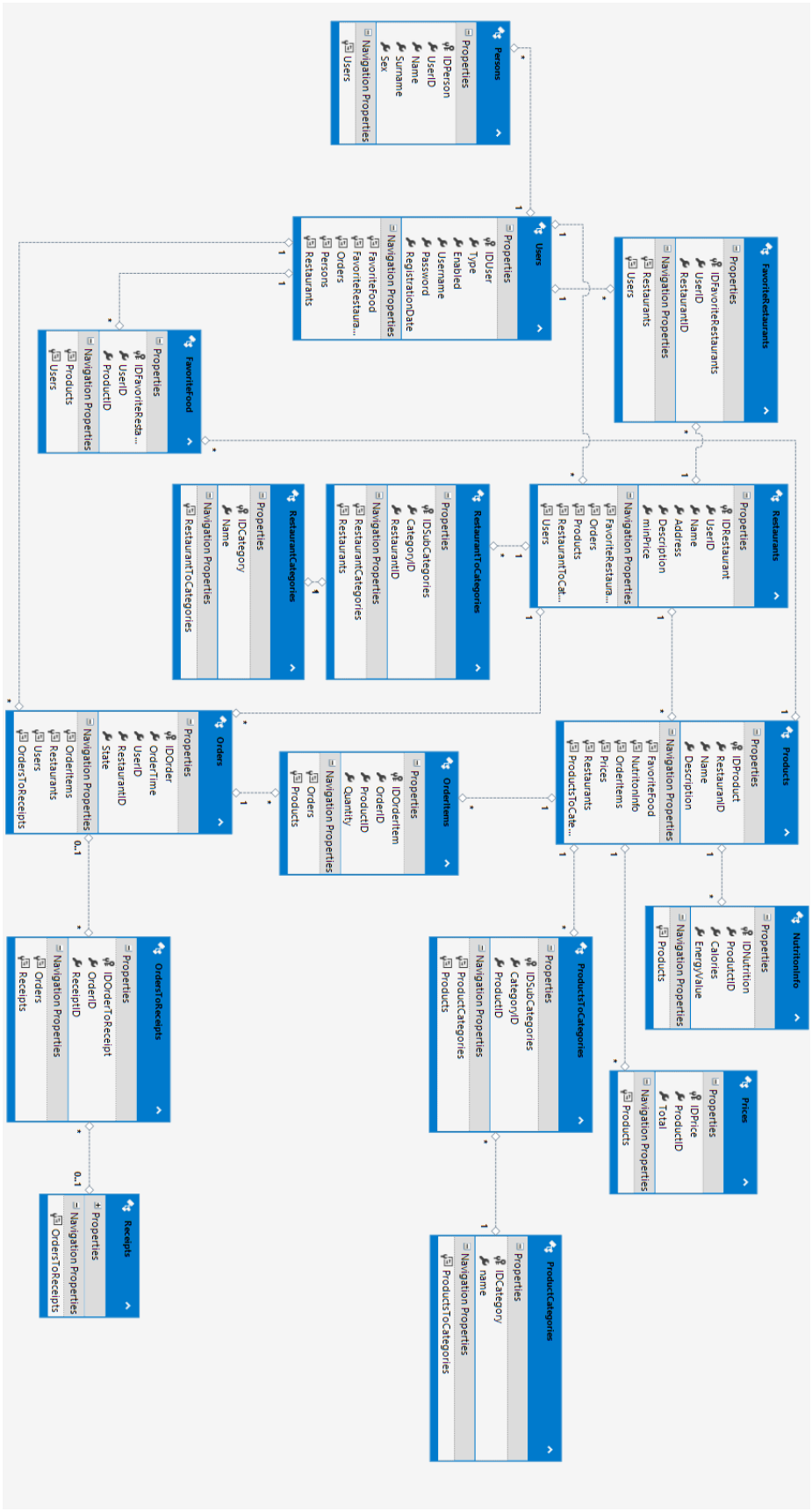
[4] *https://www.guru99.com/introduction-to-database-sql.html*

3.1. ER model baze podataka

[5]ER Model (engl. *entity-relationship model*) baze podataka je konceptualni način prikazivanja povezanosti među podacima u bazi podataka. ERM je tehnika modeliranja dostupna u bazi podataka koja generira apstraktni dijagram tj. vizualnu interpretaciju međuodnosa podataka. Osnovni elementi ovog dijagrama su entiteti i veze. Entiteti označavaju objekte od kojih se naknadno izrađuju modeli u aplikaciji te se odnose na konkretne, prepoznatljive pojmove o kojima se čuvaju podaci u bazi. Primjer: restorani, osobe, korisnici, proizvodi. Veze u bazi služe kako bi se različiti entiteti povezali s točno onim podacima koji pripadaju njima ili se odnose na njih. Najjednostavniji način povezivanja više entiteta je preko ključeva koji mogu biti primarni ili strani, ovisno o tipu entiteta i tome vezujemo li ga ili vezujemo nešto na njega. [3]

Slika 3. prikazuje ER dijagram baze podataka. Prilikom registracije, korisnik upisuje svoje osobne podatke koji služe za njegovu identifikaciju. Uz osobne podatke, korisnik je dužan unijeti email adresu s kojom mora potvrditi svoj identitet, korisničko ime te lozinku koje će koristiti prilikom prijave u aplikaciju. Postoje više vrsti korisnika aplikacije pa se tako poslovni korisnici mogu registrirati kao restorani na način da upišu svoje osnovne podatke potrebne za identifikaciju te isto tako željeno korisničko ime i lozinku za prijavu u aplikaciju. Entitet restorana vezan je na proizvode koje ima u ponudi, izvršene narudžbe, te kategorije restorana kojima pripada. Proizvodi su entiteti koji se sadrže informacije o imenu, opisu, cijeni i kategoriji proizvoda. Narudžba je entitet koji opisuje i povezuje korisnika s restoranom iz kojega je kupio hranu na način da se za svaki zapis narudžbe zapisuje korisnik koji je naručio hranu, restoran iz kojega je korisnik naručio hranu te se dodaju objekti entiteta stavke (engl. *order items*) koji vežu proizvode na narudžbu. Ovisno o narudžbi, kreira se zapis entiteta Račun koji u sebi sadrži cijenu i broj računa koji se također veže na entitet narudžbe.

[5] *https://beginnersbook.com/2015/04/e-r-model-in-dbms/*



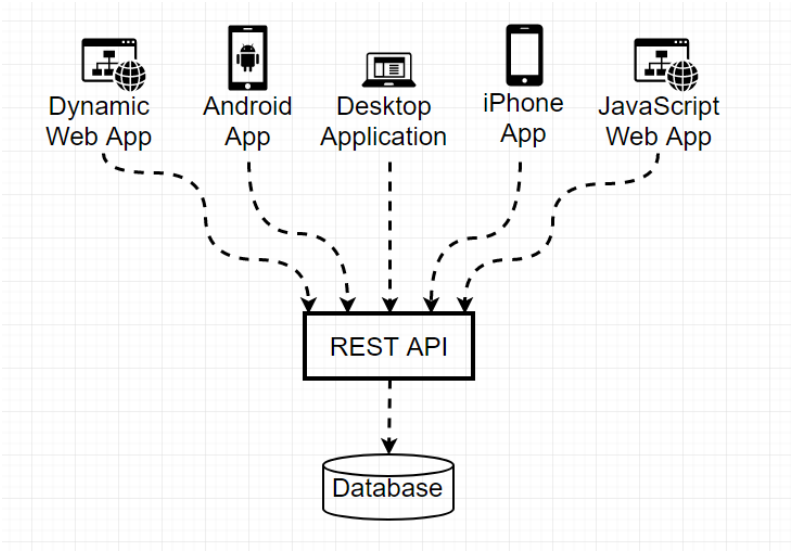
Slika 3. ER dijagram baze podataka

3.2. Model baze podataka na web serveru

Uz lokalnu pohranu baze podataka, baza će biti dostupna i na web serveru. U današnje vrijeme dostupno je mnoštvo servisa za pohranu baze u oblaku. Prednosti toga su široka dostupnost, mogućnost dodatne tehničke podrške, jednostavnost korištenja i pristupačna cijena. Takav oblik organizacije omogućava komunikaciju s bazom neovisno o platformi, jedini je uvjet da aplikacija ima pristup internetu. U slučajevima gdje postoje mobilne aplikacije s istim funkcionalnostima kao i web aplikacije, zbog korištenja baze podataka pohranjene u oblaku, omogućen im je rad s istim podacima uz izvršavanje istih akcija na bazi. Komunikacija se odvija preko REST servisa koji bi izvršavao interakciju s bazom.

4. Primjena REST oblika komunikacije

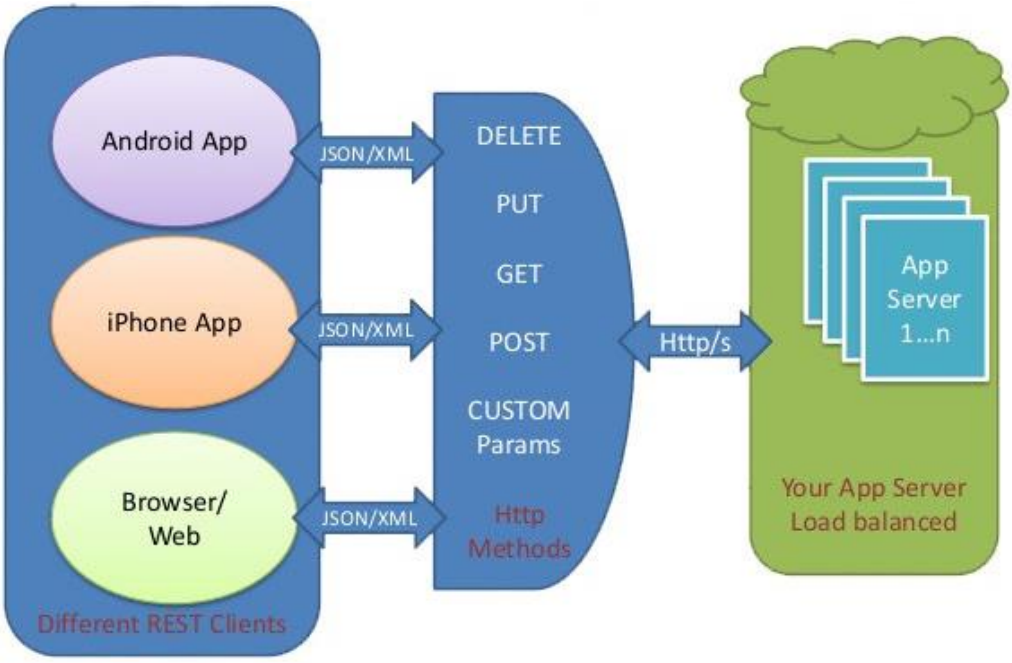
[6]Reprezentacijski prijenos stanja (engl. *represetational state transfer*) označava stil arhitekture za pružanje standarada između računalnih sustava na webu, olakšavajući različitim sustavima međusobnu komunikaciju. Može se reći da REST također označava način komunikacije između poslužitelja i klijenta prilikom korištenja mrežnih resursa uz pomoć HTTP protokola. U arhitektonskom stilu REST-a, implementacija klijenta i implementacija poslužitelja su dvije ne povezane cjeline i mogu se izvršiti samostalno, a da druga strana za to ne zna. Sve dok svaka strana zna koji format poruka mora poslati drugoj strani, može ih se držati odvojenima. Odvajajući problematiku korisničkog sučelja od problematike pohrane podataka, unaprjeđuje se fleksibilnost sučelja na svim platformama i poboljšava se skalabilnost pojednostavljivanjem komponenata poslužitelja. Uz to, razdvajanje omogućuje svakoj komponenti sposobnost da se samostalno razvija. Korištenjem REST sučelja, različiti klijenti dohvaćaju iste REST krajnje točke, izvode iste radnje te primaju iste odgovore. Kao glavne karakteristike REST servisa izdvajaju se ne postojanje stanja (engl. *stateless*), mogućnost keširanja (engl. *cacheable*), jedinstveno sučelje (engl. *uniform interface*), izričito korištenje HTTP metoda te korištenje XML i/ili JSON formata poruke za komuniciranje. [4]



Slika 4. Prikaz komunikacije različitih tehnologija s REST servisom[7]

[6] *https://www.codecademy.com/articles/what-is-rest*
[7] *https://happycoding.io/tutorials/java-server/images/rest-api-1.png*

Na slici 4. vidljivo je kako se zapravo aplikacije nevezano od platforme spajaju na isti REST servis na način na koji znaju komunicirati s njime. REST servis zaprimi zahtjev od klijenta, izvršava upit na bazu podataka, zaprima podatke od baze te ih vraća klijentu koji je imao upit. Upravo ovakvi primjeri govore o interoperabilnosti informacijskih sustava.



Slika 5. REST API arhitektura[8]

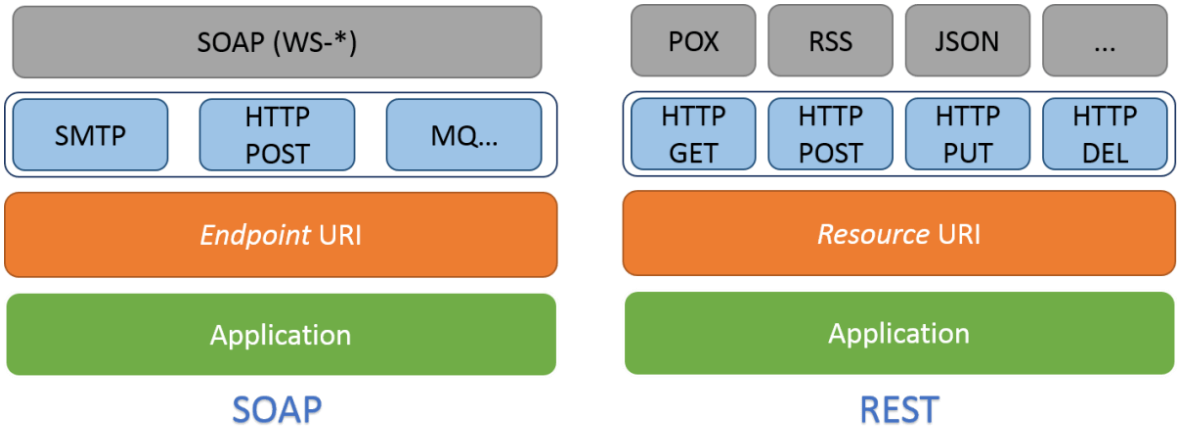
Na slici 5. vidljiva je arhitektura REST API-ja. API se sastoji od HTTP metoda koje su namijenjene za komunikaciju s bazom, a pozivaju se iz klijenata s kojima REST servis komunicira. Najčešći upiti na bazu su dohvaćanje, umetanje, brisanje te izmjena podataka. Nakon što primi odgovor od baze, API šalje podatke klijentu u XML ili JSON formatu.

4.1. SOAP vs. REST

REST web servisi se danas smatraju kao jedan od najrasprostranjenijih način komunikacije putem interneta. Za razliku od REST API-ja, koji opisuju web servise bazirane na REST arhitekturi, postoje i jednostavni protokoli za komunikaciju. Jedan od njih je SOAP protokol koji opisuje jednostavnu komunikaciju tekstualnim sadržajem preko HTTP komunikacijskog kanala. Neovisan o platformi, baziran na XML formatu uz jedini uvjet da

[8] *https://shareurcodes.com/photos//rest-api.jpg*

aplikacija za komunikaciju može koristiti HTTP protokol. SOAP je definiran nizom pravila koja se moraju poštivati kako bi komunikacija bila moguća. Za razliku od REST-a, SOAP protokol ima strogo definirane oblike poruka u komunikaciji, baziran je na sučeljima kojima izlaže svoje funkcionalnosti, koristi WSDL (engl. *Web Service Description Language*) file za pružanje informacija o funkcijama te općenito servisu i onome što pruža. SOAP zahtijeva veću internetsku propusnost od REST-a iz razloga što se u poruci koja se šalje nalaze još brojne druge informacije i pravila o samoj poruci. U današnje vrijeme REST servisi su sve više rasprostranjeni jer sve velike društvene i socijalne mreže raspolažu s velikim brojem svojih javnih servisa. Javni REST servisi služe kako bi ostali razvojni inženjeri (engl. *software developers*) lakše implementirali svoje aplikacije na ciljanu platformu.[5]



Slika 6. Razlika između komunikacije u SOAP i REST web servisu[9]

Na slici 6. vidljivo je kako u različitim izvedbama web servisa oni koriste različite protokole i metode zahtjeva za komunikaciju. U aplikaciji izrađenoj za potrebe ovog rada, komunikacija je omogućena putem REST web servisa zbog jednostavnosti same izvedbe, zbog bržeg rada same aplikacije te zbog lakše implementacije komunikacije na druge platforme ukoliko se javi potreba za time.

4.2. Formati za razmjenu podataka u REST API-ju

[10]REST servisi nisu ograničeni samo na jedan format poruke za komunikaciju. Najučestaliji su JSON i XML format. Oba formata su čitljiva kako računalo tako i čovjeku. Ovisno o situaciji i aplikaciji, razvojni inženjer odabire koji način komunikacije želi implementirati u

[9] *https://i2.wp.com/www.thistechnologylife.com/wp-content/uploads/2019/04/SOAP-vs-REST-1.png*
[10] *https://restfulapi.net/*

web servis. Svaki od oba formata ima prednosti i mane nad drugim o čemu će se govoriti u nastavku rada.

4.2.1. XML

[11]XML jezik za označavanje podatka tj. *Extensible Markup Language*. Osnovna ideja bila je stvoriti jezik koji će biti lako čitljiv i ljudima i računalnom programu. Princip realizacije temelji se na tome da odgovarajući sadržaj treba uokviriti odgovarajućim oznakama koje ga pobliže opisuju i imaju poznato ili lako shvatljivo značenje. Danas je XML vrlo raširen i koristi se u nizu slučajeva, a neki od njih su odvajanje podataka od prezentacije, razmjena podataka, pohrana podataka, povećanje dostupnosti podataka i izrada novih specijaliziranih jezika za označavanje. Prednosti nad JSON-om su to što podržava meta podatke u samoj poruci, upisane u obliku atributa. Kako je XML format duže u upotrebi, većina preglednika ga pregledava i procesira na vrlo pregledan, čitljiv i organiziran način. Struktura stabla korištena u XML-u odlično se prilagođava ovom oblikovanju i omogućuje korisnicima lakši pregled elemenata. XML također ima sposobnost komuniciranja različitog sadržaja unutar istog podatkovnog čvora. [6] U kodu 1. vidljiv je podatak o bankovnom računu upisan u XML formatu.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
   <BankAccount>
      <AccountHolder>
         <FirstName>Darshan</FirstName>
         <LastName>Singh</LastName>
      </AccountHolder>
      <Balance>25382.20</Balance>
      <Number>1234</Number>
      <OpenDate>11/04/1974</OpenDate>
   </BankAccount>
</root>
```

Kod 1 . Primjer XML poruke

[11] *https://www.w3schools.com/xml/*

4.2.2. JSON

[12]JSON označava JavaScript objekt (engl. *JavaScript Object Notation*), a ujedno je i jedan od lakših tekstualnih standarada dizajniran za razmjenu podataka. Izrazito lako je čitljiv te je jednostavan za rukovanje. Programski je neovisan što znači da se jednostavno može koristiti s bilo kojim programskim jezikom. JSON format bazira se na uređenim parovima imena i vrijednosti ili listama podataka univerzalnih struktura podataka korištenih u gotovo svim programskim jezicima. Kao prednosti, a ponekad se može reći i mana, može se izdvojiti činjenica da JSON ne sadrži meta podatke te time znatno smanjuje veličinu poruke te omogućuje bržu komunikaciju. Za definiciju JSON zapisa potrebno je znatno manje informacija nego kod XML-a jer koristi samo informacije o gotovim objektima u zapisu. Možda najznačajnija prednost JSON-a je to što je on u podskupu JavaScripta pa se generalno obrađivanje podataka vrlo prirodno uklapa u JS kod. [6]

```
{
    "BankAccount":{
        "Number": "1234",
        "OpenDate‟: "11/04/1974",
        "Balance": "25382.20",
        "AccountHolder": {
            "LastName": "Singh",
            "FirstName": "Darshan"
        }
    }
}
```

Kod 2 . Primjer JSON poruke

Na kodu 2. vidljivo je kako je zapisan isti zapis s koda 1., ali u JSON formatu. Usporedbom JSON i XML formata zapisa vidljivo je kako su oba vrlo lako čitljiva čovjeku i stroju, ali isto tako da JSON ima jednostavniji način zapisa u odnosu na XML. Izbor formata može se olakšati saznanjem da je XML više namijenjen za označavanje podataka te je idealan izbor u situaciji kada su meta podaci bitni dio podataka. JSON-ova svrha je strukturirana razmjena

[12] *https://www.json.org/json-en.html*

podataka što se odnosi na jednostavnu razmjenu potrebnih podataka kada meta podaci i označavanje podataka u dokumentu nisu uvjet.

4.3. Praktična primjena u aplikaciji

Za rad aplikacije potrebno je omogućiti komunikaciju s bazom podataka. Trenutna aplikacija napravljena je u obliku web aplikacije. Upravo zbog različitosti platforma na kojima će se potencijalno nalaziti aplikacija i s kojih će se pristupati podacima u aplikaciji izabrana je komunikacija pomoću vanjskog web servisa. Web servis koji je poslužen na internetu (engl. *hosting*) zadužen je za svaki oblik komunikacije s bazom te definira način komunikacije s istom. Funkcije koje su definirane u servisu logički raspodijeljene u kontrolere koji su zaduženi za njih. Aplikacija se spaja na akcije servisa putem pristupnih URL-ova te im prosljeđuje podatke potrebne za uspješnu komunikaciju. Nakon dohvata podataka iz baze, servis prosljeđuje aplikaciji podatke koji su bili traženi. Sva komunikacija s bazom odvija se preko REST poziva. Primjer: `UserController` zadužen je za sve akcije vezane za registriranog korisnika aplikacije, tj. dohvat podataka iz baze, izmjenu podataka u bazi, brisanje podataka iz baze.

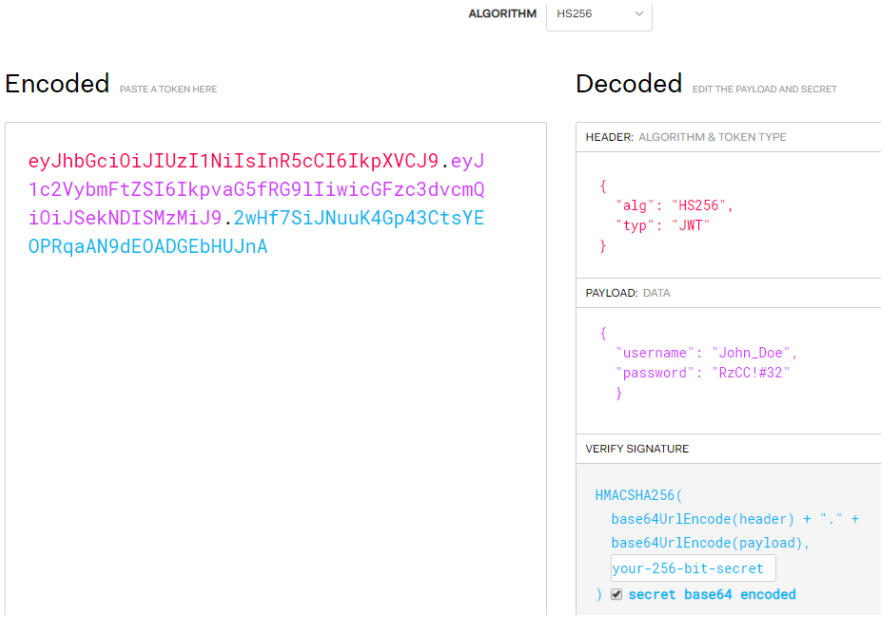
```
namespace API.Controllers
{
    public class UsersController : ApiController
    {
      private FoodOrderingAppEntities db = new FoodOrderingAppEntities();

      [ResponseType(typeof(Users))]
      public async Task<IHttpActionResult> GetUserWithID(int id)
        {
            Users users = await db.Users.FindAsync(id);

            if (users == null)
            {
                return NotFound();
            }

            return Ok(users.Username);
        }

      // GET: api/Users/5
      [ResponseType(typeof(Users))]
      public async Task<IHttpActionResult> GetUserWithLogin(string
      username, string password)
        {
            IQueryable<Users> users = db.Users;

            if (users == null)
            {
                return NotFound();
            }

            UserModel user = new UserModel();
            user.IDUser = (users.FirstOrDefault(e => e.Username
            ==username && e.Password == password)).IDUser;
            user.Username = (users.FirstOrDefault(e => e.Username ==
            username && e.Password == password)).Username;
            user.Password = (users.FirstOrDefault(e => e.Username ==
            username && e.Password == password)).Password;
            user.Type = (users.FirstOrDefault(e => e.Username == username
            && e.Password == password)).Type;
            user.RegistrationDate = (users.FirstOrDefault(e => e.Username
            == username && e.Password == password)).RegistrationDate;

            return Ok(user);
        }
}
```

Kod 3 . Primjer REST poziva u aplikaciji

Kod 3. prikazuje kod aplikacije koji se odnosi na *`UserController`*. Konkretni dio koda zadužen je za dohvat korisnika iz baze, kako po ID-ju tako i po *login* podacima. Ovisno o akcijama korisnika na vizualnom sučelju aplikacije pozivaju se metode povezane s akcijama na formi. Određene komunikacijske metode pozivati će se sinkrono, dok će pojedine metode biti pozivane i asinkrono. Kroz rad će biti govora o prednostima asinkronih poziva te će ponašanje biti objašnjeno na primjeru iz aplikacije.

4.4. JWT Token

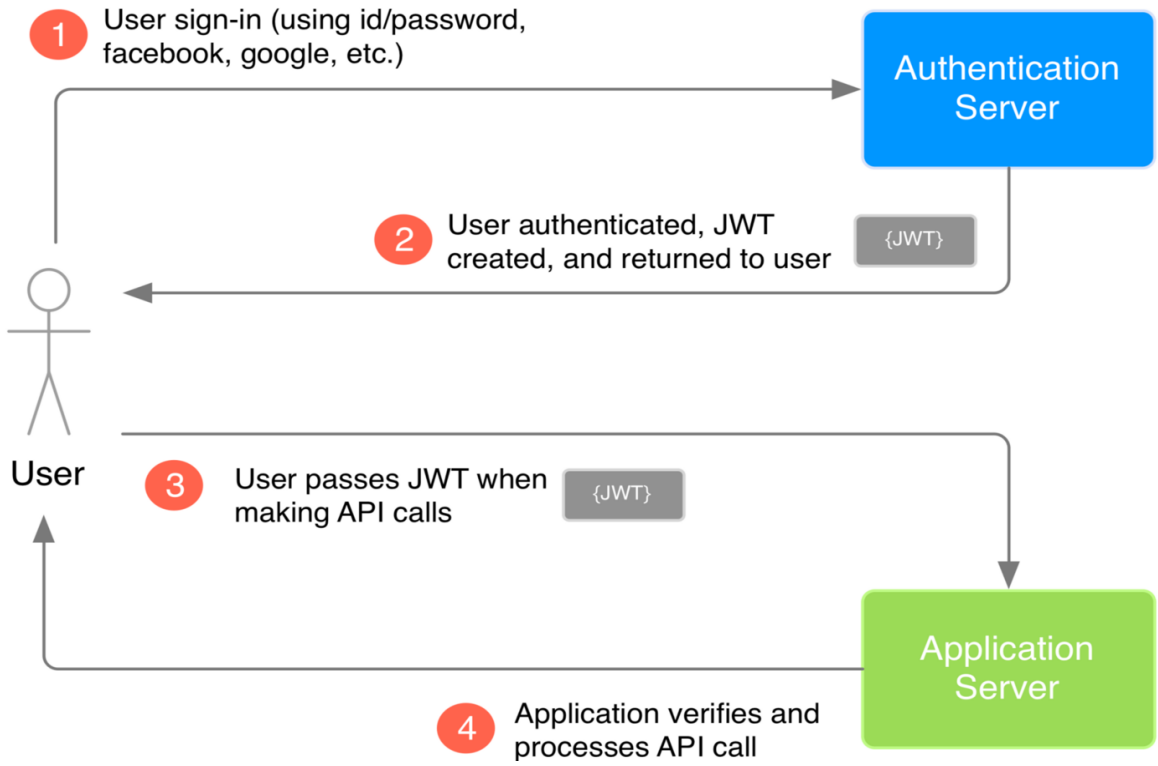
Kolika je zapravo sigurnost aplikacije i samih podataka pitanje je koje se postavlja gotovo uz svaku korisničku aplikaciju dostupnu na tržištu. Sigurnosno pitanje biti će riješeno korištenjem [13]JWT tokena koji označava JSON web token. Opisuje i definira kompaktan i samostalan način sigurnog prijenosa informacija u obliku JSON zapisa između dvije strane. Podaci razmijenjeni korištenjem JWT token-a su digitalno potpisani i vjerodostojni. JWT tokeni mogu biti potpisani pomoću privatnih (npr. HMAC algoritam) ili parom privatnih i javnih ključeva pomoću RSA ili ECDSA algoritama. Očekivano korištenje je u situacija gdje aplikacija radi s osjetljivim podacima. Autorizacija korisnika na internetu je možda i najpoznatiji slučaj koji ima potrebu za korištenjem JWT tokena. Jednom kada se korisnik prijavi u aplikaciju, svaki sljedeći zahtjev uključuje JWT što mu ujedno omogućava pristup rutama, uslugama i resursima koji su dozvoljeni tim znakom što je vidljivo na slici 8. Drugi jednako tako popularni slučaj je generalna razmjena informacija u web aplikaciji. Ukoliko aplikacija radi s osjetljivim podacima, najbolja praksa bila bi zaštita svih podataka. Kako je JWT tokene moguće potpisati parovima privatnih i javnih ključeva, omogućuju visoku razinu sigurnosti za podatke. Uz to, kako se potpis izračunava pomoću zaglavlja i poruke moguće je provjeriti je li sadržaj bio neovlašteno ugrožen. [7]



Slika 7. Prikaz šifrirane JWT poruke

[13] *https://jwt.io/*

Struktura JSON web tokena sastoji se od zaglavlja, poruke i digitalnog potpisa što je moguće vidjeti na slici 7. Uz to, prikazano je kako izgleda šifrirana poruka nastala uz pomoć JWT tokena. Poruka je kriptirana HS256 algoritmom.



Slika 8. Proces komunikacije uz pomoć JWT tokena[14]

Slika 8. prikazuje proces komunikacije korištenjem JWT tokena. Komunikacija se odvija na način da korisnik prilikom prijave na aplikaciju, ukoliko su podaci za prijavu ispravno uneseni, kao odgovor na prijavu dobiva JWT token koji je generiran pomoću javnog i privatnog ključa. Korisnik će prilikom svakog idućeg upita na API slati svoj JWT token kako bi API znao da se radi o postojećem registriranom korisniku te kako bi se osigurala sigurnost u komunikaciji na internetu. Nakon što API potvrdi da su upit i korisnički token ispravni, API komunicira s bazom te potom prosljeđuje podatke korisniku.

[14] *https://miro.medium.com/max/2800/1*WrMX8PcUWaRwF00deECtzQ.png*

5. Događaji unutar aplikacije

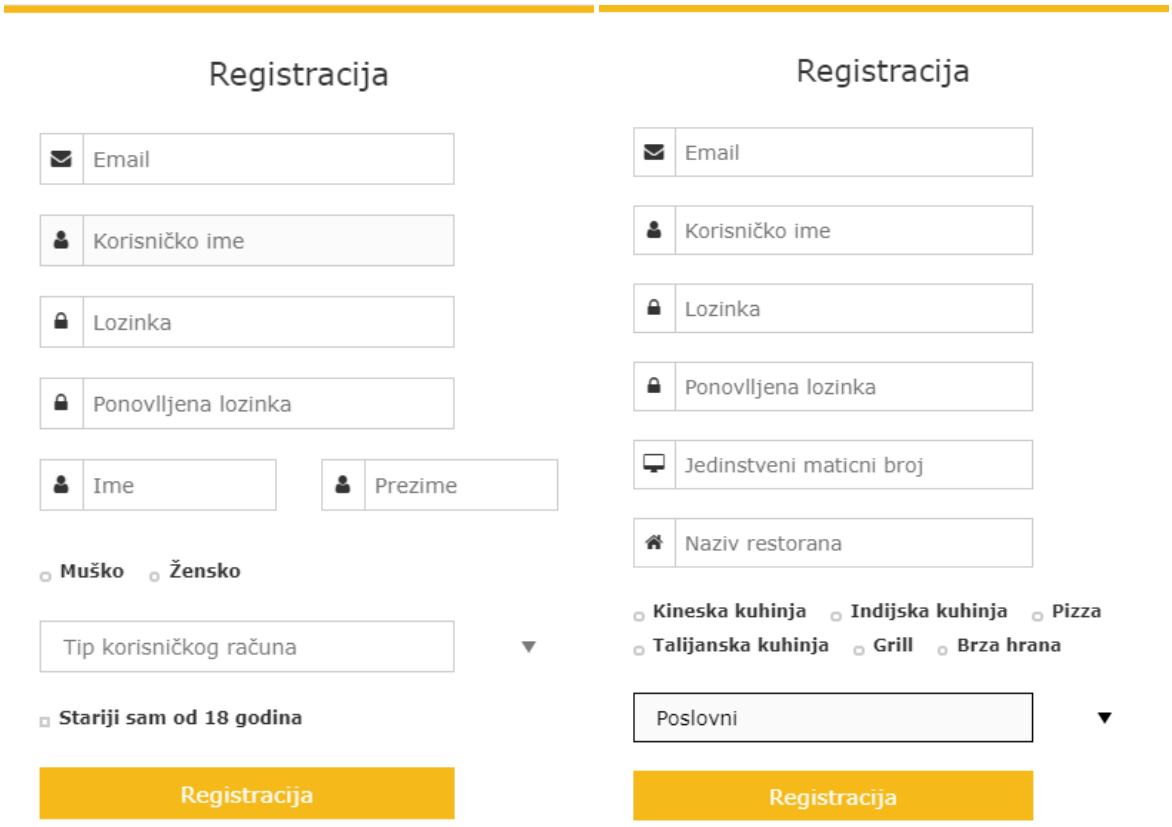
Aplikaciju karakteriziraju mnoge mogućnosti i implementirano je mnoštvo različitih funkcionalnosti koje se ujedno mogu promatrati i kao zasebni elementi aplikacije. Ovaj dio rada govoriti će o različitim događajima u aplikaciji te će pobliže biti objašnjen način na koji određene funkcionalnosti rade. Opisan će biti proces prijave i registracije u aplikaciju, bit će opisan način prikazivanja i pretraživanja restorana i proizvoda, proces narudžbe u aplikaciji kako s korisničke tako i sa strane restorana. Plaćanje će biti moguće preko postojećih API-ja za kartično plaćanje.

5.1. Registracija, prijava i rad s podacima

Registracija u aplikaciju sastoji se od nekoliko koraka koji će biti opisani u nastavku. U ovom dijelu će se proći kroz sve korake registracije i prijave u aplikaciju, bit će opisano dohvaćanje i spremanje podataka, korištenje JWT tokena u aplikaciji te sve biti popraćeno primjerima iz aplikacije.

5.1.1. Registracija u aplikaciju

Aplikacija podržava način rada kako za prijavljene tj. registrirane korisnike tako i za anonimne korisnike. Prilikom registracije, korisnik ispunjava registracijsku formu u koju unosi osnovne podatke o sebi potrebne za kreiranje korisničkog računa u aplikaciji. Registrirati se mogu poslovni i privatni korisnici. Nakon registracije, privatni korisnici imaju mogućnost pretraživanja gotovih jela, restorana i recepata. Mogu pratiti povijest svojih narudžbi, izrađivati favorizirane liste jela i restorana te ostavljati recenzije kako bi drugi korisnici vidjeli zadovoljstvo drugih korisnika iskazanih recenzijama. Princip rada za poslovne korisnike odnosi se na to da su korisnici u mogućnosti dodavati i izmjenjivati ponudu, prihvaćanje ili odbijanje narudžbe dobivene kroz aplikaciju i mogu pratiti povijest svojih prodaja putem aplikacije. Registracija za poslovne korisnike, za razliku od privatnih, zahtjeva odobrenje administratora kako se nebi desilo lažno registriranje i gomilanje lažnih poslovnih računa u aplikaciji. Registracijska forma za poslovne korisnike nešto je drugačija od forme za privatne korisnike jer sadrži polja za unos dodatnih podataka vezanih za restorane. Nakon unesenih podataka, podaci se zapisuju u bazu pozivom akcijske metode koja prosljeđuje podatke vanjskom API-ju koji komunicira s bazom.



Slika 9. Registracijske forme u aplikaciji

Slika 9. prikazuje razlike u registracijskim formama za privatne i poslovne korisnike. Forma je dinamički generirana ovisno o opciji izabranoj u padajućem izborniku s tipovima korisničkog računa. Uobičajeno ponašanje je registracijska forma za privatne korisnike.

5.1.2. Prijava u aplikaciju

Na primjeru prijave korisnika u aplikaciju vidjet će se međusobna komunikacija između sva tri sloja aplikacije. Nakon unošenja korisničkog imena i lozinke, aktivira se akcijska metoda koja primljene podatke s `LoginView`-a prosljeđuje u API koji komunicira s bazom. Nakon što API završi provjeru podataka u bazi, aplikacija prima povratnu informaciju o tome jesu li uneseni podaci ispravni. Ukoliko je prijava ispravna te ukoliko nije bilo nikakve druge greške, podaci o prijavljenom korisniku zapisuju se u web sesiju (engl. *session*) te dok god je prijavljen, korisniku su dozvoljene akcije za prijavljene korisnike. Kod 3. prikazuje metodu implementiranu u web servisu koja služi za provjeru ispravnosti dobivenih podataka.

Servis dohvati podatke iz baze te ovisno o postojanju zapisa u bazi, prosljeđuje potrebne informacije u aplikaciju.

```
[HttpPost]
  public ActionResult Login(string username, string password)
        {
            var successuful = false;
            using (var client = new HttpClient())
            {
                client.BaseAddress = GetUri();

                  var getDataTask = client.GetAsync(client.BaseAddress +
                  "getUserWithLogin?username=" + username + "&password="
                  + password).ContinueWith(response =>
                {
                    var result = response.Result;

                    if (result.IsSuccessStatusCode)
                    {
                      successuful = true;
                              HttpCookie appCookie = new
                              HttpCookie("LoggedInUser");
                              appCookie.Value =
                              result.Content.ToString();
                              appCookie.Expires =
                              DateTime.Now.AddMinutes(4);
                              appCookie.Path = "/Login";
                              Response.Cookies.Add(appCookie);
                              ViewBag.LoggedInUser = result.Content;
                              This.session[“loggedInUser”] =
                              result.Content;
                              var deserializedObject =
                              JsonConvert.DeserializeObject<User>(result.
                              Content.ReadAsStringAsync().Result);
                    }
                });
                getDataTask.Wait();
            }
            if (successuful)
                return View("Index");
            else
            {
                  ModelState.AddModelError(string.Empty, "Wrong username
                  or password. Try again!");
                return View();
            }

        }
```

Kod 4 . Akcijska metoda `Login`

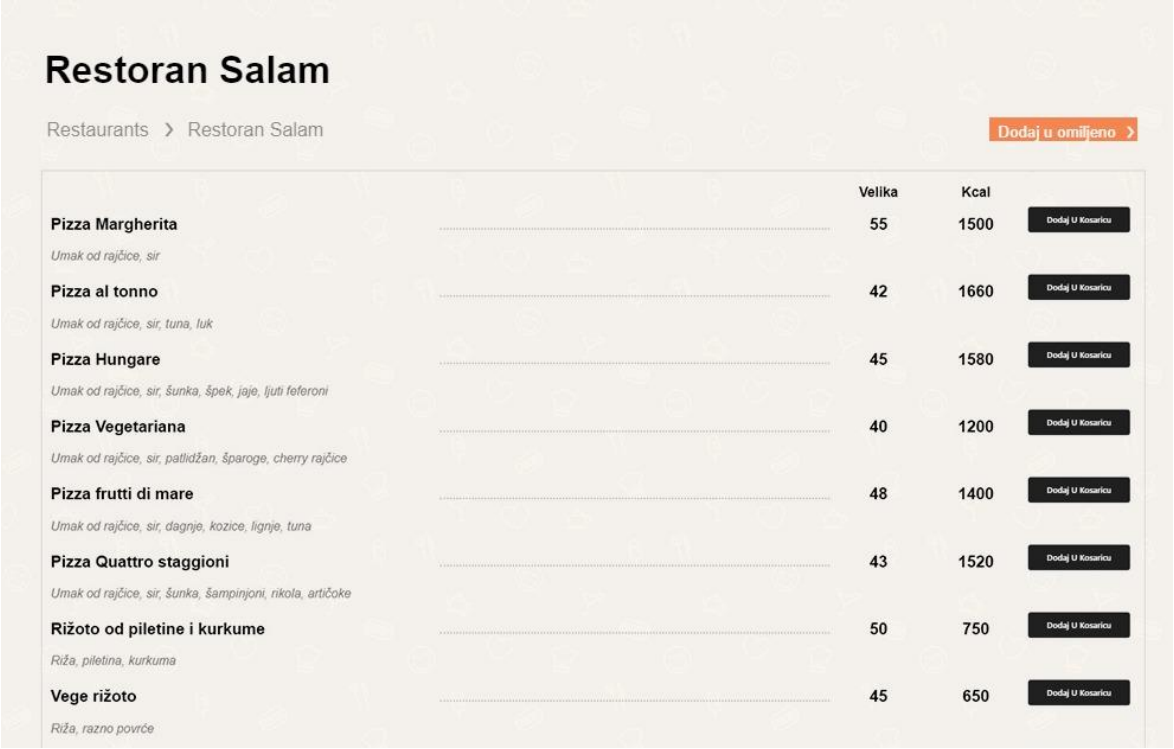
Na kodu 4. prikazana je akcijska metoda koja komunicira sa servisom te provjerava jesu li uneseni podaci za prijavu ispravni. Ovisno o rezultatu, poduzimaju se dodatne akcije. Ako su podaci točni, korisnik je uspješno prijavljen, a ukoliko su krivi, ispisuje se poruka o ne ispravnosti unesenih podataka.

5.2. Pretraživanje restorana i njihovih ponuda

Pregled dostupnih restorana i njihovih ponuda jedna je od funkcija aplikacije.

5.2.1. Dohvaćanje i prikaz podataka

Svi podaci vezani uz restorane i njihove proizvode čuvaju se u bazi podataka. Podaci se prilikom inicijalnog učitavanja stranice učitavaju iz baze i pohranjuju se u trenutnu sesiju web stranice. Podaci su logički organizirani i raspodijeljeni u liste objekata na kojima je moguće primjenjivati već implementirane funkcije za sortiranje, pretraživanje i povezivanje. Na pregledu namijenjenom za restorane korisnik može vidjeti sve restorane, primijeniti određenu filter funkciju kako bi dobio samo određene restorane, može izabrati između svojih naj omiljenijih restorana te može vidjeti sve informacije vezano za restorane. Nakon izbora restorana, podaci se prikazuju po `RestaurantView` shemi.

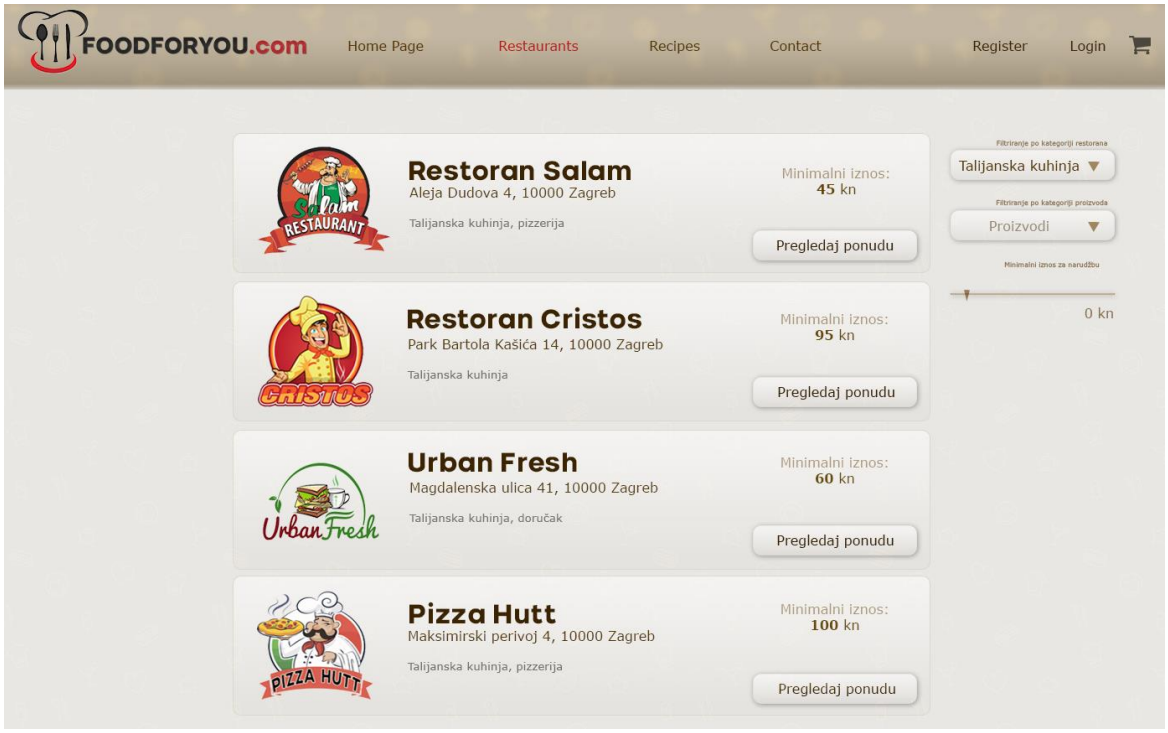


Slika 10. Prikaz menu-a restorana

Slika 10. prikazuje `RestaurantView` za odabrani restoran. Prikazuju se informacije o gotovim jelima dostupnima za narudžbu. Uz svako jelo nalaze se informacije o namirnicama koje se nalaze u istome, istaknuta je kalorijska vrijednost jela te je za svako jelo istaknuta cijena. Poslovni korisnici nakon prijave u aplikaciju mogu uređivati svoju ponudu. Definiranje novih proizvoda vrši se kroz `NewProductView.`

5.2.2. Akcije vezane uz restorane i proizvode

Kako bi se povećala jednostavnost pretrage po restoranima i proizvodima, implementirano je mnoštvo filtera za pretragu. Filteri po kategorijama restorana i jela, cijeni proizvoda i minimalnom iznosu narudžbe samo naj učestaliji su od njih. Korisnik ima mogućnost dodavanja jela i restorana u svoj popis favorita kako bi lakše i brže mogao doći do njih.



Slika 11. `RestaurantsView` nakon filtriranja

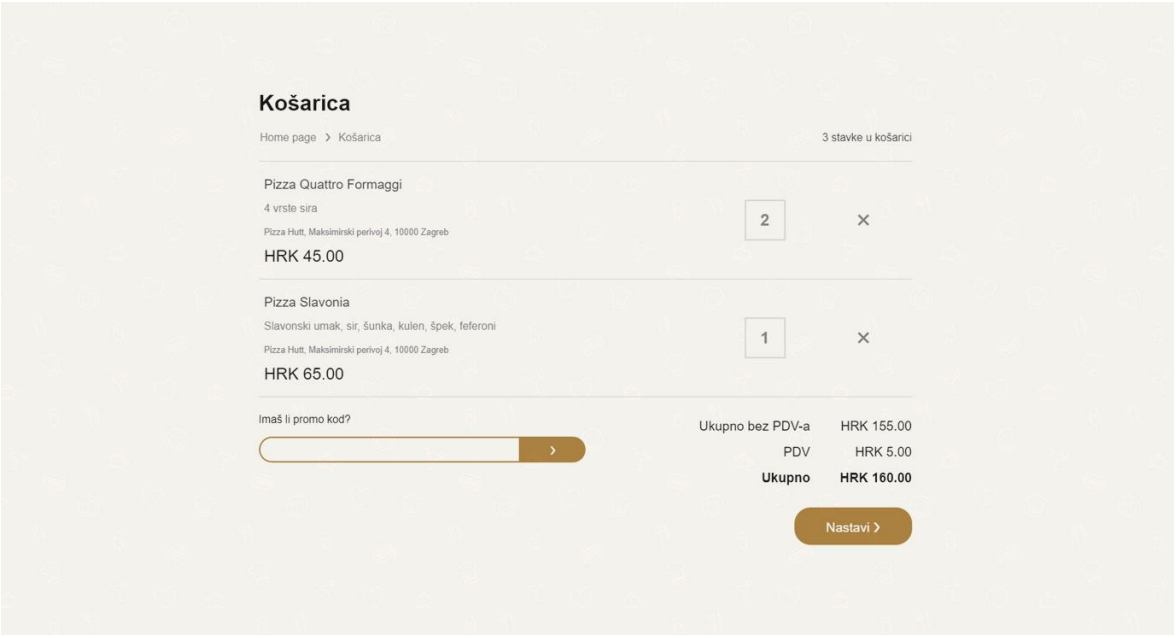
Na slici 10. vidljivo je kako izgleda prikaz restorana nakon primjene filtera. Funkcija za sortiranje radi na način da za svaki restoran provjeri kojoj kategoriji pripada, te ako je u kategoriji izabranoj u padajućem izborniku, restoran se i dalje prikazuje na formi. Restorani mogu imati definiran minimalni iznos narudžbe iznad kojeg vrše dostavu, jedna opcija filtriranja omogućava pregled restorana sortiranih ovisno o tom parametru. Ostale opcije odnose se na sortiranje proizvoda iz jelovnika restorana, ovisno o cijeni ili kategoriji.

5.3. Košarica i proces narudžbe unutar aplikacije

Funkcionalnost košarice u aplikaciji korisniku uvelike pomaže prilikom narudžbe jer omogućava praćenje stavki koje korisnik želi naručiti.

5.3.1. Košarica

Košarica je jedna od ključnih funkcionalnosti aplikacije zbog kojeg je narudžba uopće moguća. Ona je dio aplikacije koji sadrži podatke o stavkama koje korisnik želi naručiti te ih nakon toga procesira i izvršava narudžbu. Implementirana je na način da sadrži informacije o željenim stavkama, njihovu količinu i njihovu cijenu. `ShopingCartView` uz osnovne informacije o stavkama, sadrži polje za unos kupona u koje korisnik može upisati promotivni kod koji mu omogućuje popust. Nakon unosa promotivnog koda, poziva se akcijska metoda `CheckCouponInDatabase` implementirana u `ShopingCartContoller`-u koja poziva REST API koji nakon komunikacije s bazom, kao povratnu informaciju šalje status ispravnosti upisanog koda.



Slika 12. Izgled modula košarice u aplikaciji

Stavke koje se nalaze u košarici podložne su osnovnim promjenama, a to su promjena količine ili jednostavno brisanje stavke iz košarice. Pritiskom na gumb za nastavak, poziva se `OrderDataView` koji je zadužen za nastavak rukovanja trenutnom narudžbom. Na Slici 12. prikazan je izgled `ShopingCartView`-a na kojem su istaknute osnovne funkcionalnosti „košarice“.

5.3.2. Proces kreiranja i slanja narudžbe

Nakon što je korisnik odabrao stavke za narudžbu i nastavio s kupovinom, otvara se `OrderDataView` kojem je glavna funkcija prikupiti podatke potrebne za plaćanje i

provedbu narudžbe. Strukturom je raspodijeljen u dvije cjeline od kojega jedna sadrži podatke o korisniku tj. podatke za dostavu, dok je druga zadužena za informacije vezane za plaćanje. Osim imena, prezimena, adrese i kontaktnih podataka korisnika postoji mogućnost unosa dodatnih napomena potrebnih dostavljaču prilikom izvršavanja dostave. Korisnik, ovisno o mogućnostima restorana, može izabrati želi li narudžbu platiti gotovinom, kreditnom karticom ili PayPal računom. Za plaćanje karticom i PayPal-om implementirani su javno dostupni API-i za provedbu plaćanja na taj način. Nakon što su svi podaci uspješno uneseni, nakon potvrde narudžbe, API komunicira s bazom te šalje upit za potvrdu narudžbe restoranu za koga je narudžba namijenjena.

5.3.3. Prihvaćanje i potvrda narudžbe

Prihvaćanje i potvrda narudžbe od strane restorana posljednji su korak do uspješne narudžbe. Nakon što korisnik zatraži narudžbu te se podaci o narudžbi upišu u bazu, API izvršava poziv za potvrdu narudžbe upućen restoranu iz koga je narudžba zatražena. Uz pretpostavku da je ciljani restoran prijavljen na svoj poslovni profil u aplikaciji, dolazi mu obavijest u obliku zahtjeva za potvrdu narudžbe kako bi se ona sigurno izvršila i odradila. Restoran može prihvatiti ili odbiti narudžbu uz obvezni unos napomene. Korisnik dobiva povratnu informaciju od API-a je li njegova narudžba uspješno zaprimljena te koje je procijenjeno vrijeme koje je potrebno dostavljaču da dostavi naručeno jelo.

5.4. Pregled recepata i narudžba namirnica

Dodatna mogućnost koju aplikacija pruža, uz narudžbu gotovih jela, je i narudžba sirovih namirnica za izradu određenog jela po definiranom receptu. Korisniku je omogućen pregled unesenih recepata te uputstva za pripremu istog. Ova funkcionalnost u aplikaciji značajna je po tome što omogućuje jednostavno učenje kuhanja kod korisnika, a ujedno je na tržištu izdvaja od ostalih aplikacija. Tijek narudžbe je jednostavniji nego kod gotovih jela iz razloga što nije potrebna komunikacija s „drugom stranom“ već se narudžba smatra valjanom čim korisnik potvrdi narudžbu i unese potrebne korisničke podatke i podatke o plaćanju. Cilj uvođenja ovog modula u aplikaciju bio je napraviti platformu za razmjenu znanja te povećanje publike kojoj je aplikacija namijenjena.

6. Pregled postojećih rješenja

Internet je golema baza aplikacija i raznih sadržaja pa tako nije čudno da već postoje aplikacije sličnih funkcionalnosti. Predstavljene će biti trenutno najkorištenije i najpoznatije aplikacije sličnih funkcionalnosti, svaka će biti individualno opisana te će se naposlijetku usporediti sve aplikacije te prikazati njihove prednosti i mane nad drugima.

6.1. Pauza.hr web aplikacija

Pauza.hr, trenutno možda i najpopularnija aplikacija za online naručivanje hrane na tržištu u Hrvatskoj, omogućuje narudžbu iz preko 150 restorana raspodijeljenih u 8 gradova. Inicijalno, bez prijave u aplikaciju, korisniku se pruža opcija unosa adrese, putem koje aplikacija sortira registrirane restorane te korisniku prikazuje samo one kojima je unesena adresa u krugu za dostavu. Aplikacija na svojoj početnoj strani ima tzv. brze filtere gdje se mogu pretražiti restorani po kvartovima ili gradovima. Nakon što aplikacija prikaže trenutno otvorene restorane, korisnik ima mogućnost pregleda meni-ja restorana te može dodavati stvari u košaricu. Nakon što završi dodavanje u košaricu, korisnik se mora prijaviti u aplikaciju ukoliko želi završiti i platiti trenutnu narudžbu. Ovisno o restoranu, moguće su različite metode plaćanja. Plaćanje gotovinom dostavljaču, plaćanje putem PayPal računa i kartično plaćanje mogućnosti su aplikacije. Nakon što korisnik završi narudžbu, ista se šalje restoranu koji mora odobriti narudžbu kako bi ona bila pravovaljana i tek kad korisnik dobi potvrdu o uspješnoj narudžbi, ona se smatra uspješnom. Aplikacija može raditi na Hrvatskom ili Engleskom jeziku. Prilikom prijave i registracije u aplikaciju, korisnik odabire je li novi ili postojeći korisnik te se ovisno o tome prikazuju različite opcije za prijavu. Prijava je moguća uz Facebook i Google+ račun također.

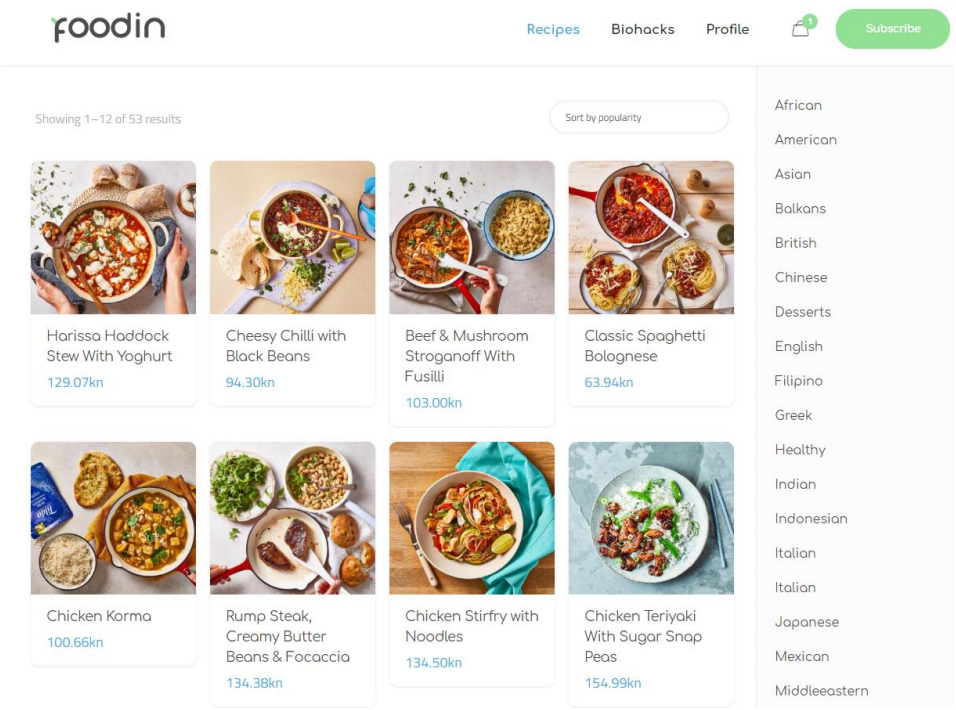
6.2. Dobartek.hr

Dobartek.hr također je aplikacija koja nudi online naručivanje hrane. Za razliku od Pauza.hr aplikacije ova aplikacija također sadrži i opciju da korisnik osobno dođe u restoran „pokupiti“ naručenu hranu. Dio aplikacije zadužen za dostavu moguć je samo u gradu Zagrebu i broji nešto više od 10 registriranih restorana. Nakon što se na početnoj stranici izabere opcija osobnog preuzimanja hrana, može se primijetiti da je rad aplikacije omogućen

a desetak drugih gradova i da se broj restorana znatno povećao. Na početnoj stranici moguće je koristiti pametne filtere kako bi korisnik lakše mogao pretraživati one restorane i jela koja on želi. Restorani se mogu favorizirati što omogućuje filtraciju restorana na „Nove“, „Sve“ i „Omiljene“ kako bi se brže moglo pristupiti omiljenom restoranu. Filteri koji se također mogu koristiti su za kategorije dostupnih jela i za način plaćanja koji može biti gotovina, online plaćanje ili kartično plaćanje. Klikom na dostupni restoran otvara se pregled (engl. *view*) definiran za njega koji prikazuje dostupna jela u tom restoranu. Nakon dodavanja stavki u košaricu, korisnik se mora prijaviti u aplikaciju kako bi mogao završiti narudžbu. Prijava je moguća putem računa kreiranog za aplikaciju i prijava uz pomoć Facebook profila korisnika. Na samom kraju moguće je izabrati i unijeti podatke za dostavu i podatke o načinu plaćanja.

6.3. Foodin.io

Foodin.io je nešto drugačija aplikacija od gore navedenih, ali zbog funkcionalnosti nove aplikacije biti će ubrojena u ovu analizu. Ova aplikacija ne nudi narudžbu gotovih jela iz restorana već omogućuje pregled recepata i narudžbu namirnica namijenjenih za izradu određenog jela. Aplikacija trenutno radi samo na području grada Zagreba. Na početnoj stranici aplikacije nalaze se informacije o cijeloj aplikaciji i način na koji aplikacija funkcionira. Glavne dvije mogućnosti aplikacije su narudžba sirovih namirnica za recepte koji su dostupni u aplikaciju i narudžba sirovih namirnica potrebnih za tjednu ishranu po određenom planu prehrane. Trenutno je ovo jedinstvena takova aplikacija na tržištu u Hrvatskoj. Za registraciju je dovoljan samo email na koji se nakon unosa u registracijsku formu šalje password koji je korisnik kasnije dužan koristiti. Prijava u aplikaciju ide putem unesenog maila prilikom registracije i passworda koji je dostavljen na tu email adresu. Nakon što korisnik izabere namirnice koje želi, mora se prijaviti kako bi završio narudžbu. Nakon unosa kontaktnih podataka i podataka vezano za dostavnu adresu, moguće je izabrati način plaćanja koji je online kartično plaćanje. Na slici 13. vidljiv je izbornik za recepte dostupan u aplikaciji.



Slika 13. Prikaz recepata dostupnih u aplikaciji[15]

6.4. Web aplikacija za dostavu hrane

Web aplikacija za dostavu hrane je aplikacija koja je nastala nakon analize postojećih aplikacija na tržištu i cilj je ujediniti različite funkcionalnosti u jedinstvenu aplikaciju. Osim narudžbe gotovih jela iz restorana, moguć je i pregled recepata za izradu određenih jela te narudžbu namirnica potrebnih za izradu istog. Uz svako jelo zabilježena je energetska i kalorijska vrijednost obroka kako bi korisnici lakše mogli pratiti svoje prehrambene navike. Početna stranica aplikacije prikazuje informativne grafike, istaknutu kategoriju restorana za brzi pregled, istaknute recepte te ostale informacije o samom korištenju aplikacije. Korisnik može pretraživati registrirane restorane te njihovu ponudu jela. Dodavanjem u košaricu, korisnik priprema narudžbu koju nakon prijave u aplikaciju može provesti. Druga bitna funkcionalnost aplikacije je pregled dostupnih recepata u aplikaciji te ukoliko želi, korisnik može naručiti namirnice potrebite za izradu tog jela. Nakon što se korisnik prijavi, u mogućnosti je završiti narudžbu klikom na „Naruči!“ gumb. Potrebno je unijeti podatke za dostavu, dodatne napomene ukoliko ih ima i način plaćanja kako bi nakon što restoran potvrdi narudžbu ona bila pravovaljana.

[15] *https://foodin.io/hr/kategorija-proizvoda/recipes/*

6.5. Usporedba rješenja

Usporedbom dostupnih aplikacija vidljivo je da se razlikuju u glavnim značajkama funkcionalnosti te da je svaka od njih jedinstvena na svoj način. Aplikacija pauza.hr usmjerena je samo na narudžbu gotovih jela iz registriranih restorana. Najpopularnija je na tržištu za tu uslugu, a to se očituje brojem registriranih restorana i brojem gradova u kojima su dostupne njihove usluge. Dobartek.hr nešto je drugačija od aplikacije „pauze“, a to je vidljivo po opcijama koje nudi. Dostava hrane ograničena je samo na nekolicinu zagrebačkih restorana, ali implementacijom opcije da korisnik osobno preuzme narudžbu u restoranu, povećali su broj poslovnih partnera s kojima surađuju. Omogućili su restoranima koji nemaju organiziranu dostavu za svoje proizvode da se bez problema mogu registrirati, promovirati i poslovati na aplikaciji. Cilj povećanja opcija bio je uključivanje većeg broja restorana u aplikaciju te da se zapravo može registrirati tko god želi neovisno o mogućnosti dostavljanja. Uz aplikacije za dostavu gotovih jela, foodin.io, nova aplikacija za narudžbu hrane, se na tržištu predstavio kao aplikacija putem koje je moguće naručiti sirove namirnice za izradu jela po receptu na izbor. Aplikacija također nudi opciju narudžbu namirnica na tjednoj bazi tj. dostupni su tjedni planovi prehrane koji imaju definirani svaki obrok kroz tjedan dana pa opcija omogućava narudžbu svih potrebnih namirnica za sve recepte u planu. Trenutni nedostatak ove aplikacije je loše izvedeno korisničko sučelje i pokoji „bug“ koji se javlja prilikom pretraživanja stranice. Nova aplikacija za dostavu hrane uključuje većinu gore navedenih funkcionalnosti i zbog toga je ona najsadržajnija što se tiče ponude. Uz opciju dostave gotovih jela iz registriranih restorana, nudi mogućnost pregled recepata u aplikaciji isto kao i narudžbu namirnica potrebnih za izradu jela. Dodatno, uz svako jelo dostupne su informacije o energetskoj i kalorijskoj vrijednosti obroka što u današnje vrijeme sve više ljudi prati i planira prehranu ovisno o tim čimbenicima.

Zaključak

U današnje vrijeme, programiranje je jedno od zanimanja koje od programera zahtijeva da konstantno razvija svoje znanje i uči nove tehnologije. Zbog stalnog dolaska novih tehnologija na tržište, programeru se pruža mogućnost da unaprjeđuje aplikaciju implementacijom novih, dosad možda ne izvedivih, mogućnosti.

Ovaj rad naučio me o važnosti planiranja, arhitekture i procjeni posla. Analiza i planiranje su prvi korak u razvoju aplikacije, a ujedno se pokazao kao jedan od naj bitnijih u cijelom procesu. Dobro isplanirana aplikacija u početku omogućuje praćenje dostizanja unaprijed definiranih ciljeva. Prilikom razvoja aplikacije, ukoliko je planiranje i analiza bila dobra, ne bi trebalo biti puno odstupanja u vremenu potrebnom za razvoj niti u samoj implementaciji. Kod lošeg ili ne isplaniranog razvoja aplikacije javlja problem o ne poznavanju opsega aplikacije. Programer u toku razvoja isprobava mogućnosti i razvija funkcionalnosti koje prije nisu bile definirane, ne zna se jasni cilj aplikacije te se ne može pratiti koliko je zapravo stvarnog posla napravljeno. Dobro isplanirani model baze podataka, klase unutar aplikacije te željene funkcionalnosti određenog modela u aplikaciji programeru daju jasniju sliku o samoj aplikaciji. Izrada ove aplikacije naučila me da ulaganjem vremena u analizu i planiranje razvoja, itekako štedim vrijeme koje bi potrošio na sami razvoj aplikacije. Izbor prave arhitekture i raspodjela aplikacije na slojeve omogućuje lakše održavanje koda, jasniju raspodjelu zadataka aplikacije te su kasnija proširivanja aplikacije puno jednostavnija. Prilikom izrade ove aplikacije najveći izazov bio mi je implementacija novih, meni dosad ne poznatih tehnologija u rad. Naučio sam da su kod programiranja najbitniji osnovni programerski koncepti i pravila te da se zapravo poznavanjem istih svaka tehnologija može vrlo jednostavno implementirati u rad sustava. Dodatni izazov mi je bio razvoj aplikacije u slojevima te razvoj dodatnog API-ja, ali dobra analiza i planiranje aplikacije su otklonili probleme koji su se potencijalno mogli pojaviti.

Dobivena znanja i iskustvo su najvrjedniji dio ovog rada i smatram ih kao dobru podlogu za daljnje razvijanje u smjeru izrade web aplikacija i programiranja općenito.

Popis slika

Popis kôdova

Literatura

[1] MVC arhitektura, https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.html

[2] Baza podataka, SQL https://www.guru99.com/introduction-to-database-sql.html

[3] ER Model https://www.techopedia.com/definition/7057/entity-relationship-model-er-model

[4] REST servis, https://www.codecademy.com/articles/what-is-rest

[5] SOAP, REST, https://www.soapui.org/learn/api/soap-vs-rest-api.html

[6] JSON, XML, https://www.guru99.com/json-vs-xml-difference.html

[7] JWT Token, https://jwt.io/

[8] Arhitektura aplikacije, 3 slojni razvoj aplikacije, RICHARDS, M., Software Architecture Patterns (2015.)

ALGEBRA
VISOKO
UČILIŠTE

WEB APLIKACIJA ZA NARUDŽBU HRANE

Pristupnik: Luka Vuković, 0321005673

Mentor: Aleksander Radovan, dipl. ing.