



УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У  
НОВОМ САДУ



Игор Јелић, ПР4/2016

ВЕБ АПЛИКАЦИЈА ЗА ДОСТАВУ

ПРОЈЕКАТ

- Примењено софтверско инжењерство (ОАС) -

Нови Сад, 24.10.2022.

## SADRŽAJ

1. OPIS REŠAVANOG PROBLEMA
2. OPIS KORIŠĆENIH TEHNOLOGIJA I ALATA
3. OPIS REŠENJA PROBLEMA
4. PREDLOZI ZA DALJA USAVRŠAVANJA
5. LITERATURA

## OPIS REŠAVANOG PROBLEMA

Cilj aplikacije za dostavu jeste realizacija kontrole i upravljanja dostavljačkim sistemom. Sam dostavljački sistem obuhvata korisnike, porudžbine i proizvode koje je moguće poručiti. Ceo sistem je namenjen za tri grupe korisnika, administratora sistema, dostavljača i potrošača, od koje zavisi i sama uloga korisnika u sistemu.

Administratorsku ulogu može imati jedan ili više nadležnih organa celokupnog sistema, čiji je osnovni zadatak zapošljavanje novih dostavljača i upravljanje raspoloživim proizvodima. Administratori, takođe, imaju i uvid u sve porudžbine u sistemu, kao i u njihov status u datom momentu.

Dostavljači jesu korisnici koji vrše same dostave porudžbina. Uslov za početak rada je verifikacija dostavljačkog profila od strane administratora. Obaveštenje o verifikaciji/blokiranju korisničkog profila, dostavljač dobija putem elektronske pošte. Kada je ulogovan u sistem, dostavljač ima mogućnost da vidi sve trenutne porudžbine koje čekaju dostavljača, da prihvati onu porudžbinu koja mu odgovara, kao i da pregleda istoriju njegovih dostavljenih porudžbina. Prihvatanjem određene porudžbine, dostavljač aktivira tajmer koji odbrojava do momenta isporuke poručenih proizvoda i nije više u mogućnosti da pregleda i bira novu porudžbinu, dokle god trenutna nije isporučena.

Potrošači predstavljaju krajnje korisnike usluga dostavljačke kompanije i njihove mogućnosti su poručivanje proizvoda, i pregled istorije primljenih porudžbina. Sam potrošač može u svakom momentu da proveriti status trenutne porudžbine. Porudžbina može biti u stanju čekanja na dostavljača ili u stanju "preuzeta", koje označava da je dostava proizvoda u toku. Kada dostavljač preuzme porudžbinu, samim tim, otpočne dostavu, potrošač na stranici trenutne porudžbine dobija tajmer koji odbrojava do momenta isporuke. Kada se isporuka izvrši, potrošač može da kreira novu porudžbinu. Potrošač može imati samo jednu aktivnu porudžbinu istovremeno.

Samo utvrđivanje identiteta kao i autorizacija se vrši prilikom prijavljivanja na sistem. Proveravaju se korisnički kredencijali, email i lozinka, a na osnovu uloge ulogovanog korisnika dodeljuju mu se određena prava pristupa delovima aplikacije i pristup odgovarajućim akcijama pozadinskog servisa.

Pre prijave na sistem, korisnici treba da se registruju i kreiraju lični nalog popunjavajući formu ličnim informacijama.

Prilikom registracije korisnik bira da li se registruje kao dostavljač ili potrošač. Nije moguća registracija novih administratora, oni se učitavaju iz sistema.

Nakon prijave, svi korisnici imaju mogućnost pregleda i modifikovanja ličnih informacija, kao i ažuriranje profilne slike korisnika.

## OPIS KORIŠĆENIH TEHNOLOGIJA I ALATA

Projekat je realizovan u Visual Studio 2019 i Visual Studio Code okruženjima, koristeći ASP.NET Core Web API za pozadinske veb-servise, Ocelot za API gateway i Angular za razvoj korisničkog interfejsa i celokupnog frontenda, baziranog na principu single-page aplikacija.

**Microsoft Visual Studio**[1] - Integrisano programsko okruženje kreirano od strane Microsoft kompanije. Unutar sebe podržava 36 različitih programskih jezika. Služi za programiranje desktop aplikacija, računarskih igara, veb-sajtova, veb-servisa i veb-aplikacija. Visual Studio takođe podržava XML/XSLT, HTML/XHTML, JavaScript, CSS, Java i J#. Microsoft pruža i besplatnu verziju programa Visual Studio pod nazivom Community Edition koja podržava dodatke i dostupna je bez ikakvih troškova.

**Ocelot**[2] - lightweight, open-source API Gateway namenjen .NET Core aplikacijama koje koriste mikroservisnu arhitekturu i potrebna im je pojedinačna pristupna tacka, na koju stižu svi HTTP zahtevi. Ocelot predstavlja skup middleware-a kroz koje prolazi svaki HTTP zahtev, određenim redosledom. Na osnovu konfiguracije, Ocelot manipuliše objektom HTTP zahteva i na kraju protočnog mehanizma samog Ocelota, zahtev se prosleđuje ka odgovarajućem DownStream servisu. Takođe, omogućava autentifikaciju i autorizaciju zasnovanu na bearer tokenima i samim claim-ovima ugrađenim u token.

**Angular**[3] - platforma za razvoj single-page veb-aplikacija, koja u osnovi koristi TypeScript jezik. Kao platforma, Angular je:

- frejmwork baziran na komponentama, za izgradnju skalabilnih veb-aplikacija
- kolekcija integrisanih biblioteka koje pokrivaju širok spektar slučajeva upotrebe poput rutiranja, rad sa formama, komunikacija klijentske aplikacije sa serverom
- skup alata koji olakšavaju razvoj, testiranje i modifikaciju koda

Dizajniran je da učini nadogradnju koda što jednostavnijom.

**ASP.NET Core Web API**[4] - podrška za kreiranje RESTful servisa, takođe poznatih kao API, koristeći C#. Za obradu HTTP zahteva, Web API koristi kontrolere. Web API kontroleri jesu klase koje nasleđuju ponašanje bazne klase ControllerBase.

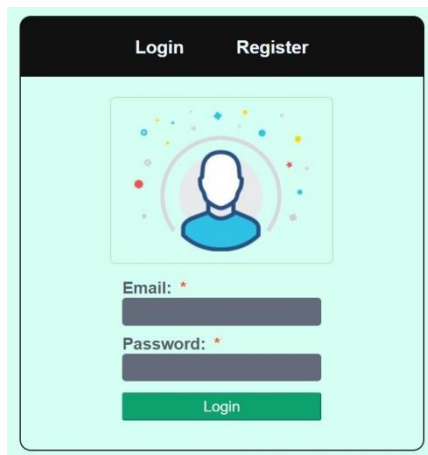
**Entity Framework Core**[5] - open source, cross-platform verzija Entity Frameworka, popularne tehnologije za pristup podacima unutar baze podataka. EF Core služi kao objektno-relacioni mapper koji omogućuje .NET programerima da rade sa bazom podataka koristeći .NET objekte. Takođe, eliminiše potrebu za pisanje većine koda potrebnog za pristup podacima iz baze podataka. Podržava mnoge tipove baza podataka poput SQL Server, PostgreSQL, MySQL, Oracle DB itd.

**AutoMapper**[6] - mapper objekata jednog tipa u objekte drugog tipa. Dokle god se poštuju ustanovljene konvencije AutoMappera, potreba za posebnim konfiguracijama za mapiranje dva tipa je gotovo nepostojeća.

## OPIS REŠENJA PROBLEMA

### Neprijavljen korisnik

Prvim ulaskom na aplikaciju za dostave, pred korisnikom se nalazi prozor sa dve komponente. Jedna nudi mogućnost za prijavu na sistem, dok druga podrazumeva formu za registrovanje novog korisnika. Prelaz sa jedne komponente na drugu se vrši pritiskom na “Login” i “Register” hiperlinkove.

The image shows a mobile application login screen. At the top, there is a black header bar with two white text links: "Login" and "Register". Below the header, the background is a light teal color. In the center, there is a white rounded rectangle containing a user profile icon (a blue circle with a white outline of a person's head and shoulders) surrounded by small, colorful dots. Below the icon, there are two input fields: "Email: \*" and "Password: \*", each with a corresponding text input box. At the bottom of the white rounded rectangle is a green button with the white text "Login".

Slika 1.1. Login forma

Forma za prijavu korisnika na sistem (Slika 1.1) sadrži polja za unos korisničkih kredencijala. Traži se unos e-mail adrese kao i odgovarajuća lozinka vezana za profil registrovan na konkretnu e-mail adresu.

Validacija sa klijentske strane je prisutna tako što dugme za prijavu na sistem nije moguće aktivirati dokle god forma za prijavu nije validna. Pored deaktiviranja dugmeta za prijavu, korisniku će crvenom bojom biti uokvirena polja koja je dodirnuo, ali unos koji je uneo ne ispunjava sve uslove validnosti konkretnog podatka.

Sa serverske strane, validacija podrazumeva proveru validnosti modela podataka koji je stigao u telu zahteva sa prednje strane. Ukoliko su podaci validni, u bazi podataka se traži kombinacija prosledjene e-mail adrese i kriptovane vrednosti unete lozinke. Za kriptovanje lozinke prilikom registracije kao i prilikom verifikovanja kriptovanih vrednosti unete lozinke prilikom prijave na sistem, koristi se biblioteka Bcrypt.Net-Next[7] i njene metode Verify i HashPassword.

Posle uspešne validacije kredencijala, na serverskoj strani se formira token u koji se upisuju klejmovi koji će kasnije služiti kao pomoć pri autorizaciji na zadnjoj strani, kao i informacije koje će koristiti klijentska aplikacija (ime, uloga korisnika, putanja do profilne slike...).

Slika 1.2. Forma za registraciju novih korisnika

Registrowanje novih korisnika se vrši putem registracione forme prikazane na slici 1.2. Od korisnika se traži unos njegovih osnovnih podataka, kao i odabir njegove uloge u sistemu. Ukoliko korisnik pokuša da izvrši registraciju pritiskom na dugme “Register”, zahtev će se poslati na server tek ukoliko je korisnik uspešno uneo sve validne podatke i sa računara učitao profilnu sliku. U suprotnom, biće obavešten putem iskaćućeg crvenog prozora sa desne strane da forma nije validna.

Kada zahtev stigne na korisnički mikroservis, proverava se da li već postoji profil sa istom e-mail adresom, kao i validnost ostalih podataka. U slučaju da je prosleđeni model uspešno validiran i sa serverske strane, korisničke informacije, zajedno sa keširanom vrednošću korisničke lozinke, se čuvanju u bazi podataka i vraća se odgovor da je registracija uspeła.

Konačno, korisnik se upućuje na stranicu za prijavu na sistem i pruža mu se prilika da se odmah prijavi na novoregistrovani profil.

## Prijavljen korisnik

Uspešnom prijavom na sistem, korisnik se preusmerava na početnu stranu aplikacije (Slika 2.1.), na kojoj vidi korisnički meni(1) i komponentu za sadržaj(2).



Slika 2.1. Početna stranica prijavljenog korisnika

**Korisnički meni** (1) se razlikuje u zavisnosti od uloge korisnika u sistemu. Sa leva na desno, korisnik ima priliku da vidi svoju profilnu sliku, ime i prezime, a zatim i linkove ka ostalim stranicama koje su dostupne njegovoj roli. Na samom kraju menija nalazi se link za odjavu sa sistema.

Samo kod dostavljačke grupe korisnika, profilna slika u meniju je uokvirena odgovarajućom bojom koja odgovara statusu njegovog profila. Zelena boja označava da je profil aktiviran od strane admina, narandžasta da se potvrda i dalje čeka, dok je crvenom bojom uokvirena profilna slika odbijenog dostavljača.

**Komponenta za sadržaj** (2) drži najveću površinu prozora prijavljenog korisnika. U njoj se nalazi glavni sadržaj koji se dinamički menja rutiranjem kroz aplikaciju pomoću hiperlinkova korisničkog menija.

Na slici 2.1. je prikazan početni prikaz nakon prijave korisnika, na kom se nalaze kartica sa osnovnim korisničkim informacijama (3) i veći prikaz korisničke profilne slike (4). U samoj kartici se nalazi i dugme “Edit Profile”, koje vodi na stranicu za ažuriranje korisničkih podataka (Slika 2.2.).



The image shows a web form for updating user profile information. It has a light green background. The fields are: 'Email:' with the value 'd2@gmail.com', 'First Name:' with 'Deliverer', 'Last Name:' with 'Two', and 'Date Of Birth:' with '12/02/1999' and a calendar icon. Below these is a button labeled 'Profile picture'. At the bottom is a green button labeled 'Save Changes'.

Slika 2.2. Forma za ažuriranje korisničkih podataka

Prilikom učitavanja komponente (Slika 2.2.), polja forme za ažuriranje korisničkih podataka su automatski popunjena trenutnim korisničkim informacijama učitanim iz baze podataka.

Dugme “Profile picture” omogućava promenu profilne slike korisnika, dok se pritiskom na dugme “Save Changes” na server šalje zahtev za ažuriranje korisničkih informacija prosleđenih kroz formu.

Uspešnim ažuriranjem profila, korisnik se preusmerava na početnu stranicu.

## Administrator

Glavne uloge administratora u sistemu aplikacije za dostavu su upravljanje dostavljačima celog sistema i dodavanje proizvoda koji su na raspolaganju prilikom potrošačkih porudžbina.

Na stranici do koje se stiže klikom na link „PRODUCTS“, nalazi se prikaz svih proizvoda sistema koji su trenutno na raspolaganju za poručivanje, kao i dugme za dodavanje novog proizvoda (Slika 3.1.).



Slika 3.1. Prikaz svih proizvoda

Proizvodi su predstavljeni karticama (1) koje drže informacije o nazivu, ceni i sastojcima konkretnog proizvoda. Klikom na dugme za dodavanje novog proizvoda (2), admin se preusmerava na stranicu koja podrazumeva formu za unos podataka o novom proizvodu (Slika 3.2.).

### NEW PRODUCT

Name:

Price:

Ingredients:

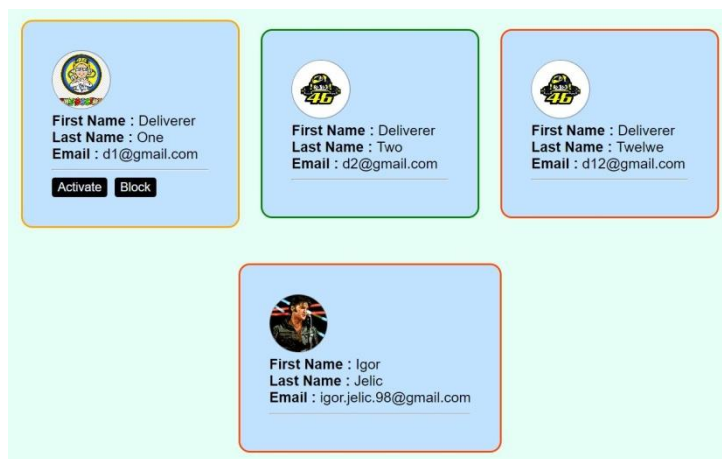
Slika 3.2. Dodavanje novog proizvoda

Proizvod nije moguće dodati ukoliko neko od polja forme ostane prazno. Prosleđeni podaci o novom proizvodu se validiraju i na serverskoj strani, uz proveru jedinstvenosti naziva proizvoda. U sistemu ne mogu postojati dva proizvoda sa istim nazivom.

Stranica za verifikaciju profila dostavljača (Slika 3.3.) se sastoji od kartica sa osnovnim informacijama o svakom dostavljaču u sistemu. Aktivirani dostavljači su uokvireni zelenom, a odbijeni crvenom bojom. Kartice dostavljača čiji profil i dalje čeka na aktivaciju su uokvireni narandžastom bojom i pored informacija o dostavljaču, sadrže i dugmad za aktiviranje/blokiranje korisnika.

Klikom na dugme za aktiviranje ili blokiranje profila dostavljača, na korisnički mikroservis se šalje zahtev za aktivaciju/blokiranje odabranog dostavljača. Nakon promene statusa profila u bazi podataka, pomoću EmailService biblioteke koja koristi NETCore.MailKit[8] nuget package, na e-mail adresu dostavljača, aplikacija šalje e-mail poruku kojom se dostavljač obaveštava da je njegov zahtev za registraciju obrađen i da je njegov profil sada aktiviran ili odbijen.





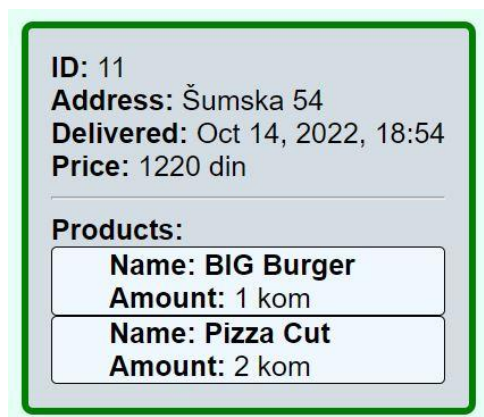
Slika 3.3. Verifikacije dostavljača

Pored ova dva zadatka, administratori imaju mogućnost da pregledaju sve porudžbine u sistemu, nebitno da li one bile isporučene, prihvaćene ili u stanju čekanja na dostavljača.

Isporučene porudžbine su uokvirene zelenom, prihvaćene narandžastom, a one koje i dalje čekaju dostavljača crvenom bojom.

Da pregledaju istoriju svojih dostavljenih, kao i uspešno preuzetih porudžbina takođe imaju dostavljači, odnosno potrošači. Na slici 3.4. je prikazan izgled kartice na kojoj se nalaze informacije o jednoj iz skupa narudžbina.

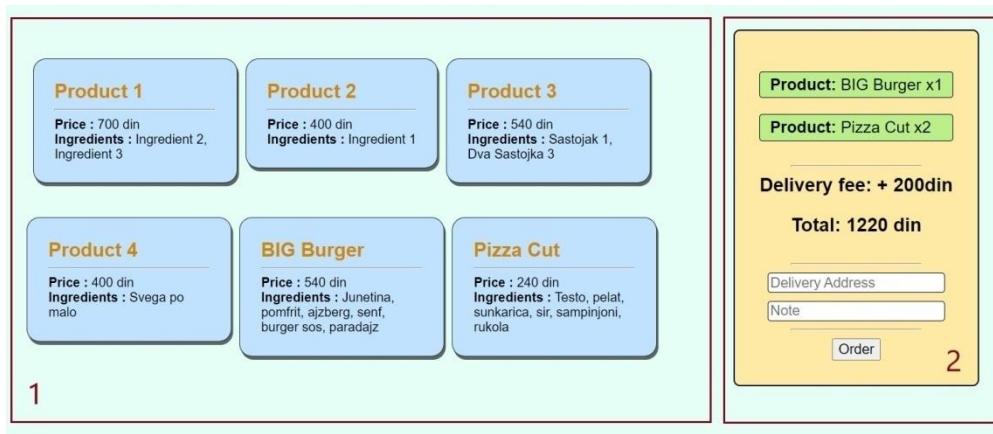
Potrošači i dostavljači imaju pravo da pregledaju jedino porudžbine čija je dostava završena, pa u njihovom slučaju kartice porudžbine nisu uokvirene nikakvom bojom.



Slika 3.4. Prikaz jedne od izvršenih porudžbina (admin)

## Potrošač

Potrošač aplikacije za dostavu pored uređivanja svog profila i pregleda dosadašnjih uspešnih porudžbina, ima mogućnost kreiranja i poručivanja novih porudžbina. Jedan potrošač u jednom trenutku može da čeka samo jednu porudžbinu.



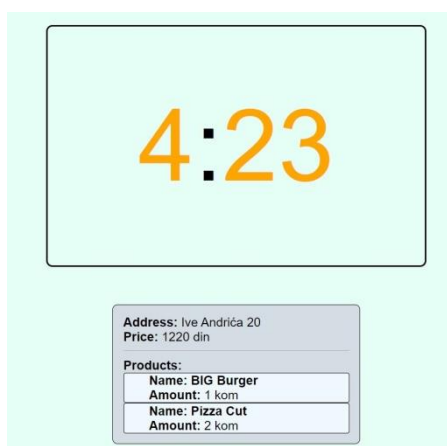
Slika 4.1. Kreiranje nove porudžbine

Prikaz stranice na kojoj potrošač kreira novu porudžbinu (Slika 4.1.) je podeljena u dve particije. Sa leve strane (1) nalaze se grupisani svi raspoloživi proizvodi u sistemu. Klikom na svaku od kartica proizvoda, proizvod se dodaje u korpu (2) koja se nalazi na krajnjoj desnoj strani prikaza. Proizvodi se podjednako lako i izbacuju iz korpe, klikom na proizvod koji se nalazi u korpi.

Aplikacija na automatski izračunatu sumu cena svih proizvoda dodaje i fiksnu cenu dostave. Pre kompletiranja porudžbine potrošač je još dužan da unese adresu dostave i komentar za dostavljača.

Nakon uspešno kreirane porudžbine, potrošač se usmerava na stranicu trenutne porudžbine gde ima priliku da vidi detalje o trenutnoj porudžbini. Ukoliko je porudžbina preuzeta, na ekranu se pojavljuje tajmer (Slika 4.2.a) koji odbrojava do momenta isporuke, u suprotnom, potrošač je obavešten da porudžbina i dalje čeka svog dostavljača (Slika 4.2.b).

Potrošaču je onemogućeno rutiranje do stranice za kreiranje nove porudžbine dokle god porudžbina ne bude prihvaćena i tajmer ne odbroji do nule, tj. dok dostava ne bude izvršena.



Slika 4.2.a Potvrđena trenutna porudžbina

The order **[ID:12]** is awaiting for the delivery person! :)

<b>Address:</b> Ive Andrića 20
<b>Price:</b> 1220 din
<b>Products:</b>
<b>Name:</b> BIG Burger
<b>Amount:</b> 1 kom
<b>Name:</b> Pizza Cut
<b>Amount:</b> 2 kom

Slika 4.2.b Porudžbina koja čeka dostavljača

## Dostavljač

Korisnik koji prilikom registracije odabere ulogu dostavljača i koji prođe proces validacije kod administratora sistema, postaje osoba koja pruža dostavljačke usluge sistema za dostavu.

Aplikacija za dostavu mu olakšava posao na način da grupiše sve trenutno dostupne porudžbine na jednom mestu i pruža mu mogućnost da se sam odluči za jednu, u zavisnosti od njegove trenutne lokacije. Slobodne porudžbine su predstavljene karticama (Slika 5.1.) koje drže osnovne informacije o svakoj od njih. U gornjem desnom uglu, pritiskom na dugme “Take”, dostavljač može da preuzme konkretnu porudžbinu i da postane odgovoran za njenu isporuku. Dostavljač može da isporučuje maksimalno jednu porudžbinu u jednom momentu. Nakon preuzimanja, dostavljač biva preusmeren na stranicu porudžbine u toku, gde ima isti prikaz kao i potrošač čiju porudžbinu dostavlja (Slika 4.1.a).

Pored preuzimanja slobodnih dostava, dostavljač može da pregleda istoriju svih lično dostavljenih porudžbina prateći link “MY ORDERS”.

<b>Take</b>
<b>ID:</b> 12
<b>Address:</b> Ive Andrića 20
<b>Price:</b> 1220 din
<b>Note:</b> Brzo!
<b>Products:</b>
<b>Name:</b> BIG Burger
<b>Amount:</b> 1 kom
<b>Name:</b> Pizza Cut
<b>Amount:</b> 2 kom

Slika 5.1. Kartica porudžbine slobodne za preuzimanje

## Detalji o rešenjima određenih izazova

Aplikacija za dostavu je dizajnirana kao mikroservisni sistem 2 mikroservisa. Ispred mikroservisnog sistema postavljen je API gateway, tako da se prednja strana obraća isključivo njemu, a on preusmerava sve zahteve ka odgovarajućem individualnom mikroservisu. Mikroservisi ne dele istu bazu podataka. U nastavku će biti prikazani primeri samog koda pisanog za rešavanja određenih problema.

Slika 6.1. prikazuje implementaciju metode za preuzimanje slobodne porudžbine od strane dostavljača sistema. Sama metoda se nalazi u interfejsu `IDelivererOrderService`, mikroservisa `UserService`, čija je implementacija smeštena u klasu `DelivererOrderService`.

```
public long ConfirmOrder(long delivererId, long orderId)
{
    var deliverer = _dbContext.Users.Find(delivererId);
    var order = _dbContext.Orders.Find(orderId);

    if (deliverer == null || order == null)
    {
        return -1;
    }

    if (deliverer.DelivererOrders == null)
    {
        deliverer.DelivererOrders = new List<Models.Order>();
    }

    Random r = new();
    int deliveryTime = r.Next(3, 10);

    order.DeliveryTime = DateTime.Now.AddMinutes(deliveryTime);
    order.Confirmed = true;

    deliverer.DelivererOrders.Add(order);

    _dbContext.SaveChanges();

    return order.Id;
}
```

Slika 6.1. Metoda za preuzimanje dostupne porudžbine

Metoda prima dva parametra, informacije o identifikatoru dostavljača i porudžbine. Identifikator dostavljača se čita iz tokena koji je stigao sa samim zahtevom, u kontroleru `OrdersController`, dok se identifikator porudžbine prosleđuje putem URL-a. Iz baze podataka se dovlače korisnik i porudžbina sa konkretnim identifikatorima, definiše se random vreme isporuke, a zatim se ažurira `Confirmed` svojstvo porudžbine na vrednost `true`. Samom dostavljaču se u listu dostavljenih porudžbina dodaje konkretna porudžbina i nove vrednosti se čuvaju u bazi podataka. Kao povratna vrednost se vraća identifikator porudžbine.

Token ugrađen u svaki zahtev koji stiže sa prednje strane u sebi drži i informaciju o ulozi korisnika. U zavisnosti od toga da li korisnik bio potrošač ili dostavljač, pozivaju se metode dva različita servisa. Slika 6.2. prikazuje metodu `MyOrders`, kontrolera `OrdersController`, u kojoj je predstavljeno na koji način aplikacija odlučuje da li će da pozove metodu `GetMyOrders` servisa dostavljača ili servisa potrošača.

```
// GET api/Orders/MyOrders
[HttpGet]
[Route("MyOrders")]
//[Authorize(Roles="customer,deliverer")]
0 references
public IActionResult MyOrders()
{
    long userId = -1;
    string userRole = "";

    var identity = HttpContext.User.Identity as ClaimsIdentity;
    if (identity != null)
    {
        userId = long.Parse(identity.FindFirst("id").Value);
        userRole = identity.FindFirst("userrole").Value;
    }
    else
    {
        return BadRequest();
    }

    if (userRole.Equals("deliverer"))
    {
        var delivererOrders = _delivererOrderService.GetMyOrders(userId);
        return Ok(delivererOrders);
    }
    else if (userRole.Equals("customer"))
    {
        var customerOrders = _customerOrderService.GetMyOrders(userId);
        return Ok(customerOrders);
    }
    else
    {
        return BadRequest();
    }
}
```

Slika 6.2. Metoda MyOrders kontrolera OrdersController

Prvi korak jeste izvlačenje vrednosti o identifikatoru korisnika i korisničkoj ulozi smeštenih unutar tokena. Zatim na osnovu uloge, koja se u bazi podataka čuva kao tekstualni podatak sa mogućim vrednostima “deliverer”, “customer” i “administrator”, aplikacija odlučuje metodu čijeg servisa će da pozove i da povratnu vrednost smesti u promenljivu. Na kraju, podaci dobijeni iz odgovarajućeg servisa se prosleđuju nazad prednjoj strani uz status 200, koji obaveštava da je sve prošlo po planu i da su podaci uspešno preuzeti sa sloja podataka i prosleđeni kroz odgovor.

Slika 6.3. prikazuje deo konfiguracije Ocelota unutar ocelot.json datoteke, gde vidimo definisanje dve rute, sa svim njenim osobinama. UpstreamPathTemplate jeste putanja koju poziva klijentska aplikacija, dok DownstreamPathTemplate predstavlja tačnu putanju do metode kontrolera na koju se zahtev odnosi. DownstreamHostAndPorts definiše host i port mikroservisa koji zahtev “gađa”. UpstreamHttpMethod navodi koje HTTP metode su dozvoljene u okviru ove rute. Ukoliko se navedu prazne zagrade, podrazumevaće se da su sve HTTP metode dozvoljene.

AuthenticationOptions pruža mogućnost autentifikacije korisnika koji pokušava da pristupi određenoj ruti. Kao AuthenticationProviderKey se navodi tekstualni ključ koji služi kao referenca na konkretno podešavanje autentifikacije konfigurisane u metodi ConfigureServices, klase Startup.cs.

RouteClaimsRequirement osobina rute omogućava da navedemo sve ključ-vrednost parove koji treba da se nalaze među klejmovima tokena, da bi trenutni korisnik imao pravo da pristupi ruti.

```
// Pregled svih product-a (LOGGED)
{
  "DownstreamPathTemplate": "/api/Products",
  "DownstreamScheme": "https",
  "DownstreamHostAndPorts": [
    {
      "Host": "localhost",
      "Port": 44349
    }
  ],
  "UpstreamPathTemplate": "/Products",
  "UpstreamHttpMethod": [ "GET" ],
  "AuthenticationOptions": {
    "AuthenticationProviderKey": "AuthKey",
    "AllowedScopes": []
  },
  "RouteClaimsRequirement": {
    "logged": "true"
  }
},
// Dodavanje novog product-a (ADMIN)
{
  "DownstreamPathTemplate": "/api/Products",
  "DownstreamScheme": "https",
  "DownstreamHostAndPorts": [
    {
      "Host": "localhost",
      "Port": 44349
    }
  ],
  "UpstreamPathTemplate": "/Products",
  "UpstreamHttpMethod": [ "POST" ],
  "AuthenticationOptions": {
    "AuthenticationProviderKey": "AuthKey",
    "AllowedScopes": []
  },
  "RouteClaimsRequirement": {
    "userrole": "administrator"
  }
},
}
```

Slika 6.3. Konfiguracija rutiranja u ocelot.json fajlu

Kod kojim se implementira kreiranje nove porudžbine potrošača nalazi se na Slici 6.4. Za početno vreme dostave se postavlja trenutno vreme uvećano za 10 godina, što predstavlja nerealno dug period čekanja na dostavljača. Rok se smanjuje na vreme od 3 do 10 minuta kada dostavljač preuzme konkretnu porudžbinu.

```
2 references
public OrderInfoDto CreateOrder(long userId, OrderCreateDto order)
{
    var user = _dbContext.Users.Include("CustomerOrders.OrderedProducts").FirstOrDefault(u => u.Id == userId);

    if (user == null)
    {
        return null;
    }

    if (user.CustomerOrders == null)
    {
        user.CustomerOrders = new List<Order>();
    }

    Order newOrder = _mapper.Map<Order>(order);

    newOrder.DeliveryTime = DateTime.Now.AddYears(10);
    newOrder.Confirmed = false;
    newOrder.Delivered = false;

    user.CustomerOrders.Add(newOrder);
    _dbContext.SaveChanges();

    OrderInfoDto retVal = _mapper.Map<OrderInfoDto>(user.CustomerOrders.Last());
    return retVal;
}
```

Slika 6.4. Kreiranje nove porudžbine

U metodu CreateOrder se prosleđuju id korisnika, kao i DataTransfer objekat posebno kreirane klase OrderCreateDto koja je preslikavanje originalne Order klase, ali koja sadrži samo polja koja su neophodna za rad metoda servisa. Korišćenjem DT objekata, smanjuje se veličina saobraćaja kroz mrežu, ubrzava se proces slanja i primanja zahteva, odnosno odgovora sa servera (mikroservisa).

Kroz celu aplikaciju, podaci se među slojevima razmenjuju isključivo u obliku odgovarajućih DT objekata. Mapiranje originalnih objekata u DT objekte je olakšano korišćenjem AutoMapper[6] nuget paketa.



Na prednjoj strani se koriste modeli koji po nazivima polja (case sensitive) odgovaraju DataTransfer objektima koji stižu sa servera. Slika 6.5. prikazuje primere pozivanja mikroservisnih kontroler metoda. Konkretno, GET metode za dobijanje trenutno slobodnih porudžbina i za preuzimanje jedne od slobodnih porudžbina, kao i POST metoda kojom potrošač kreira novu porudžbinu.

```
availableOrders(): Observable<Order[]> {  
  return this.http.get<Order[]>(environment.serverURL + '/Orders/Available');  
}  
  
takeOrder(orderId: number): Observable<number> {  
  return this.http.get<number>(  
    environment.serverURL + `/Orders/TakeOrder/${orderId}`  
  );  
}  
  
createOrder(order: Order): Observable<Object> {  
  return this.http.post<Object>(  
    environment.serverURL + '/Orders/MyOrders',  
    order  
  );  
}
```

Slika 6.5. Pozivanje serverskih metoda iz klijentske aplikacije

Osnovna adresa servera serverURL je smeštena u environment.ts fajlu i na nju se nadovezuju potrebne adresne ekstenzije u zavisnosti od toga kom mikroservisu i kontroleru mikroservisa pristupamo, kao i metodi samog kontrolera. Nakon naziva HTTP metode, u izlomljene zagrade je smešten tip podatka koji se očekuje kao odgovor sa zadnje strane.

Na slici 6.6. je prikazana upotreba createOrder metode, order servisa, koja se nalazi unutar orders-create-order komponente. Injekcijom zavisnosti kroz konstruktor komponente, prosleđuje se objekat order servisa. Poziva se metoda, na čiju se povratnu vrednost pretplaćujemo i u zavisnosti od uspešnosti odgovora, aplikacija reaguje na dva moguća načina. Ukoliko je odgovor uspešno isporučen, u data objekat se smešta povratna vrednost metode servisa zadnje strane. U ovom slučaju primljeni podaci se ispisuju na konzoli, a potom se ispisuje obaveštenje korisniku da je kreiranje porudžbine uspešno završeno. Ako je došlo do bilo kakve greške prilikom odgovora, korisnik će dobiti informaciju o poruci same greške.

```
this.orderService.createOrder(this.order).subscribe(  
  (data) => {  
    console.log(data);  
    this.toastr.success('Porudzbina uspeła!');  
    window.location.reload();  
  },  
  (error: HttpResponse) => {  
    this.toastr.error(error.message);  
  }  
);
```

Slika 6.6. Poziv metode createOrder

## PREDLOZI ZA DALJA USAVRŠAVANJA

Trenutna verzija aplikacije za dostavu čini jednu stabilnu i pouzdanu osnovu i ima veliki potencijal za usavršavanje i uvođenje novih funkcionalnosti sa ciljem što boljeg i sigurnijeg iskustva svake od grupa korisnika koji je koriste.

Najveći potencijal za dalja usavršavanja postoji u funkcionalnostima aplikacije koje su namenjene dostavljačima.

Trenutan način raspodele porudžbina po dostavljačima je neefikasan i prepušten samim dostavljačima koji ne mogu u datom momentu da znaju da li je drugi kolega dostavljač možda u boljoj poziciji za konkretnu porudžbinu koju planira da preuzme na sebe. Sam sistem bi trebao da koristi algoritam koji uzima u obzir lokacije svih trenutno aktivnih dostavljača i da optimizuje čitav tok rada dostavljača kako bi njihovi pređeni putevi bili što manji i samim tim, ušteda vremena veća.

Pored toga, korisna stvar za zadovoljstvo potrošača bi bilo uvođenje prioriteta samih porudžbina, na način da bi porudžbine koje već duže vreme čekaju na prihvatanje, dobile status većeg prioriteta, plus neki tip bonusa za dostavljača koji je prihvati. Na taj način bi se stimulisalo prihvatanje porudžbina kreiranih na manje atraktivnim lokacijama i smanjio bi se broj nezadovoljnih potrošača.

Povećanjem broja raspoloživih proizvoda, stranica za kreiranje nove porudžbine postaje nepregledna. U budućnosti bi svakako trebalo obratiti pažnju na bolji način organizovanja proizvoda u vidu tabele ili slično, uz mogućnost sortiranja, filtriranja itd.

Pozitivno poboljšanje bi bila i mogućnost dostavljača da prihvatanjem neke porudžbine, sâm da procenu vremena potrebnog za izvršenje same dostave.

Takođe, za dodatno unapređenje potrošačkog iskustva, trebalo bi mu omogućiti da u svakom momentu ima uvid u trenutnu lokaciju dostavljača njegove porudžbine.

Jedno od naprednijih usavršavanja bilo bi svakako i uvođenje mogućnosti onlajn plaćanja poručenih porudžbina.

U sistem bi mogao da se uvede i proces ocene dostavljača prilikom svake dostave, koja bi se odnosila na zadovoljstvo uslugom, brzinu dostave, ljubaznost itd. Bonusi za bolje ocenjene dostavljače bi bila dodatna motivacija za profesionalnost samih dostavljača.

Sa druge strane, za poboljšanje dostavljačkog iskustva, potrebno je implementirati opciju podrške za dostavljače u vidu četa sa administratorima, kao i mogućnost mobilnog poziva ka samom potrošaču koji čeka dostavu, iz same aplikacije (u slučaju jednostavnijeg problema, npr. ne radi interfon).

Trenutno, administrator nema na osnovu čega da donese odluku da li će da aktivira ili prihvati novoregistrovanog dostavljača. Ovaj problem bi u većoj meri bio rešen ukoliko bi dostavljač prilikom registracije imao opciju da pored svih ličnih informacija, ima i dodatno polje u kojem će da ukratko opiše svoju biografiju, kao i dugme za upload CV-ja.

Još jedan pravac usavršavanja aplikacije za dostavu jeste uvođenje više jezika na kojima aplikacija može da radi, kao i mogućnost korisnika da odabere njemu odgovarajuć jezik. U trenutnom stanju, svi linkovi, dugmad i informacije pisani su na engleskom jeziku.



## LITERATURA

- [1] [https://sr.m.wikipedia.org/sr-el/Microsoft\\_Visual\\_Studio](https://sr.m.wikipedia.org/sr-el/Microsoft_Visual_Studio)
- [2] <https://ocelot.readthedocs.io/en/latest/index.html>
- [3] <https://angular.io/docs>
- [4] <https://learn.microsoft.com/en-us/aspnet/core/web-api>
- [5] <https://learn.microsoft.com/en-us/ef/core/>
- [6] <https://docs.automapper.org/en/stable/Getting-started.html>
- [7] <https://www.nuget.org/packages/BCrypt.Net-Next/>
- [8] <https://www.nuget.org/packages/NETCore.MailKit/2.1.0>