A picture containing icon

Description automatically generated

Web aplikacija za narudžbe i dostavu hrane

Uživaoci Alhamijado književnosti

* [Kovačević Hamza](https://github.com/hkovacevic2)
* [Kukuruzović Haris](https://github.com/hkukuruzov1)
* [Mahmić Adnan](https://github.com/amahmic3)

Contents

[Opis projekta 3](#_Toc74254110)

[Scenariji upotrebe 4](#_Toc74254111)

[Dijagram slučajeva upotrebe 14](#_Toc74254112)

[Dijagrami aktivnosti 15](#_Toc74254113)

[MVC dijagram 20](#_Toc74254114)

[ER dijagram 21](#_Toc74254115)

[Prototipi 22](#_Toc74254116)

[SOLID Principi 32](#_Toc74254117)

[Design patterni 33](#_Toc74254118)

[Dijagrami sekvenci 38](#_Toc74254119)

[Dijagrami komponenti, paketa i raspoređivanja 41](#_Toc74254120)

# Opis projekta

*Trofí* je platforma za dostavu hrane. Kupci mogu iz udobnosti svog doma, posla ili nekog drugog mjesta preko web aplikacije naručiti bilo koji proizvod iz ponude. Dostava stiže na Vašu adresu brzo i jednostavno. Prati uživo svoju dostavu u realnom vremenu. Prednost ove aplikacije je ta što je napravljena na način da ćete biti u toku sa statusom narudžbe u stvarnom vremenu tj. od momenta narudžbe do dolaska na željenu adresu. Mogućnost plaćanja online i prilikom dostave.

# Scenariji upotrebe

1. Registracija korisnika kao naručioca

|  |  |
| --- | --- |
| Naziv scenarija | Registracija naručioca |
| Opis | Registracija osobe koja želi da koristi aplikaciju kao naručilac |
| Posljedice – uspješan završetak | Potvrda o uspješnoj registraciji accounta na sistem |
| Posljedice – neuspješan završetak | Neuspješna validacija korisničkih podataka, samim tim i registracija accounta na sistem |
| Primarni akteri | Neregistrovani korisnik |
| Ostali akteri | Sistem |
| Glavni tok | Neregistrovani korisnik popunjava formu s ličnim podacima potrebnim za kreiranje novog korisničkog računa, forma se dalje validira, te u slučaju uspješne validacije kreira se novi korisnički račun. |
| Alternativni tok | Neuspješna validacija korisničkih podataka |

Glavni tok događaja

|  |  |
| --- | --- |
| Neregistrovani korisnik | Sistem |
| 1. Pristup formi za registrovanje |  |
|  | 1. Prikaz forme |
| 1. Popunjavanje forme |  |
| 1. Submit forme |  |
|  | 1. Uspješna validacija korisničkih podataka |
|  | 1. Slanje email potvrde o uspješnoj registraciji |

Alternativni tok događaja

|  |  |
| --- | --- |
| Neregistrovani korisnik | Sistem |
|  | 1. Neuspješna validacija korisničkih podataka |
|  | 1. Upozorenje korisniku |
| 1. Povratak na korak 3. glavnog toka događaja |  |

|  |  |
| --- | --- |
| Naziv scenarija | Registracija dostavljača |
| Opis | Registracija osobe koja želi da koristi aplikaciju kao dostavljač |
| Posljedice – uspješan završetak | Potvrda o uspješnoj registraciji accounta na sistem |
| Posljedice – neuspješan završetak | Neuspješna validacija korisničkih podataka, samim tim i registracija accounta na sistem |
| Primarni akteri | Neregistrovani korisnik |
| Ostali akteri | Sistem i administrator |
| Glavni tok | Neregistrovani korisnik popunjava formu s ličnim podacima potrebnim za kreiranje novog korisničkog računa, forma se dalje validira, te u slučaju uspješne validacije kreira se novi korisnički račun. |
| Alternativni tok | Neuspješna validacija korisničkih podataka, odbijanje zahtjeva od strane administratora |

1. Registracija dostavljača  
     
     
   Glavni tok događaja

|  |  |  |
| --- | --- | --- |
| Neregistrovani korisnik | Sistem | Administrator |
| 1. Pristup interfejsu za registrovanje dostavljača |  |  |
|  | 1. Prikaz forme |  |
| 1. Popunjavanje forme |  |  |
| 1. Submit forme |  |  |
|  | 1. Uspješna validacija korisničkih podataka |  |
|  |  | 1. Prihvata zahtjev za kreiranje novog accounta dostavljača |
|  | 7. Slanje email potvrde o uspješnoj registraciji |  |

Alternativni tok događaja

|  |  |
| --- | --- |
| Neregistrovani korisnik | Sistem |
|  | 1. Neuspješna validacija korisničkih podataka |
|  | 1. Upozorenje korisniku |
| 1. Povratak na korak 3. glavnog toka događaja |  |

Alternativni tok događaja

|  |  |  |
| --- | --- | --- |
| Neregistrovani korisnik | Sistem | Administrator |
|  | Uspješna validacija korisničkih |  |
|  |  | Odbijanje zahtjeva za registraciju zbog nepoštivanja pravilnika aplikacije |
|  | Slanje obrazloženja e-mail porukom |  |

3. Registracija restorana

|  |  |
| --- | --- |
| Naziv scenarija | Registracija restorana |
| Opis | Registracija osobe koja želi da doda svoj restoran u sistem |
| Posljedice – uspješan završetak | Potvrda o uspješnoj registraciji restorana na sistem |
| Posljedice – neuspješan završetak | Neuspješna validacija podataka potrebnih za registrovanje restorana u sistem |
| Primarni akteri | Neregistrovani korisnik |
| Ostali akteri | Sistem i administrator |
| Glavni tok | Neregistrovani korisnik popunjava formu s podacima potrebnim za kreiranje novog korisničkog računa, forma se dalje validira, te u slučaju uspješne validacije restoran se dodaje u sistem. |
| Alternativni tok | Neuspješna validacija podataka, odbijanje zahtjeva od strane administratora |

Glavni tok događaja

|  |  |  |
| --- | --- | --- |
| Neregistrovani korisnik | Sistem | Administrator |
| 1. Pristup interfejsu za registrovanje restorana |  |  |
|  | 1. Prikaz forme |  |
| 1. Popunjavanje forme |  |  |
| 1. Submit forme |  |  |
|  | 1. Uspješna validacija korisničkih podataka |  |
|  |  | 1. Prihvata zahtjev za kreiranje novog restorana |
|  | 7. Slanje email potvrde o uspješnoj registraciji |  |

Alternativni tok događaja

|  |  |
| --- | --- |
| Neregistrovani korisnik | Sistem |
|  | 1. Neuspješna validacija korisničkih podataka |
|  | 1. Upozorenje korisniku |
| 1. Povratak na korak 3. glavnog toka događaja |  |

Alternativni tok događaja

|  |  |  |
| --- | --- | --- |
| Neregistrovani korisnik | Sistem | Administrator |
|  | 1. Uspješna validacija korisničkih podataka |  |
|  |  | 1. Odbijanje zahtjeva za registraciju zbog nepoštivanja pravilnika aplikacije |
|  | 1. Slanje obrazloženja e-mail porukom |  |

1. Narudžba

|  |  |
| --- | --- |
| Naziv | Naručivanje jela |
| Opis slučaja upotrebe | Naručilac jela putem Web interfejsa bira restoran, pregleda jelovnik, ubacuje stavke u svoju online korpu, zaključuje narudžbu te plaća obračunati iznos. |
| Preduvjet | Da bi se narudžba zaključila korisnik treba biti registrovan i proći proces validacije adrese za dostavu. |
| Posljedice - uspješan završetak | Narudžba je zaključena i počinje proces dostave. |
| Posljedice - neuspješan završetak | Narudžba se odbija i korisnik se vraća na korak zaključivanja. |
| Primarni akteri | Naručilac |
| Ostali akteri | Sistem |
| Glavni tok | Korisnik bira restoran iz kojeg želi naručiti proizvode. Nakon što željene proizvode doda u korpu, kreće proces zaključivanja narudžbe. Ukoliko korisnik nije prijavljen na sistem biva preusmjeren na formu za prijavu, u suprotnom unosi adresu za dostavu i kreće proces validacije. Nakon validacije adrese korisnik bira način plaćanja. Narudžba se dodaje u red za dostavu. |
| Proširenja/Alternative | Neuspješna validacija adrese za dostavu, Plaćanje online |

Glavni tok događaja:

|  |  |
| --- | --- |
| Korisnik | Sistem |
| 1. Pristupanje interfejsu za naručivanje |  |
|  | 2. Prikaz trenutne ponude restorana |
| 3. Izbor restorana. |  |
|  | 4. Prikaz proizvoda odabranog restorana |
| 5. Dodavanje proizvoda u korpu |  |
| 6. Započinje proces zaključivanja narudžbe |  |
|  | 7. Ukoliko korisnik nije prijavljen prosljeđuje ga na formu za prijavu, u suprotnom prikazuje formu za zaključivanje narudžbe |
| 8. Unosi adresu za dostavu narudžbe |  |
|  | 9. Validacija adrese za dostavu |
| 10. Bira način plaćanja |  |
|  | 11. Narudžba se dodaje u red za dostavu |
|  | 12. Šalje naručiocu poruku da je narudžba uspješno zaključena i da on može pratiti njeno stanje |

Prošireni tok događaja: Plaćanje online

|  |  |  |
| --- | --- | --- |
| Korisnik | Sistem | PayPal |
| 1.Bira način plaćanja online |  |  |
|  | 2. Preusmjerava korisnika na PayPal formu za plaćanje |  |
|  |  | 3. Prikaz fome za plaćanje |
| 4. Popunjava formu za plaćanje |  |  |
|  |  | 5. Validacija uplate |
|  | 6. Ako je validacija uspješna nastavlja se tok događaja od koraka 11., u suprotnom vraća se na korak 10. u glavnom toku događaja |  |

Alternativni tok događaja: Neuspješna validacija adrese

|  |  |
| --- | --- |
| Korisnik | Sistem |
|  | 1. Narudžba ne prolazi validaciju adrese |
|  | 2. Upozoravanje korisnika u čemu je problem. |
| 3. Nastavak na koraku 8. glavnog toka događaja. |  |

1. Dostava

|  |  |
| --- | --- |
| Naziv | Dostava |
| Opis slučaja upotrebe | Dostavljač pristupa interfejsu narudžbi koje čekaju na isporuku, bira narudžbu koju želi isporučiti, te započinje proces narudžbe. |
| Posljedice - uspješan završetak | Narudžba uspješno dostavljena |
| Primarni akteri | Dostavljač |
| Ostali akteri | Sistem, restoran, naručilac |
| Glavni tok | Dostavljač s interfejsa sistema bira narudžbu za isporuku, sistem obavještava restoran i naručioca da je započet proces dostave. Dostavljač pokupi narudžbu u restoranu. Dostavljač isporučuje narudžbu naručiocu i time zaključuje proces dostave. |
| Proširenja/alternative | Restoran otkazuje narudžbu |

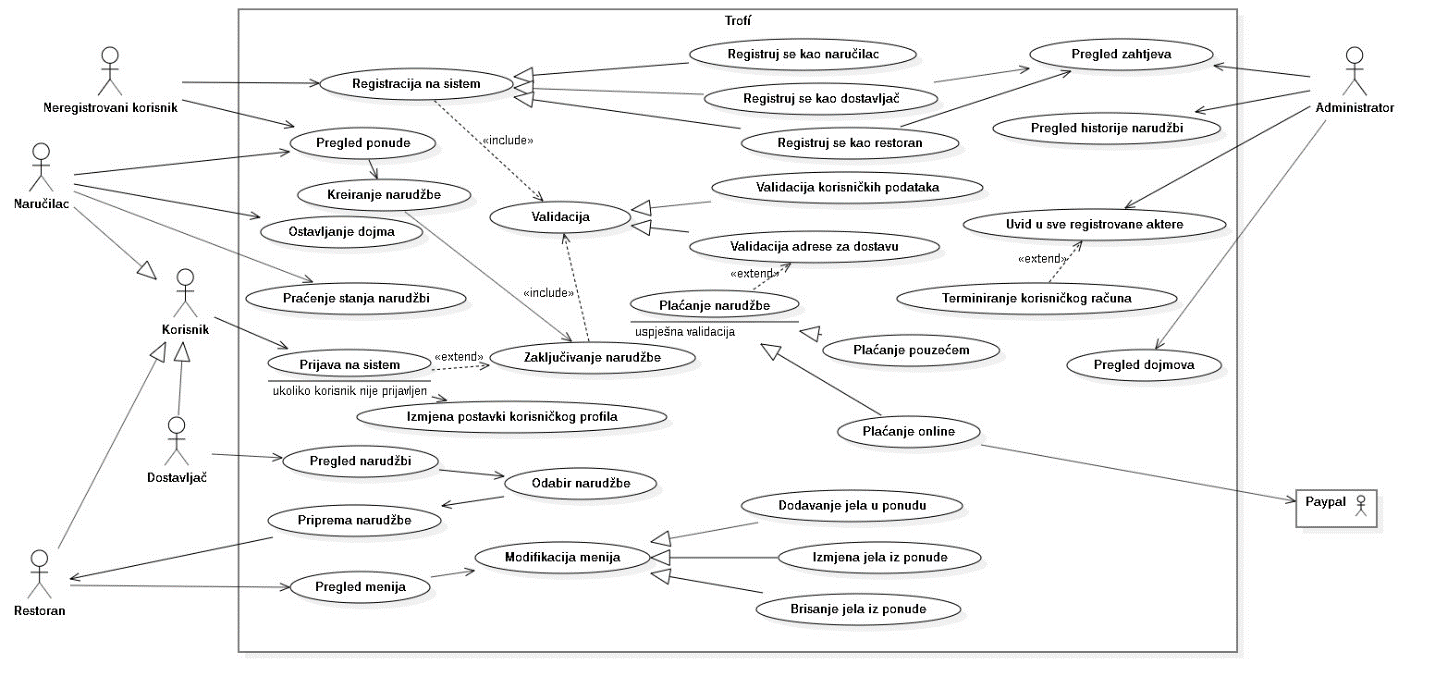
Glavni tok

|  |  |  |  |
| --- | --- | --- | --- |
| Dostavljač | Sistem | Restoran | Naručilac |
| 1.Pristup interfejsu za neisporučene narudžbe |  |  |  |
| 2. Odabir narudžbe za isporuku |  |  |  |
| 3. Započinje proces dostave |  |  |  |
|  | 4. Obavještava restoran i naručioca o stanju narudžbe |  |  |
| 5. Posjeta restoranu iz kojeg je naručena narudžba |  |  |  |
|  |  | 6. Predaja narudžbe dostavljaču |  |
|  | 7. Obavještava naručioca o stanju narudžbe |  |  |
| 8. Odlazak na adresu naručioca |  |  |  |
| 9. Predaja narudžbe naručiocu |  |  |  |
|  |  |  | 10. Primanje narudžbe |
|  | 11. Obavještava restoran o stanju narudžbe |  |  |

Alternativni tok: Restoran otkazuje narudžbu

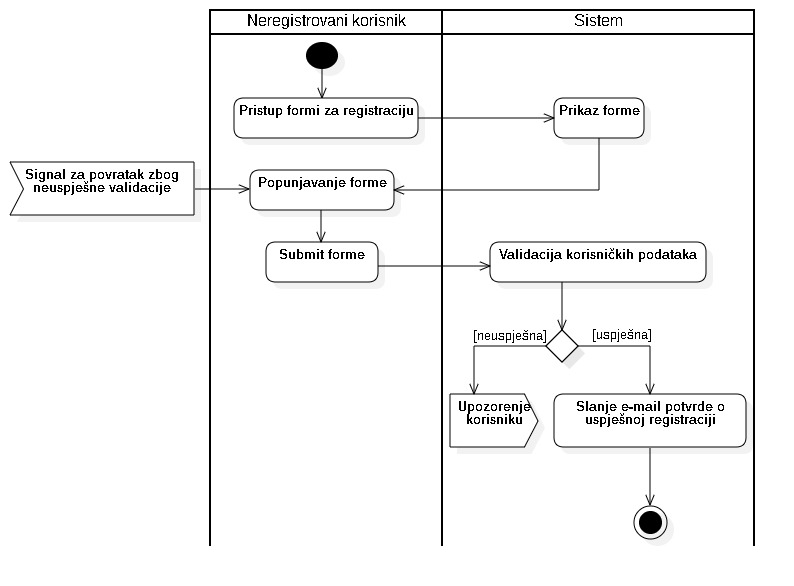
|  |  |  |
| --- | --- | --- |
| Dostavljač | Sistem | Restoran |
| 1. Započinje proces narudžbe |  |  |
|  | 2. Obavještava restoran i naručioca o stanju narudžbe |  |
|  |  | 3. Otkazuje narudžbu |
|  | 4. Obavještava dostavljača i naručioca o otkazivanju narudžbe |  |
| 5. Povratak na korak 2. glavnog toka scenarija |  |  |

# Dijagram slučajeva upotrebe

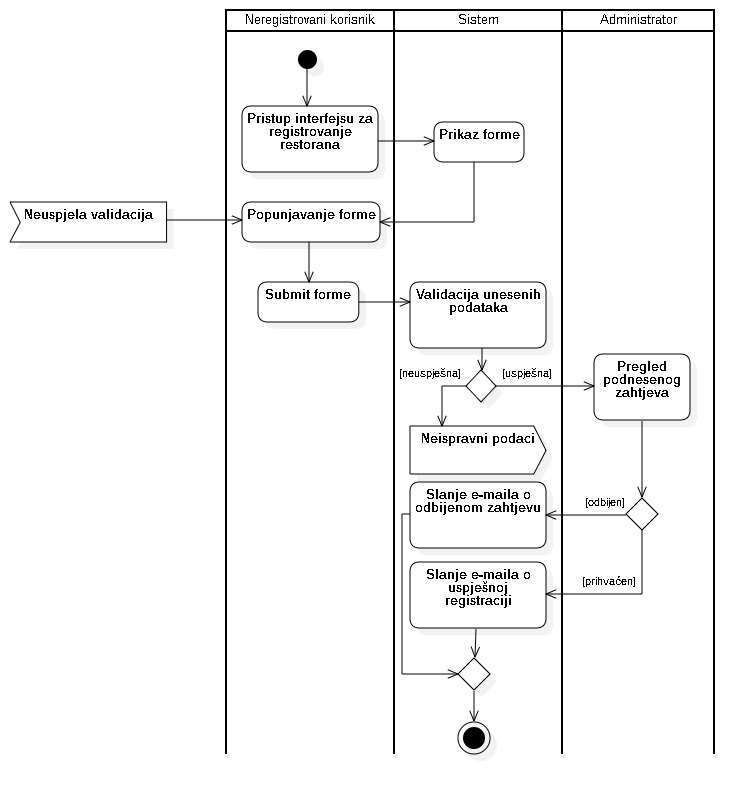


# Dijagrami aktivnosti

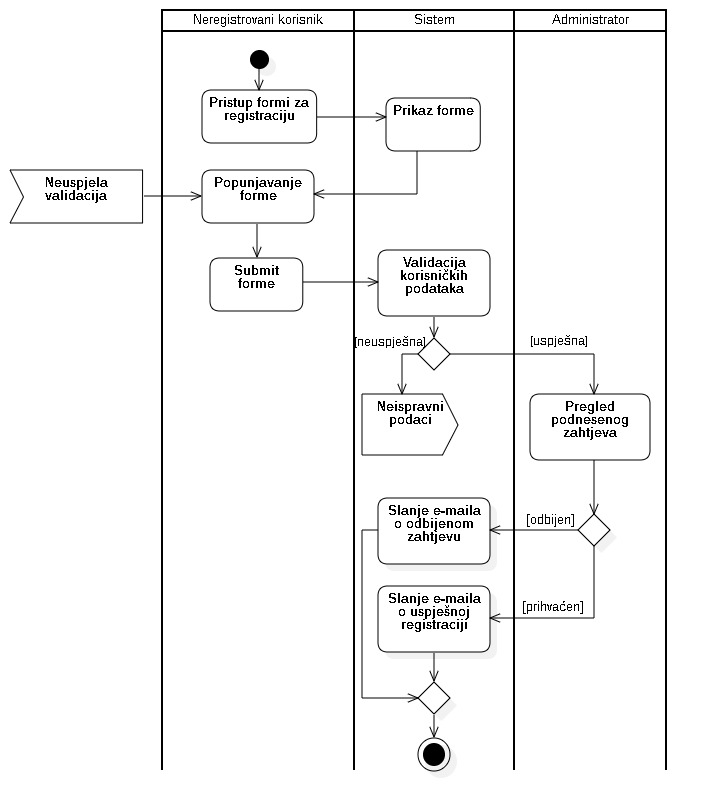
1. Registracija naručioca



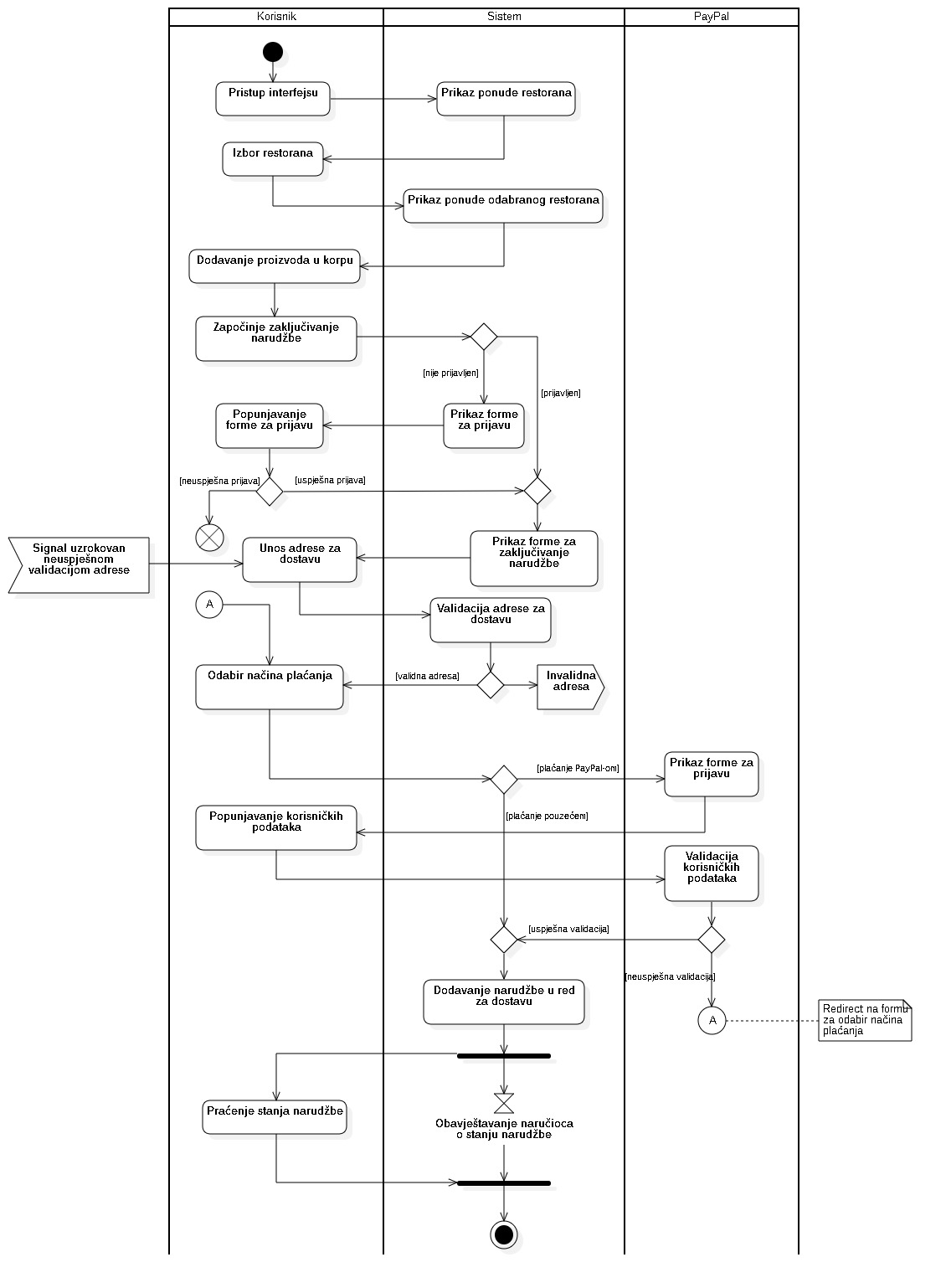
1. Registracija restorana



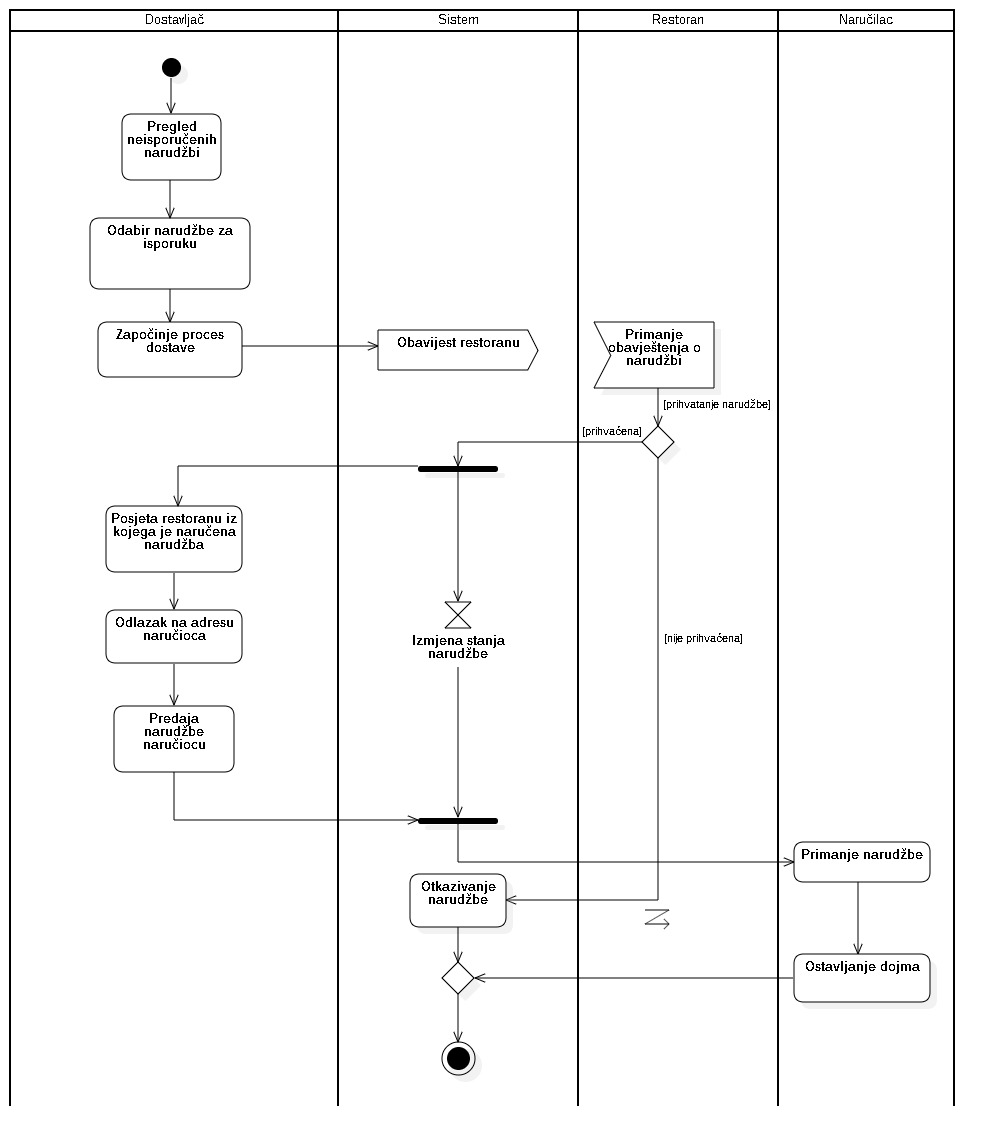
1. Registracija dostavljača



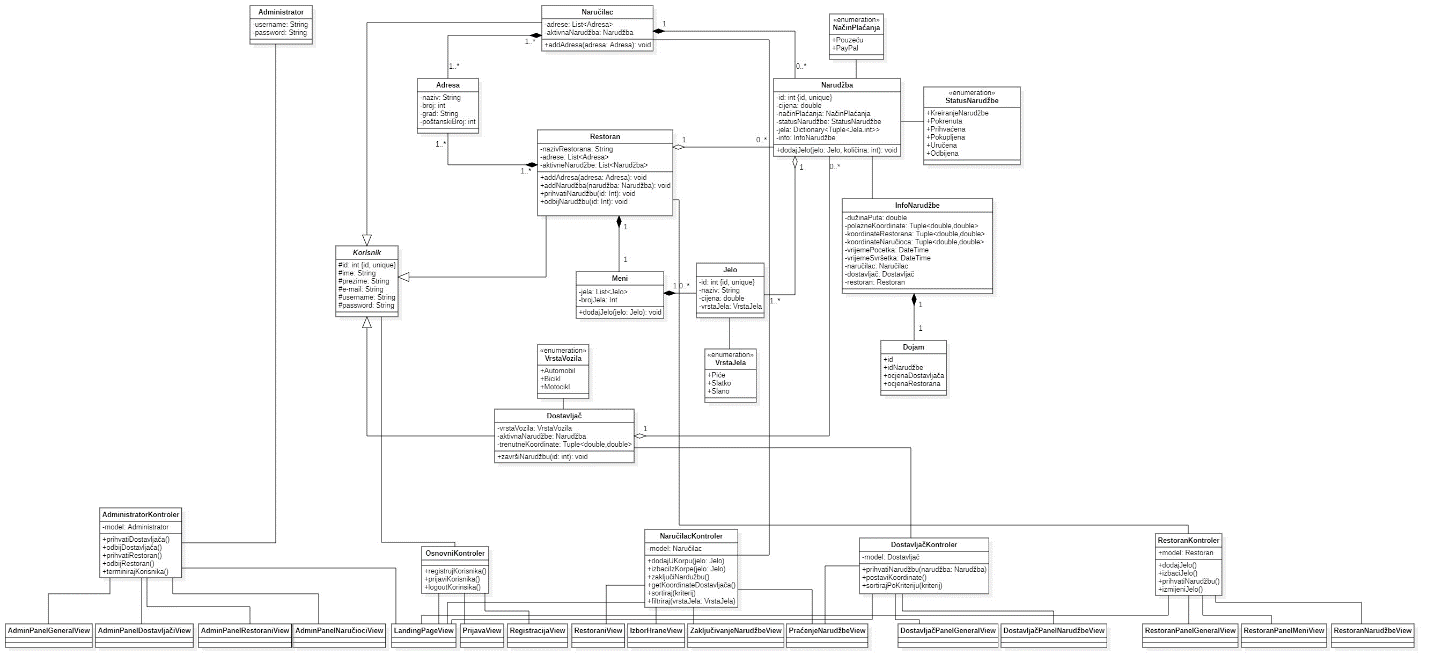
1. Naručivanje



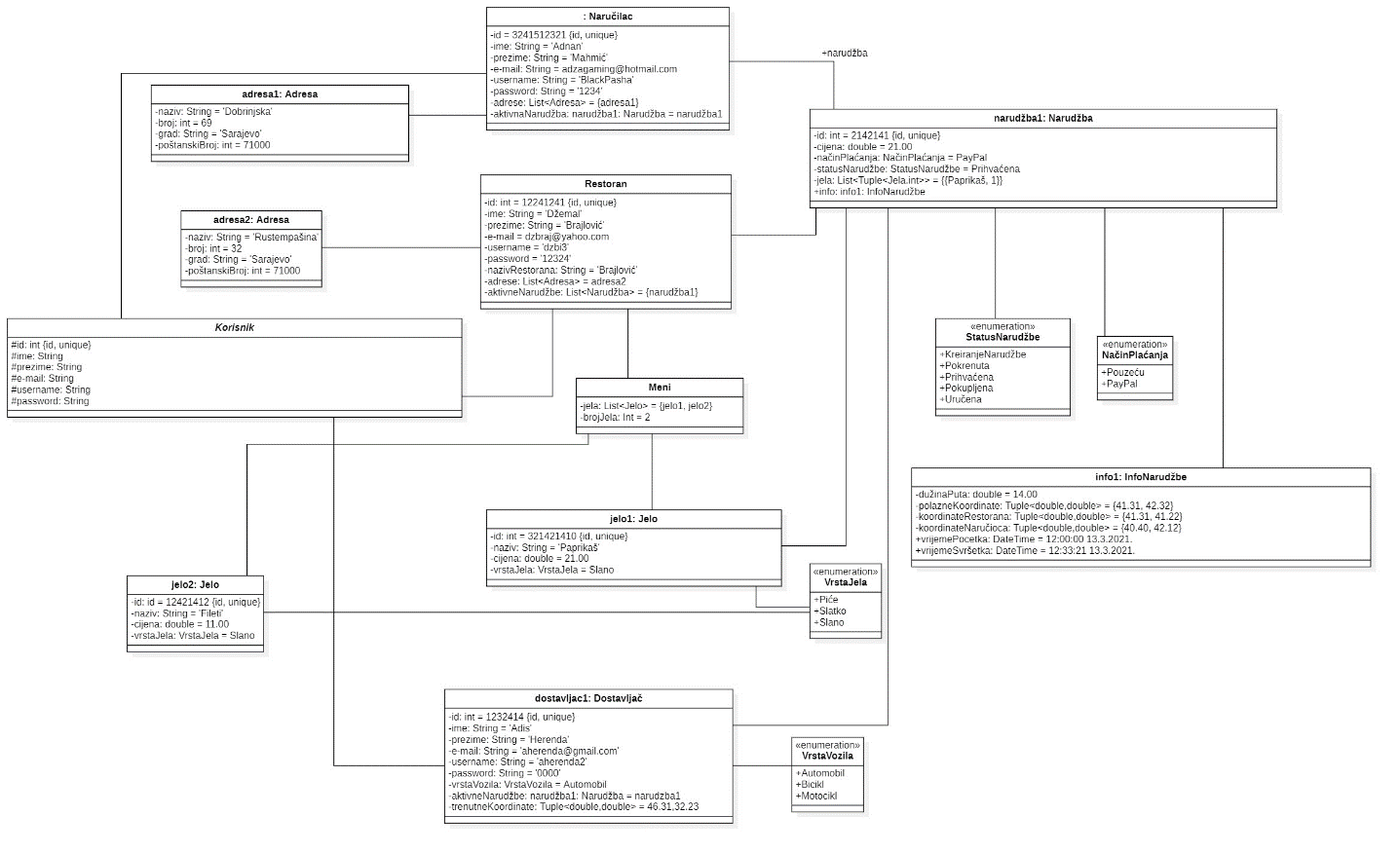
1. Dostava



# MVC dijagram



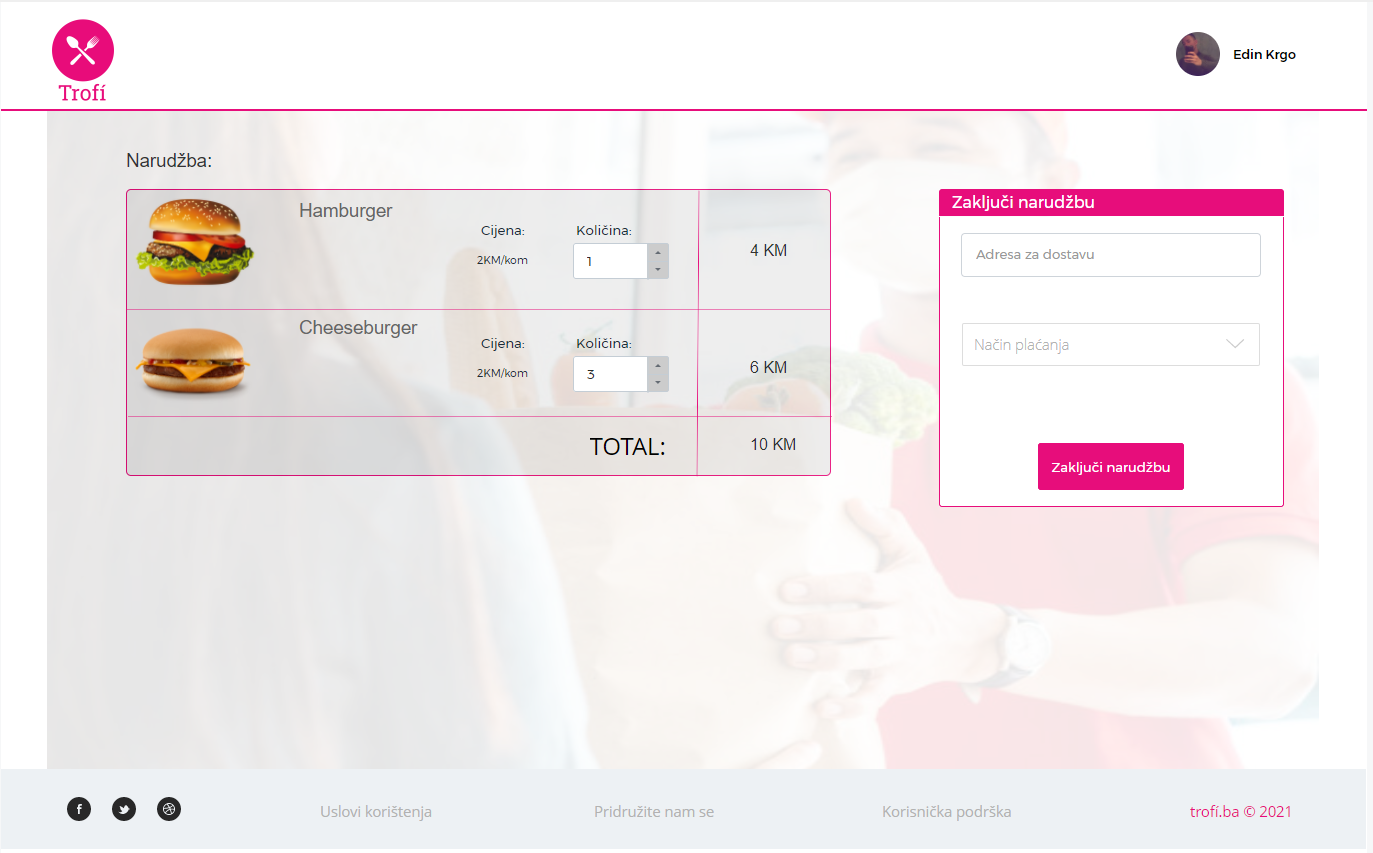
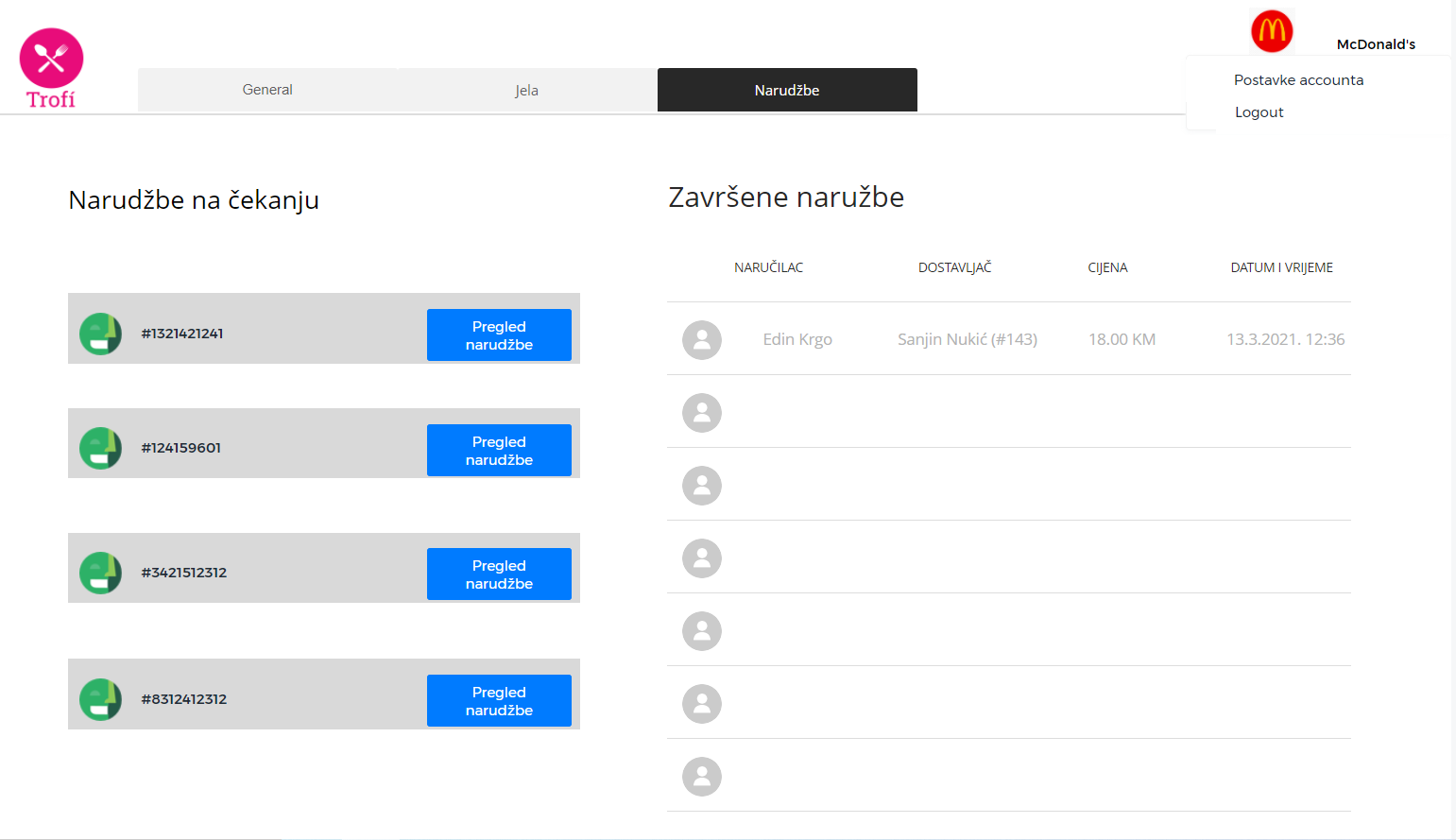
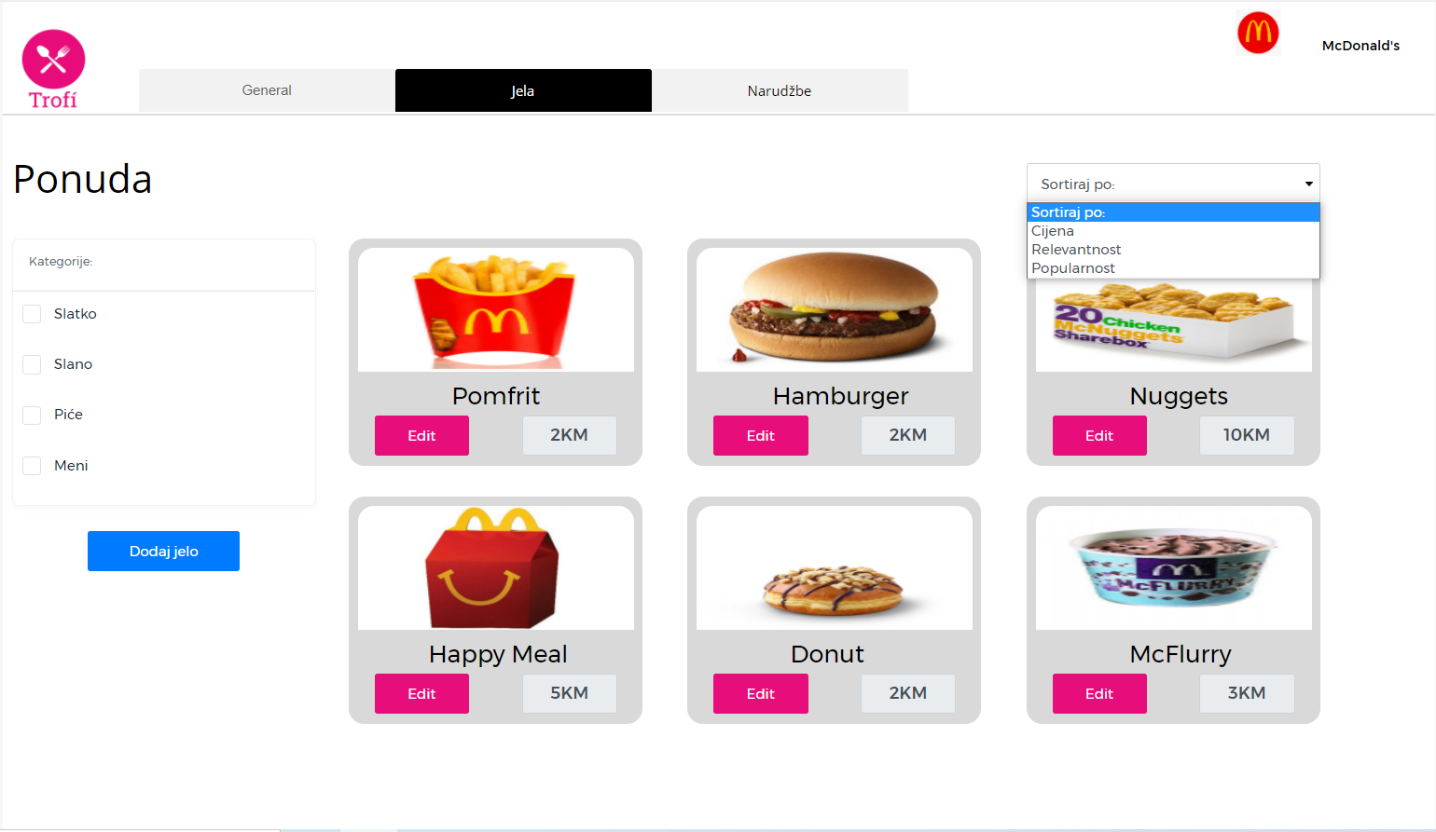
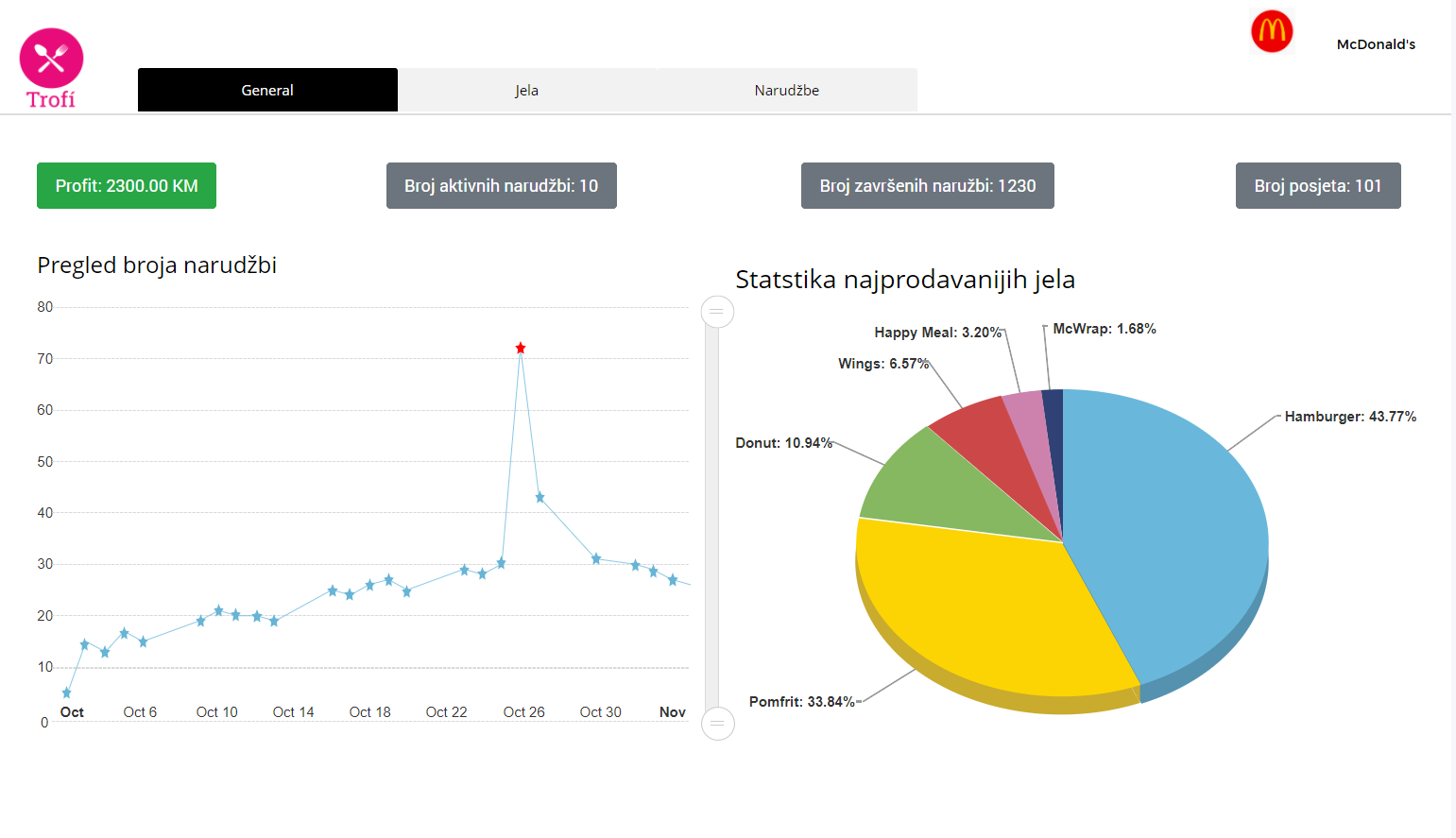
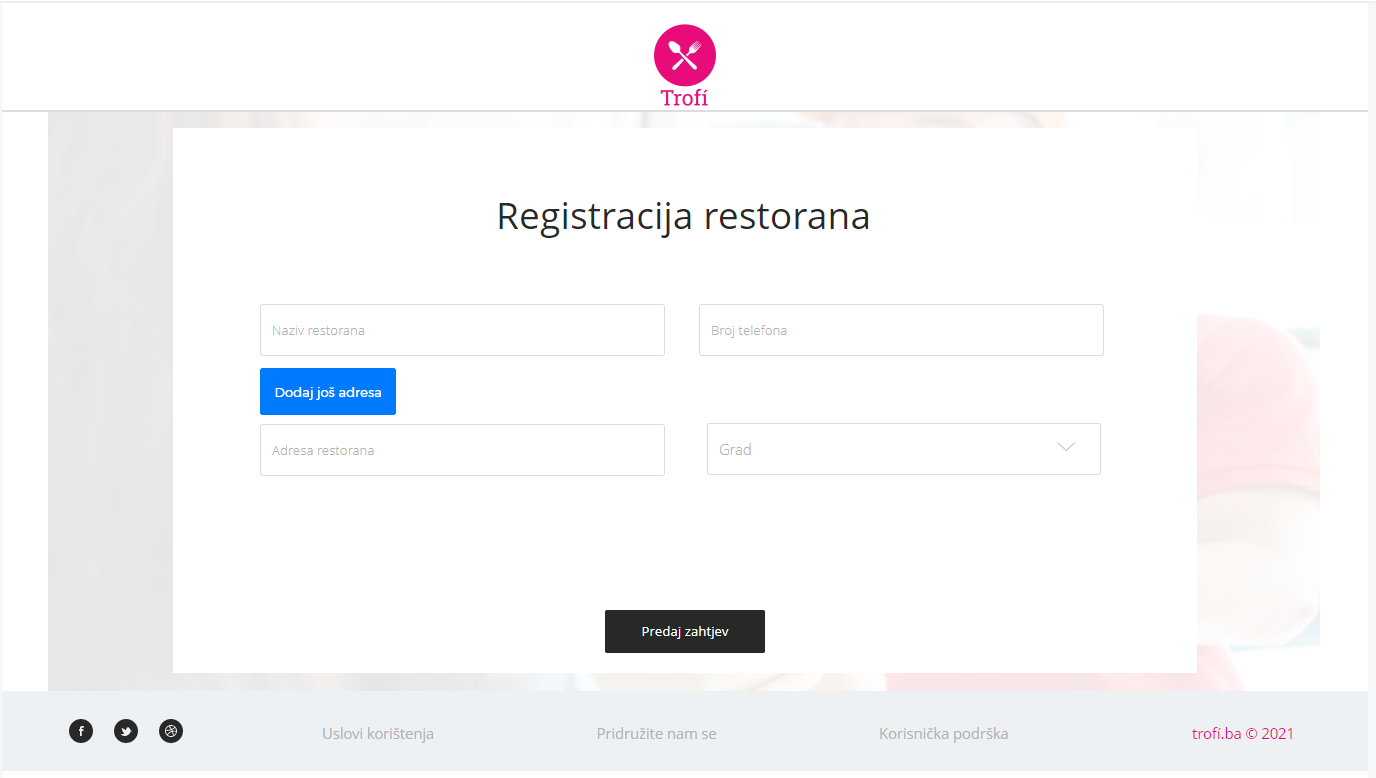
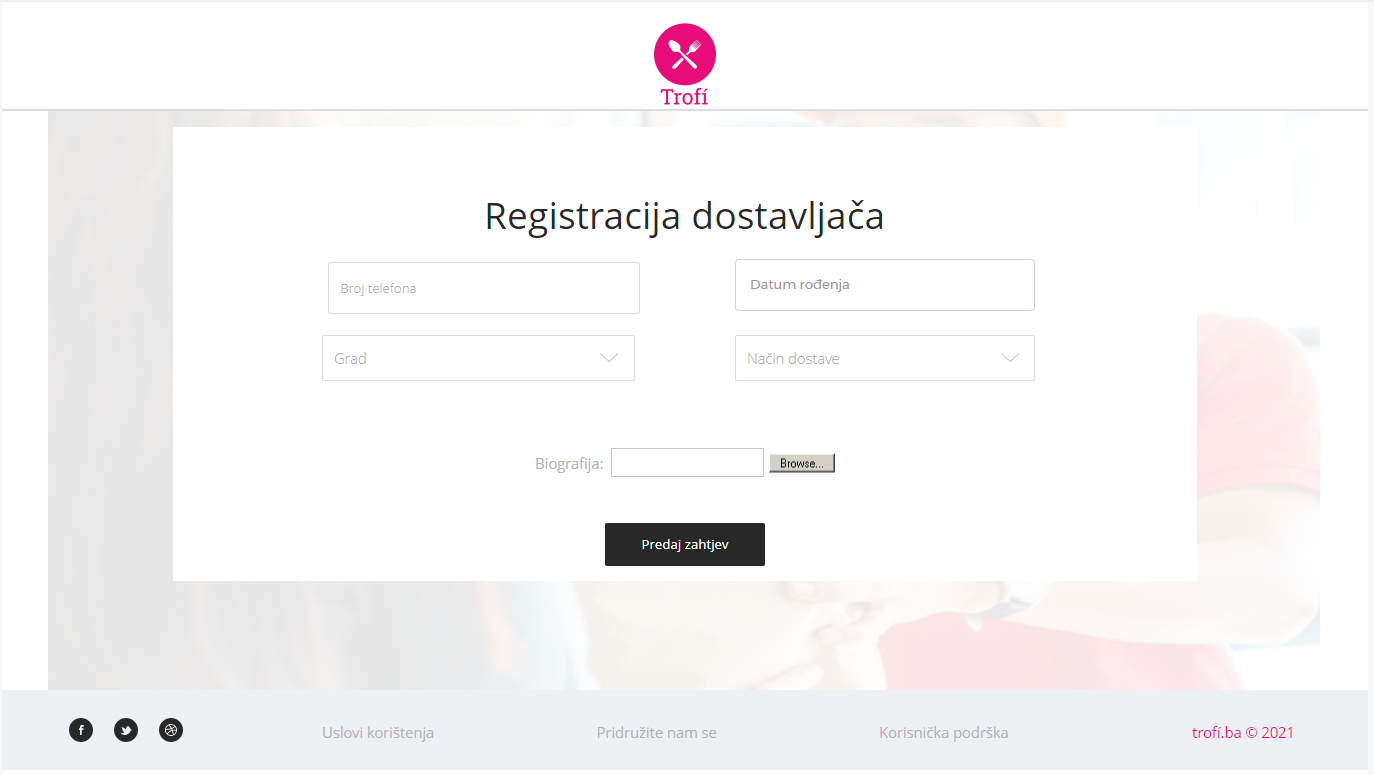
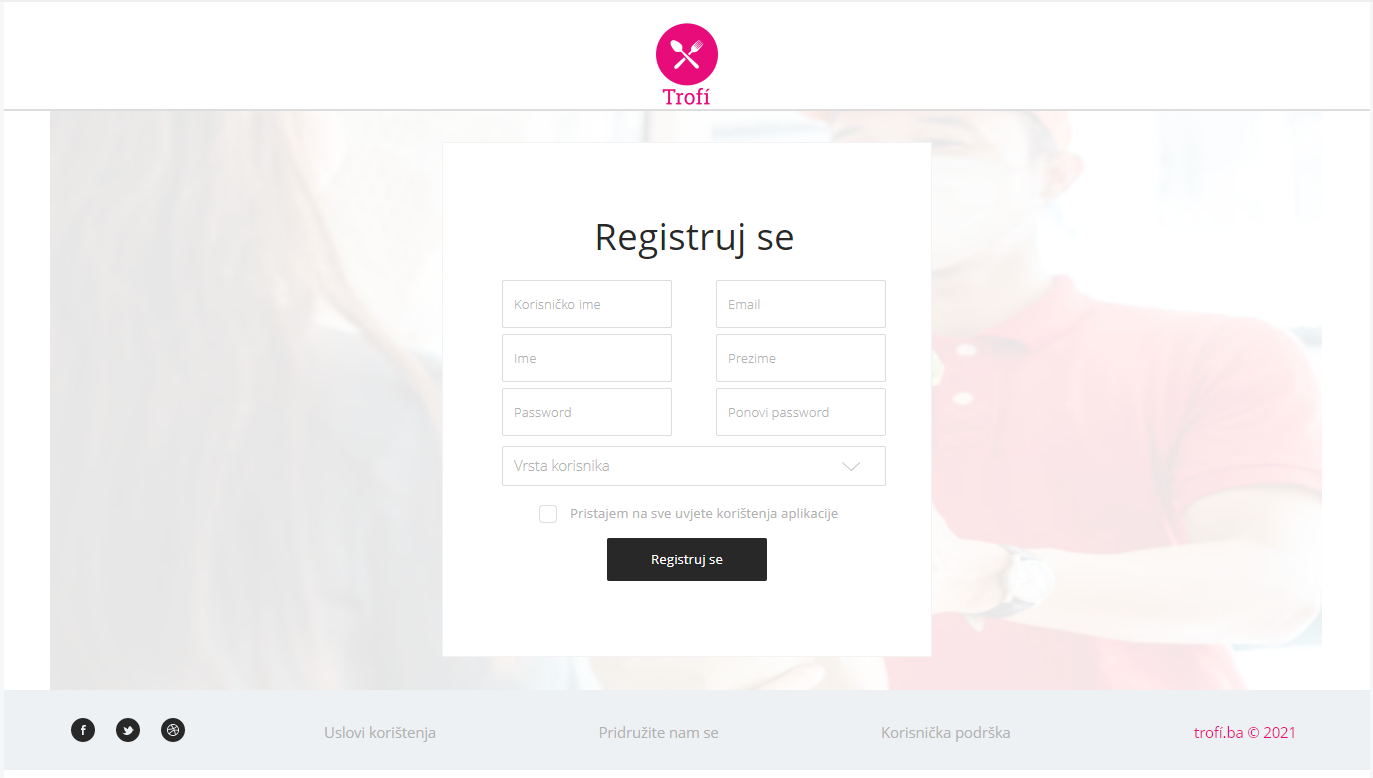
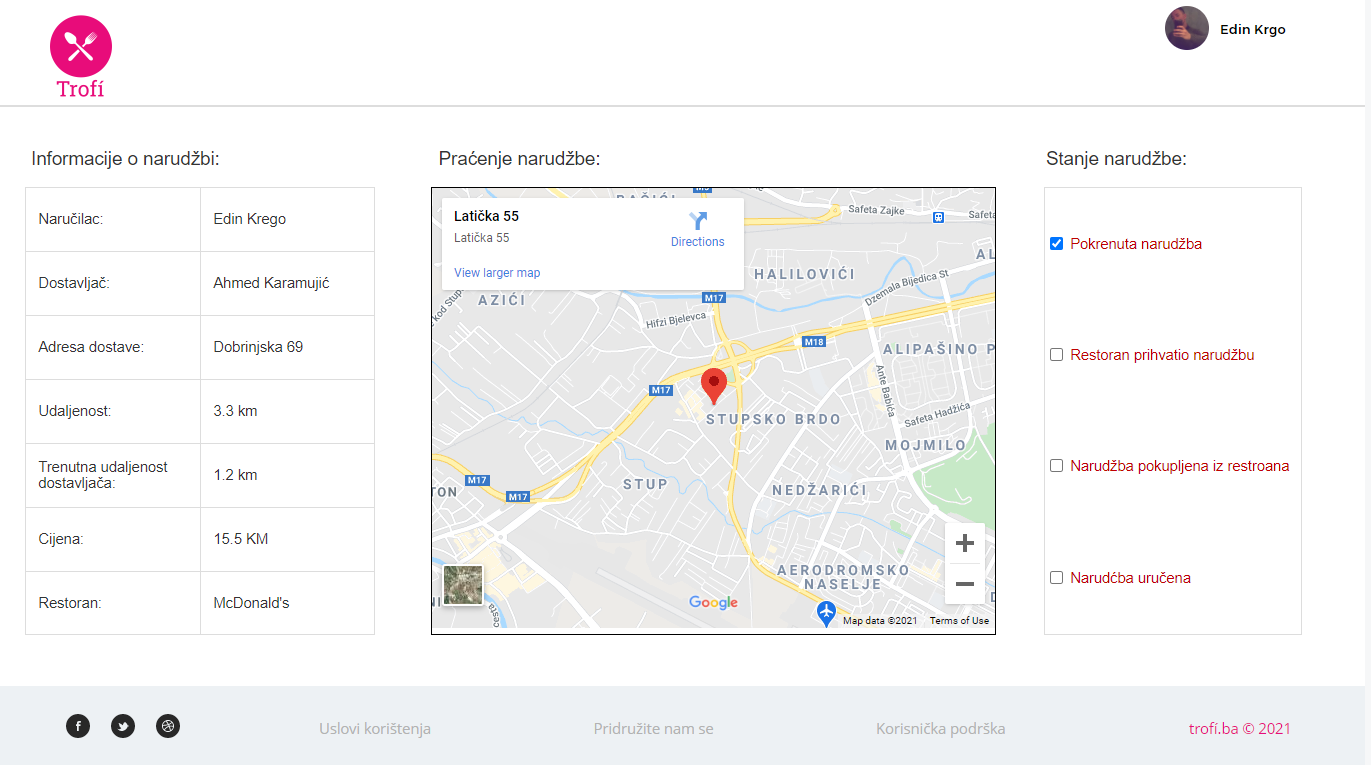
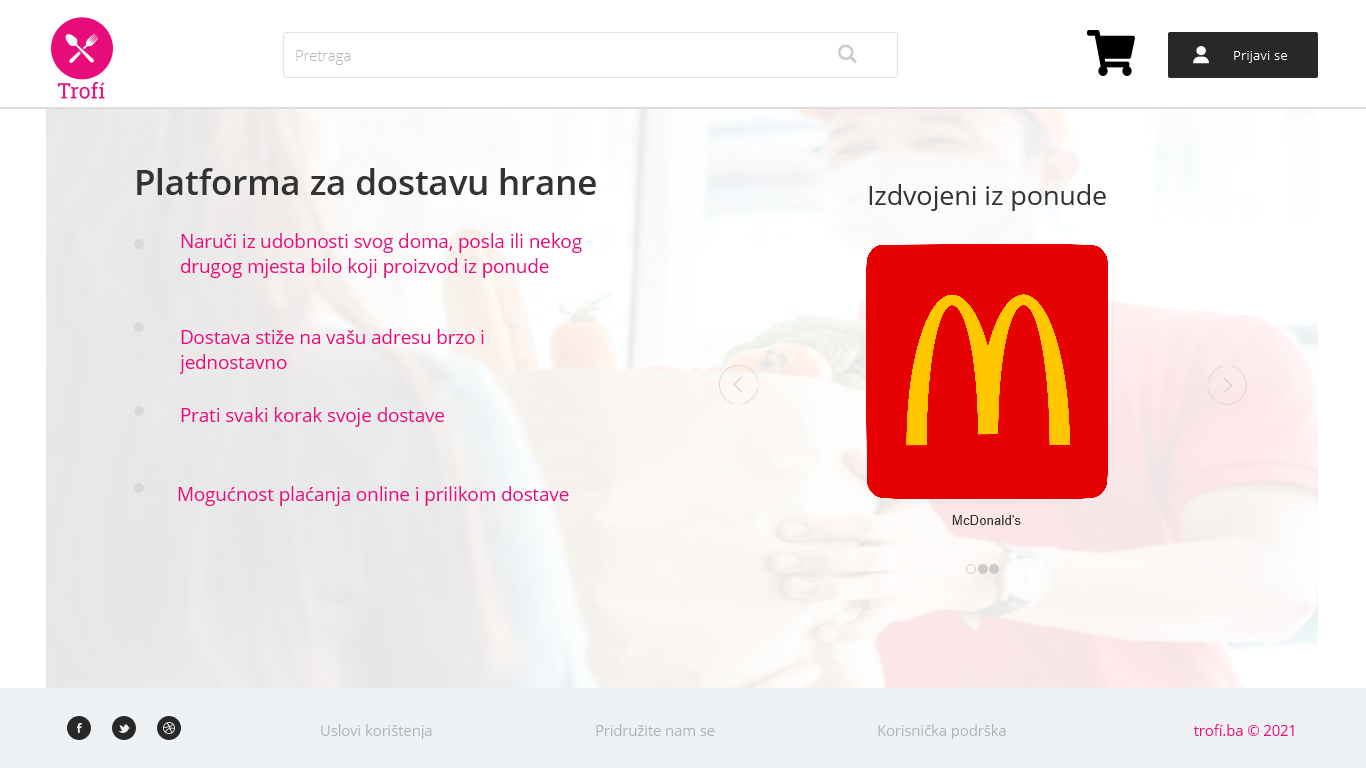
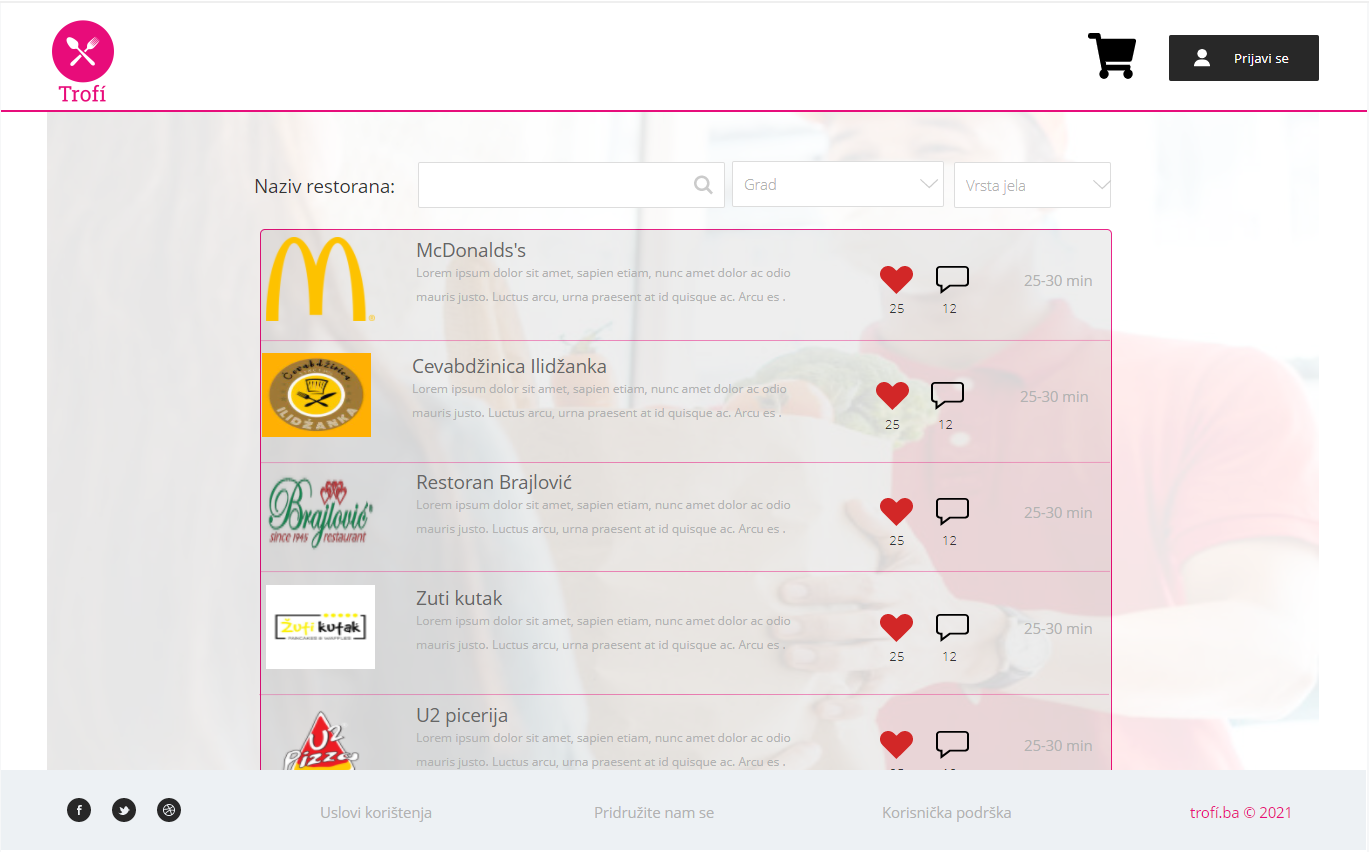
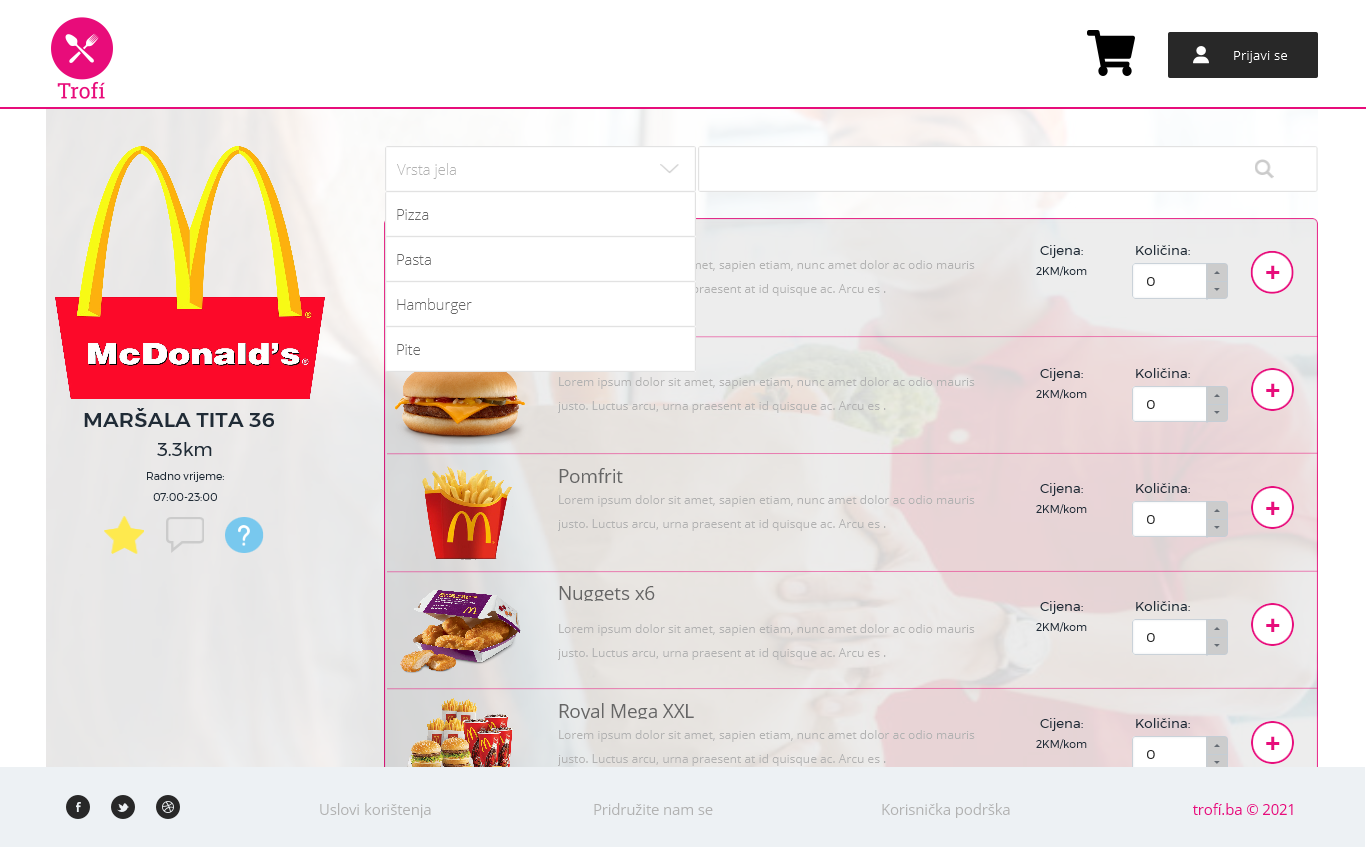
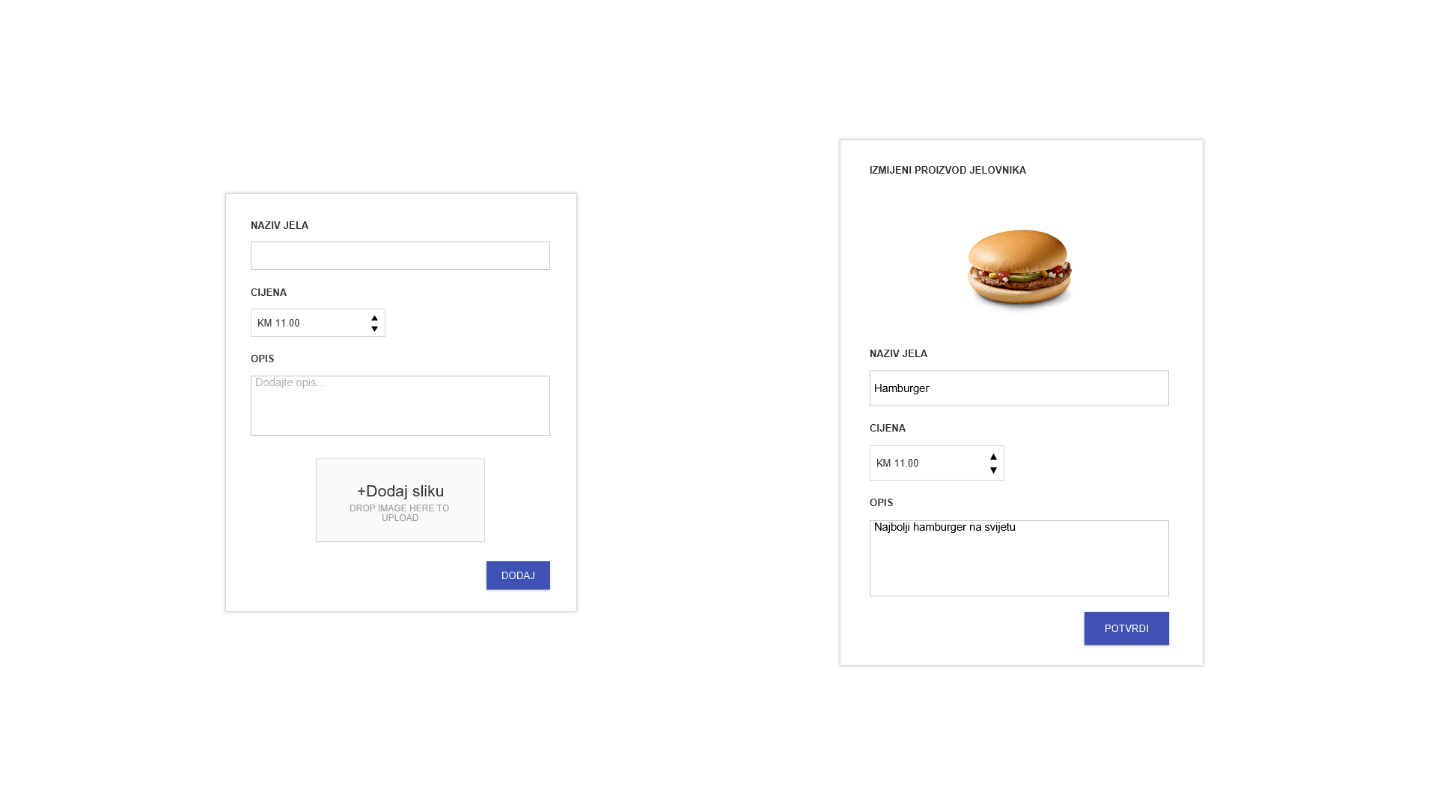
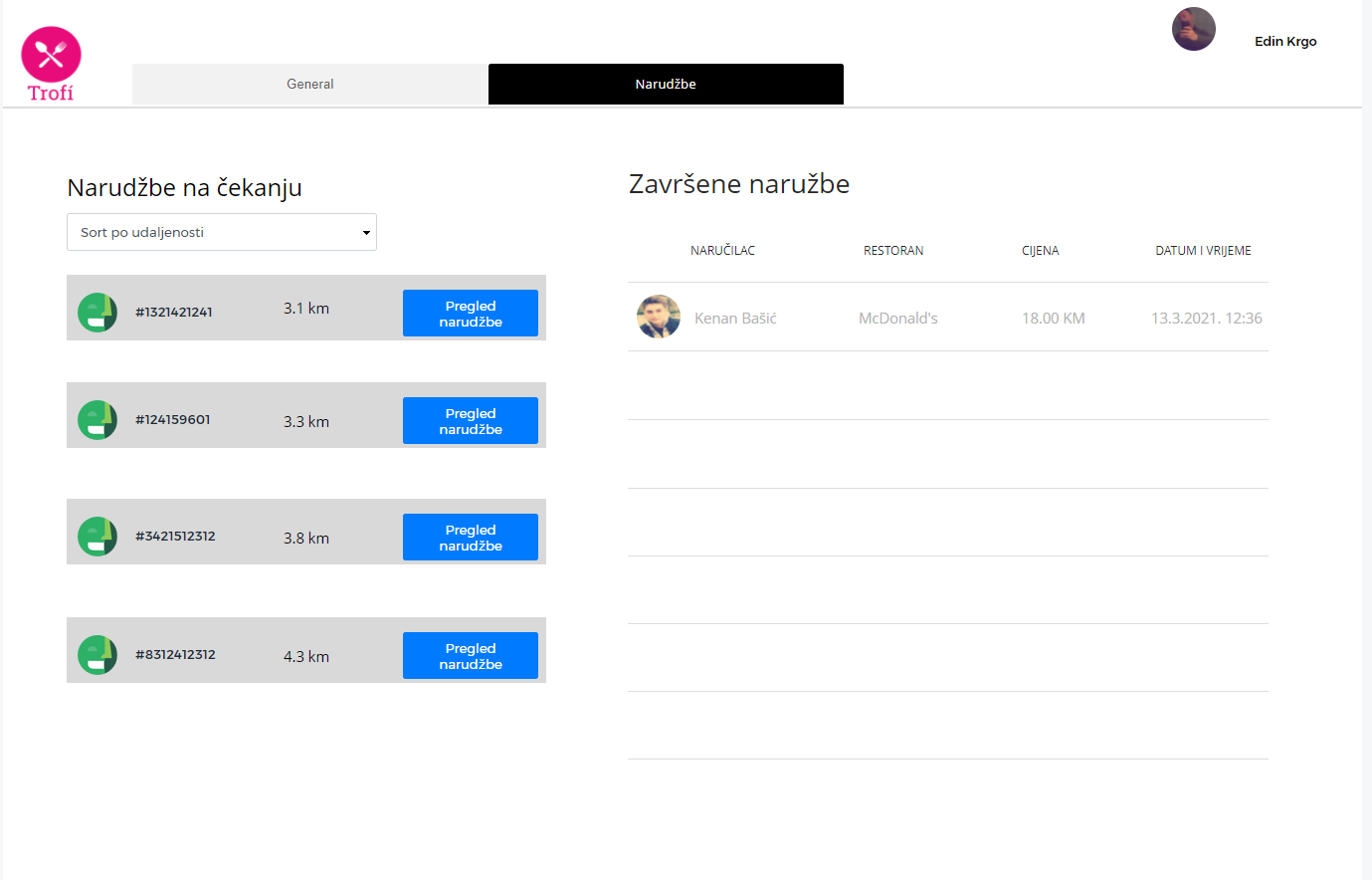
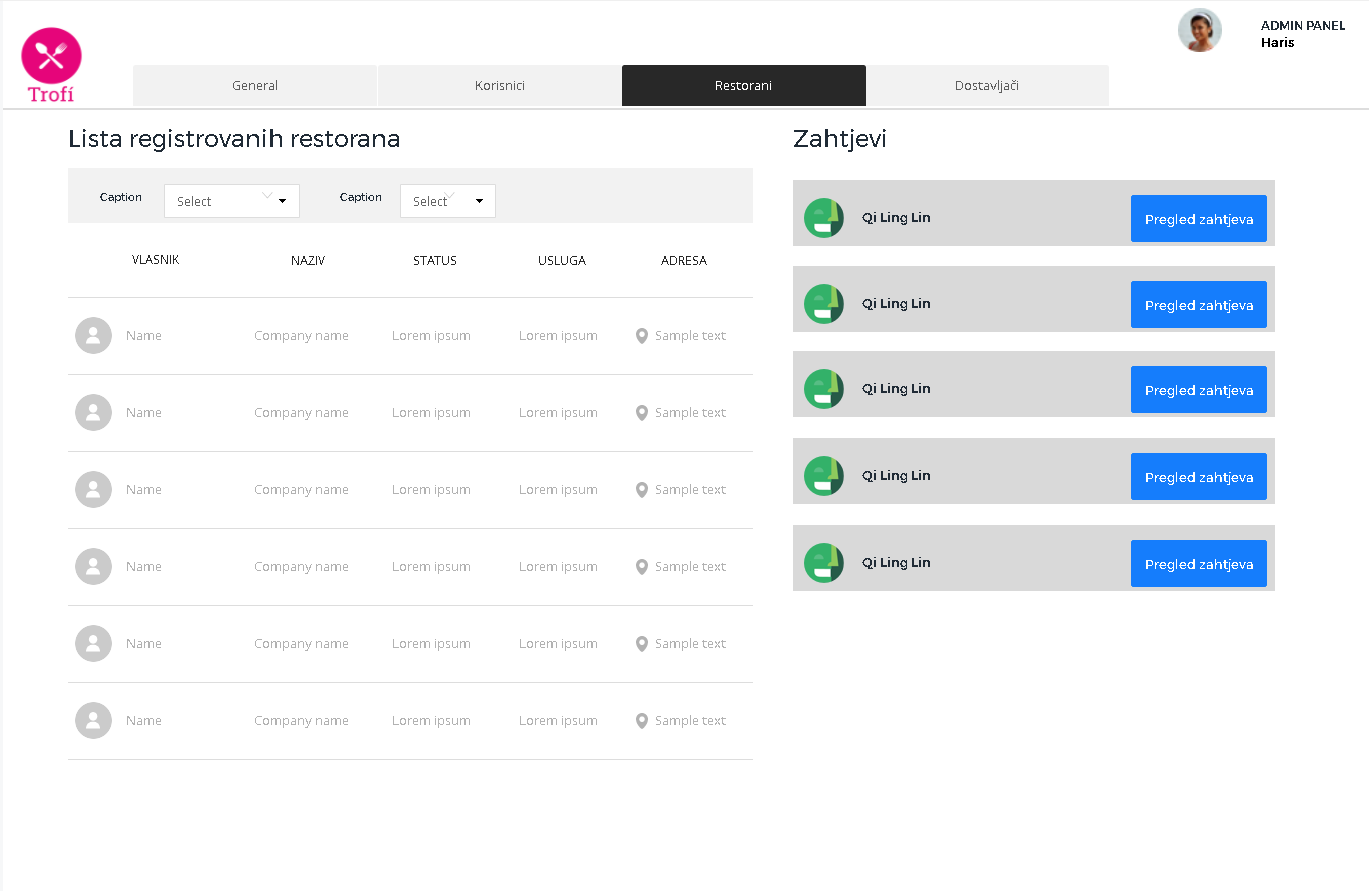
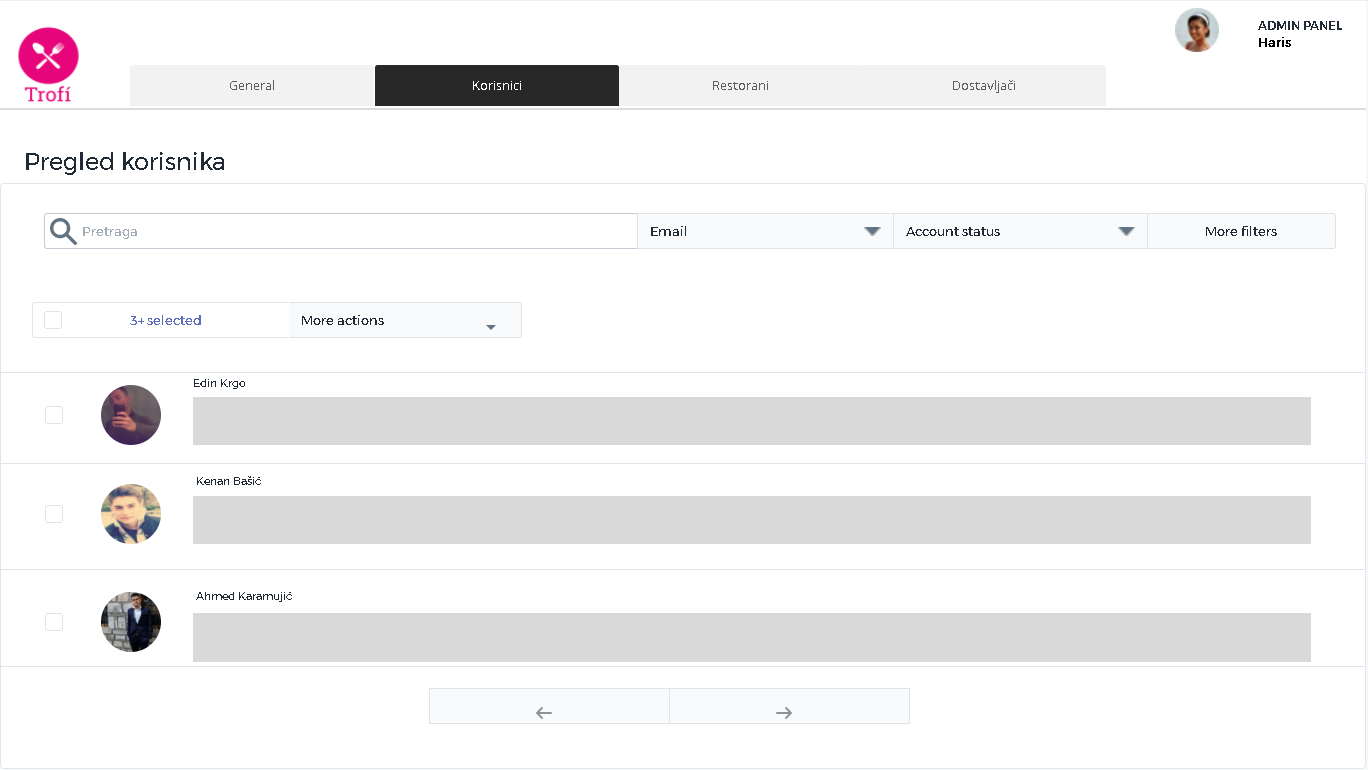
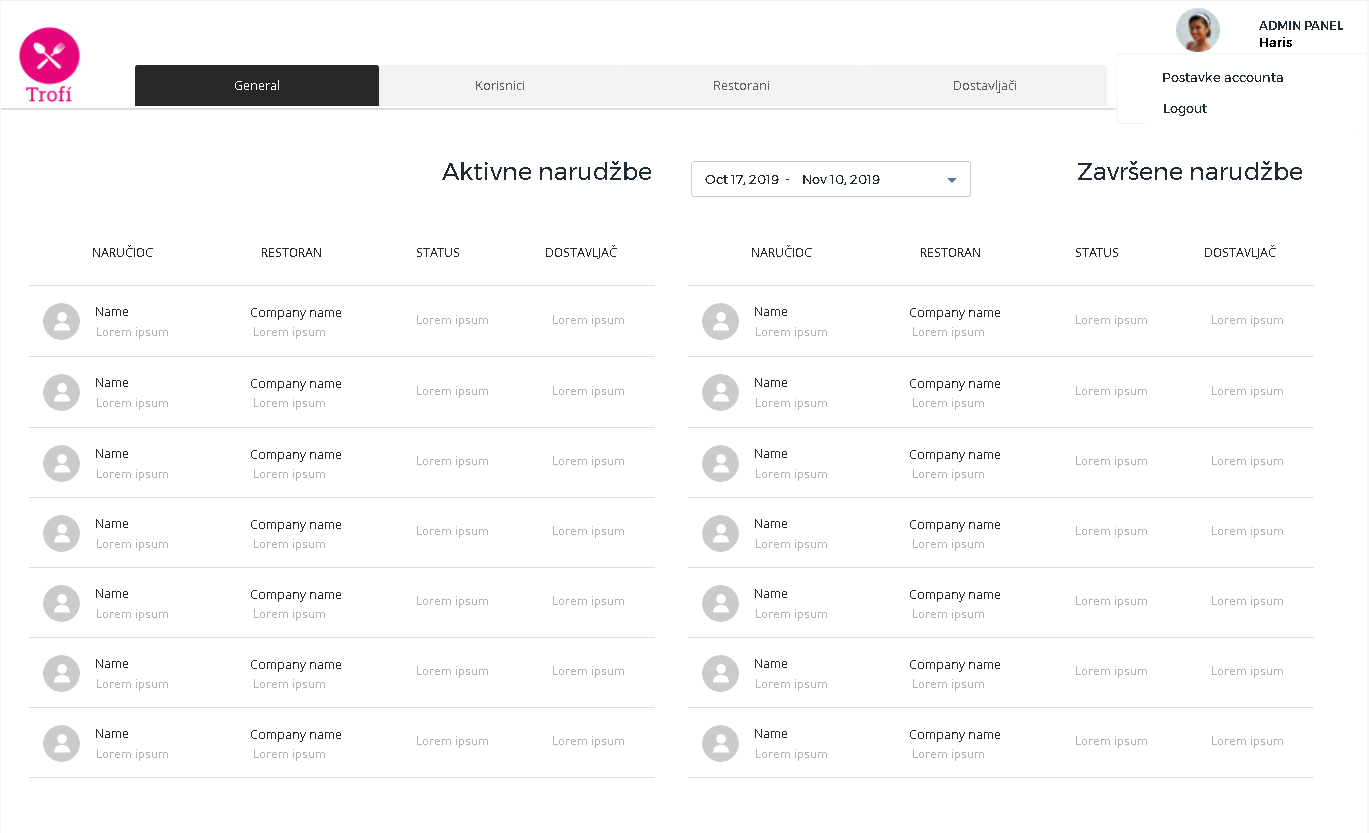
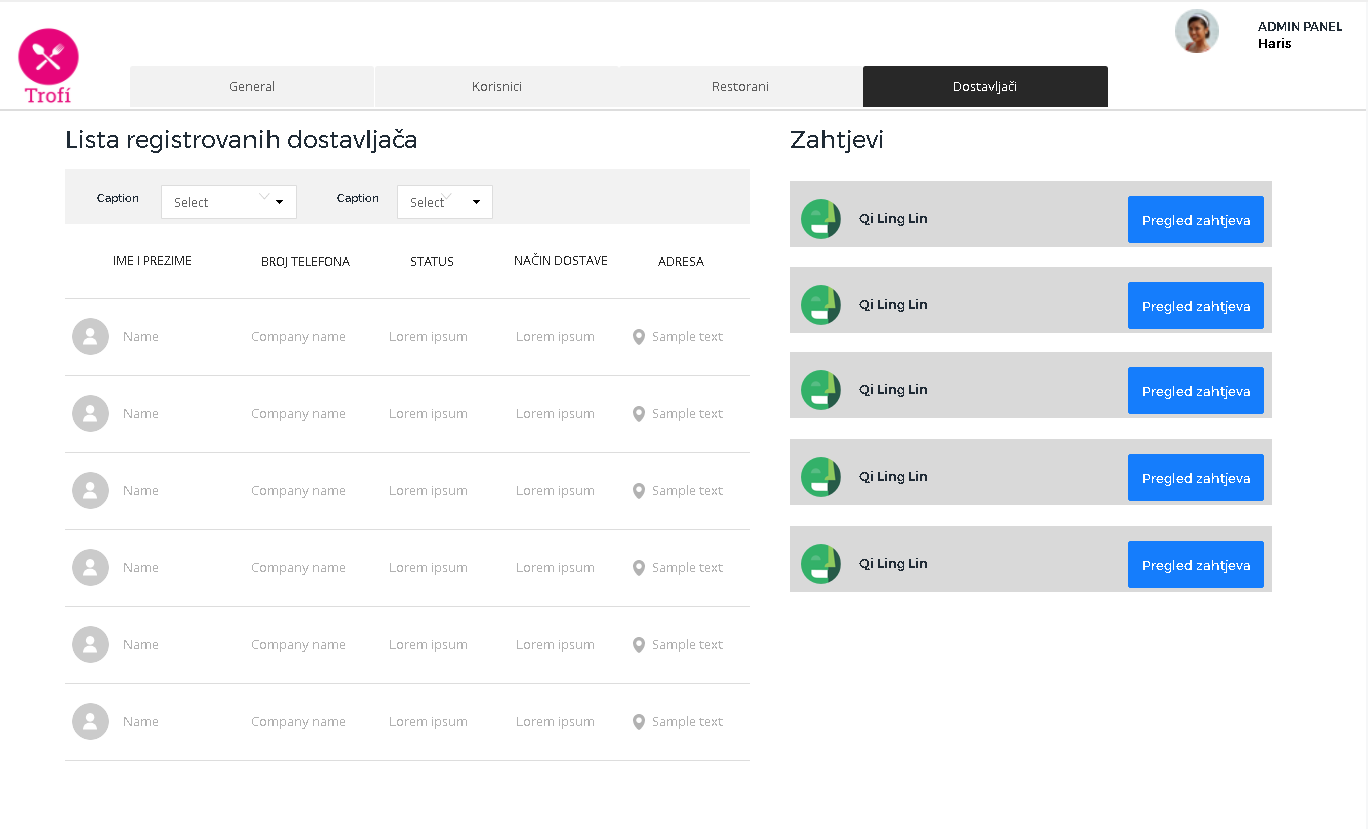
Dijagram objekata



# ER dijagram



# Prototipi



# SOLID Principi

1. Single Responsibility Principle - Princip pojedinačne

odgovornosti

Princip je ispoštovan zato što većina klasa sadrži samo svoje atribute i

CRUD metode. Svaka klasa se brine samo za jednu stvar, samim tim ima

samo jednu odgovornost.

2. Open Closed Principle - Otvoreno zatvoren princip

Nijedna klasa ne narušava navedeni SOLID princip jer i u slučajevima

kada jedna klasa ima instancu druge klase kao atribut ili je neka vrsta

kontejnerske klase za drugu klasu, moguća je samo modifikacija

okruženja odnosno atributa instanci klase.

3. Lisk Substitution Principle -Liskov princip zamjene

Liskov princip zamjene je ispoštovan uvodeći Korisnik apstraktnu klasu.

Na mjestima gdje se koristi navedena apstraktna klasa moguće je bez

problema koristiti bilo koju od izvedenih klasa (Naručilac, Restoran,

Dostavljač).

4. Interface Segregation Principle -Princip izoliranja interfejsa

Princip je u suštini ispoštovan jer klase ne implementiraju nikakve

dodatne interfejse (izuzmemo li komunikaciju s bazom podataka).

5. Dependency Inversion Principle - Princip inverzije ovisnosti

Princip je zadovoljen koliko je moguće što se tiče implementacije

apstraktnih klasa jer je to bilo moguće samo za potrebe korisničkih klasa

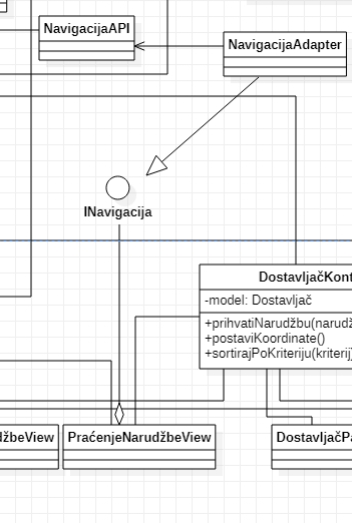
(Naručilac, Restoran, Dostavljač).

# Design patterni

1. Strukturalni

Adapter patern

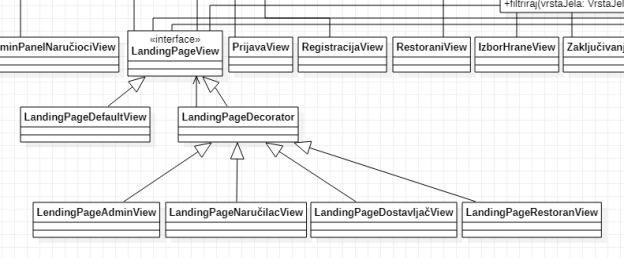
Ako korisnik želi da koristi neki drugi servis za navigaciju, možemo k reirati adapter patern koji će ispuniti želju korisnika. Ovim ćemo postići veću fleksibilnost i sami korisnici neće ovisiti o jednom servisu.



Fasadni patern

Prilikom implemenatacije dijagrama klasa, ovaj pattern smo iskoristili kreiranjem klase Narudžba.

Dekorator patern

Ovaj patern dodaje određene akcije objektu preko dekorator klase. Dekorator smo iskoristili kod landing page-a, jer footer je svakom viewu identičan, dok se navbar mijenja u odnosu na vrstu korisnika.

Bridge patern

S obzirom da imamo samo jednu vrstu nasljeđivanja ovaj patern u našem slučaju ne možemo iskoristiti. Kompozitni patern

Ne vidimo situaciju u kojoj bi razumno mogli iskoristiti ovaj patern.

Proxy patern

Ideja je da kada korisnik klikne na sliku od jela da mu se prikaže slika sa većom rezolucijom preko proxy klase.

Flyweight patern

Ukoliko želimo da administrator vidi trenutni položaj svih dostavljača na mapi, dovoljno je kreira ti samo jedan grafički prikaz dostavljača. Svaki dostavljač će imati atribut grafičkog prikaza I u zavisnosti od njegovih koordinata poziocionirat ćemo ga na mapi.

1. Kreacijski

Singleton pattern

Osigurava da je samo jedna instanca objekta kreirana te osigurava pristup objektu na globalnom nivou.

U suštini sve klase koje vrše razmjenu resursa (klase koje komuniciraju s bazom podataka, razmjenu podataka između servera i korisnika)

bit će implementirane kao Singleton klase čime će biti iskorišten Singleton design pattern.

Prototype pattern

U slučaju da je kreiranje novog objekta neke klase zahtjevno (ometa i/ili opterećuje rad sistema, iziskuje mnogo memorije, šteti performansama sistema)

a znajući da minorne prepravke generalizovanog objekta ispunjavaju naše zahtjeve, koristi se Prototype design pattern. U našem slučaju, opisani pattern

bi najviše imalo smisla ukomponovati ili kod kreiranja novog jela od strane restorana ili kod kreiranja nove narudžbe od strane korisnika.

Factory Method pattern

Pattern je poznat i pod nazivom Virtual Constructor pattern, a koristan je i poželjno ga je implementirati ukoliko neka podklasa odlučuje o tome

koji objekat će biti kreiran. Konkretno, u našoj aplikaciji bi implementacija Factory Method patterna mogla biti ispunjena uz pomoć klase

Korisnik Factory koja u zavisnosti od podataka koje prima kreira objekat neke od klasa izvedenih iz apstraktne klase Korisnik.

Abstract Factory pattern

Nudi interface koji služi za kreiranje sličnih objekata bez eksplicitnog navođenja tipa. Koristi se kod manjka modularizacije sistema, kod skupina

objekata koji su dizajnirani da rade zajedno, te kada je sistem potrebno izolovati od načina na koji rade programi kojima se koristi.

Ne vidimo konkretnu primjenu datog patterna kada je u pitanju naša aplikacija.

Builder pattern

Navedeni pattern je veoma koristan jer pruža interface za kreiranje kompleksnijih objekata tj. objekata koji koriste resurse i podatke

postojećih objekata za svoju "izgradnju". Da bi smanjili kompleksnost kreiranja nekog objekta implementira se Builder pattern kojim fazu kreiranja razbijamo u više dijelova.

Builder pattern u našoj aplikaciji bi bilo prikladno iskoristiti kod objekata klase InfoNarudžba.

1. Patterni ponašanja

Strategy pattern

Korisno je implementirati strategy pattern da bi se različiti algoritmi koji služe za rješavanje nekog problema izdvojili u posebne klase.

Moguće bi ga bilo iskoristiti kod algoritama sortiranja proizvoda odabranog restorana.

State pattern

Predstavlja dinamičku verziju strategy patterna i postiže se promjenom podklase unutar hijerarhije klasa. U našem slučaju prikladno bi bilo

implementirati ga kod promjene stanja narudžbe (pokrenuta, prihvaćena itd.).

Template method pattern

Navedeni pattern omogućava izdvajanje određenih koraka nekog algoritma u upotrebi u više zasebnih podklasa pri čemu se struktura algoritma

ne mijenja. Nismo pronašli potrebu za izvedbom navedenog patterna.

Observer pattern

Observer pattern definiše one-to-many zavisnost između objekata, takvu da kada se promjeni stanje jednog objekta, zavisni objekti bivaju

obaviješteni i ažurirani. U našoj aplikaciji klasa Narudžba bi imala ulogu Observera te, u zavisnosti od lokacije dostavljača,

ažurirala status narudžbe.

Iterator pattern

Potreba za iterator patternom se javlja kod sekvencijalnog pristupa kolekciji objekata bez otkrivanja interne strukture objekata

prisutnih u kolekciji i poznavanja strukture kolekcije. Hipotetska izvedba ovog patterna bi bila implementirana za klasu Meni,

tačnije sekvencijalni pristup kolekciji jela nekog restorana.

Chain of responsibility pattern

Pattern omogućuje objektu da pošalje instrukciju bez znanja koji objekat će primiti i obraditi poslanu instrukciju. Svaki objekat uvezan

u lanac(chain) može obraditi instrukciju, proslijediti je drugom objektu ili oboje. Kad korisnik kreira narudžbu mogla bi se, kao kod

primjera sa CV-em sa laboratorijske vježbe, provesti obrada narudžbe.

Mediator pattern

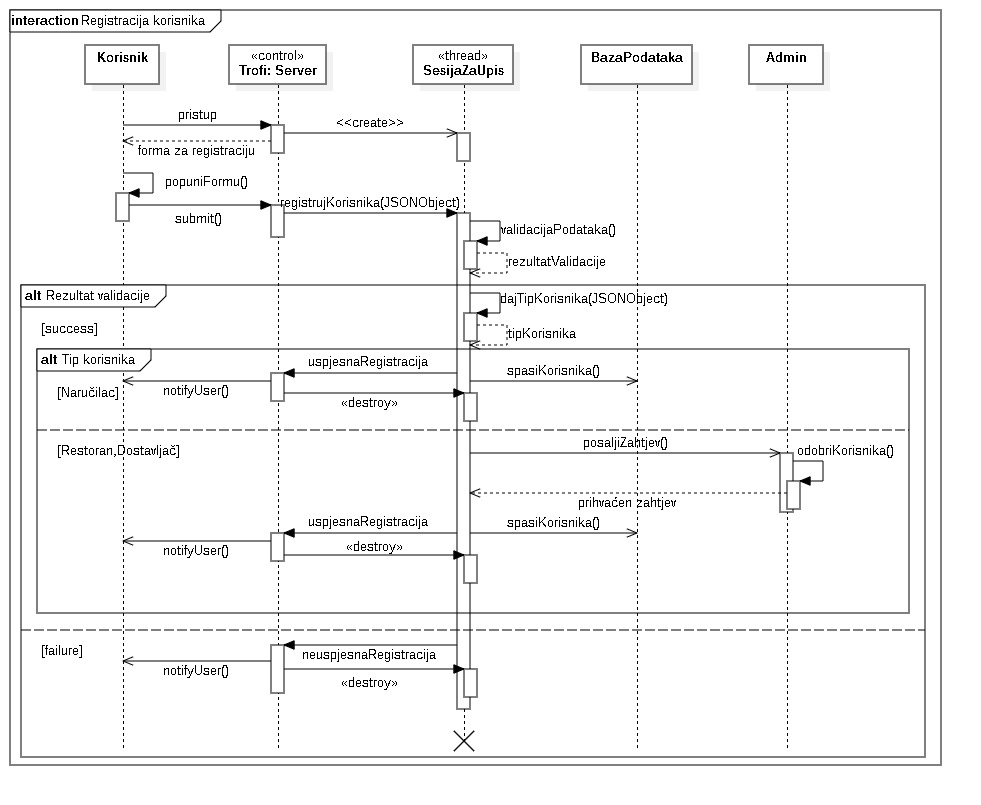
Izvedba mediator patterna definiše objekat koji enkapsulira interakciju skupa objekata. Rješava se problem tight coupling-a jer se izbjegava

direktna komunikacija između objekata, te omogućava presretanje i modifikaciju same interakcije odvojeno od objekata.

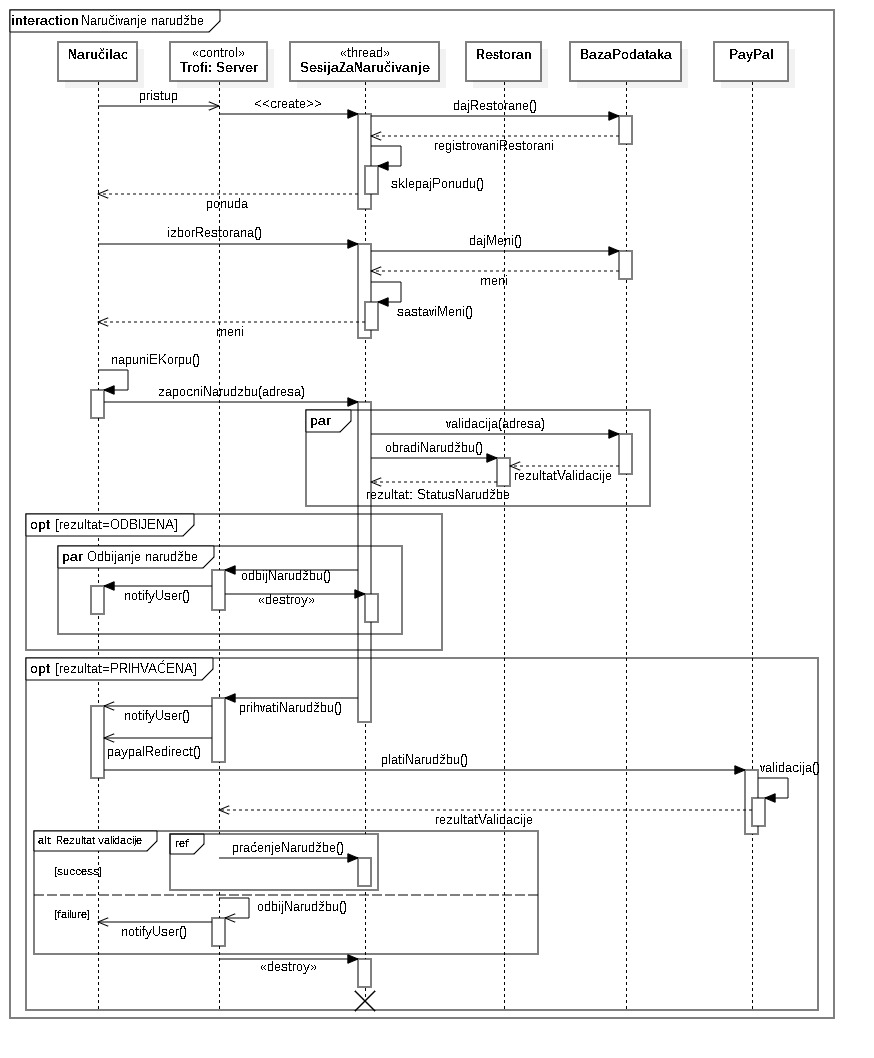
Mediator pattern je poželjno izvesti kod bilo kakve upotrebe formi (log in, sign up...).

# Dijagrami sekvenci

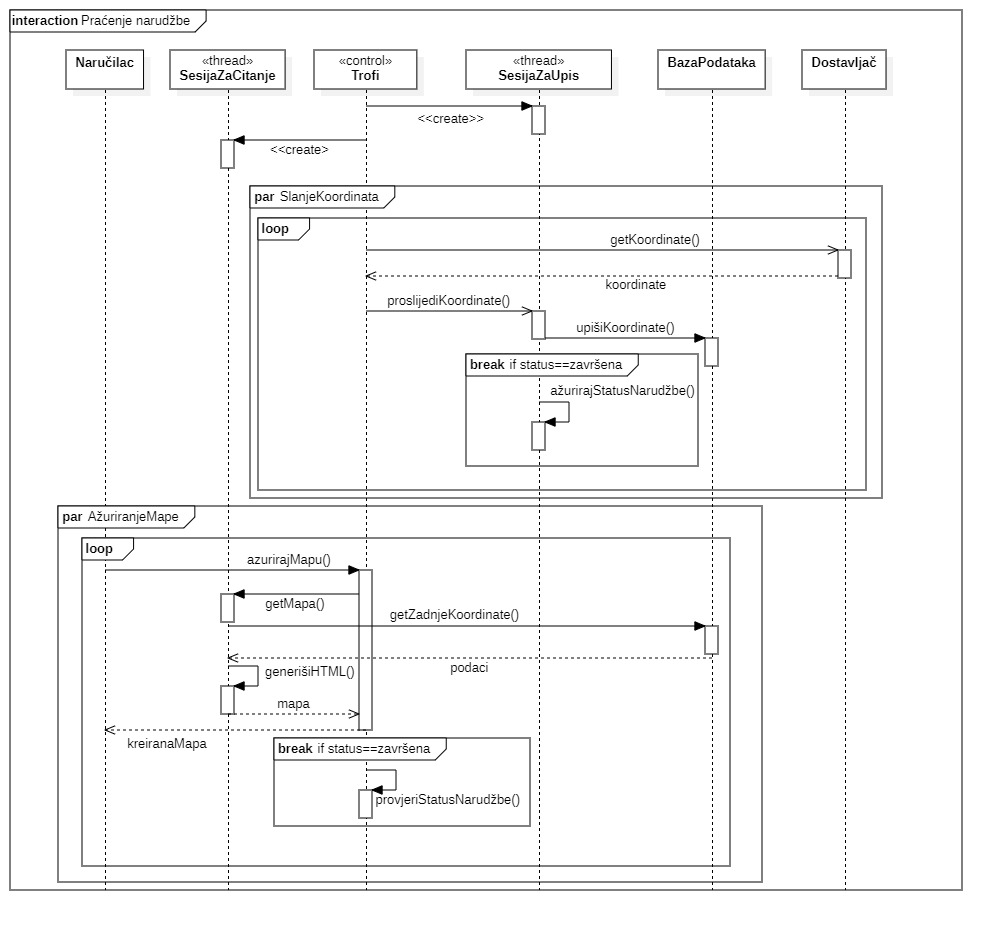
1. Registracija korisnika



1. Naručivanje narudžbe

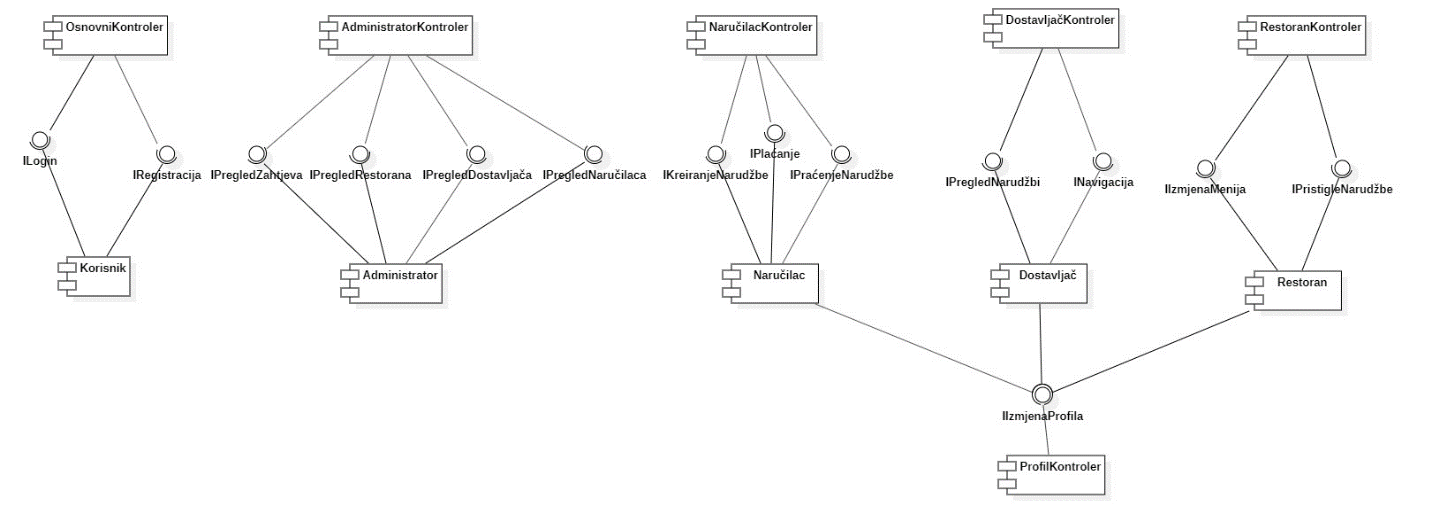


1. Praćenje narudžbe

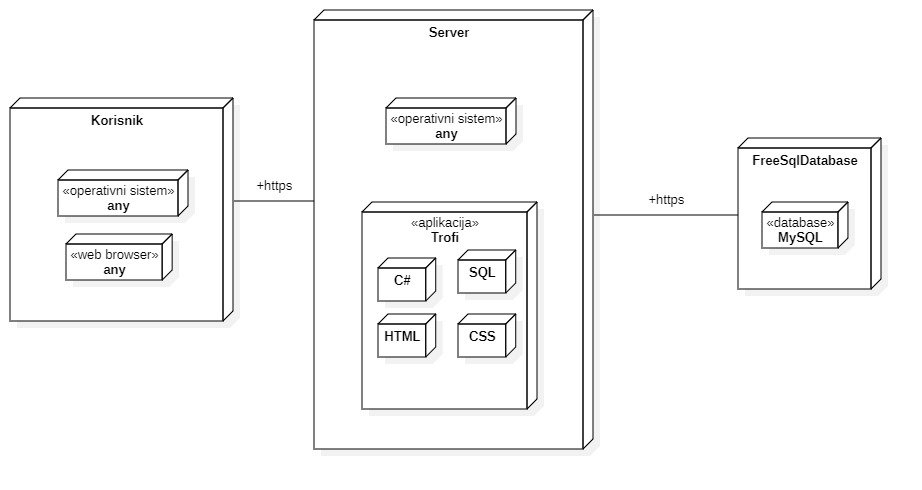


# Dijagrami komponenti, paketa i raspoređivanja

1. Dijagram komponenti



1. Dijagram raspoređivanja



1. Dijagram paketa

