

## Specifikacija projekta

### 1. Osnovne informacije o sistemu

**Naziv teme:** QuickBite

**Logo:**



**Naziv tima:** zenCoders (Tim 17)

**Nastavna grupa:** Grupa 5

**Link na repozitorij tima:** <https://github.com/OOAD-2023-2024/QuickBite>

**Članovi tima:**

1. Elma Tirak
2. Umihana Šećunović
3. Mahir Rešidović

**Namjena sistema:**

*Opisati sistem i njegovu namjenu sa maksimalno sedam rečenica. U okviru ovog polja potrebno je objasniti šta sistem treba raditi na apstraktном nivou, bez detaljnog objašnjavanja pojedinačnih funkcionalnosti i načina razlikovanja aktera sistema (što je predmet daljih poglavlja).*

**QuickBite** je sistem koji povezuje korisnike, ugostiteljske jedinice i tržne centre u cilju omogućavanja kupovine, primanja i slanja različitih proizvoda unutar grada.

Ovaj sistem je zamišljen kao odgovor na sve brzi život ljudi u velikim gradovima, te nakon napornog dana ili u nedostatku vremena tokom izvršavanja svakodnevnih obaveza omogućava pristup svakoj tački na mapi Vašeg grada.

Bez obzira da li vam se jedu tradicionalni čevapi, savijača sa jabukom, moderni internacionalni obrok ili ste možda zaboravili kupiti brašno, QuickBite povezuje Vas sa najboljom ponudom u gradu.

Opisani sistem vodi računa o Vašoj sigurnosti i privatnosti, nudeći sigurne opcije plaćanja i transparentnost u svakom koraku procesa naručivanja.

## 2. Funkcionalnosti (poslovni procesi) Sistema

*Opisati 6 do 8 najznačajnijih funkcionalnosti sistema (u zavisnosti od broja članova u timu). Funkcionalnosti sistema predstavljaju usluge koje sistem pruža korisnicima. Sve funkcionalnosti pripadaju nekoj od različitih vrsta: u svrhu ostvarivanja krajnje usluge sistema, perzistencija podataka (CRUD operacije), operacije koje koriste principe asinhronne obrade zahtjeva, operacije koje koriste specifične algoritme obrade podataka i operacije u kojima se vrši korištenje vanjskih uređaja. Neophodno je nавesti barem po jednu funkcionalnost svake od različitih vrsta.*

- 1) **Naziv funkcionalnosti:** Registracija ili login korisnika

**Vrsta funkcionalnosti:** Perzistencija podataka (CRUD operacija)

**Opis funkcionalnosti:**

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Ukoliko se korisnik prethodno registrovao u sistemu, ponudit će mu se opcija login-a koja će sadržavati unos korisničkog imena i lozinke.

U slučaju da korisnik nije registrovan u sistemu, otvorit će se prozor za registraciju koji će sačinjavati sljedeća polja: ime, prezime, e-mail adresa, korisničko ime i lozinka.

- 2) **Naziv funkcionalnosti:** Pretraživanje kategorije i detalji narudžbe

**Vrsta funkcionalnosti:** Usluga sistema

**Opis funkcionalnosti:**

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Kategorije su dizajnirane tako da korisnici mogu lako odabratи različite opcije, poput 'Namirnice', 'Gotova hrana', 'Pokloni', 'Slatki kutak' i druge specifične kategorije.

Unutar svake kategorije, korisnici mogu koristiti filtere kako bi sortirali proizvode po cijeni, vremenu dostave ili drugim parametrima.

Korisnici mogu unijeti naziv jela, restorana ili bilo kojeg ključnog pojma povezanog s proizvodima ili uslužnim jedinicama koje žele pronaći.

- 3) **Naziv funkcionalnosti:** Izračunavanje popusta na osnovu frekventnosti narudžbi

**Vrsta funkcionalnosti:** Operacija sa specifičnim algoritmom obrade

**Opis funkcionalnosti:**

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Sistem automatski prati i analizira historiju kupovine svakog registriranog korisnika kako bi identificirao frekventne kupce.

Na temelju utvrđene frekvencije kupnje, sistem generiše personalizirane kodove za popust ili neposredne popuste koji se primjenjuju na buduće transakcije.

- 4) **Naziv funkcionalnosti:** Slanje obavijesti o primljenoj narudžbi kuriru

**Vrsta funkcionalnosti:** Korištenje vanjskog uređaja

**Opis funkcionalnosti:**

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Kada se narudžba stvori ili označi kao spremna za dostavu, sistem automatski šalje poruku odgovarajućem kuriru s detaljima narudžbe i uputama za preuzimanje.

Ovo uključuje podatke poput adrese za preuzimanje, adrese isporuke, predviđenog vremena dostave, i posebnih instrukcija.

Integracija sa sistemom za upravljanje dostavama koji koristi e-mail, SMS, ili aplikacijske push notifikacije za brzo i efikasno slanje informacija kuriru.

- 5) **Naziv funkcionalnosti:** Određivanje lokacije korisnika na mapi

**Vrsta funkcionalnosti:** Asinhrona operacija

**Opis funkcionalnosti:**

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Sistem dobavlja i prikazuje trenutnu lokaciju korisnika na mapi unutar aplikacije bez ometanja ostalih interakcija korisnika s aplikacijom.

Ova funkcionalnost omogućava korisnicima da nastave s pregledom aplikacije dok se njihova lokacija ažurira u pozadini.

- 6) **Naziv funkcionalnosti:** Dovršavanje košarice i odabir vrste naplate

**Vrsta funkcionalnosti:** Operacija sa specifičnim algoritmom obrade

**Opis funkcionalnosti:**

*Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.*

Korisnik može pregledati sve stavke u košarici, izvršiti potrebne izmjene, a zatim nastaviti na sigurno plaćanje kroz integrirani sistem naplate. Pri završetku, potvrda o uspješnoj kupovini se prikazuje i šalje korisniku.

### 3. Akteri sistema

Potrebno je navesti najmanje tri aktera sistema. Neophodno je navesti barem po jednog aktera za svaku od različitih vrsta.

*Korisnici usluga sistema*

a) **Naziv aktera: Kupac**

**Vrsta aktera:** Korisnik usluge

**Funkcionalnosti u kojima akter učestvuje:**

Funkcionalnost sistema	Način učešća
1) Registracija ili login korisnika	Mogućnost uređivanja
2) Pretraživanje kategorije i detalja narudžbe	Mogućnost uređivanja
3) Dovršavanje košarica i odabir vrste naplate	Mogućnost uređivanja

b) **Naziv aktera: Kurir**

**Vrsta aktera:** Zaposlenik sistema

**Funkcionalnosti u kojima akter učestvuje:**

Funkcionalnost sistema	Način učešća
1) Slanje obavijesti o primljenoj narudžbi kuriru	Mogućnost pregleda
2) Određivanje lokacije korisnika na mapi	Mogućnost uređivanja
3) Registracija ili login korisnika	Mogućnost uređivanja

c) **Naziv aktera:** Sistemski administrator

**Vrsta aktera:** Administrator

**Funkcionalnosti u kojima akter učestvuje:**

Funkcionalnost sistema	Način učešća
1) Registracija ili login korisnika	Mogućnost uređivanja
2) Slanje obavijesti o primljenoj narudžbi kuriru	Mogućnost uređivanja
3) Izračuvanje popusta na osnovu frekventnosti narudžbi	Mogućnost uređivanja

#### 4. Nefunkcionalni zahtjevi sistema

*Opisati najmanje tri najznačajnija nefunkcionalna zahtjeva sistema. Nefunkcionalni zahtjevi predstavljaju ograničenja koja sistem mora zadovoljiti kako bi mogao ispravno obavljati svoje funkcionalnosti. Validacije polja za unos vrijednosti ne predstavljaju nefunkcionalne zahtjeve.*

1) **Naziv nefunkcionalnog zahtjeva:** Sigurnost podataka korisnika

**Opis:**

*Opisati ograničenje sistema i način na koje se ono ispoljava.*

Sistem mora osigurati visok nivo sigurnosti korisnika, zbog zaštite ličnih podataka, kao i eventualnih podataka o detaljima plaćanja. Pomenuta sigurnost se može postići na više načina od kojih bi najefikasniji bili verifikacija putem e-maila ili jednokratni pristupni kod.

2) **Naziv nefunkcionalnog zahtjeva:** Jednostavnost korištenja za krajnje korisnike

**Opis:**

*Opisati ograničenje sistema i način na koje se ono ispoljava.*

Važno je da sistem bude prilagođen različitim kategorijama krajnjih korisnika, što podrazumijeva mogućnost upotrebe aplikacije od strane korisnika koji pripadaju različitim dobnim skupinama ili nemaju tehničko iskustvo korištenja savremenih aplikacija.

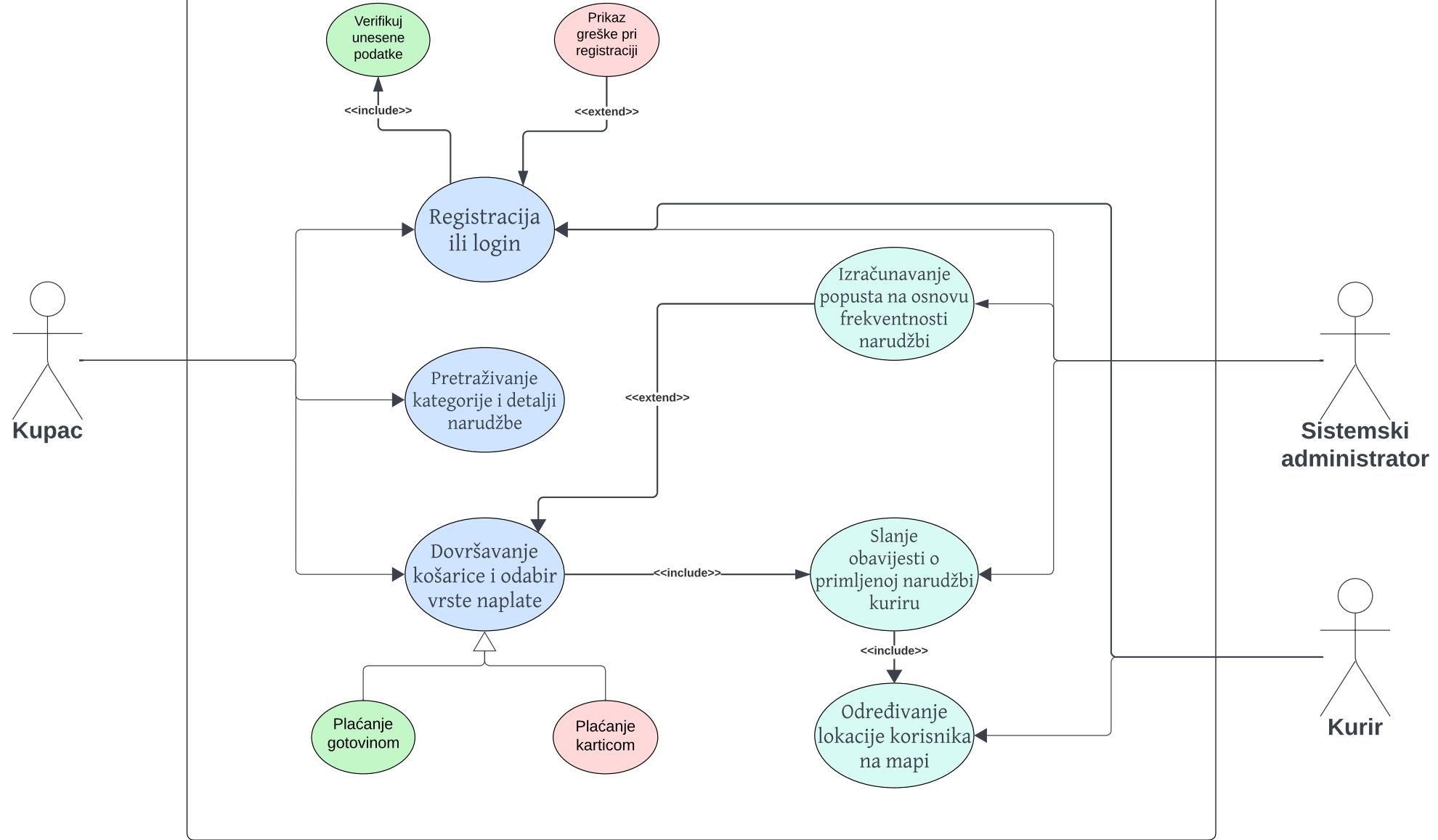
3) **Naziv nefunkcionalnog zahtjeva:** Prateća dokumentacija i napomene

**Opis:**

*Opisati ograničenje sistema i način na koje se ono ispoljava.*

Važno je da se korisnik pri upotrebi funkcionalnosti aplikacije na vrijeme obavijesti o sljedećim koracima koji bi mogli izazvati tehničke ili logičke nedostatke u dalnjem izvršavanju programa. Korisniku će u toku korištenja programa biti dostupna često postavljana pitanja (FAQ) i preporuke sistema za daljnje korake.

# QuickBite



## **DIJAGRAMI SLUČAJEVA UPOTREBE – SCENARIJI**

### **Scenario 1**

<b>Naziv slučaja upotrebe</b>	<b>Registracija ili login korisnika</b>
<b>Opis slučaja upotrebe</b>	Sistem omogućava korisnicima da kreiraju novi korisnički nalog i da se uloguju u sistem kako bi koristili usluge aplikacije
<b>Vezani zahtjevi</b>	-
<b>Posljedice -uspješan završetak</b>	Kupac je registrovan i prijavljen u sistem, te ima pristup personalizovanom korisničkom interfejsu
<b>Posljedice -neuspješan završetak</b>	Kupac nije registrovan ili prijavljen u sistem zbog neispravnih podataka ili greške u sistemu
<b>Primarni akteri</b>	Kupac
<b>Ostali akteri</b>	-
<b>Glavni tok</b>	Kupac bira opciju za registraciju, unosi tražene informacije (ime, e-mail, lozinka), pristaje na uslove korištenja, bira opciju "Registruj se", a zatim validira unesene podatke i šalje e-mail za verifikaciju. Sistem potvrđuje uspješnu registraciju. Kupac bira opciju "Login" i unosi svoje podatke koje zatim sistem provjerava i prijavljuje korisnika.

### **Tok dogadaja 1.1 - Uspješan završetak**

<b>Kupac</b>	<b>Sistem</b>
1. Biranje opcije za registraciju, unošenje informacija i slanje registracionog formulara	
	2. Prihvatanje i provjera korisničkih podataka (slanje e-maila za verifikaciju)
3. Verifikacija putem e-maila	
4. Prijavljivanje na sistem	

### Tok događaja 1.2 - Neuspješan završetak

Kupac	Sistem
1. Biranje opcije za registraciju, unošenje informacija i slanje registracionog formulara	
	2. Prihvatanje i validacija podataka, te eventualno otkrivanje greške (npr. korisničko ime već postoji, lozinka je slaba, podaci su nepotpuni)
3. Prikaz poruke o grešci, ponovni unos (korekcija) podataka i slanje	
	4. Otkrivanje nove ili iste greške
5. Odustajanje od registracije ili ponovni pokušaj	

### Scenario 2

<b>Naziv slučaja upotrebe</b>	<b>Pretraživanje kategorije i detalji narudžbe</b>
<b>Opis slučaja upotrebe</b>	Kupac pretražuje različite kategorije i detalje proizvoda koje želi naručiti
<b>Vezani zahtjevi</b>	-
<b>Preduslovi</b>	Kupac je registrovan i prijavljen u aplikaciju
<b>Posljedice - uspješan završetak</b>	Kupac uspješno nalazi i selektira proizvode za narudžbu
<b>Posljedice - neuspješan završetak</b>	Kupac ne može pronaći ili odabrati proizvode
<b>Primarni akteri</b>	Kupac
<b>Ostali akteri</b>	-
<b>Glavni tok</b>	Kupac unosi ključne riječi u pretraživač i pregleda ponuđene opcije ili učitava novu stranicu sa proizvodima koji svojim imenom ili detaljima odgovaraju pomenutoj riječi, ukoliko u ponuđenim opcijama nije pronašao proizvod koji je tražio
<b>Alternative/proširenja</b>	-

### Tok dogadaja 2.1 - Uspješan završetak

Kupac	Sistem
1. Otvara prozor koji se nudi poslije registracije i bira opciju za pretragu	
3. Unosi naziv kategorije ili jela u pretragu	2. Prikazuje interfejs za pretragu
	4. Izlistava kategorije ili jela koje odgovaraju unesenim terminima
5. Bira željenu kategoriju i dodaje jelo u korpu	
	6. Prikazuje detalje izabranog jela ili druge kategorije proizvoda (omogućen prikaz korisničke korpe)

### Tok dogadaja 2.2 - Neuspješan završetak

Kupac	Sistem
1. Otvara prozor koji se nudi poslije registracije i bira opciju za pretragu	
	2. Prikazuje interfejs za pretragu
3. Unosi nejasan ili nepostojeći termin u pretragu	
	4. Ne pronalazi rezultate i prikazuje poruku o grešci
5. Pokušava prilagoditi pretraživačke termine	
	6. Ponovo ne uspijeva da pronađe rezultate ili izlistava neke nerelevantne rezultate
7. Odustaje od pretrage	
	8. Vraća se na početni ekran aplikacije

### Scenario 3

<b>Naziv slučaja upotrebe</b>	<b>Izračunavanje popusta na osnovu frekventnosti narudžbe</b>
<b>Opis slučaja upotrebe</b>	Sistem automatski prati i analizira historiju kupovine svakog registriranog korisnika kako bi identificirao frekventne kupce
<b>Vezani zahtjevi</b>	-
<b>Preduslovi</b>	Korisnik mora biti postojeći kupac koji koristi usluge već nekoliko puta
<b>Posljedice – uspješan završetak</b>	Kupac je ostvario popust
<b>Posljedice – neuspješan završetak</b>	Kupac nije uspio da ostvari popust na osnovu svoje historije narudžbi
<b>Primarni akteri</b>	Sistemska administrator
<b>Ostali akteri</b>	Kupac
<b>Glavni tok</b>	Nakon što kupac obavi kupovinu, sistem bilježi podatke o novoj narudžbi, uključujući identifikator kupca, proizvode, cijene i datum narudžbe. Sistem računa broj prethodnih narudžbi u određenom vremenskom razdoblju na osnovu čega procenat popusta može da se izračuna. Izračunati popust se primjenjuje na ukupnu cijenu narudžbe, te se ažurirana cijena s popustom prikazuje kupcu tokom procesa plaćanja.
<b>Alternative/proširenja</b>	-

### Tok događaja 3.1 - Uspješan završetak

<b>Sistemska administrator</b>	<b>Kupac</b>
	1. Obavljanje narudžbe
2. Registracija narudžbe	
3. Analiziranje historije narudžbi tog korisnika kako bi se utvrdilo koliko često je koristio usluge sistema	
4. Izračunavanje odgovarajućeg popusta na osnovu prethodno dobijene frekvencije narudžbi	
5. Primjena popusta i prikaz konačne cijene s primjenjenim popustom tokom procesa plaćanja	

### Tok događaja 3.2- Neuspješan završetak

<b>Sistemska administrator</b>	<b>Kupac</b>
	1. Obavljanje narudžbe
2. Registracija narudžbe	
3. Analiziranje historije narudžbi korisnika i utvrđivanje da je novoregistrovani korisnik	
4. Prikaz konačne cijene bez popusta	

## Scenario 4

<b>Naziv slučaja upotrebe</b>	<b>Slanje obavijesti o primljenoj narudžbi kuriru</b>
<b>Opis slučaja upotrebe</b>	Kada se narudžba napravi ili označi kao spremna za dostavu sistem automatski šalje poruku odgovarajućem kuriru s detaljima narudžbe i uputama za preuzimanje
<b>Vezani zahtjevi</b>	-
<b>Preduslovi</b>	Postojanje povezanosti između sistema za upravljanje narudžbi i sistema za upravljanje dostavom, kao i dostupnost kurira za preuzimanje obavijesti i realizaciju dostave
<b>Posljedice – uspješan završetak</b>	Kurir dobija obavijest o narudžbi
<b>Posljedice – neuspješan završetak</b>	-
<b>Primarni akteri</b>	Sistemske administrator
<b>Ostali akteri</b>	Kurir
<b>Glavni tok</b>	Nakon bilježenja sistem označava narudžbu kao spremnu za dostavu i automatski generira obavijest koja sadrži detalje narudžbe kao što su adresa dostave, kontakt podaci kupca i vrijeme narudžbe. Generirana obavijest se automatski šalje kuriru putem odgovarajućeg kanala komunikacije. Kurir potvrđuje dolazak obavijesti o novoj narudžbi i dolazi na mjesto preuzimanja narudžbe koju treba dostaviti, nakon čega je dostavlja na odgovarajuću adresu.
<b>Alternative/proširenja</b>	-

### Tok događaja 4.1- Uspješan završetak

<b>Sistemske administrator</b>	<b>Kurir</b>
1. Uspješno bilježenje svih detalja narudžbi	
2. Sistem generiše sve relevantne detalje poput adrese isporuke, broja telefona, itd.	
	3. Primanje obavijesti o novoj narudžbi
	4. Dostavljanje narudžbe korisniku

## Scenario 5

<b>Naziv slučaja upotrebe</b>	<b>Određivanje lokacije korisnika na mapi</b>
<b>Opis slučaja upotrebe</b>	Sistem dobavlja i prikazuje trenutnu lokaciju korisnika na mapi
<b>Vezani zahtjevi</b>	-
<b>Preduslovi</b>	Unos adrese od strane korisnika
<b>Posljedice – uspješan završetak</b>	Omogućen uvid u lokaciju kuriru
<b>Posljedice – neuspješan završetak</b>	Kurir ne može dostaviti narudžbu
<b>Primarni akteri</b>	Kurir
<b>Ostali akteri</b>	-
<b>Glavni tok</b>	Kupac u određenom koraku u aplikaciji unosi trenutnu ili adresu stanovanja. Unesenu adresu kurir prosljeđuje na mapu, koja daje tačnu lokaciju sa prikazanim uputama i najkraćim putem, bilo da se radi o dostavi automobilom ili na neki drugi način.
<b>Alternative/proširenja</b>	-

### Tok događaja 5.1 – Uspješan završetak

<b>Kurir</b>	<b>Sistem</b>
	1. Prosljeđivanje unesene adresе kuriru
2. Unos lokacije pomoću aplikacije kao što je npr. Google Maps	3. Prikazivanje udaljenosti kurira od trenutne lokacije korisnika unutar aplikacije
4. Pokretanje izračunavanja najkraćeg puta do odabrane lokacije i odabir načina dostave	

### Tok događaja 5.2 – Neuspješan završetak

<b>Kurir</b>	<b>Sistem</b>
	1. Prosljeđivanje unesene adresе kuriru
2. Unos lokacije pomoću aplikacije kao što je npr. Google Maps	3. Obavještavanje korisnika o neispravnim podacima unutar polja adresе
4. Unos ispravljenih podataka u aplikaciju vezanu za mape (npr. Google Maps)	5. Prikazivanje udaljenosti kurira od trenutne lokacije korisnika unutar aplikacije
6. Pokretanje izračunavanja najkraćeg puta do odabrane lokacije i odabir načina dostave	

## Scenario 6

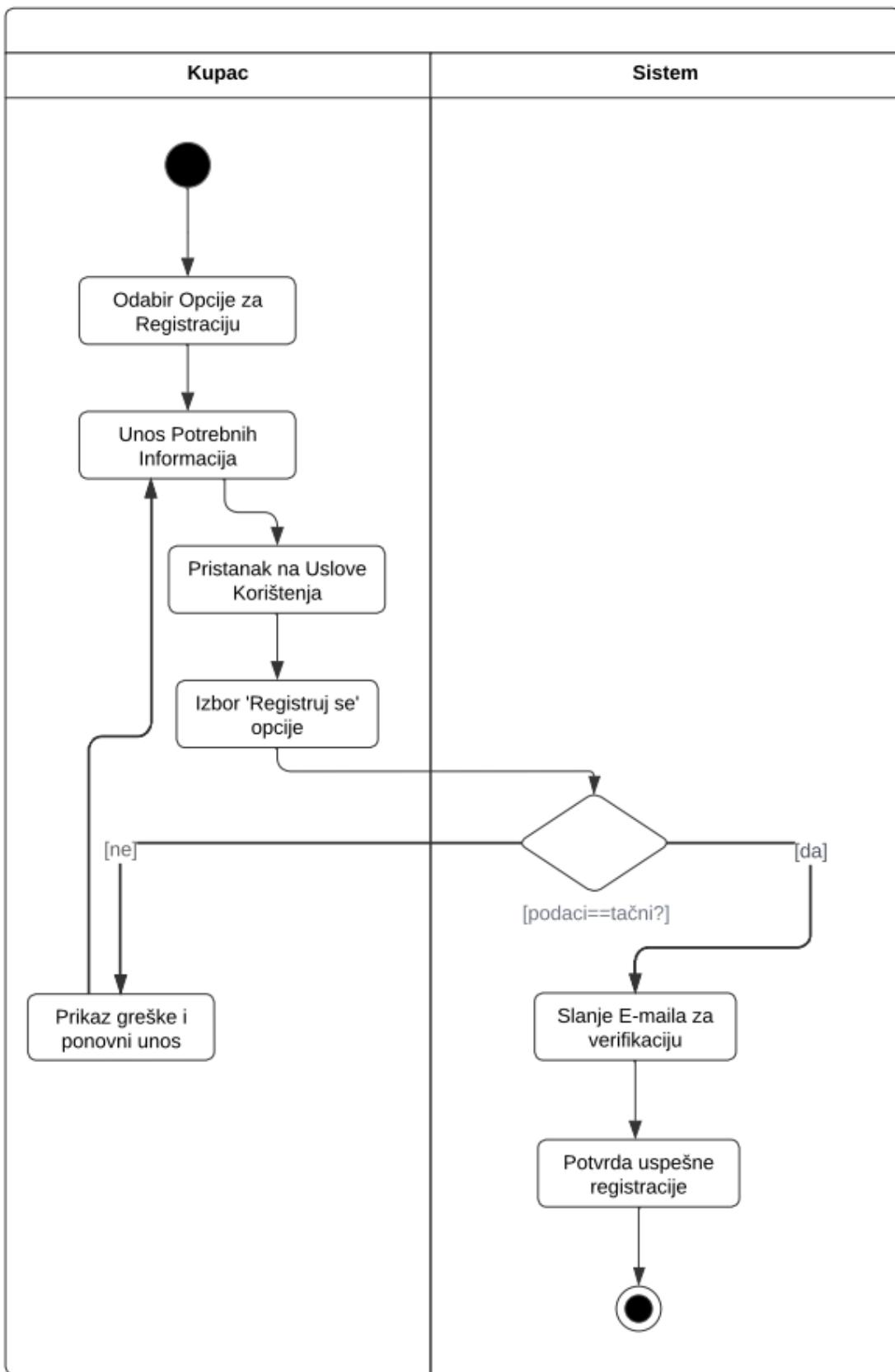
<b>Naziv slučaja upotrebe</b>	<b>Dovršavanje košarice i odabir vrste naplate</b>
<b>Opis slučaja upotrebe</b>	Pregled stavki u košarici, izvršavanje potrebnih izmjena i plaćanje kroz integrirani sistem naplate.
<b>Vezani zahtjevi</b>	-
<b>Preduslovi</b>	Odabir jednog ili više proizvoda
<b>Posljedice – uspješan završetak</b>	Narudžba spremna za isporuku
<b>Posljedice – neuspješan završetak</b>	-
<b>Primarni akteri</b>	Kupac
<b>Ostali akteri</b>	-
<b>Glavni tok</b>	Kupac mora urediti detalje proizvoda koje je odabrao i poslao u košaricu (npr. količina, začini, dodaci, pribor za jelo, itd). Nakon odabira željenih detalja, korisnik mora izabrati opciju plaćanja. Ako je opcija plaćanja prilikom preuzimanja, onda je korisnik završio sa narudžbom, ali ako korisnik odabere opciju plaćanja karticom, bit će preusmjerena na stranicu sa detaljima plaćanja, odnosno podacima o bankovnoj kartici, pri čemu će se u oba slučaja plaćanja izračunati popust, ukoliko ga je korisnik ostvario frekventnošću narudžbi.
<b>Alternative/proširenja</b>	-

### Tok događaja 6.1 – Uspješan završetak

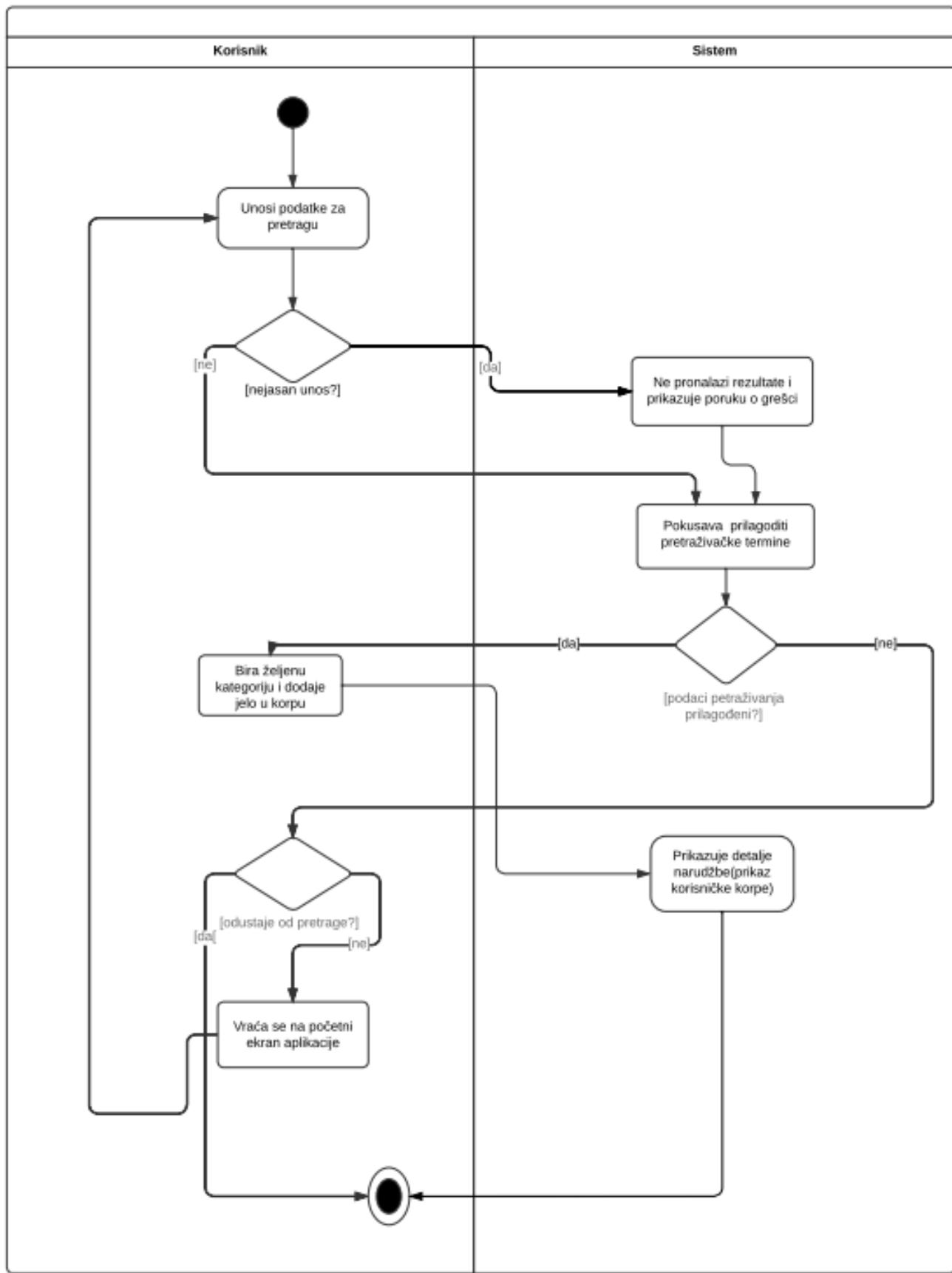
Kupac	Sistem
1. Odabir detalja proizvoda	
2. Odabir vrste plaćanja	
3. Unos podataka o bankovnoj kartici*	
	4. Slanje poruke o uspješno izvršenoj narudžbi
	5. Prosljедivanje narudžbe na uvid kuriru
6. Povratak na glavni meni aplikacije	

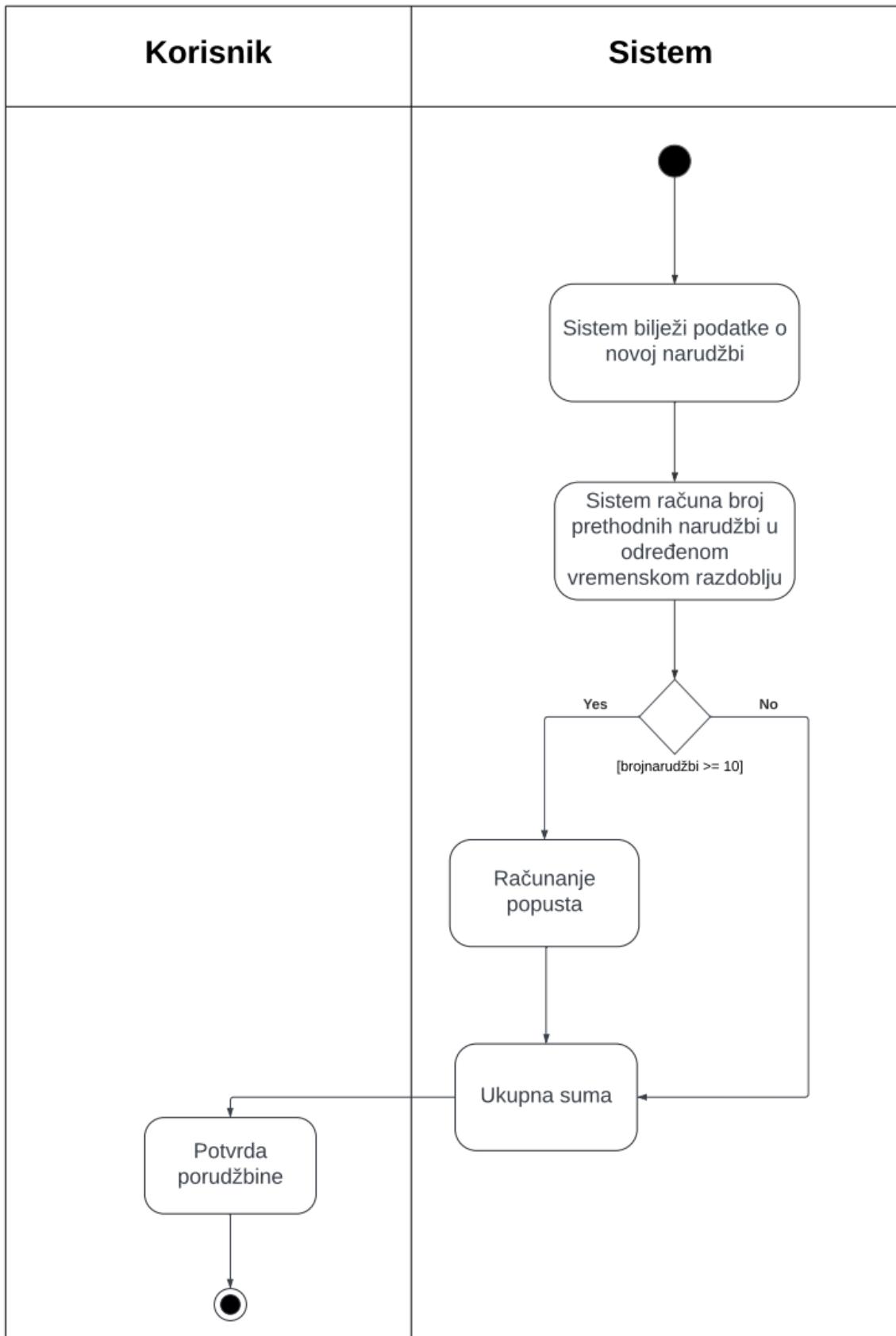
\* (opcionalno)

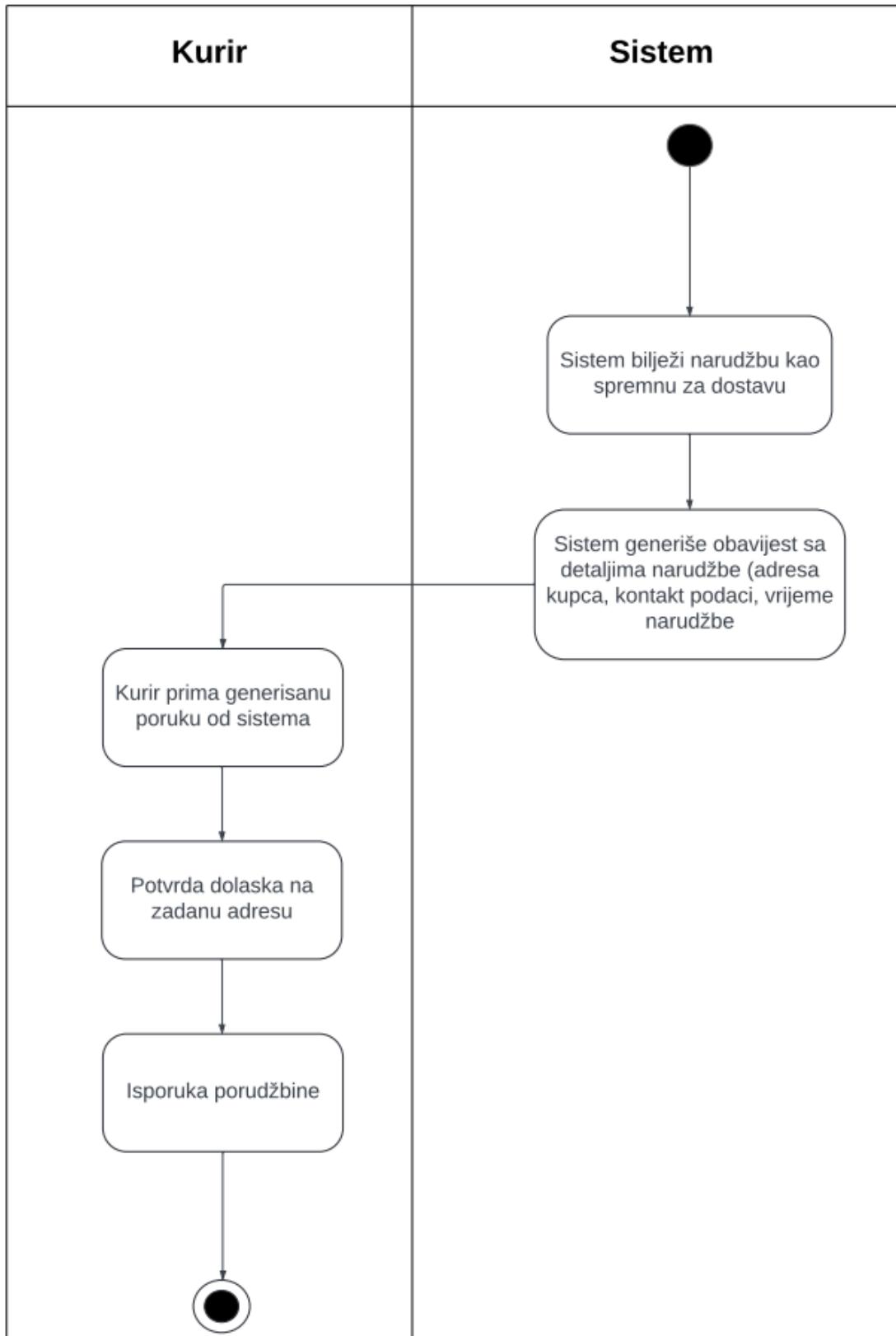
## Dijagram aktivnosti za scenario 1- Registracija ili login korisnika



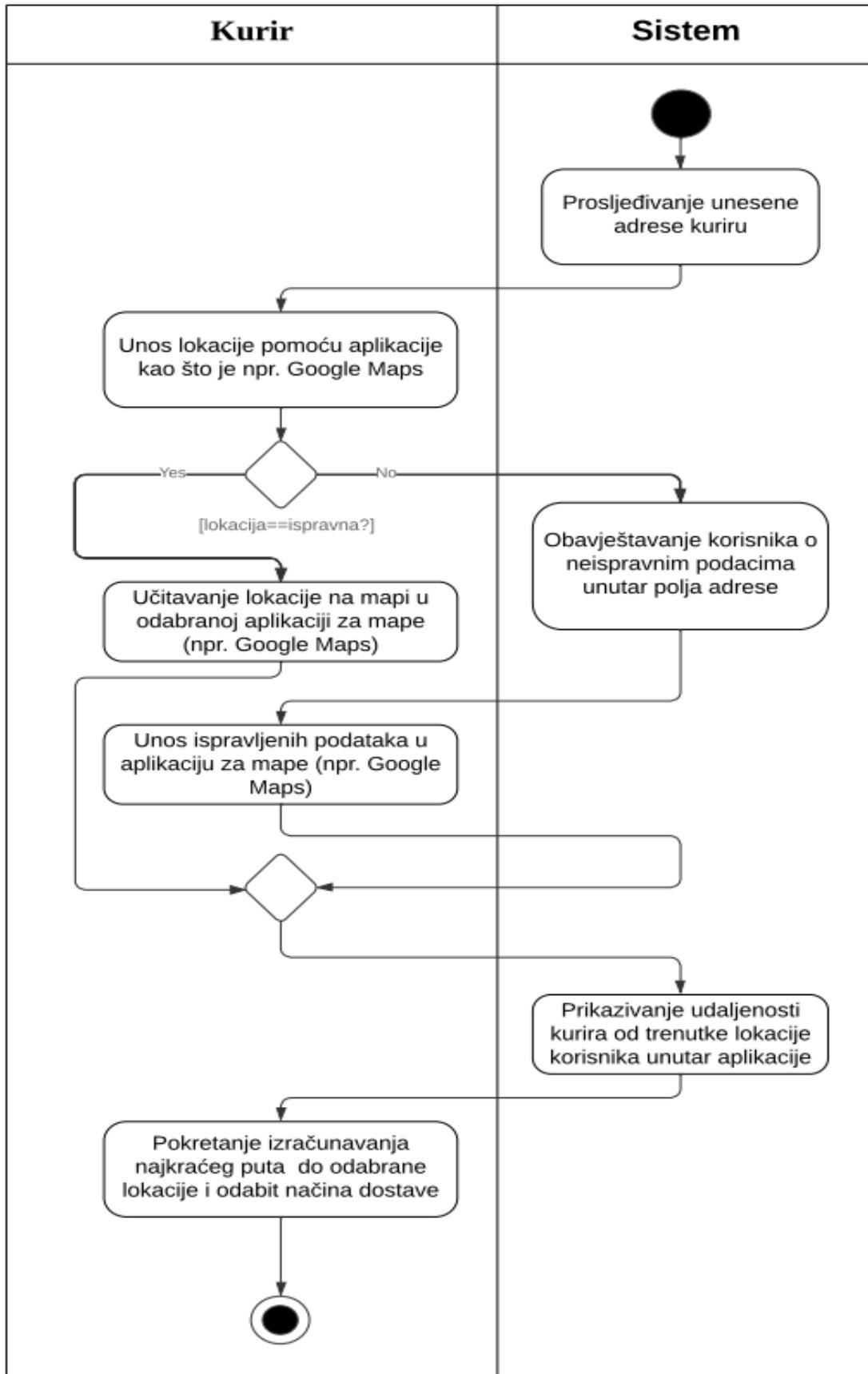
## Dijagram aktivnosti za scenario 2- Pretraživanje kategorije i detalji narudžbi



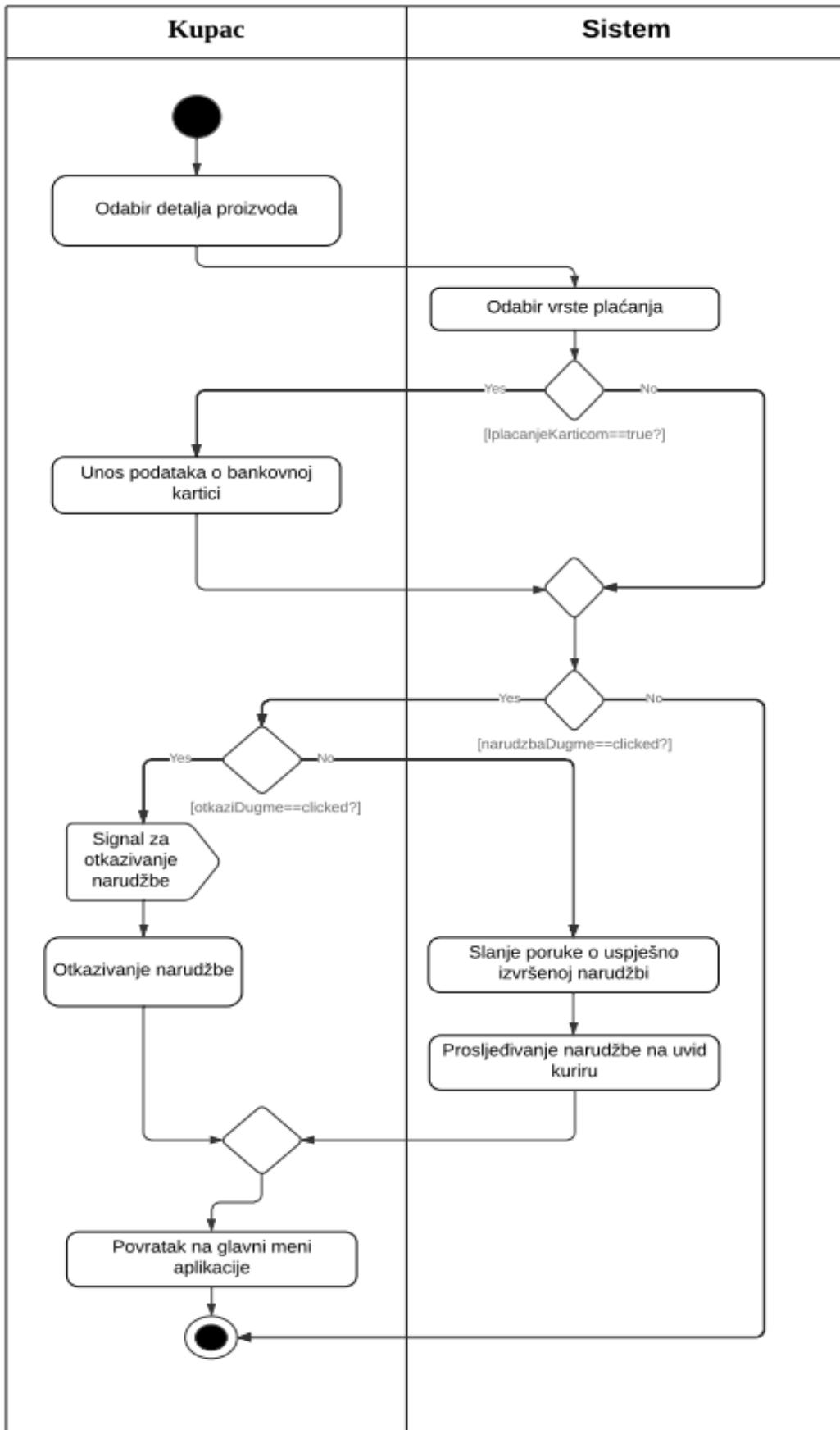
**Dijagram aktivnosti za scenario 3- Izračunavanje popusta na osnovu frekventnosti narudžbe**

**Dijagram aktivnosti za scenario 4- Slanje obavijesti o primljenoj narudžbi kuriru**

### Dijagram aktivnosti za scenario 5- Određivanje lokacije korisnika na mapi



## Dijagram aktivnosti za scenario 6 – Dovršavanje košarice i odabir vrste naplate



## Prototipi

- Login



Dobro došli!

Email ili korisničko ime

Lozinka

Novi korisnik u QuickBite? [Registrirajte se](#)

Prijavi se

- Registracija



Registracija

Email

Korisničko ime

Lozinka

Pristajete na [Uslove korištenja](#)

Registruj me

- Verifikacija



## Verifikacija email adrese

Ukoliko želite nastaviti,  
potvrdite svoju email adresu.

**Potvrdi email adresu**



- Lokacija

**QuickBite**

Vaša adresa:

Unesi adresu

**Potvrdi lokaciju**

- Početna stranica

**QuickBite**

POČETNA RESTORANI SUPERMARKETI BRZA HRANA KOŠARICA O NAMA

Pretraži

Uštedi vrijeme.  
Pozovi QuickBite.

Specijalna ponuda

- Izbor restorana

QuickBite

1 2 3

Čevabdžinica Hodžić Metropolis Perla restoran Arigato sushi bar

←

- Izbor supermaketa

### QuickBite

[1](#)    [2](#)    [3](#)



Mercator



Bingo



Konzum



Amko komerc



- Izbor brze hrane

### QuickBite

[1](#)    [2](#)    [3](#)



Burger Bar



Chipas



Žeks doner



Bash



- Izbor proizvoda

**QuickBite**



 Ćevapi 7.50 KM <a href="#">DODAJ</a>	 Pileći ražnjići 6.00 KM <a href="#">DODAJ</a>	 Tortilja piletina 6.00 KM <a href="#">DODAJ</a>	 Punjena pljeskavica 8.00 KM <a href="#">DODAJ</a>
 Pileći file na žaru 6.00 KM <a href="#">DODAJ</a>	 Sudukice juneće 7.90 KM <a href="#">DODAJ</a>	 Teleća jetra 8.50 KM <a href="#">DODAJ</a>	 Hamburger 7.50 KM <a href="#">DODAJ</a>



- Košarica

**QuickBite**

<b>Ukupan iznos narudžbe:</b> <input type="text" value="42.00"/> KM	<b>Proizvod</b>	<b>Količina</b>	<b>Cijena</b>
<b>Ostvareni popust:</b> <input type="text" value="6.00"/> KM	Ćevapi	<input type="text" value="2"/>	<input type="text" value="12.50"/> KM
<b>Ukupno:</b> <input type="text" value="36.00"/> KM	Kebab	<input type="text" value="1"/>	<input type="text" value="7.50"/> KM
	Tiramisu	<input type="text" value="4"/>	<input type="text" value="22.00"/> KM

  
**Izaberite način plaćanja:**

GOTOVINA  
 KARTICA



- O nama

**QuickBite**

**ABOUT US**



**QuickBite** je aplikacija koja povezuje korisnike, ugostiteljske jedinice i tržne centre u cilju omogućavanja kupovine, primanja i slanja različitih proizvoda unutar grada.

Bez obzira da li vam se jedu tradicionalni čevapi, savijača sa jabukom, moderni internacionalni obrok ili ste možda zaboravili kupiti brašno, QuickBite povezuje Vas sa najboljom ponudom u gradu.

**Jednostavnost** 

**Učinkovitost**

**Užitak** 

*Contact us!*

←

- Admin panel

**QuickBite-Administrator**

**korisničko ime/broj narudžbi**

mresidovic1	3
usecunovic1	5
etirak1	9
user555	12

**kuriri**

Ime1 Prezime1	Ukloni
Ime2 Prezime2	Ukloni
Ime2 Prezime2	Ukloni
Ime2 Prezime2	Ukloni

**Dodaj novog kurira**

**ugostiteljske jedinice**

KFC	Ukloni
Mrkva	Ukloni
Metropolis	Ukloni
Mrvica	Ukloni

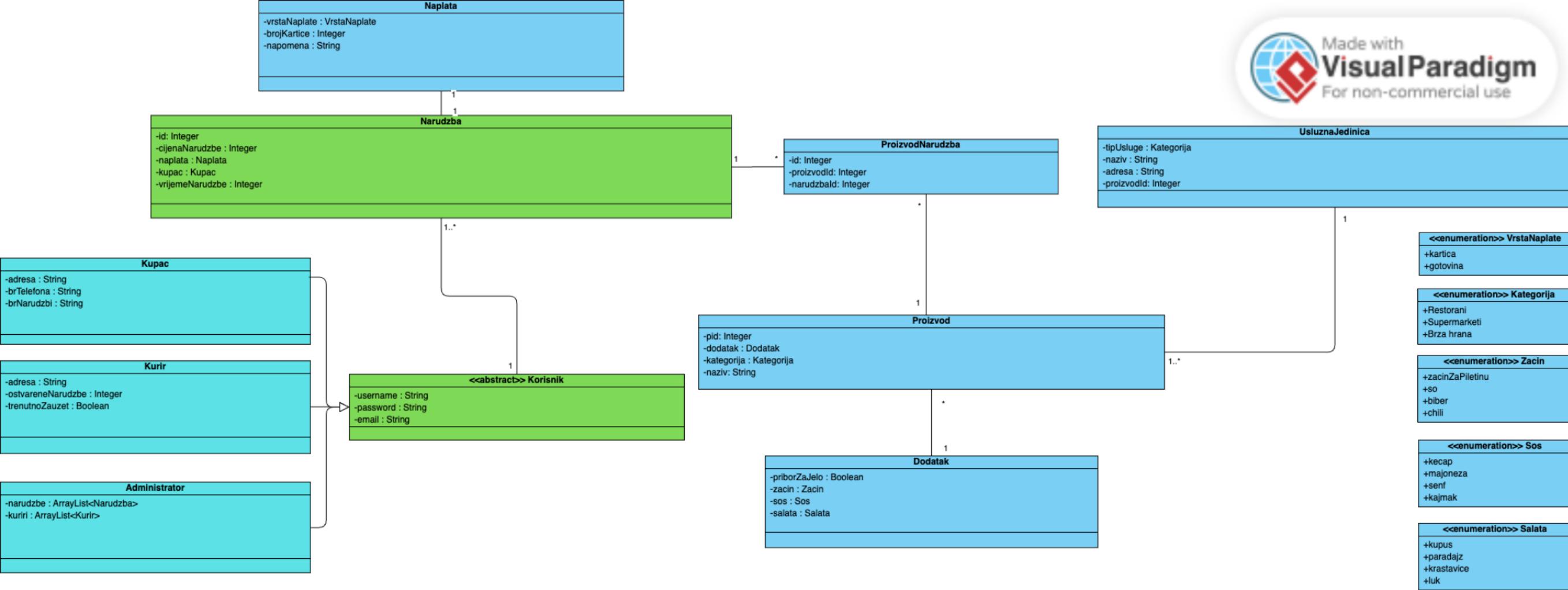
**Dodaj novo mjesto**

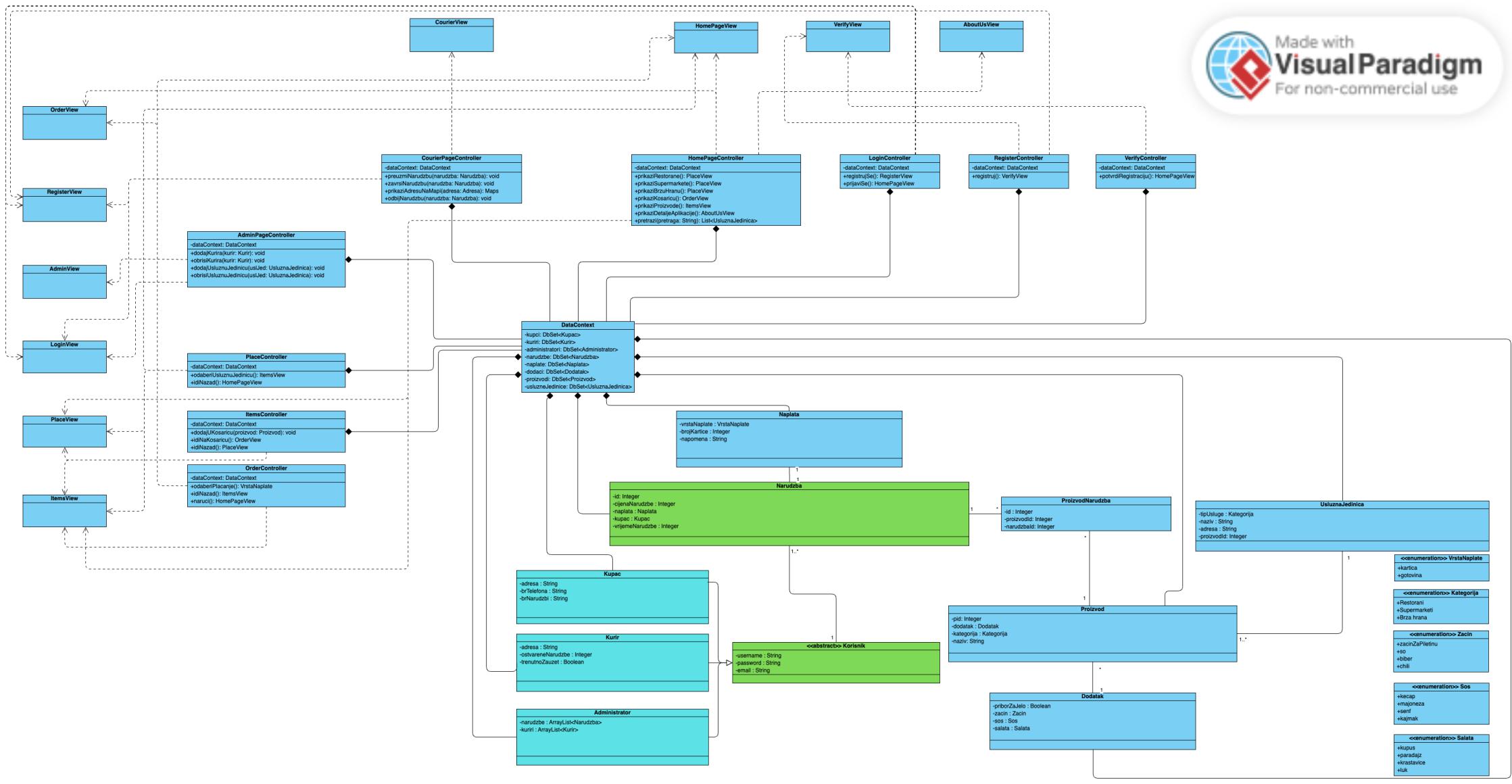
- Kurir panel

**QuickBite-Kurir**

**Aktivne narudžbe**

mresidovic1	KFC	adresa1	Preuzmi	Odbij	Dostavljeno	Prikaži na mapi
usecunovic1	KFC	adresa2	Preuzmi	Odbij	Dostavljeno	Prikaži na mapi
etirak1	Metropolis	adresa3	Preuzmi	Odbij	Dostavljeno	Prikaži na mapi
user55	Mrvica	adresa4	Preuzmi	Odbij	Dostavljeno	Prikaži na mapi





Naplata	
Id	
BrojKartice	
Napomena	
VrstaNaplate	

FK\_Narudzba\_Naplata\_NaplaId



Narudzba	
Id	
VrijemeNarudzbe	
Naplatald	
Cijena	
KupacId	

FK\_ProizvodNarudzba\_Narudzba\_NarudzbaId



ProizvodNarudzba	
Id	
ProizvodId	
Narudzbald	



Korisnik	
Id	
Adresa	
BrojNarudzbi	
BrojTelefona	
Email	
OstvareneNarudzbe	
Password	
TrenutnoZauzet	
Username	

FK\_Narudzba\_Korisnik\_KupacId



Proizvod	
Id	
Kategorija	
DodatakId	
Naziv	

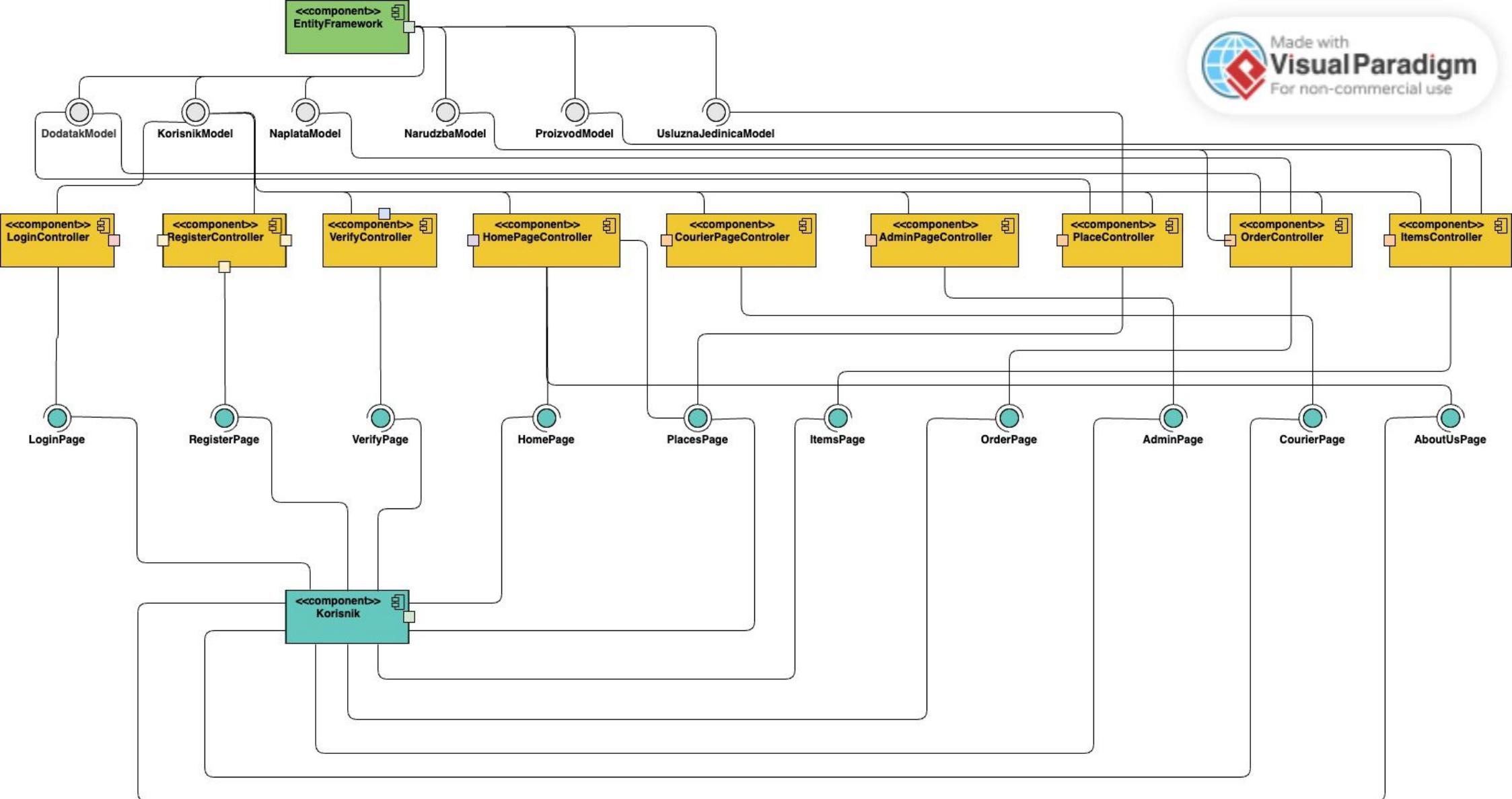


Dodatak	
Id	
PriborZajelo	
Salata	
Sos	
Zacin	

UsluznaJedinica	
Id	
TipUsluge	
Adresa	
Naziv	
ProizvodId	

FK\_UsluznaJedinica\_Proizvod\_ProizvodId







## Analiza i dizajn sistema

U nastavku je potrebno definisati sve potencijalne klase koje će se koristiti u sistemu. Za određivanje klasa koje će biti neophodne za rad sistema potrebno je koristiti specifikaciju sistema i prethodno kreirane dijagrame.

Template za jednu klasu potrebno je iskopirati onoliko puta koliko je neophodno da bi se definisale sve klase u sistemu.

### Definicija klasa u sistemu

**Naziv klase:** Korisnik

#### Funkcionalni zahtjevi u kojima klasa učestvuje:

- FZ br. 01: Registracija ili login korisnika
- FZ br. 02: Pretraživanje kategorije i detalja narudžbe
- FZ br. 03: Dovršavanje košarice i odabir vrste naplate
- FZ br. 04: Određivanje lokacije korisnika na mapi
- FZ br. 05: Slanje obavijesti o primljenoj narudžbi kuriru
- FZ br. 06: Izračunavanje popusta na osnovu frekventnosti narudžbi

#### Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
username	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
email	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
password	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
adresa	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
brojTelefona	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
brojNarudzbi	Integer	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
ostvareneNarudzbe	Integer	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
trenutnoZauzet	Boolean	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration

narudzbe	List<Narudzba>	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
kuriri	List<Kurir>	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** Narudzba

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

FZ br. 01: Pretraživanje kategorije i detalja narudžbe

FZ br. 02: Dovršavanje košarice i odabir vrste naplate

FZ br. 03: Ostvarivanje popusta na osnovu frekventnosti narudžbi

FZ br. 04: Slanje obavijesti o primljenoj narudžbi kuriru

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
proizvodi	List<Proizvod>	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
cijenaNarudzbe	Integer	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
naplata	Naplata	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
kupac	Kupac	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
vrijemeNarudzbe	Integer	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase (enum):** Kategorija

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

FZ br. 01: Pretraživanje kategorije i detalja narudžbe

FZ br. 02: Dovršavanje košarice i odabir vrste naplate

FZ br. 03: Slanje obavijesti o primljenoj narudžbi kuriru

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
naziv	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** Proizvod

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

FZ br. 01: Pretraživanje kategorije i detalja narudžbe

FZ br. 02: Dovršavanje košarice i odabir vrste naplate

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
naziv	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
kolicina	Integer	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
kategorija	Kategorija	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
cijena	Integer	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
dodatak	Dodatak	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** Dodatak

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

FZ br. 01: Pretraživanje kategorije i detalja narudžbe

FZ br. 02: Dovršavanje košarice i odabir vrste naplate

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
priborZaJelo	Boolean	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

zacin	Zacin	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
sos	Sos	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
salata	Salata	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase (enum):** Zacin

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

FZ br. 01: Pretraživanje kategorije i detalja narudžbe

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
naziv	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase (enum):** Sos

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

FZ br. 01: Pretraživanje kategorije i detalja narudžbe

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
naziv	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase (enum):** Salata

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

FZ br. 01: Pretraživanje kategorije i detalja narudžbe

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
naziv	String	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** Naplata

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

FZ br. 01: Dovršavanje košarice i odabir vrste naplate

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
vrstaNaplate	VrstaNaplate	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
brojKartice	Integer	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
napomena	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** VrstaNaplate

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

FZ br. 01: Dovršavanje košarice i odabir vrste naplate

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
naziv	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase:** UsluznaJedinica

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

FZ br. 01: Slanje obavijesti o primljenoj narudžbi kuriru

FZ br. 02: Pretraživanje kategorije i detalja narudžbe

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
tipUsluge	Djelatnost	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
naziv	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
adresa	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
proizvodi	List<Proizvod>	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

**Naziv klase (enum):** Djelatnost

**Funkcionalni zahtjevi u kojima klasa učestvuje:**

FZ br. 01: Pretraživanje kategorije i detalja narudžbe

**Atributi koje klasa posjeduje:**

Naziv atributa	Tip varijable	Dodatne napomene
naziv	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

## **1. S-Single Responsibility Principle (Princip pojedinačne odgovornosti)**

Ovaj princip je ispoštovan jer sve klase imaju pojedinačne odgovornosti, odnosno svaki posao se veže za jednu klasu.

Ako posmatramo klasu Korisnik, možemo primijetiti da ona posjeduje samo one atribute koje svaki korisnik mora imati, te će klasa posjedovati samo one metode koje će moći vraćati i modifikovati te atribute.

Očito je da ova klasa poznaje samo jednu stvar (korisnika) jer sadrži samo atribute i metode koji su usko vezani uz samog korisnika.

Izmjena ove klase može izazvati samo potreba za promjenom Korisnika.

## **2. O-Open/Closed Principle (Princip otvorenosti/zatvorenosti):**

Ovaj princip je ispostovan u primjeru klase Korisnik i klase koje su naslijedene iz nje. Klasu "Kupac" mogli bismo dodati bez da modifikujemo postojeći kod.

Klase su otvorene za proširenje, ali zatvorene za modifikaciju. Također, ako bismo željeli dodati novu vrstu korisnika poput "Korisnik", trebalo bi moći dodati novu klasu "UposlenikUsluznogObjekta" bez mijenjanja postojećeg koda u klasama "Korisnik" ili "Narudzba".

## **3. L-Liskov Substitution Principle (Princip zamjene Liskov):**

Svaka podklasa trebalo bi da bude zamjenjiva s njihovom baznom klasom bez narušavanja ispravnosti programa. To znači da bi "Kurir", "Administrator" i "Korisnik" trebali biti zamjenjivi gdje god se očekuje instanca klase "Korisnik".

Za primjer možemo da uzmemo situaciju u kojoj, prilikom izmjene username-a ili nekih drugih korisničkih podataka, oni se mogu promijeniti za instance svih naslijedjenih klasa, bez narušavanja ispravnosti programa i ponavljanja koda. Naravno, ovo važi uz prepostavku da je takva funkcionalnost vezana isključivo za baznu klasu.

#### **4. I-Interface Segregation Principle (Princip izoliranja interfejsa)**

U našem primjeru imamo 3 vrste korisnika: kupac, kurir, administrator, pri čemu svaki od njih obavlja svoje funkcije. Na primjer, kupac ne može da se miješa u promjenu cijena bilo kojih proizvoda, pa to zahtjeva posebne interfejse, čime je ovaj princip zadovoljen.

#### **5. Dependency Inversion Principle (Princip inverzije ovisnosti)**

Klasa Korisnik bi trebalo da posjeduje interfejs u kom će postojati metode koje su zajedničke za klase Administrator, Kupac i Kurir pa je time ispoštovan ovaj princip. Na primjer, klasa Korisnik sadrži metodu izvrsi\_placanje, a klase Administrator, Kupac i Kurir zatim nasljeđuju Korisnik klasu i implementiraju tu metodu na svoj način.

## KREACIJSKI PATERNI

Potreba za dodavanjem kreacijskih paterna u sistem se prvo bitno ogleda u cilju povećanja nivoa principa enkapsuliranja, kako bi se dovelo do što većeg odvajanja korištenja sistema i u ovom slučaju kreiranja klase koje sistemu trebaju. Cilj ovog postupka je sličan kao i kod mnogih drugih paterna, a to jeste smanjenje mogućnosti greške, te ako se greška i desi, pokušati ćemo minimizirati njen utjecaj. Želimo da greška uvijek utiče na aspekte sistema, a ne na čitav sistem.

U našem slučaju kreacijski paterni koje ćemo primjeniti su:

- Singleton patern
- Builder patern

### 1. Singleton pattern

-ovaj patern osigurava da postoji samo jedna instanca određene klase koja je dostupna globalno. U softver inženjeringu, singleton pattern je dizajn pattern koji ograničava instanciranje klase na jednu jedinu instancu. Ovo je korisno kada je potreban tačno jedan objekt za koordinaciju akcija u cijelom sistemu, i koristi se tamo gdje je potrebna samo jedna instanca klase za kontrolu radnje tokom cijelog izvršenja. Singleton klasa ne bi trebala imati više instanci ni u kom slučaju i po svaku cijenu. Singleton patern možemo primjeniti na obavijesti koja se prosljedjuje svim korisnicima koji uspješno izvrše narudžbu, tj. kako korisnik vidi obavijest, nije potrebno praviti zasebnu instancu za svakog korisnika.

### 2. Builder pattern

-koristi se za konstruisanje složenih objekata korak po korak. **Builder pattern** se koristi u slučaju kreiranja kompleksnih objekata gdje nerijetko dolazi do pojave velikog broja parametara (konstruktor pretežno) od kojih vecina može biti neiskoristena. Također, može biti iskoristen u situacijama kada se razliciti objekti kreiraju na slike nacine. Koristan je kada objekat ima mnogo opcionalnih parametara ili složene konfiguracije. Primjer builder paterna u našoj aplikaciji mogao bi se ogledati u kreiranju kompleksnog objekta narudžbe koja može imati različite dodatke, napomene i posebne zahtjeve itd.

Ostali paterni bi mogli biti iskorišteni :

### 1. PrototypePatern:

- **Primjena:** Prototype patern koristi se za kloniranje postojećih objekata umjestokreiranja novih od nule. Ovaj patern može biti koristan kada imamo objekte sa složenom konfiguracijom koje često ponavljamo.
- **Primjer:** Kloniranje prethodnih narudžbi korisnika kako bi se ubrzalo kreiranje nove porudžbine sa sličnim stavkama.
- Ovaj patern nismo implementirali jer nam je zahtijevao viši nivo kompleksnosti od našeg ciljnog nivoa.

## **2.Factory Method patern**

Uloga **Factory Method paterna** je da omogući kreiranje objekata na način da podklase odluče koju klasu instancirati. Različite podklase mogu na različite načine implementirati interfejs. Factory Method instancira odgovarajuću podklasu(izvedenu klasu) preko posebne metode na osnovu informacije od strane klijenta ili na osnovu tekućeg stanja.

U nasem slučaju, mozemo iskoristiti **Factory Method** prilikom kupovine, gdje ukoliko vrsimo kupovinu proizvoda, metoda se izvrsava sa uracunatim popustom, dok sa druge strane, metoda se izvrsava na drugi nacin (nema takvu funkcionalnost) prilikom kupovine bez popusta, jer niko nije u mogućnosti ostvariti popust bez prethodnih narudžbi.

3.

## **Abstract factory patern**

Kupovina je omogućena registrovanim i neregistrovanim korisnicima. Međutim, samo registrovani korisnici mogu izvršiti i kupovinu uz popust .Možemo napraviti apstraktnu factory klasu iz koje ćemo naslijediti dvije factory klase, za registrovane i neregistrovane korisnike. Shodno tome kakav korisnik je u pitanju, overrideana metoda koja kreira Kupovinu bi u jednom slučaju vraćala instancu Kupovine sa popustom, a u drugom Kupovina bez popusta(Obe ove klase su naslijedene iz Kupovina).

Ako bismo u budućnosti željeli omogućiti specifičan način kupovine nekoj trećoj kategoriji korisnika, to bi nam sada bilo znatno olakšano. Naravno, ako bismo još neki segment sem kupovine smatrali pogodnim za „personalizaciju“ po ovim kategorijama korisnika, to bismo mogli riješiti tako što dodamo još jedan sistem sličan već postojećem koji se tiče kupovine i još jednu apstraktnu metodu unutar factory klase koja bi kreirala instance takvih objekata.

## PATERNI PONAŠANJA

### 1) Strategy patern

Uloga **strategy paterna** jeste da izdvaja algoritam iz matične klase i uključuje ga u posebne klase. Pogodan je kada postoje različiti primjenjivi algoritmi (strategije) za neki problem. **Strategy patern** omogućava klijentu izbor jednog od algoritma iz familije algoritama za korištenje. Algoritmi su neovisni od klijenata koji ih koriste. Podržava *open-closed* princip.

U nasem slučaju, **strategy patern** bi se mogao iskoristiti prilikom odabira nacina sortiranja izbora uslužne jedinice (restorana, supermarketa ili fast-fooda), gdje bi algoritam djelovao na osnovu strategije, odnosno na osnovu izbora korisnika.

### 2) State patern

State Pattern je dinamička verzija Strategy paterna. Objekat mijenja način ponašanja na osnovu trenutnog stanja. Postiže se promjenom podklase unutar hijerarhije klasa.

Status dostupnosti restorana može biti modeliran pomoću State obrasca, s različitim stanjima kao što su "Otvoren", "Zatvoren" ili "Na pauzi". Ovo stanje može se dinamički mijenjati ovisno o radnom vremenu restorana ili drugim faktorima, a aplikacija može prilagoditi prikaz restorana ovisno o trenutnom stanju.

State patern možemo koristiti i u opisu statusa narudžbe korisniku.

### 3) TemplateMethod patern

Omogućava **izdvajanje određenih koraka algoritma u odvojene podklase**. **Struktura algoritma se ne mijenja** - mali dijelovi operacija se izdvajaju i ti se dijelovi mogu implementirati različito.

U nasem slučaju, templateMethod patern bi se mogao iskoristiti prilikom kupovine, gdje u zavisnosti da li je korisnik prijavljen ili ne, prilikom kupovine narudžbe uračunavamo popust. Tok narudžbe i druge mogućnosti će ostati jednake, jedina razlika će se ogledati u popustu, zavisno od podklase korisnika.

Osnovna klasa korisnika ima implementirane *default* verzije metoda, a *override* je za popust (koji u općem slučaju vraća 0) napravljen samo u klasi koja omogućava popust pri kupovini.

### 4) Observer patern

Uloga **Observer paterna** je da **uspostavi relaciju između objekata tako kada jedan objekat promijeni stanje drugi zainteresirani objekti se obavještavaju**.

Kada korisnik napravi narudžbu, sistem može koristiti Observer obrazac kako bi obavijestio korisnika o promjenama statusa njihove narudžbe. Na primjer, korisnik može biti obaviješten putem e-maila,

SMS-a ili notifikacije u aplikaciji kada se status narudžbe promijeni (npr. "Priprema", "Dostava", "Isporučeno").

## 5) Iterator patern

**Iterator patern** omogućava sekvenčijalni pristup elementima kolekcije **bez poznavanja kako je kolekcija strukturirana**.

U nasem slučaju, **iterator patern** bi mogli iskoristiti prilikom korisnikove „posjete“ web aplikaciji. Korisnik bi mogao da bira nacin listanja artikala, bilo kroz „shuffle“ mode (slučajni redoslijed artikala), „cijena“ (ide kroz articke prema nekom algoritmu koji uzima u obzir cijenu) itd.

## 6) Mediator patern

**Mediator patern** enkapsulira protokol za komunikaciju medju objektima dozvoljavajući da objekti komuniciraju bez medjusobnog poznavanja interne strukture objekta.

U nasem slučaju, mediator patern bi mogli iskoristiti ukoliko bi **upravljali narudžbama**. Mediator može poslužiti kao centralizirani posrednik za upravljanje narudžbama. Kada korisnik izvrši narudžbu, različiti dijelovi sistema kao što su administrator, kuriri i sistem naplate mogu komunicirati putem medijatora radi obrade narudžbe.

## 7)

**Memento patern** omogućava da spasimo i vratimo prijašnje stanje objekta bez otkrivanja detalja njegove implementacije.

U našem slučaju, **memento patern** bi mogli iskoristiti prilikom kupovine karte, gdje bi korisnik bio u mogućnosti da se pritiskom na *back* vrati na prethodno odabranu količinu ili vrstu jela ili neke druge promjene u narudžbi.

## **Adapter dizajn patern**

Adapter patern dizajn služi da omogući da se interfejs već postojeće klase korsiti kao drugi interfejs, ali često omogućava i to da postojeće klase rade sa drugim bez izmjene njihovog izvornog koda. Osnovna namjena je da omogući širu upotrebu već postojećih klasa. Kreira novu adapter klasu koja povezuje originalnu klasu i željeni interfejs. Na taj način se dobija željena funkcionalnost bez ikakvih izmjena na originalnoj klasi.

Ovaj patern se u našoj aplikaciji može pronaći prilikom izvršavanja naplate sa aplikacijom. Korisnik ima dvije mogućnosti izbora naplate, pouzećem ili preko svog bankovnog računa. Time dajemo aplikaciji dodatnu komunikaciju između ta dva sistema tako da prilagodi i prevodi zahtjeve i odgovore između njih.

Svaki adapter bi implementirao isti interfejs plaćanja koji očekuje ova aplikacija, ali bi interno koristio specifične interfejse za komunikaciju sa različitim procesorima plaćanja. Na ovaj način, aplikacija može da koristi isti kod za obradu plaćanja, bez obzira na to koju metodu naplate korisnik odabere.

## **Façade dizajn patern**

Ovo je strukturalni obrazac dizajna koji obezbjeđuje pojednostavljeni interfejs za skup interfejsa u podsistemu čineći ga lakšim za korišćenje. Olakšava klijentima da koriste funkcionalnosti aplikacije bez potrebe za poznavanjem unutrašnjeg složenog sistema. Na primjer, u našem slučaju složeni interfejs za naručivanje i dostavu bi obuhvatao različite komponente poput upravljanja narudžbama, plaćanja, komunikacije sa korisnicima, upravljanje dostavom, itd. Ovaj patern bi omogućio korisnicima interakciju sa ovakvim sistemom kroz jednostavan i intuitivan interfejs. Dakle, umjesto što bi korisnici morali da pozivaju različite servise ili klase za kreiranje narudžbi, praćenje statusa istih ili komunikaciju s korisnicima, fasada bi pružila jednostavne metode koje obavljaju ove zadatke na osnovu korisničkog zahtjeva. Ovakav patern je koristan iz razloga što skriva kompleksnost podistema od klijenata. Klijenti koriste jednostavan interfejs, a pri tome ne poznaju interne detalje implementacije.

## **Decorator dizajn patern**

Služi da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Takođe, omogućava i da se funkcionalnost podijeli između klasa u zavisnosti od područja interesa. Ovaj patern nasljeđuje originalnu klasu, ali se ne oslanja na nasljeđivanje prilikom dodavanja novih atributa i objekata, već kreira nove. Na primjer, u našem slučaju ovaj patern bismo mogli iskoristiti ukoliko želimo da korisniku pružimo mogućnost promjene njegovih podataka, npr. promjena emaila-a ili drugih podataka.

## **Proxy design patern**

U našoj aplikaciji ovaj patern bi mogao da se odnosi na samo ograničenje pristupa. Može se koristiti za implementaciju ovog koncepta tako što će djelovati kao posrednik između korisnika aplikacije i resursa, provjeravajući i primjenjujući određena pravila ili ograničenja prije nego što omogući pristup resursima. Primjer može biti autentifikacija i autorizacija. Što se autentifikacije tiče, ovaj patern može da provjeri identitete korisnika prije nego što im omogući pristup određenim funkcionalnostima ili podacima. Nakon autentifikacije, ovaj patern može odobriti ili odbiti pristup određenim resursima ili funkcionalnostima na temelju njihovih privilegija. Proxy može da provjeri je li korisnik administrator ili obični korisnik prije nego što im dopusti pristup administratorskim funkcijama.

## **Bridge design patern**

Kada bi naša aplikacija podržavala različite načine prikaza informacija, na primjer na web stranici, mobilnoj aplikaciji, itd. Svaki od ovih načina prikaza bi zahtijevao različitu implementaciju, ali bi dijelio zajedničke apstrakcije poput liste restorana, prodavnica, ili prikaza detalja restorana i mogućnosti naručivanja hrane, odnosno namirnica. Obzirom da pravimo samo web aplikaciju ovaj patern neće postojati u našem primjeru, zbog kompleksnosti implementacije.

## **Flyweight patern**

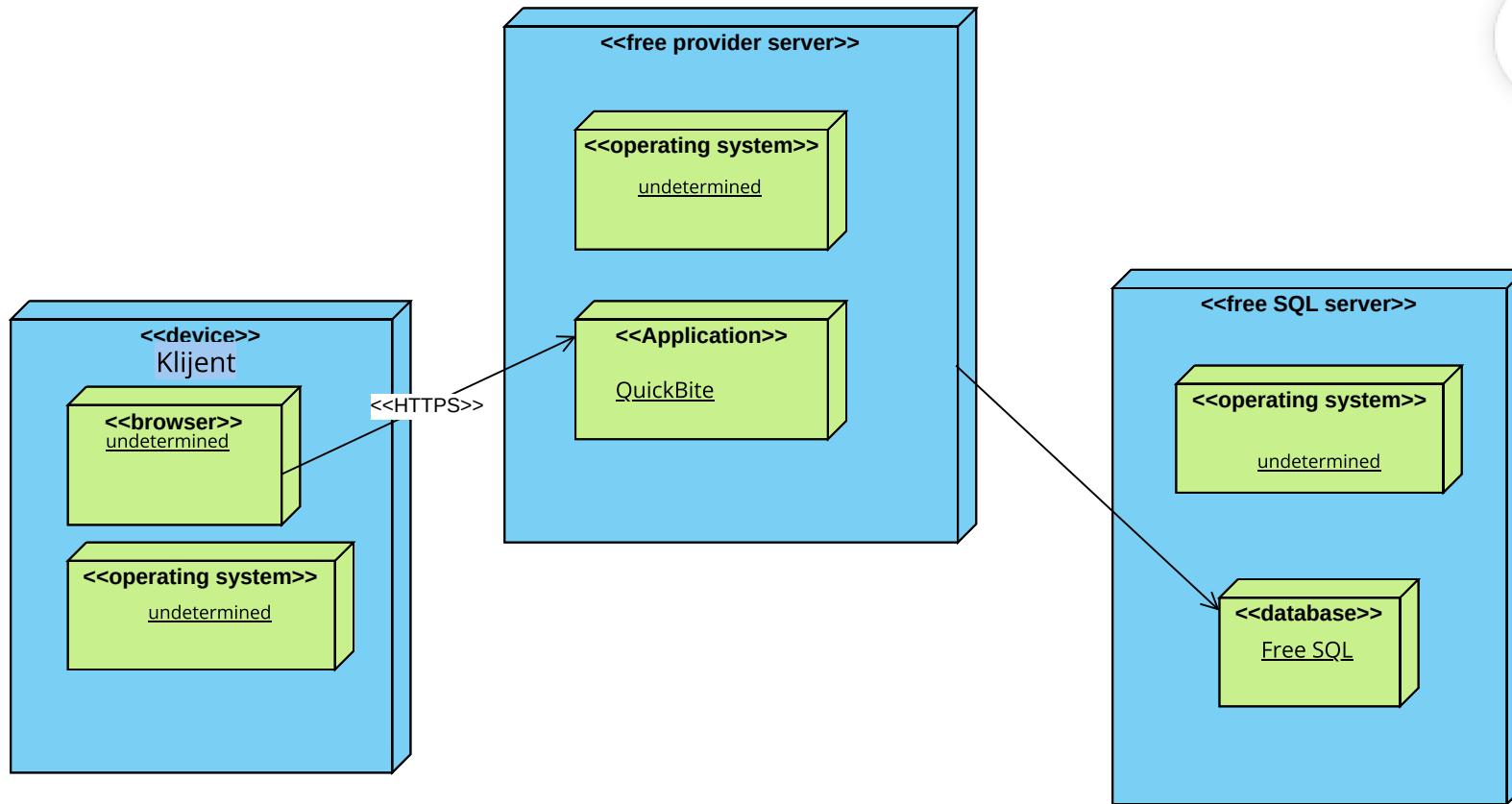
Strukturalni patern koji bismo mogli iskoristiti kada bi naši modeli imali osobinu koja se naknadno postavlja, te joj se dodjeljuje neka defaultna vrijednost koja se nalazi u samom sistemu radi smanjenog korištenja memorije. Flyweight obrazac može biti posebno koristan za optimizaciju upravljanja stavkama menija, detaljima restorana, korisničkim profilima i drugim ponavljajućim podacima. To može dovesti do efikasnije upotrebe memorije i boljih performansi, posebno u scenarijima gdje je potrebno upravljati velikim brojem sličnih objekata istovremeno. Spomenuta osobina nije značajna samo za razvoj toka aplikacije. Trenutno dizajnirani sistem

kojeg imamo, koristi podatke koji su zaista potrebni aplikaciji, te iz tog razloga ne vidimo neku potrebu za ovim paternom. Međutim, mogli bismo ovaj patern iskoristiti za default popust prilikom kreiranja nove narudžbe starog korisnika.

## **Composite patern**

Ovaj patern služi da bismo postigli hijerarhiju objekata, tako što ćemo kreirati strukturu stabla pomoću klase, u kojoj se kompozicije individualnih objekata, kao i oni sami ravnopravno tretiraju, odnosno moguće je pozvati zajedničku metodu nad svim klasama. Korištenje kompozitnog obrasca omogućuje nam da jednostavno manipuliramo i obrađujemo cijele grupe objekata na isti način kao i pojedinačne objekte, što olakšava upravljanje složenim strukturama podataka. Osim toga, omogućuje nam fleksibilnost u dodavanju ili uklanjanju novih elemenata u hijerarhiji bez potrebe za promjenom koda koji se koristi za obradu tih elemenata. Kao ideju za ovaj patern možemo omogućiti da se korisniku pokaže konačni račun narudžbe, te se cijena računa na osnovu popusta(ako postoji), dalje obrađuje na osnovu odabira načina plaćanja. Potrebno je naglasiti da bi se onda morale praviti posebne klase vrste plaćanja koje bi se na različit način implementirale.

Jednostavniji način implementacije ovog paterna jeste taj da smo omogućili vrlo lako dodavanje dodataka narudžbi u vidu sosa, priloga, itd. Na primjer, začinima vlo lako možemo dodati neki kojeg nemamo, a da ne mijenjamo ostatak koda.



# Narudžba jela

