

# Objektno oblikovanje

Akadska godina 2017/2018.

*IOrder*

Dokumentacija, Rev. 2

Tim: *IOrder*

Vođa tima: *Shimun Matić*

Članovi: *Borna Grgić, Martin Mihalić, Matija Vukić*

Datum predaje: 4<sup>th</sup> veljača 2018.

## Sadržaj

Sadržaj .....	2
1. Opis projekta .....	3
2. Funkcionalni zahtjevi .....	4
2.1. Korisnički zahtjevi .....	4
2.2. Zahtjevi administratora uslužnih objekata .....	4
2.3. Zahtjevi zaposlenika uslužnog objekta .....	4
2.4. Zahtjevi administratora sustava IOrder .....	4
2.1 Opis use caseova korisnika .....	5
1.1. 2.2 Opis use caseova IOrder administratora .....	11
1.2. 2.3. Opis use caseova administratora uslužnog objekta .....	12
1.3. 2.4. Opis use caseova zaposlenika uslužnog objekta .....	15
3. Ostali zahtjevi .....	17
4. Backend API .....	18
5. Pregled ekrana .....	24
5.1 Mobilna aplikacija Android .....	24
5.2 Desktop aplikacija .....	27
5.3 Web aplikacija .....	29
6. Arhitektura i dizajn sustava .....	33
6.1. Arhitektura sustava .....	33
6.2. Arhitektura backend poslužitelja .....	34
Implementacija perzistencije .....	35
6.3. Slojevi .....	37
Repository .....	37
Converter .....	40
Service .....	40
Controller .....	41
Business model .....	42
6.4. Sigurnost .....	42
6.5. Obavijesti .....	43
6.6. Komunikacijski servis .....	44
6.7. Baza podataka .....	45
Tablice .....	45
6.8. Arhitektura Desktop aplikacije .....	47
Opis modela razreda .....	47
Opis interakcije sa sustavom .....	47
6.9. Arhitektura Web aplikacije .....	48
6.10. Arhitektura Android aplikacije .....	49
Opis modela razreda .....	49
Opis modula .....	50
Opis interakcije sa sustavom .....	51
7. Korištene tehnologije i alati .....	52

## 1. Opis projekta

Brzim rastom populacije javila se potreba za digitalizacijom različitih procesa unutar sektora uslužnih djelatnosti koji uzimaju visoki udio vremena kako poslodavcima, tako i korisnicima istih. Kako bi se ti određeni procesi automatizirali, razvijen je sustav *IOrder* na jednoj apstraktnoj razini u kojoj može biti iskorišten u gotovo svakoj uslužnoj djelatnosti današnjice. Ukoliko ste ikad dugo čekali da vas konobar posluži u kafiću ili ste u nekom velikom trgovačkom centru zalutali na krivi odjel i morate za sobom vući puna kolica proizvoda kako biste došli do određenog proizvoda, ovaj sustav će vam uvelike olakšati probleme.

Sustav korisnicima pruža sučelje preko mobitela (Android) kojim oni mogu skenirati QR kod i putem njega identificirati uslužni objekt te njegov odjel (ukoliko postoji). Automatski se prikazuje lista kategorija i proizvoda u meniju uslužnog objekta (i određenog odjela ukoliko postoji). Korisnici zatim mogu naručivati proizvode i poslati narudžbu koju sustav procesuirat će obavještava zaposlenike o novoj tekućoj narudžbi.

Sustav administratorima daje mogućnosti prijave svojeg uslužnog objekta i definiranja njegove strukture kao i kategorija i proizvoda koje on nudi. Uslužni objekt može imati vlastito skladište koje se evidentira od strane administratora kako bi uvijek bila dostupna većina proizvoda. Administratori sustava zaduženi su i za kontaktiranje dostupnih dobavljača.

Zaposlenici unutar sustava imaju zadaću procesirati narudžbe korisnika putem vlastite aplikacije na Webu. Zaduženi su za evidenciju aktivnih narudžbi, kao i za izdavanje računa već procesuiranim narudžbama.

## 2. Funkcionalni zahtjevi

### Dionici:

- Korisnici
  - Gosti/mušterije uslužnih objekata
- Administratori uslužnih objekata
  - Vlasnici uslužnih objekata
- Zaposlenici uslužnih objekata
- IOrder administratori
  - Osobe koje odobravaju usluge vlasnicima objekata

### 2.1. *Korisnički zahtjevi*

- Korisnik se može registrirati u sustav *IOrder*.
- Korisnik može skenirati QR kodove dostupne na lokacijama uslužnog objekta.
- Korisnik nakon skeniranja ima uvid u sve kategorije i proizvode uslužnog objekta.
- Korisnik može nesmetano odabrati željene proizvode i naručiti ih putem narudžbe.
- Korisnik može pregledavati povijest narudžbi za uslužni objekt u kojem se trenutno nalazi.
- Korisnik ima uvid u cijene proizvoda i račun za čitav asortiman narudžbe.

### 2.2. *Zahtjevi administratora uslužnih objekata*

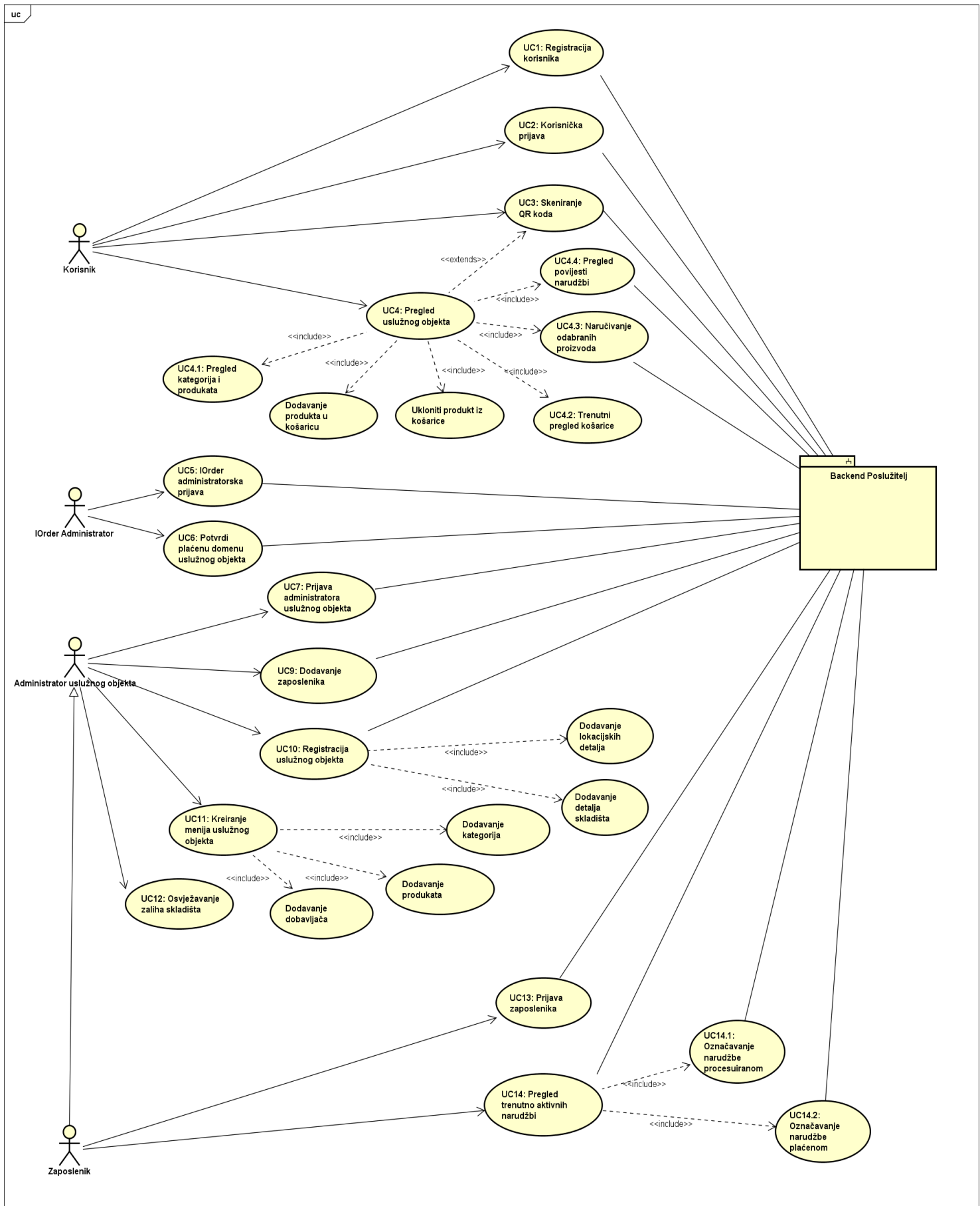
- Administrator može registrirati svoj uslužni objekt u sustav *IOrder*.
- Administrator može definirati skladište za svoj uslužni objekt.
- Administrator može definirati vlastite odjele/lokacije unutar uslužnog objekta.
- Administrator može dodavati kategorije i proizvode u svoj uslužni objekt.
- Administrator može imati pregled trenutnog stanja skladišta i osvježiti ga slanjem zahtjeva dobavljaču.
- Administrator može registrirati vlastitog zaposlenika u sustavu.

### 2.3. *Zahtjevi zaposlenika uslužnog objekta*

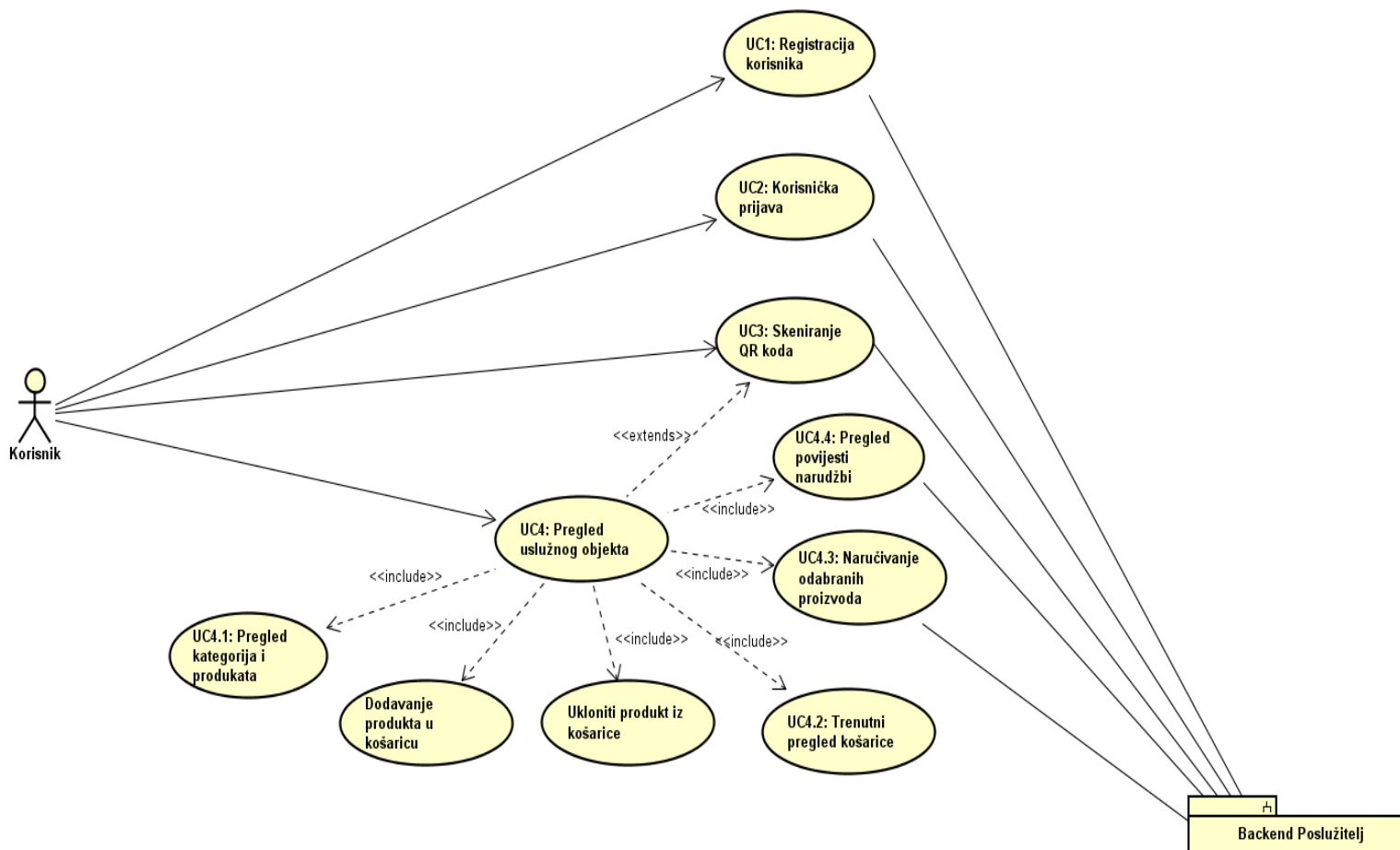
- Zaposlenik može imati pregled svih aktivnih narudžbi unutar uslužnog objekta.
- Zaposlenik može procesirati aktivnu narudžbu i pripremiti račun.

### 2.4. *Zahtjevi administratora sustava IOrder*

- Administrator sustava može verificirati i validirati uslužni objekt dajući mu domenu ukoliko je vlasnik podmirio račun.



## 2.1 Opis use caseova korisnika



Use case dijagram za korisnika

- UC1 – *Registracija korisnika*
  - **Glavni sudionik:** Anonimni korisnik.
  - **Cilj:** Ispravno ispuniti formu za registraciju u sustav.
  - **Ostali sudionici:** Backend poslužitelj, baza podataka.
  - **Rezultat:** Korisnik može pristupiti sustavu *IOrder* i vidjeti početni zaslon koji ga usmjerava u skeniranje QR koda uslužnog objekta.
  - **Željeni scenarij:**
    1. Korisnik otvara formu za registraciju.
    2. Upisuje podatke poput emaila, korisničkog imena, lozinke itd.
    3. Podaci su ispravno upisani i poslužitelj uspješno pohranjuje novog korisnika.

- UC2 – *Korisnička prijava*
  - **Glavni sudionik:** Registrirani korisnik.
  - **Cilj:** Ispravno ispuniti formu za prijavu u sustav.
  - **Ostali sudionici:** Backend poslužitelj, baza podataka.
  - **Rezultat:** Korisnik može pristupiti sustavu *IOrder* i vidjeti početni zaslon koji ga usmjerava u skeniranje QR koda uslužnog objekta.
  - **Željeni scenarij:**
    1. Korisnik otvara formu za prijavu.
    2. Upisuje korisničko ime i lozinku (ako već nije zapamćeno).
    3. Podaci su ispravno upisani i poslužitelj uspješno prijavljuje korisnika.
  
- UC3 – *Skeniranje QR koda*
  - **Glavni sudionik:** Registrirani korisnik
  - **Cilj:** Ispravno usmjeriti kameru mobitela i očitati QR kod.
  - **Ostali sudionici:** Backend poslužitelj, baza podataka, uslužni objekt.
  - **Rezultat:** Korisnik dobiva podatke o trenutnom uslužnom objektu ovisno o QR kodu kojeg je skenirao.
  - **Željeni scenarij:**
    1. Korisnik usmjerava kameru mobitela prema QR kodu.
    2. Podaci o uslužnom objektu su uspješno učitani.
    3. Korisnik ima uvid u trenutno stanje uslužnog objekta.

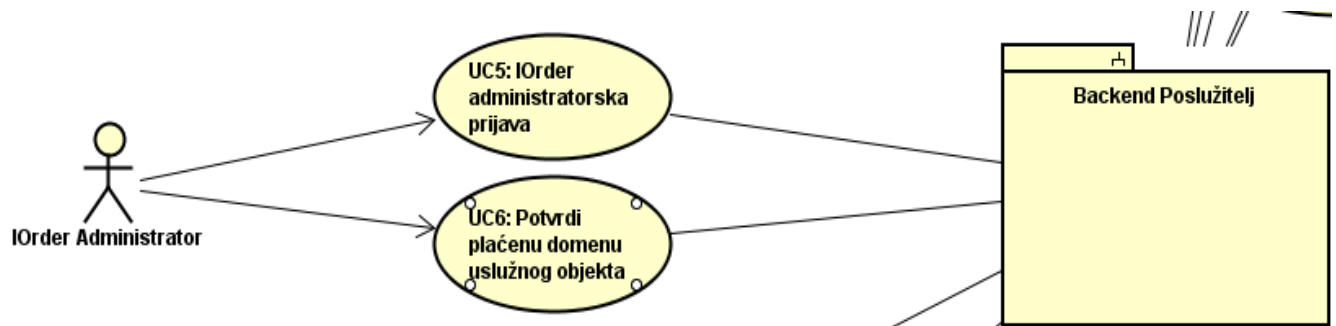
- UC4– *Pregled uslužnog objekta*
  - **Glavni sudionik:** Registrirani korisnik.
  - **Cilj:** Prikazati korisniku trenutno stanje uslužnog objekta.
  - **Ostali sudionici:** /
  - **Rezultat:** Korisnik može pregledavati opcije uslužnog objekta.
  - **Željeni scenarij:**
    1. Korisniku je prikazano sučelje kojim može djelovati prema uslužnom objektu.
  
- UC4.1 – *Pregled kategorija i produkata*
  - **Glavni sudionik:** Registrirani korisnik
  - **Cilj:** Prikazati korisniku trenutni meni uslužnog objekta.
  - **Ostali sudionici:** Backend poslužitelj, baza podataka.
  - **Rezultat:** Korisnik može pregledavati kategorije i produkte uslužnog objekta.
  - **Željeni scenarij:**
    1. Korisniku su prikazane liste kategorija koje proširenjem daju listu produkata pod određenom kategorijom.
    2. Moguća daljnja manipulacija produktima (dodavanje/brisanje iz košarice).



- UC4.2. – *Trenutni pregled košarice*
  - **Glavni sudionik:** Registrirani korisnik
  - **Cilj:** Uvidjeti trenutno stanje košarice.
  - **Ostali sudionici:** /
  - **Rezultat:** Korisnik može pregledavati proizvode i njihove detalje prije same narudžbe.
  - **Željeni scenarij:**
    1. Korisnik nakon svakog odabranog produkta dobije obavijest da je on uspješno dodan u košaricu.
    2. Otvaranjem košarice prikazuje se lista odabranih produkata i njihovi detalji te opcija narudžbe.
- UC4.3. – *Naručivanje odabranih proizvoda*
  - **Glavni sudionik:** Registrirani korisnik
  - **Cilj:** Naručiti proizvode koji su odabrani u košarici.
  - **Ostali sudionici:** Backend poslužitelj, baza podataka.
  - **Rezultat:** Korisnik šalje narudžbu zaposlenicima uslužnog objekta.
  - **Željeni scenarij:**
    1. Korisnik nakon što je otvorio košaricu šalje narudžbu pritiskom na gumb.
    2. Narudžba se zatim validira i ukoliko je uspješno procesuirana, korisnik dobije obavijest.

- UC4.4 – *Pregled povijesti narudžbi*
  - **Glavni sudionik:** Registrirani korisnik
  - **Cilj:** Prikazati korisniku svu povijest narudžbi za trenutni uslužni objekt.
  - **Ostali sudionici:** Backend poslužitelj, baza podataka.
  - **Rezultat:** Korisnik otvara ekran u kojem se nalazi lista prošlih narudžbi za trenutni uslužni objekt.
  - **Željeni scenarij:**
    1. Korisnik nakon pritiska na ikonicu koja predstavlja povijest narudžbi, dobije listu svih proteklih narudžbi.
    2. Pritiskom na bilo koju prošlu narudžbu, moguće ju je ponovno iskoristiti -> UC4.3.

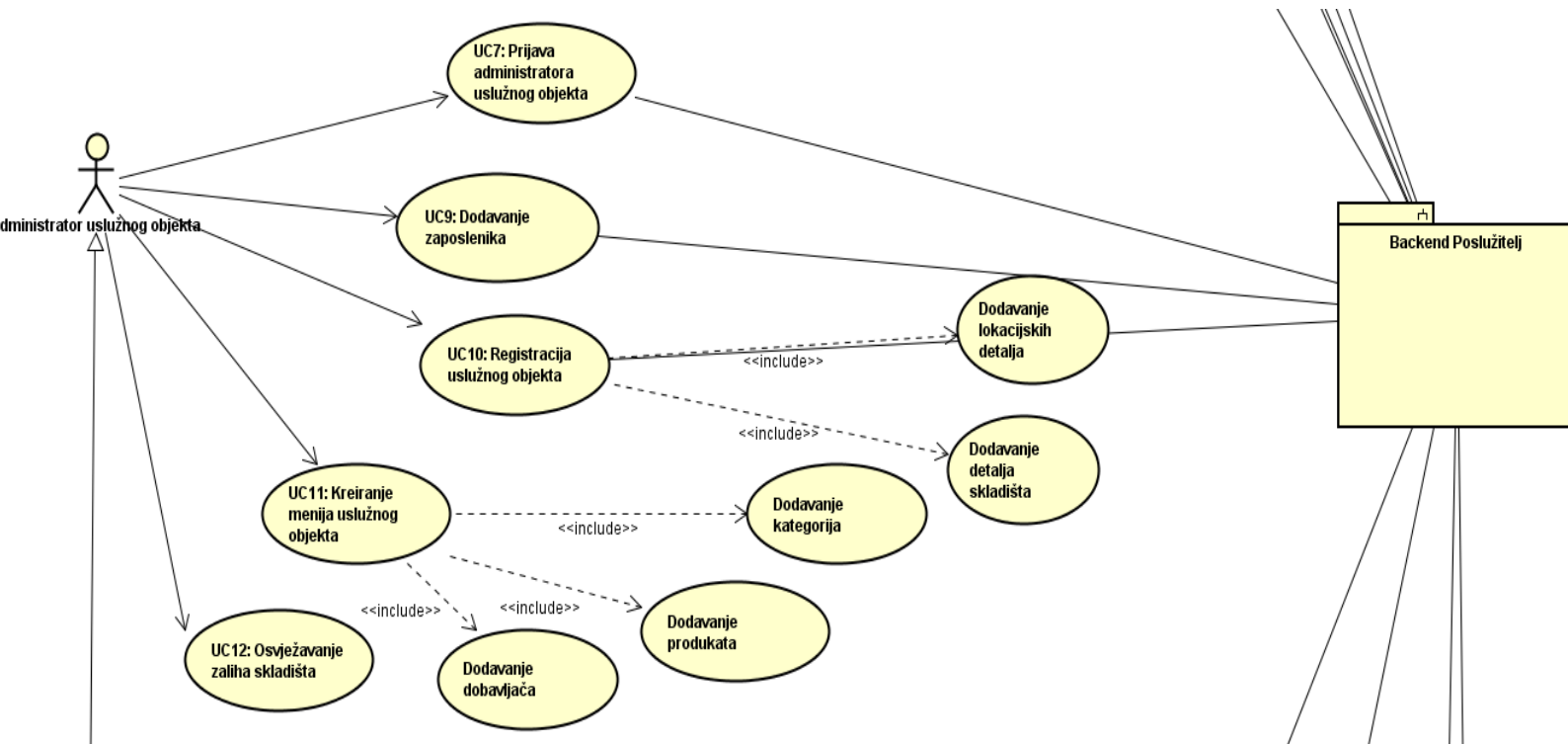
## 1.1. 2.2 Opis use caseova IOrder administratora



Use case dijagram za IOrder administratora

- UC5– *IOrder administratorska prijava*
  - **Glavni sudionik:** IOrder administrator
  - **Cilj:** Prijava u sustav.
  - **Ostali sudionici:** Backend poslužitelj, baza podataka.
  - **Rezultat:** IOrder administratoru se prikazuje prozor koji mu daje pravo potvrđivanja ovlasti uslužnih objekata u sustavu.
  - **Željeni scenarij:**
    1. IOrder administrator otvara formu za prijavu u sustav i upisuje svoje podatke.
    2. Podaci su ispravno uneseni i otvara se novi administratorski prozor
- UC6– *Potvrdi plaćenu domenu uslužnog objekta*
  - **Glavni sudionik:** IOrder administrator
  - **Cilj:** Potvrditi zahtjev vlasnika uslužnog objekta za preuzimanjem prava IOrder sustav.
  - **Ostali sudionici:** Registrirani korisnik, Backend poslužitelj, baza podataka.
  - **Rezultat:** IOrder administrator dodjeljuje administratorska prava vlasnika uslužnog objekta nad tim objektom u sustavu.
  - **Željeni scenarij:**
    1. IOrder administrator potvrđuje zahtjev vlasnika uslužnog objekta.
    2. Vlasnik uslužnog objekta ulazi u sustav IOrder i može uređivati podatke svog uslužnog objekta.

## 1.2. 2.3. Opis use caseova administratora uslužnog objekta



Use case dijagram za administratora uslužnog objekta

- **UC7 - Prijava administratora uslužnog objekta**
  - **Glavni sudionik:** administrator uslužnog objekta
  - **Cilj:** Prijava u IOrder desktop aplikaciju.
  - **Ostali sudionici:** desktop aplikacija, Backend poslužitelj, baza podataka.
  - **Rezultat:** Administratoru se prikazuje prozor za unos podataka potrebnih za prijavu u aplikaciju.
  - **Željeni scenarij:**
    1. Administrator uslužnog objekta otvara formu za prijavu u sustav i upisuje svoje podatke.
    2. Podaci su ispravno uneseni i otvara se glavni prozor aplikacije

- *UC9 - Dodavanje zaposlenika*

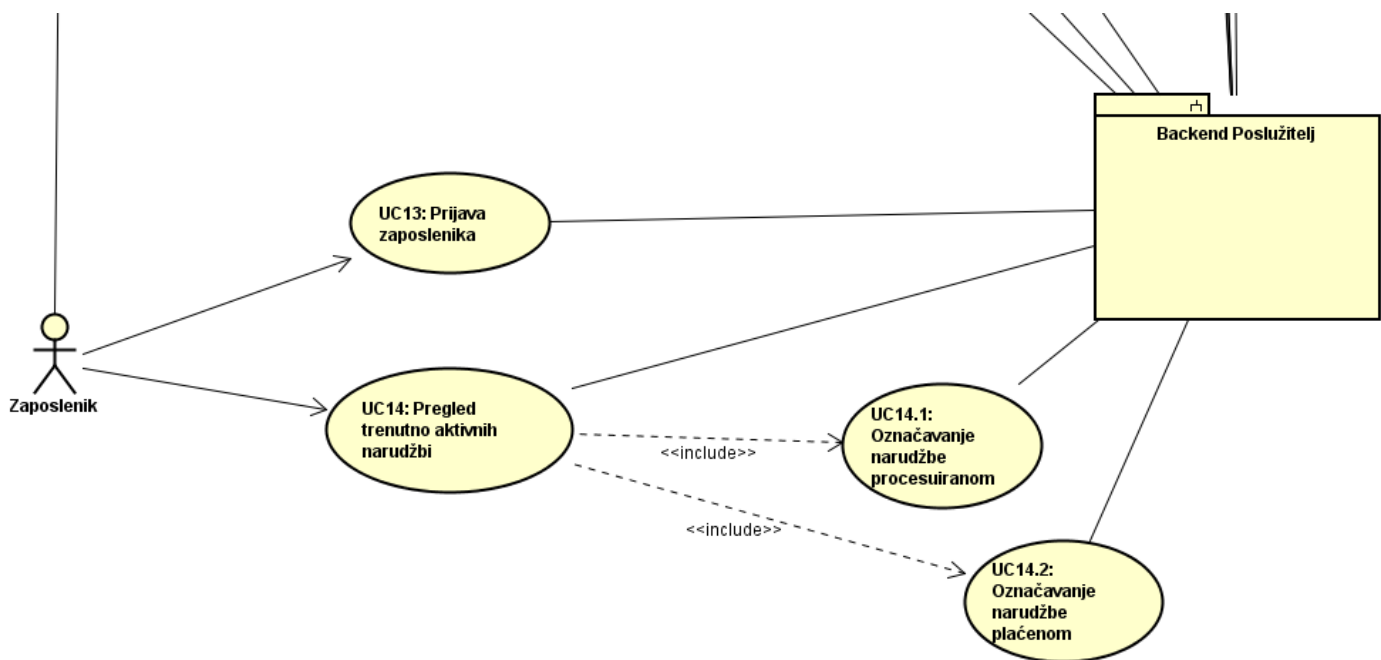
- **Glavni sudionik:** administrator uslužnog objekta
- **Cilj:** Dodati novog zaposlenika u sustav.
- **Ostali sudionici:** desktop aplikacija, zaposlenik, Backend poslužitelj, baza podataka.
- **Rezultat:** Administrator dodaje zapis o novom zaposleniku u sustav.
- **Željeni scenarij:**
  1. Administrator uslužnog objekta otvara formu za rad sa zapisima zaposlenika, otvara prozor za dodavanje novog zaposlenika i upisuje podatke o zaposleniku.
  2. . Podaci su ispravno uneseni i otvara se glavni prozor aplikacije

- *UC10 - Registracija uslužnog objekta*

- **Glavni sudionik:** administrator uslužnog objekta
- **Cilj:** Dodati novi uslužni objekt u sustav.
- **Ostali sudionici:** desktop aplikacija, Backend poslužitelj, baza podataka.
- **Rezultat:** Administrator dodaje zapis o novom uslužnom objektu u sustav.
- **Željeni scenarij:**
  1. Administrator uslužnog objekta otvara formu za rad sa zapisima uslužnih objekata, otvara prozor za dodavanje novog uslužnog objekta
  2. Upisuje ime objekta, lokaciju objekta i skladište koje čuva robu za objekt te pritišće tipku Confirm
  3. . Podaci su ispravno uneseni i otvara se glavni prozor aplikacije

- *UC11 - Kreiranje menija uslužnog objekta*
  - **Glavni sudionik:** administrator uslužnog objekta
  - **Cilj:** Dodavanje kategorija pića, dodavanje proizvoda i dobavljača proizvoda u sustav
  - **Ostali sudionici:** desktop aplikacija, zaposlenik, Backend poslužitelj, baza podataka.
  - **Rezultat:** Administrator dodaje zapis o novom proizvodu u sustav
  - **Željeni scenarij:**
    1. Administrator uslužnog objekta otvara formu za rad sa zapisima kategorija, otvara prozor za dodavanje nove kategorije i upisuje podatke o kategoriji.
    2. Administrator uslužnog objekta otvara formu za rad sa zapisima dobavljača, otvara prozor za dodavanje novog dobavljača i upisuje podatke o dobavljaču.
    3. . Nakon što su u sustav uneseni podaci o kategoriji i dobavljaču administrator može dodati zapis o novom proizvodu i povezati ga sa željenom kategorijom i željenim dobavljačem.
    4. Podaci su ispravno uneseni i otvara se glavni prozor aplikacije
  
- *UC12 - Osvježavanje zaliha skladišta*
  - **Glavni sudionik:** administrator uslužnog objekta
  - **Cilj:** Dodati proizvode u određeno skladište.
  - **Ostali sudionici:** desktop aplikacija, zaposlenik, Backend poslužitelj, baza podataka.
  - **Rezultat:** Administrator dodaje proizvode u određenoj količini u željeno skladište.
  - **Željeni scenarij:**
    1. Administrator uslužnog objekta bira željeni proizvod iz asortimana proizvoda koji su uneseni u sustav
    2. Proizvodu dodjeljuje skladište, količinu i cijenu te posprema proizvode u skladište.

### 1.3. 2.4. Opis use caseova zaposlenika uslužnog objekta



Use case dijagram za zaposlenika uslužnog objekta

- *UC13 - Prijava zaposlenika uslužnog objekta*
  - **Glavni sudionik:** zaposlenik uslužnog objekta
  - **Cilj:** Prijava u IOrder web aplikaciju.
  - **Ostali sudionici:** web aplikacija, Backend poslužitelj, baza podataka.
  - **Rezultat:** Zaposlenik je uspješno ulogiran u iOrder web aplikaciju.
  - **Željeni scenarij:**
    1. Zaposlenik otvara lokaciju iOrder web aplikacije
    2. Ispunjava priloženu formu za njegovo korisničko ime i njegovu šifru
    3. Prilikom završetka uspješne obrade para imena i šifre web aplikacije će otvoriti svoj glavni prozor.
- *UC14 - Pregled trenutnih narudžaba*
  - **Glavni sudionik:** zaposlenik uslužnog objekta
  - **Cilj:** Pregled neobrađenih narudžaba.
  - **Ostali sudionici:** web aplikacija, Backend poslužitelj, baza podataka.
  - **Rezultat:** Zaposlenik uspješno vidi popis narudžaba.
  - **Željeni scenarij:**
    1. Autorizacija korisnika
    2. U aplikaciji postoji prozor s listom gdje se u stvarnome vremenu prikazuju trenutne narudžbe

- *UC15 - Pregled određene narudžbe*
  - **Glavni sudionik:** zaposlenik uslužnog objekta
  - **Cilj:** Pregled određene narudžaba.
  - **Ostali sudionici:** web aplikacija, Backend poslužitelj, baza podataka.
  - **Rezultat:** Zaposlenik uspješno vidi detalje o narudžbi.
  - **Željeni scenarij:**
    1. Autorizacija korisnika
    2. Odabir narudžbe u listi neobrađenih narudžbi
    3. U posebnome dijelu aplikacije se mogu vidjeti detalji o toj narudžbi kao i interakcijski elementi za tu narudžbu
  
- *UC16 - Interakcija s narudžbom - Obrada narudžbe*
  - **Glavni sudionik:** zaposlenik uslužnog objekta
  - **Cilj:** Potvrđivanje da je narudžba obrađena.
  - **Ostali sudionici:** web aplikacija, Backend poslužitelj, baza podataka.
  - **Rezultat:** Narudžba je uspješno naplaćena.
  - **Željeni scenarij:**
    1. Odabir narudžbe
    2. Klik na kontrolu Plaćeno
    3. Narudžba nestaje iz liste neobrađenih narudžbi
  
- *UC17 - Interakcija s narudžbom - Generiranje računa*
  - **Glavni sudionik:** zaposlenik uslužnog objekta
  - **Cilj:** Pregled neobrađenih narudžaba.
  - **Ostali sudionici:** web aplikacija, Backend poslužitelj, baza podataka.
  - **Rezultat:** Zaposlenik uspješno vidi popis narudžaba.
  - **Željeni scenarij:**
    1. Odabir narudžbe
    2. Klik na kontrolu Račun će prikazati trenutnu narudžbu u obliku pogodnome za ispis



### 3. Ostali zahtjevi

- Korištenje sustava mora biti omogućeno za velik broj korisnika u stvarnom vremenu.
- Sustav mora biti izgrađen u okviru objektno-orijentirane paradigme sa SOLID principima.
- Najbitnije opcije sustava bi u aplikacijama trebale biti lako dostupne.
- Desktop aplikacija bi trebala biti dostupna za bilo koje stolno računalo operacijskog sustava Windows.
- Web aplikacija bi trebala biti podržana u svim glavnim pretraživačima.
- Mobilna aplikacija bi trebala biti podržana na verzijama Androida 5.0 na više.
- Dobavljanje podataka iz baze podataka ne bi trebalo trajati duže od 10 sekundi.
- Dobavljanje podataka preko interneta bi trebalo završavati u normalnom intervalu vremena s opisnom porukom uspjeha ili pogreške.
- Implementacija dodatnih opcija unutar sustava ne bi trebalo utjecati na postojeće opcije.
- Anonimni korisnici ne bi smjeli prodrijeti do nedozvoljenih dijelova aplikacija.

## 4. Backend API

Authentication			
Ruta [Ovlast]	Varijable	Tijelo zahtjeva	Odgovor
Dohvaćanje tokena [Anonymous] //POST /api/Auth		Objekt koji sadrži korisničko ime i lozinku	Token

User			
Ruta [Ovlast]	Varijable	Tijelo zahtjeva	Odgovor
Dohvaćanje svih zaposlenika vlasnika [ADMIN] //GET: api/User/Employee			Lista zaposlenika
Registracija korisnika [Anonymous] // POST: api/User/Customer		Objekt User	OK za uspješnu registraciju
Registracija zaposlenika [ADMIN] // POST: api/User/Employee		Objekt User	OK za uspješnu registraciju
Brisanje računa korisnika [CUSTOMER] // DELETE: api/User/Customer			OK za uspješno brisanje

Brisanje računa zaposlenika [ADMIN] // DELETE: api/User/Employee/{shemso}			OK za uspješno brisanje
---	--	--	----------------------------

Category			
Ruta [Ovlast]	Varijable	Tijelo zahtjeva	Odgovor
Dohvaćanje svih kategorija za vlasnika [ADMIN] // GET: api/Category			Lista kategorija
Dohvaćanje svih kategorija za skladište [ADMIN] // GET: api/Category/Warehouse/5	Id skladišta		Lista kategorija
Nova kategorija [ADMIN] // POST: api/Category		Objekt Category	OK za uspješno stvaranje nove kategorije
Ažuriranje kategorije [ADMIN] // PUT: api/Category/5	Id kategorije	Objekt Category	OK za uspješno ažuriranje
Brisanje kategorije [ADMIN] // DELETE: api/Category/5	Id kategorije		OK za uspješno brisanje

Establishment			
Ruta [Ovlast]	Varijable	Tijelo zahtjeva	Odgovor

Dohvaćanje svih uslužnih objekata [CUSTOMER, ADMIN] // GET: api/Establishment			Lista uslužnih objekata
Dohvaćanje svih uslužnih objekata jednog vlasnika [ADMIN, CUSTOMER] // GET: api/Establishment/Owner	Id vlasnika		Lista uslužnih objekata
Dohvaćanje uslužnog objekta po id-u [ADMIN, CUSTOMER] // GET: api/Establishment/5	Id uslužnog objekta		Objekt uslužnog objekta
Novi uslužni object [ADMIN] // POST: api/Establishment		Objekt Establishment	OK za uspješno stvaranje
Ažuriranje uslužnog objekta [ADMIN] // PUT: api/Establishment/5	Id uslužnog objekta	Objekt Establishment	OK za uspješno ažuriranje
Brisanje uslužnog objekta [ADMIN] // DELETE: api/Establishment/5	Id uslužnog objekta		OK za uspješno brisanje

Location			
Ruta [Ovlast]	Varijable	Tijelo zahtjeva	Odgovor
Dohvaćanje svih lokacija vlasnika [ADMIN] // GET: api/Location			Lista lokacija
Nova lokacije [ADMIN] // POST: api/Location		Objekt Location	OK za uspješno stvaranje

Ažuriranje lokacije[ADMIN] // PUT: api/Location/5	Id lokacije	Objekt Location	OK za uspješno ažuriranje
Brisanje lokacije[ADMIN] // DELETE: api/Location/5	Id lokacije		OK za uspješno brisanje

Order			
Ruta [Ovlast]	Varijable	Tijelo zahtjeva	Odgovor
Dohvaćanje povijesti za korisnika za uslužni objekt [CUSTOMER] // GET: api/Order/CustomerHistory/5	Id uslužnog objekta		Lista narudžbi
Dohvaćanje povijesti narudžbi za uslužni objekt [EMPLOYEE,ADMIN] // GET: api/Order/EstablishmentHistory/5	Id uslužnog objekta		OK za uspješno stvaranje
Nova narudžbe [CUSTOMER] // POST: api/Order		Objekt Order	OK za uspješno ažuriranje
Brisanje narudžbe [CUSTOMER,ADMIN,EMPLOYEE] // DELETE: api/Order/5	Id narudžbe		OK za uspješno brisanje

Supplier			
Ruta [Ovlast]	Varijable	Tijelo zahtjeva	Odgovor
Dohvaćanje svih dobavljača vlasnika [ADMIN] // GET: api/Supplier			Lista dobavljača

Novi dobavljač [ADMIN] // POST: api/Supplier		Objekt Supplier	OK za uspješno stvaranje
Ažuriranje dobavljača [ADMIN] // PUT: api/Supplier/5	Id dobavljača	Objekt Supplier	OK za uspješno ažuriranje
Brisanje dobavljača [ADMIN] // DELETE: api/Supplier/5	Id dobavljača		OK za uspješno brisanje

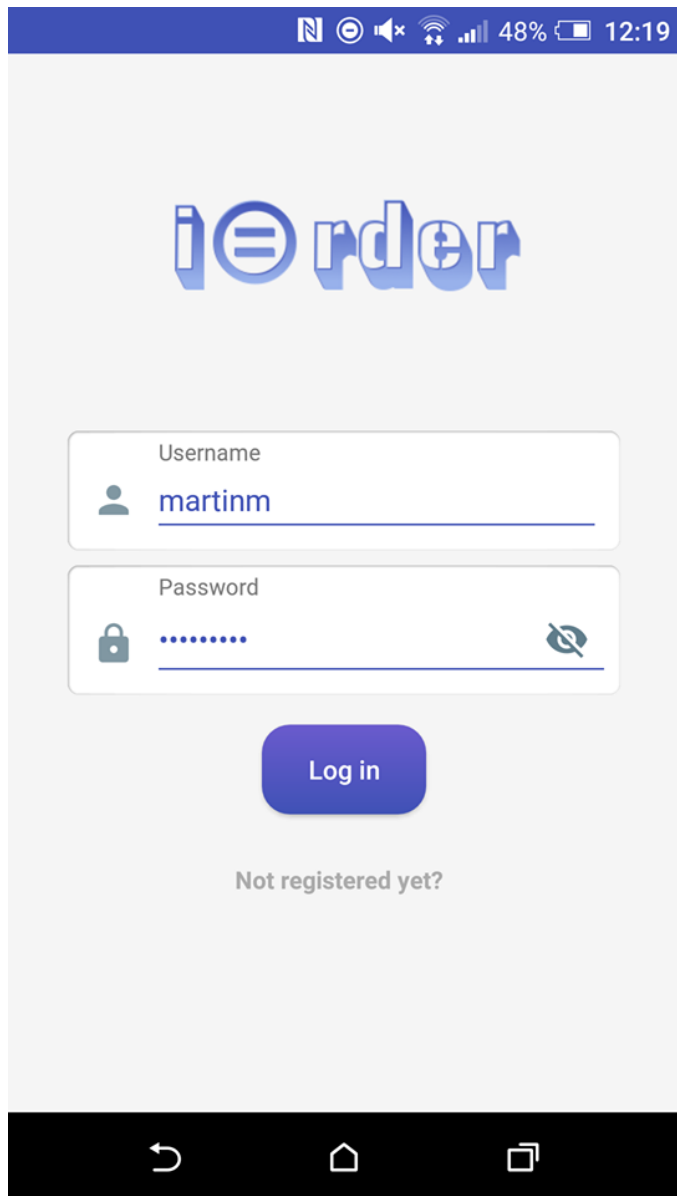
Warehouse			
Ruta [Ovlast]	Varijable	Tijelo zahtjeva	Odgovor
Dohvaćanje svih skladišta vlasnika [ADMIN] // GET: api/Warehouse			Lista skladišta
Novo skladište [ADMIN] // POST: api/Warehouse		Objekt Warehouse	OK za uspješno stvaranje
Ažuriranje skladišta [ADMIN] // PUT: api/Warehouse/5	Id skladišta	Objekt Warehouse	OK za uspješno ažuriranje
Brisanje skladišta [ADMIN] // DELETE: api/Warehouse/5	Id skladišta		OK za uspješno brisanje

Product			
Ruta [Ovlast]	Varijable	Tijelo zahtjeva	Odgovor
Dohvaćanje svih proizvoda vlasnika [ADMIN] // GET: api/Product			Lista proizvoda

Novi proizvod [ADMIN] // POST: api/Product		Objekt Product	OK za uspješno stvaranje
Ažuriranje proizvoda [ADMIN] // PUT: api/Product/5	Id proizvoida	Objekt Product	OK za uspješno ažuriranje
Brisanje proizvoda [ADMIN] // DELETE: api/Product/5	Id proizvoida		OK za uspješno brisanje
Dodavanje proizvoda u skladište [ADMIN] // POST: api/Product/Storage		Objekt WarehouseProductPair	OK za uspješno dodavanje
Dohvaćanje svih proizvoda za skladište [ADMIN] // GET: api/Product/Storage/1			Lista proizvoda

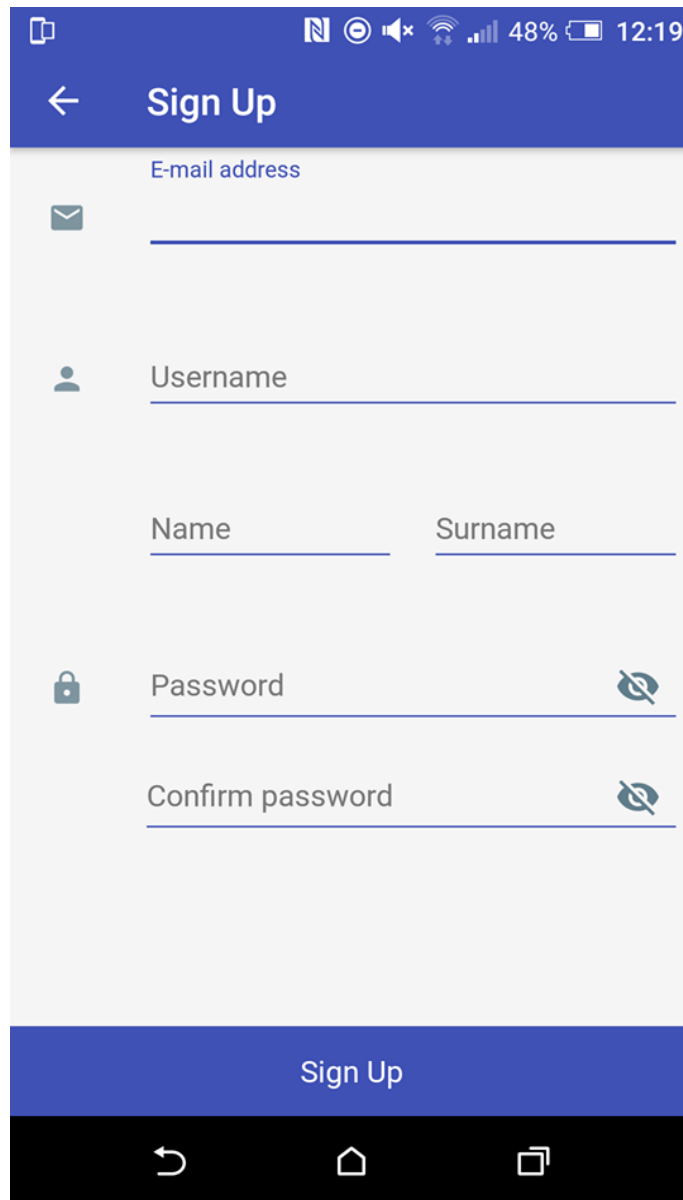
## 5. Pregled ekrana

### 5.1 Mobilna aplikacija Android



The screenshot shows the login interface of the iOrder mobile application. At the top, there is a status bar with various icons and the time 12:19. Below the status bar, the iOrder logo is displayed in a stylized blue font. The main content area features two input fields: 'Username' with the text 'martinm' and 'Password' with masked characters. A blue 'Log in' button is positioned below the password field. At the bottom of the form, there is a link that says 'Not registered yet?'. The entire screen is framed by a dark blue header and a black Android navigation bar at the bottom.

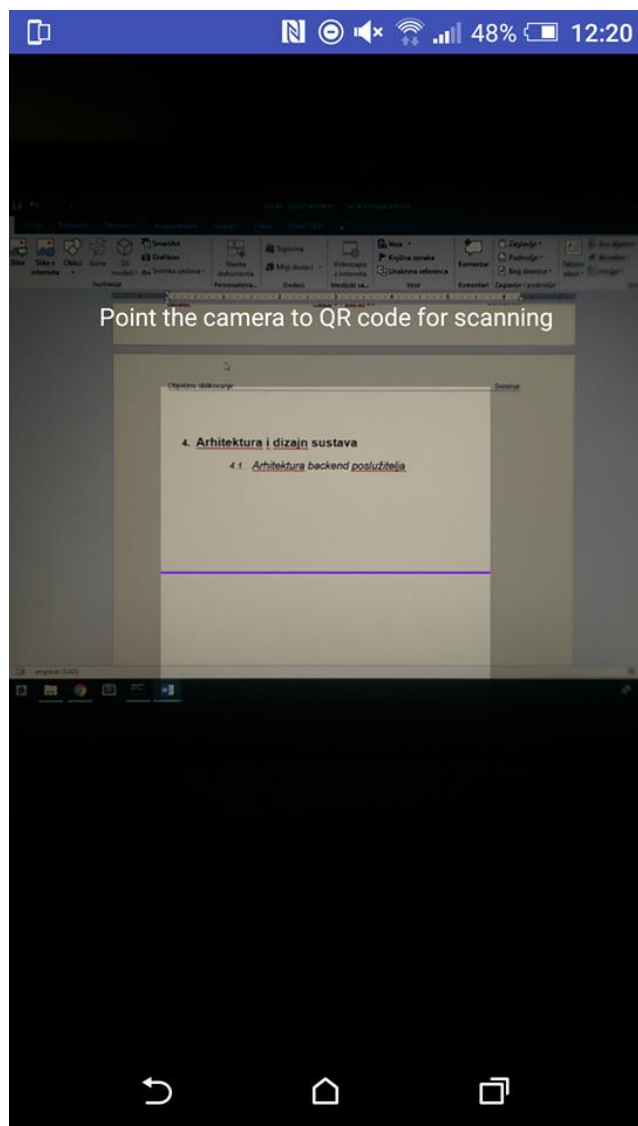
Slika 1. Forma prijave



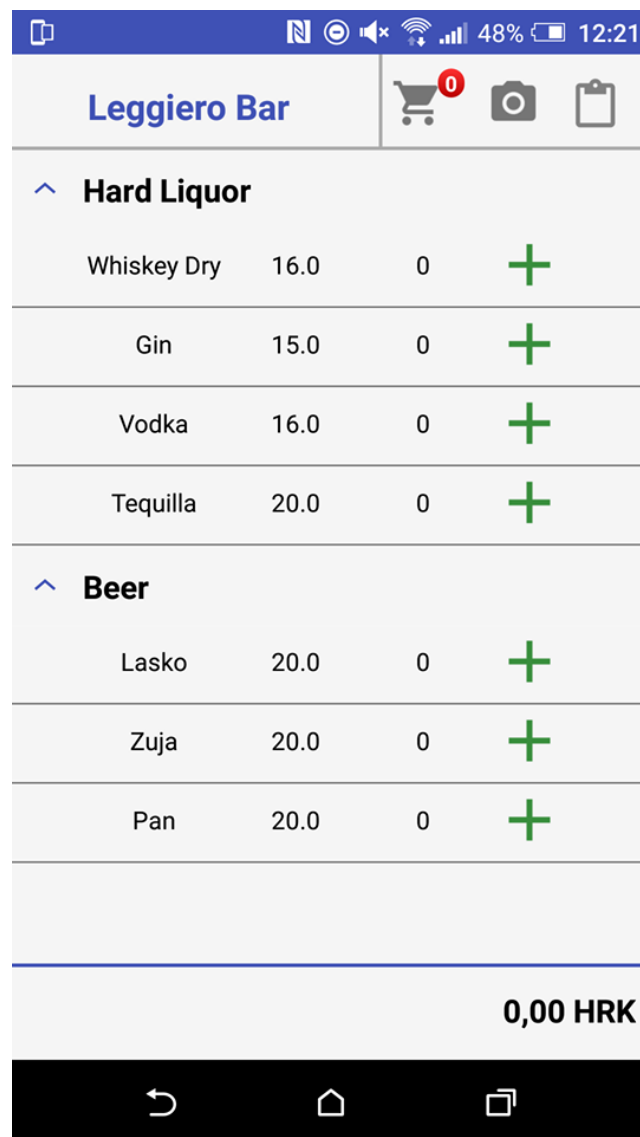
The screenshot shows the registration interface of the iOrder mobile application. The title bar at the top is blue with a back arrow and the text 'Sign Up'. The form contains four input fields: 'E-mail address', 'Username', 'Name' (with a 'Surname' field next to it), and 'Password' (with a 'Confirm password' field next to it). Each password field has a toggle icon to show or hide the text. A blue 'Sign Up' button is located at the bottom of the form. The screen is framed by a dark blue header and a black Android navigation bar at the bottom.

Slika 2. Registracijska forma

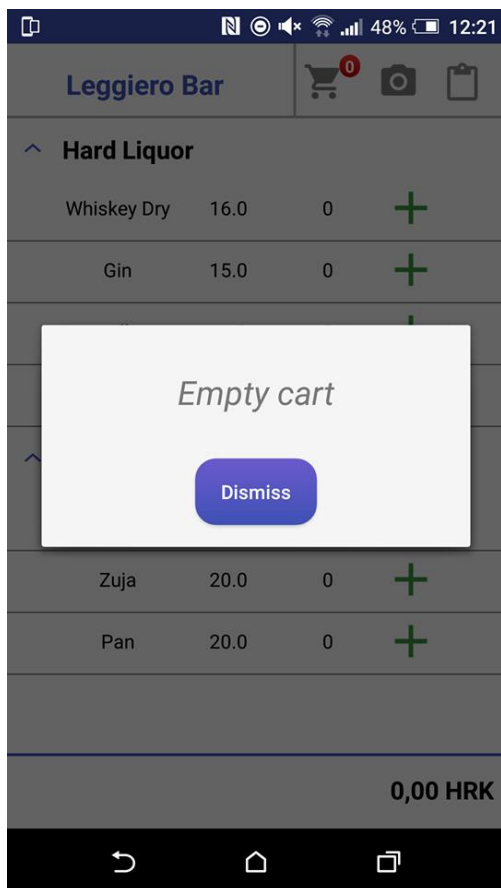




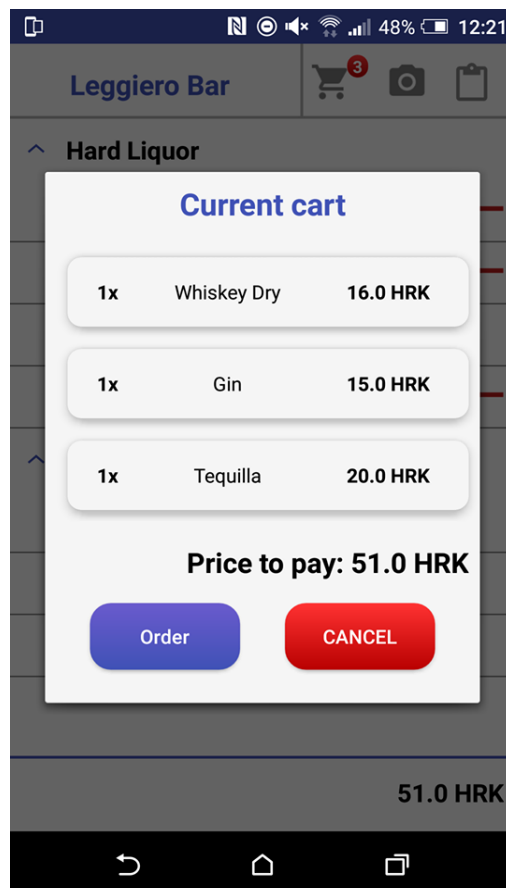
Slika 3. Pokrenut skener



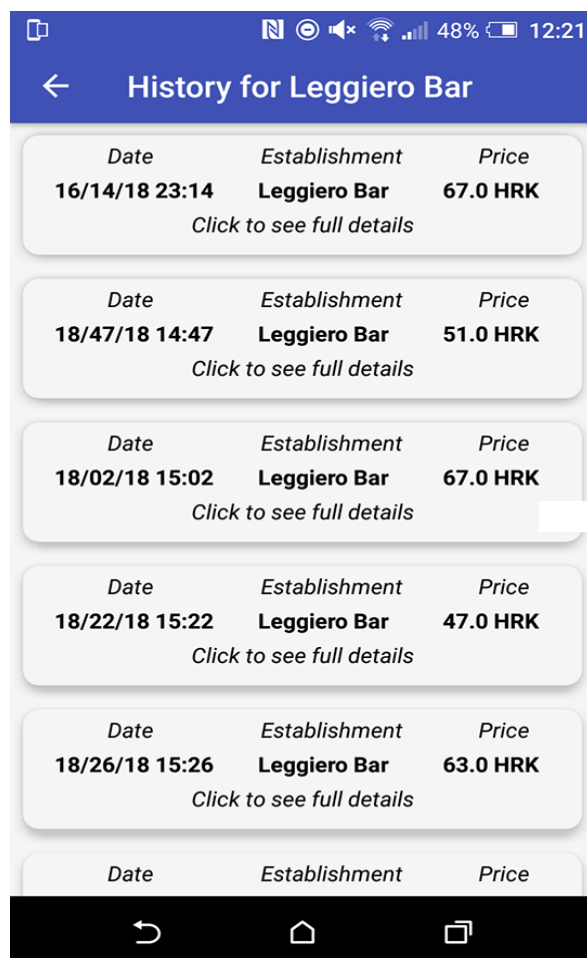
Slika 4. Pregled kategorija i proizvoda



Slika 5. Prazna košarica

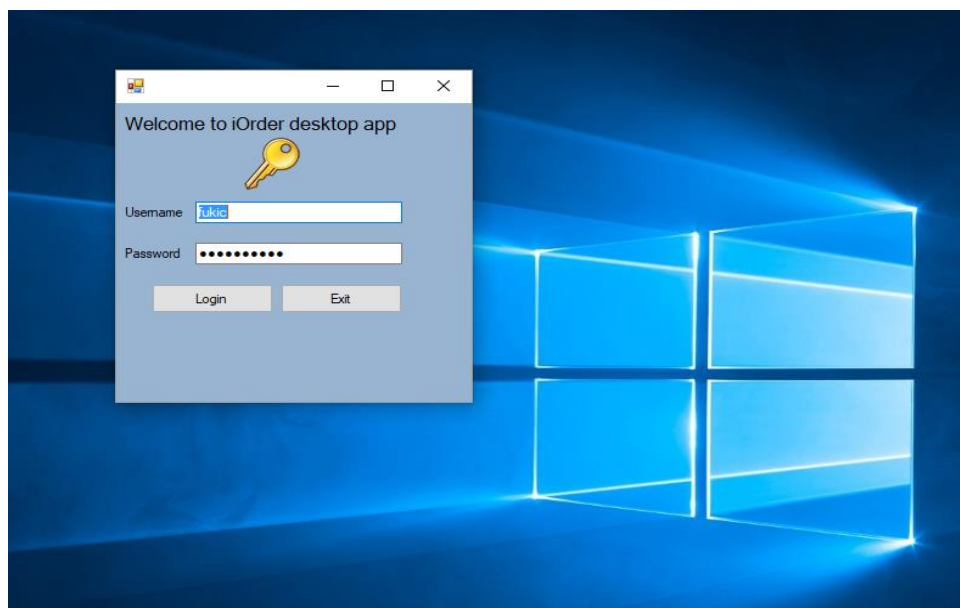


Slika 6. Košarica s proizvodima



Slika 7. Povijest narudžbi

## 5.2 Desktop aplikacija



Slika 8. Forma prijave

DesktopApp

Warehouse | Category | Product | Location | Establishment | Employee

	Id	CategoryId	Name	BuyingPrice	OwnerId	SupplierId
▶	1	7	Whiskey Dry	10,0	fukic ...	2
	2	7	Gin	9,0	fukic ...	2
	3	7	Vodka	15,0	fukic ...	2
	4	7	Tequilla	15,0	fukic ...	3
	5	8	Lasko	10,0	fukic ...	4
	6	8	Zuja	10,0	fukic ...	3
	7	8	Pan	10,0	fukic ...	3
	20	8	Tomislav	11,0	fukic ...	3

Name  Warehouse

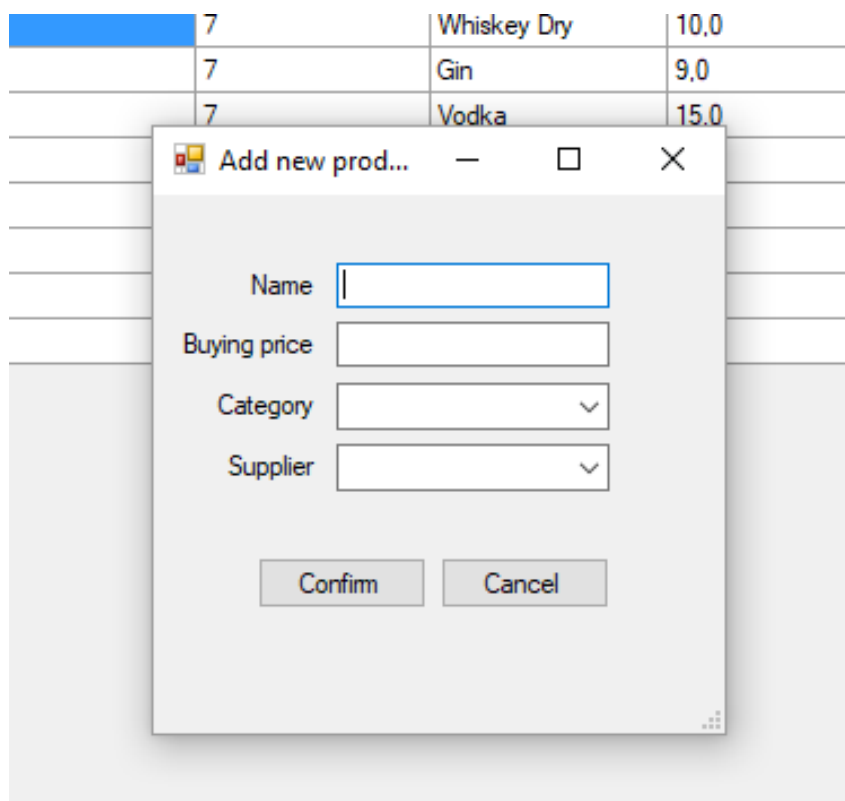
Buying price  Quantity

Category  Selling Price

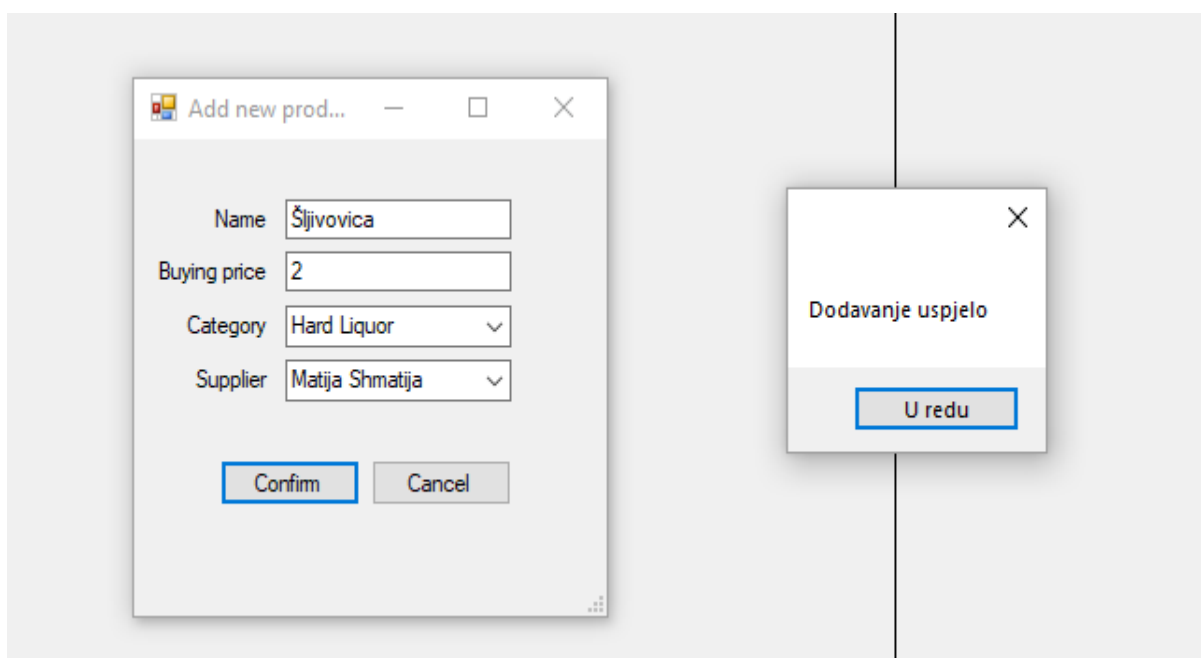
Supplier

id

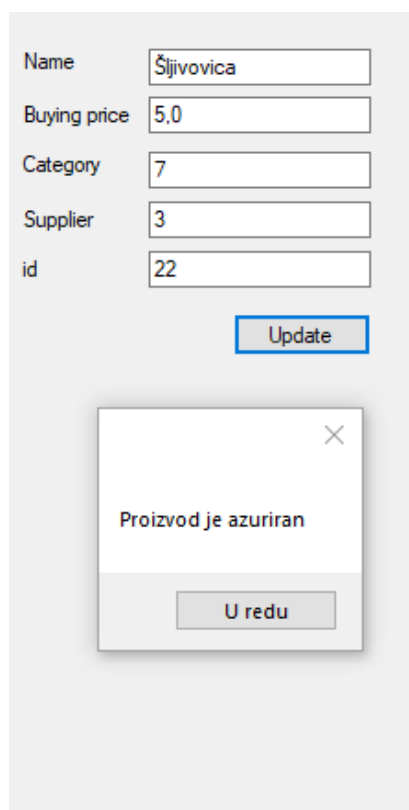
Slika 9. Pregled svih produkata



Slika 10. Dodaj novi proizvod

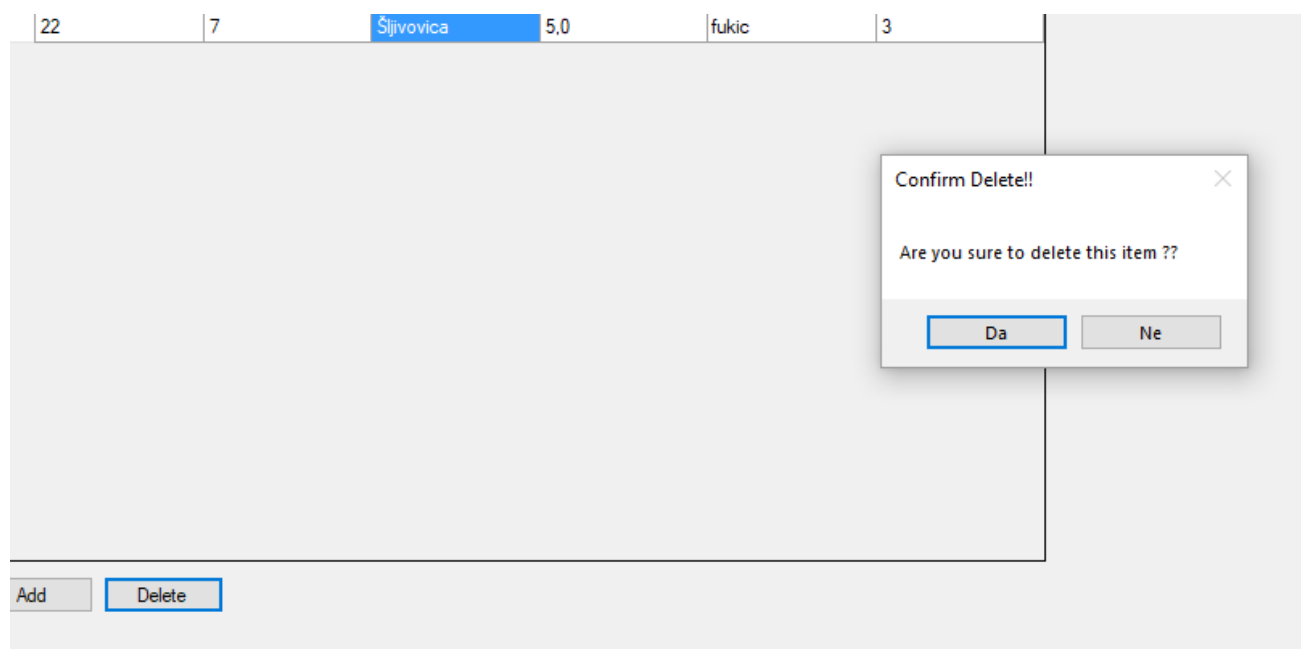


Slika 11. Uspješno dodavanje proizvoda



A screenshot of a web application interface for updating a product. The form contains five input fields: 'Name' with the value 'Šljivovica', 'Buying price' with '5,0', 'Category' with '7', 'Supplier' with '3', and 'id' with '22'. Below the fields is a blue 'Update' button. A modal dialog box is displayed in the center, containing the text 'Proizvod je azuriran' and a grey 'U redu' button.

Slika 7. Ažuriranje postojećeg proizvoda



A screenshot of a web application interface showing a table of products. The table has six columns: 'id', 'category', 'name', 'price', 'supplier', and 'status'. The first row contains the values '22', '7', 'Šljivovica', '5,0', 'fukic', and '3'. The 'name' cell is highlighted in blue. Below the table are two buttons: 'Add' and 'Delete'. A modal dialog box titled 'Confirm Delete!!' is open, asking 'Are you sure to delete this item ??' with 'Da' and 'Ne' buttons.

id	category	name	price	supplier	status
22	7	Šljivovica	5,0	fukic	3

Slika 8. Brisanje proizvoda

## 5.3 Web aplikacija

### iOrder login

kolinda

.....

Log in

Slika 9 Login prikaz

[History](#) [Order](#) [QR generate](#) [History](#) [About](#) [Logout 'kolinda'](#)

Sank 1	10:35:39 1
Sank 1	10:35:19 1
Sank 1	10:35:10 2

Customer: 'shimun' Location: Sank 1

Pay Bill

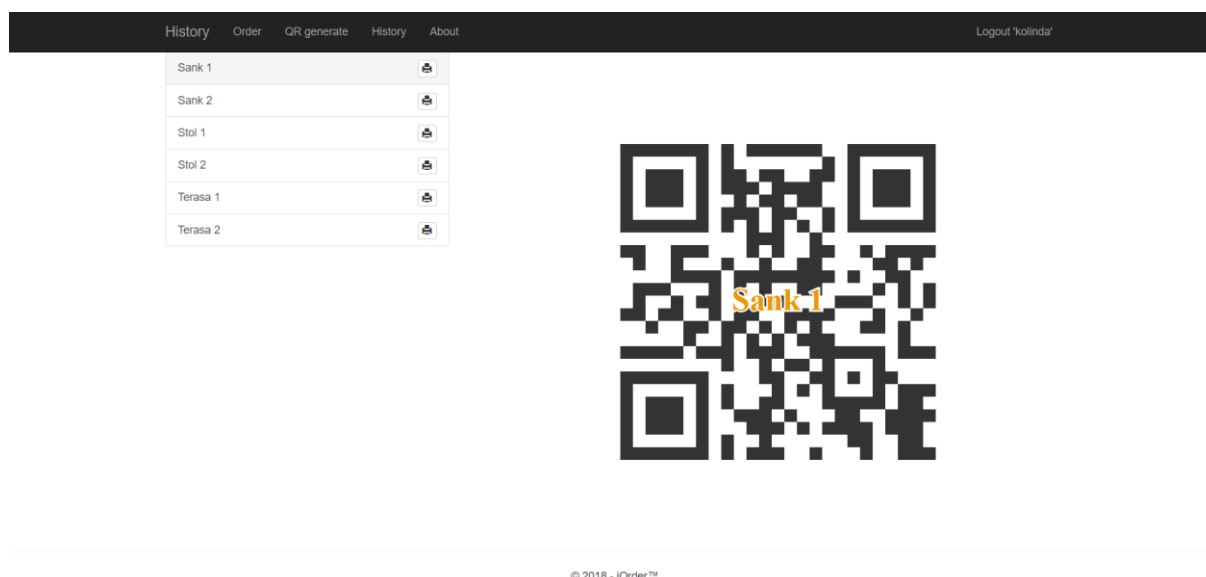
[Details](#) <

[Products](#) <

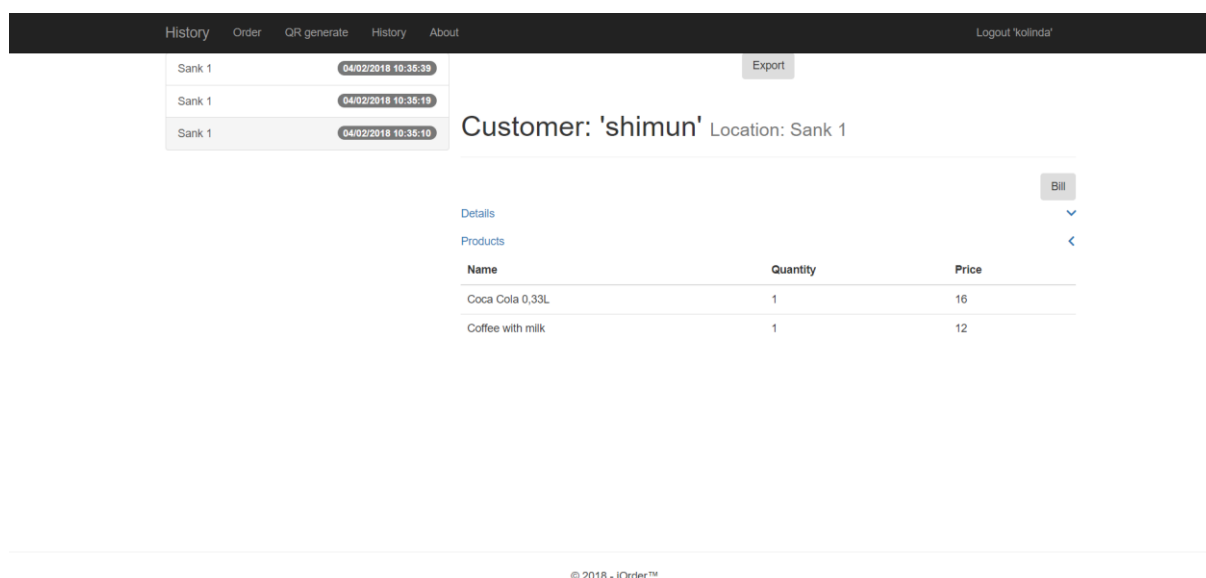
Name	Quantity	Price
Franziskaner	1	28

© 2018 - iOrder™

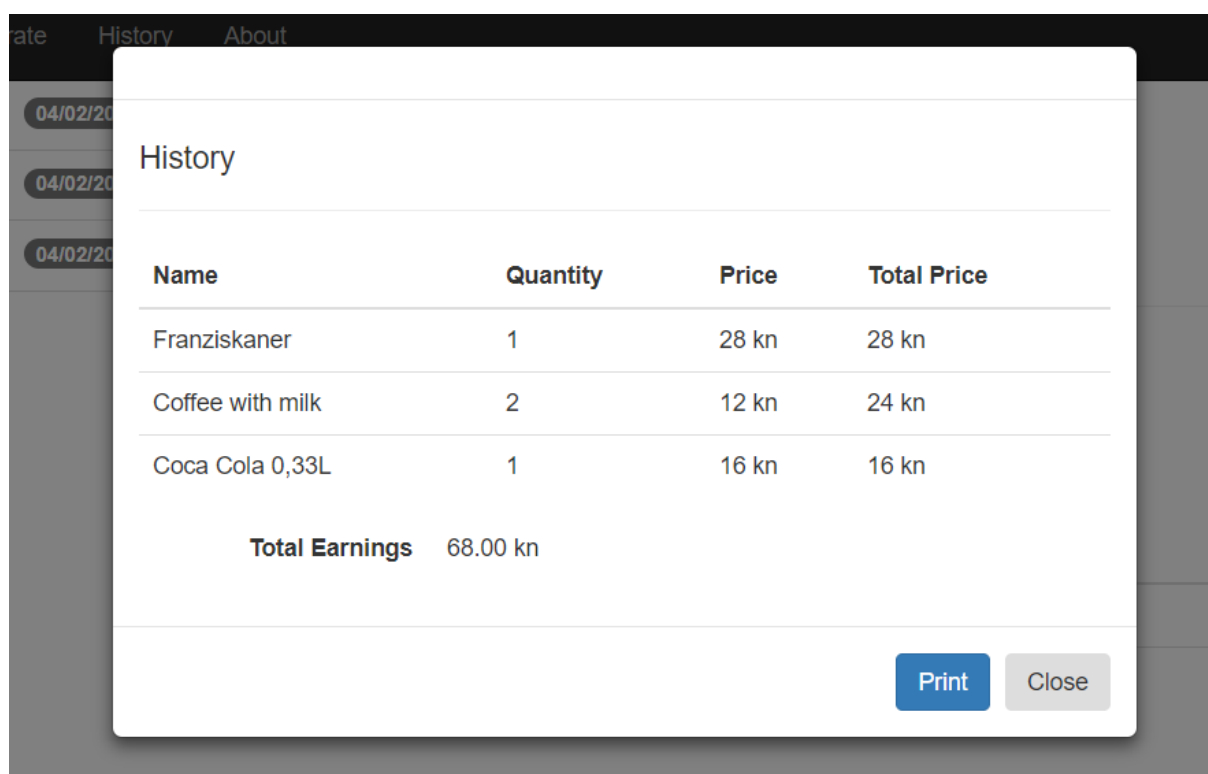
Slika 10 Glavni prikaz trenutnih narudžbi



Slika 11 Generator QR kodova lokacija



Slika 12 Povijest svih narudžbi za trenutni lokal



History

Name	Quantity	Price	Total Price
Franziskaner	1	28 kn	28 kn
Coffee with milk	2	12 kn	24 kn
Coca Cola 0,33L	1	16 kn	16 kn
Total Earnings		68.00 kn	

Print Close

Slika 13 Sažetak povijesti po ponudama

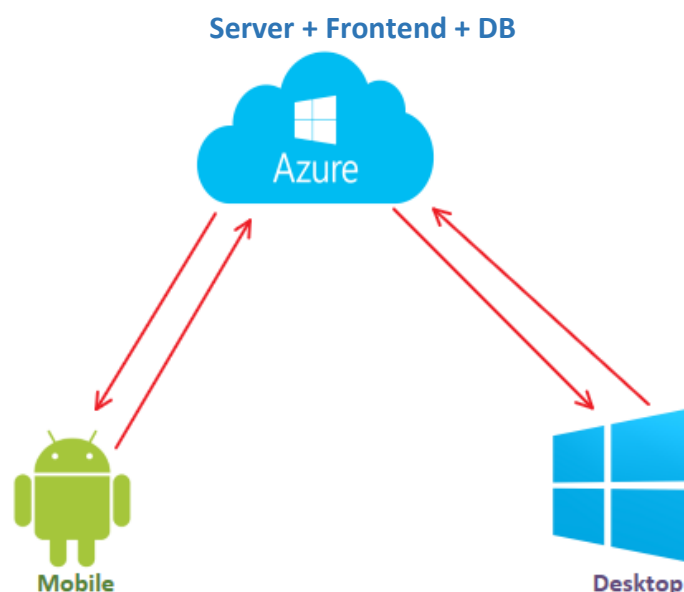


## 6. Arhitektura i dizajn sustava

### 6.1. Arhitektura sustava

Arhitektura cjelokupnog iOrder sustava sastoji se od nekoliko cjelina. Sustav čine Backend, Frontend, Mobile, Desktop te DB cjeline.

- **Backend** – poslužiteljska cjelina sustava koja obavlja cijelu logiku sustava te se brine o dohvaćanju i spremanju podataka u bazu. Također Backend služi kao REST sučelje Mobile i Desktop cjelini dok pomoću svog servisnog sloja pruža Frontend cjelini veće mogućnosti kontrole. Backend je objavljen na Microsoftovom cloud servisu Azure.
- **Frontend** – Frontend „živi“ u cloud-u Azure-a zajedno s Backend-om. Time je omogućeno korištenje servisnog sloja Backend-a bez spajanja preko REST sučelja. Frontend pruža klijentima iOrder sustava uvidi u trenutne narudžbe sa svaki od svojih uslužnih objekata te u cijelu povijest pojedinog objekta.
- **Mobile** – Mobile cjelina je ostvarena kao Android aplikacija. Aplikacija komunicira s Backend-om pomoću REST API sučelja koje pruža Backend. Aplikacija je namijenjena za korisnike klijenata iOrder sustava koji mogu bez čekanja u redu naručiti proizvode putem mobilne aplikacije.
- **Desktop** – Desktop cjelina kao i Mobile, komunicira s poslužiteljem preko REST sučelja. Ova cjelina je namijenjena za vlasnike proizvoda iOrder koji mogu dodavati, brisati te kontrolirati sve svoje resurse.
- **DB** – SQL baza podataka koja „živi“ u Microsoft SQL Serveru na Azure cloud servisu. Na bazu se spaja jednio Backend i to preko NHibernate ORM-a.

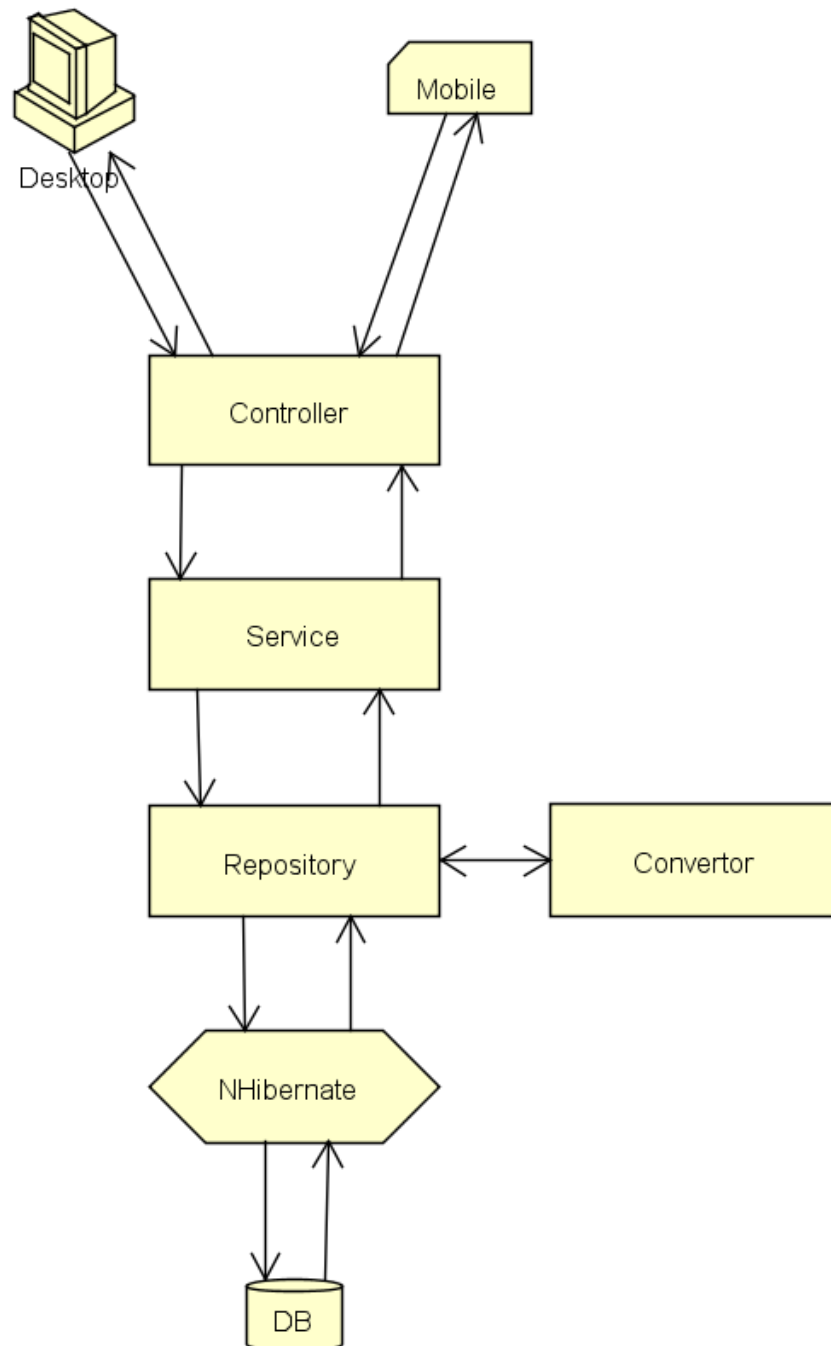


Slika 14 Arhitektura sustava iOrder

## 6.2. Arhitektura backend poslužitelja

Arhitektura poslužitelja sastoji se od logički odvojenih slojeva. Svaki sloj ima svoju zadaću i obavlja samo nju. Time se pridržavamo SoC načela (Seperation of Concerns principle).

Odvajanjem logički drugačijih radnji programa u slojeve uvelike olakšava testiranje pojedinog sloja te iskorištava sve prednosti TDD (Test Driven Development)



Slika 15 Arhitektura Backend-a

Poslužitelj je podijeljen u četiri sloja:

1. **NHibernate** – Sloj perzistencije, odnosno sloj koji spaja podatke jedne tablice iz baze u objekt jednog razreda koji odgovara tablici. Ovaj sloj je implementiran pomoću NHibernate ORM-a (Object-Relational-Mapper). Za mapiranje tablica i klasa korišten je Fluent NHibernate framework koji je dostupan kroz NuGet Package Manager u Visual Studio-u.
2. **Repository** – Sloj koji obavlja sve CRUD operacije za svaki model te dodatne operacije dohvaćanja, spremanja i ažuriranja pojedinačno za svaki model. Sloj repozitorija koristi sloj Convertor za pretvaranje Entity modela u Business modele.
3. **Service** – Sloj koji obavlja svu business logiku sustava. Service koristi repozitorije za dohvaćanje podataka potrebnih za obavljanje neke operacije ili zadatka, spremanje, ažuriranje ili brisanje nekog podatka. Važno je napomenuti da niti jedan sloj osim Service sloja nema pristup Repository sloju. Dakle svaka operacija koja se želi obaviti nad podacima mora ići preko Service sloja koji onda s obzirom na Business logiku pravilno izvršava željenu operaciju.
4. **Controller** – Sloj koji daje vanjskim jedinicama (Mobile i Desktop) pristup mogućnostima koje pruža poslužitelj. Poslužitelj te mogućnosti pruža vanjskim jedinicama preko REST API sučelja. Controller sloj se sastoji od nekoliko REST kontrolera, koji svaki pruža pristup operacijama za jedan tip podataka, odnosno jedan model domene.

Veza između slojeva je ostvarena tehnikom DI (Dependency Injection) koja se definira u Startup.cs razredu projekta. Startup.cs je razred koji je odgovoran za cjelokupnu konfiguraciju poslužitelja.

Primjer definiranja DI:

```
services.AddScoped<IConverter<OrderEntity, Order>, OrderEntityToModelConverter>();  
services.AddScoped<IConverter<Order, OrderEntity>, OrderModelToEntityConverter>();  
services.AddScoped<IOrderRepository, OrderRepository>();  
services.AddScoped<IOrderService, OrderService>();
```

Ovim odsječkom koda za DI su konfigurirana četiri sučelja zajedno sa svojim implementacijama. Za korištenje tih sučelja, razredu je dovoljno imati u konstruktoru argument koji je tipa kao i željeno sučelje i .NET Core će automatski injektirati implementaciju koju smo definirali u Startup.cs razredu za odabrano sučelje.

## ***Implementacija perzistencije***

Spajanje baze podataka s poslužiteljskom logikom i slojem repozitorija implementirano je korištenjem NHibernate ORM-a. Za mapiranje klasa s tablicama baze podataka, korišten je alat Fluent NHibernate. On nam omogućuje lakše

mapiranje jer koristi C# kod umjesto XML-a, što je preglednije te koristi mogućnosti IntelliSense-a.

Za mapiranje potrebno je proširiti razred `ClassMap<T>` iz namespace-a `FluentNHibernate-a`, gdje je `T` razred koji želimo mapirati. Sam kod za mapiranje se nalazi unutar praznog konstruktora proširenog razreda.

```
public class OrderMap : ClassMap<OrderEntity>
```

Primjeri nekih od razreda mapiranja:

`OrderMap` – razred mapiranja za razred `OrderEntity`

```
public OrderMap()
{
    Table("[order]");
    Schema("dbo");
    Id(o => o.Id).Column("id").GeneratedBy.Native();
    Map(o => o.Date).Column("date");
    Map(o => o.Paid).Column("paid");
    Map(o => o.EstablishmentId).Column("establishment_id");
    Map(o => o.CustomerId).Column("customer_id");
    Map(o => o.EmployeeId).Column("employee_id");
    Map(o => o.LocationId).Column("location_id");
    HasMany(x => x.OrderPairs).Not.LazyLoad()
        .Inverse()
        .Cascade.SaveUpdate();
}
```

Opis metoda mapiranja:

**Table()** - odabir tablice u bazi koju želimo mapirati

**Schema()** - odabir scheme unutar koje se nalazi naša tablica

**Id()** - mapiranje primarnog ključa s stupca u bazi na atribut u razredu

**Column()** - označava ime stupca u kojemu se nalaze podaci za odabrani atribut

**GeneratedBy.Native()** – prepušta generiranje primarnog ključa bazi

**Map()** – mapira jedan na jedan atribut i stupac u tablici

**HasMany()** – govori NHibernate-u da ovaj razred ima više djece u tablici s kojom je mapiran razred koji se nalazi u listi `OrderPairs`, a u ovom slučaju je to `OrderPairEntity`.

**Not.LazyLoad()** – sva djeca će biti učitana u potpunosti i tek će onda biti dostupan objekt razreda `OrderEntity`

**Inverse()** – označava da dijete ima referencu na roditelja

**Cascade.SaveUpdate()** – prilikom spremanja ili ažuriranja razreda `OrderEntity`, promjene će se odnositi i na `OrderPairEntity`.

`OrderPairMap` - razred mapiranja za razred `OrderPairEntity`

```
public OrderPairMap()
{
    Table("order_pair");
    Schema("dbo");
    Id(op => op.Id).Column("id").GeneratedBy.Native();
    Map(op => op.Quantity).Column("quantity");
    References(op => op.Order).Column("order_id").Not.LazyLoad();
    Map(op => op.ProductId).Column("product_id");
    References(op =>
op.Product).Column("product_id").Not.LazyLoad().ReadOnly();
}
```

Opis metoda mapiranja:

**References()** – tablicu kojoj odgovara strani ključ mapira u razred koji je odabran unutar zagrada

**ReadOnly()** – govori NHibernate-u da prilikom spremanja, ažuriranja ili brisanja zanemari ovaj atribut

### 6.3. Slojevi

#### Repository

Implementacija Unit of Work je realizirana kroz repozitorije i NHibernate ISession. Razred koji se bavi općenitim spremanjem, brisanjem, dohvaćanjem te ažuriranjem podataka odnosno CRUD operacijama je BaseRepository<T>. Gdje T označava razred nad kojim vršimo operacije.

Primjer implementacije CRUD operacija:

- Spremanje:

```
public virtual object Save(T t)
{
    using (var db = NHibernateHelper.OpenSession())
    {
        using (var transaction = db.BeginTransaction())
        {
            var savedId = db.Save(t);
            transaction.Commit();
            return savedId;
        }
    }
}
```

- Dohvaćanje:

```
public virtual IEnumerable<T> GetAll()
{
    using (var db = NHibernateHelper.OpenSession())
    {
        return db.Query<T>().ToList();
    }
}
```

```
public virtual T GetById(object Id)
{
    var t = new Object();
    using (var db = NHibernateHelper.OpenSession())
    {
        t = db.Get<T>(Id);
    }
    return (T) t;
}
```

- Ažuriranje:

```
public virtual object Update(object Id, T t)
{
    using (var db = NHibernateHelper.OpenSession())
    {
        using (var transaction = db.BeginTransaction())
        {
            db.Update(t);
            transaction.Commit();
            return Id;
        }
    }
}
```

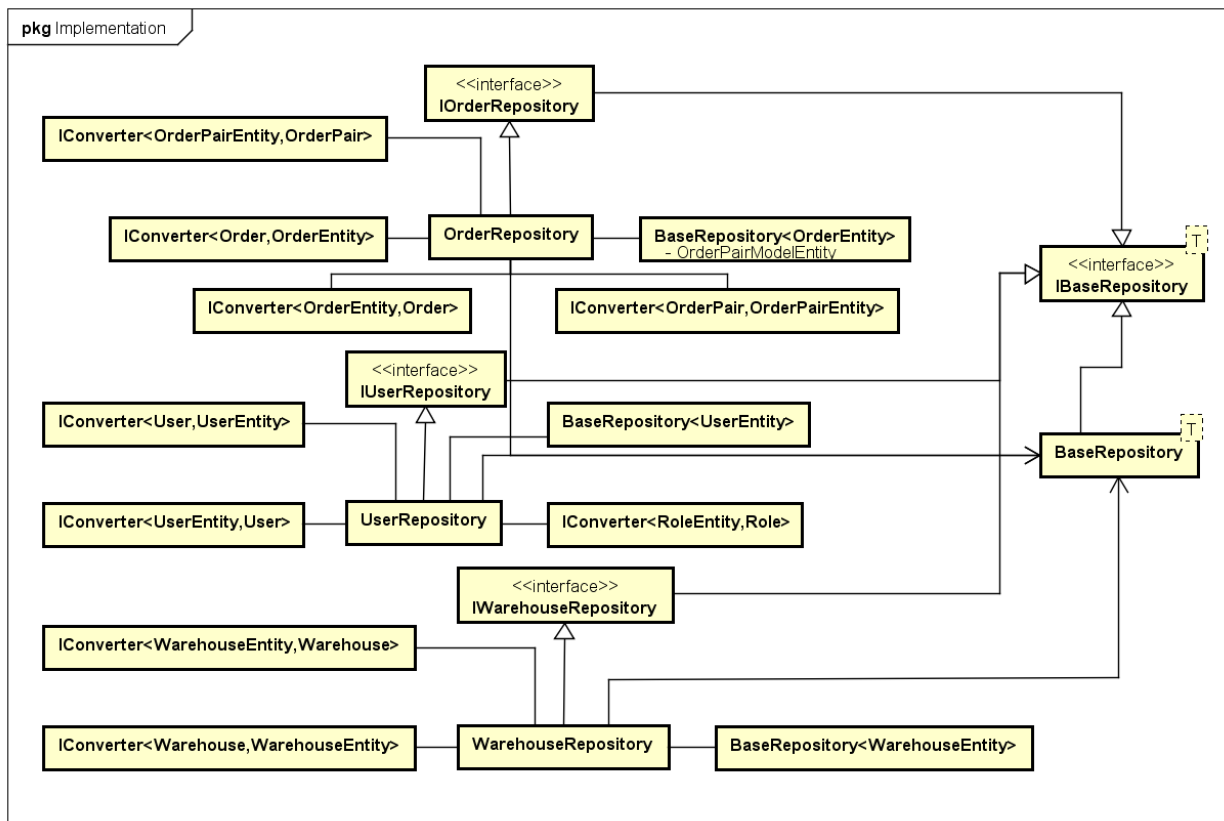
- Brisanje:

```
public virtual bool Delete(T t)
{
    using (var db = NHibernateHelper.OpenSession())
    {
        using (var transaction = db.BeginTransaction())
        {
            try
            {
                db.Delete(t);
                transaction.Commit();
                return true;
            }
            catch (Exception e)
            {
                return false;
            }
        }
    }
}
```

Za vrijeme bilo koje interakcije s bazom potrebno je otvoriti sesiju. Za otvaranje sesija koristi se razred NHibernateHelper koji vraća otvorenu sesiju ISession s kojom dalje možemo raditi Query i ostale pozive. Ukoliko se vrši akcija koja zahtjeva mijenjanje podataka potrebno je otvoriti transakciju. To se radi metodom BeginTransaction() nad otvorenom sesijom. Nakon završetka operacije potrebno je zaključati i završiti transakciju pozivom metode Commit() nad otvorenom transakcijom. Nakon toga, promjene su sinkronizirane s bazom, odnosno u bazi se nalaze napravljene promjene.

Repozitorije koriste servisi i to pomoću DI. Repozitoriji servisima šalju business modele dok NHibernate radi s entity modelima. Tim razdvajanjem pospješujemo načelo SoC.

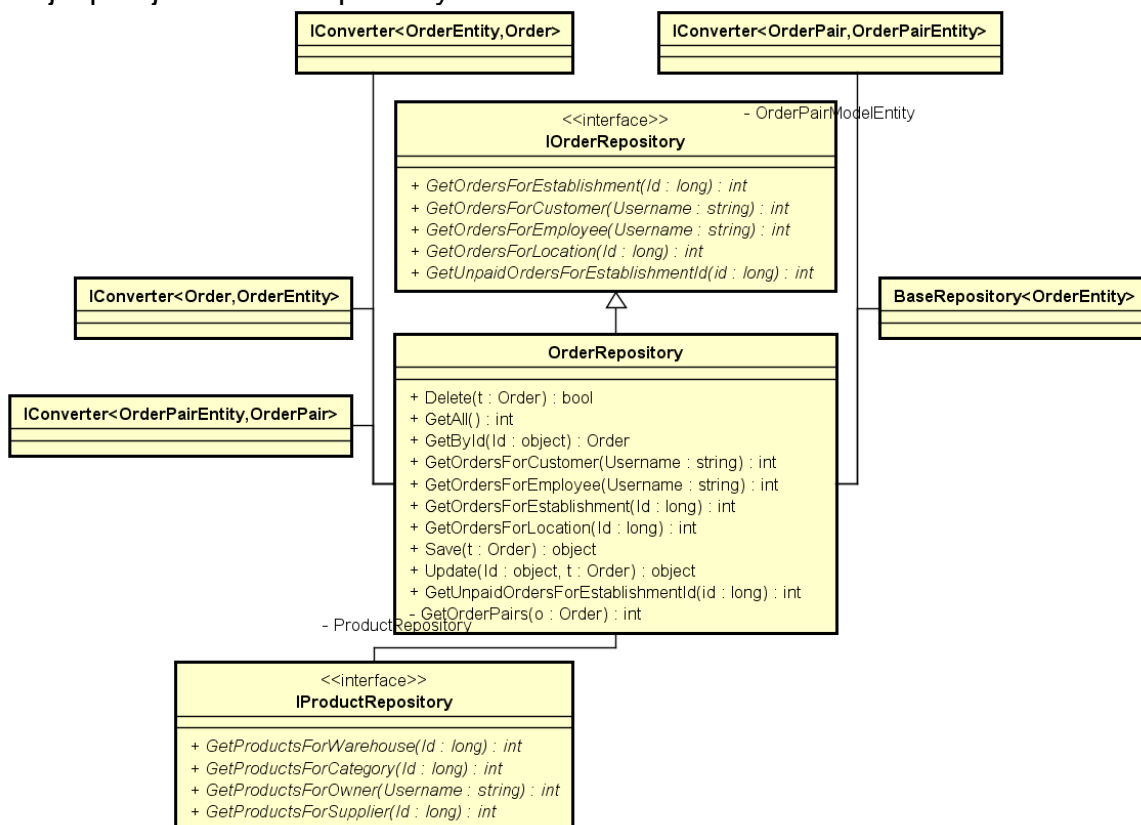
Kako bi vratili dobre tipove objekata servisima repozitoriji koriste pretvarače odnosno Converter sloj, čija je zadaća pretvoriti entity model u business i obrnuto. Primjer organizacije razreda za sloj repozitorija:



Slika 16 Dijagram razreda repozitorija

Na dijagramu su prikazani samo neki od postojećih repozitorija kako bi se povećala vidljivost.

Detaljni primjer - OrderRepository:

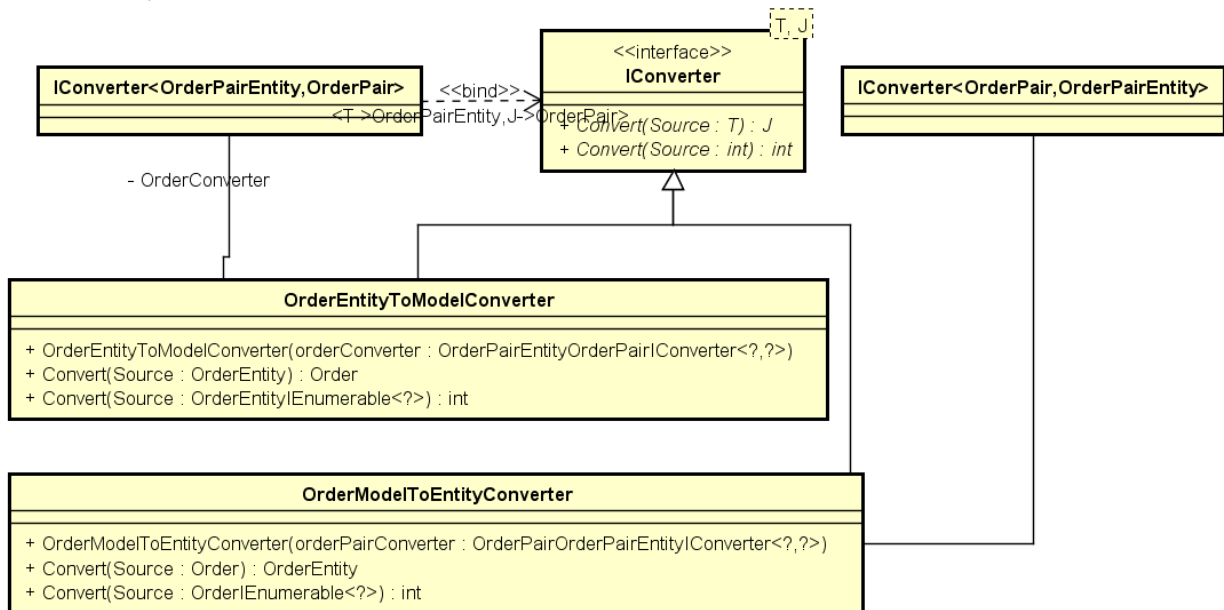


Slika 17 Dijagram razreda OrderRepository

## Converter

Uloga sloja Converter je jednostavna, a to je pretvaranje modela iz business u entity i obrnuto. Ovim slojem je skinuta odgovornost s Repository sloja. Ukoliko se desi promjena u entity ili business modelu, potrebno je promijeniti samo sloj pretvarača i to samo na dva mjesta, jedan u smjeru entity - business i drugi u obrnutom. Time smo se osigurali od promjena baze i business modela.

Primjer organizacije razreda za sloj pretvarača:



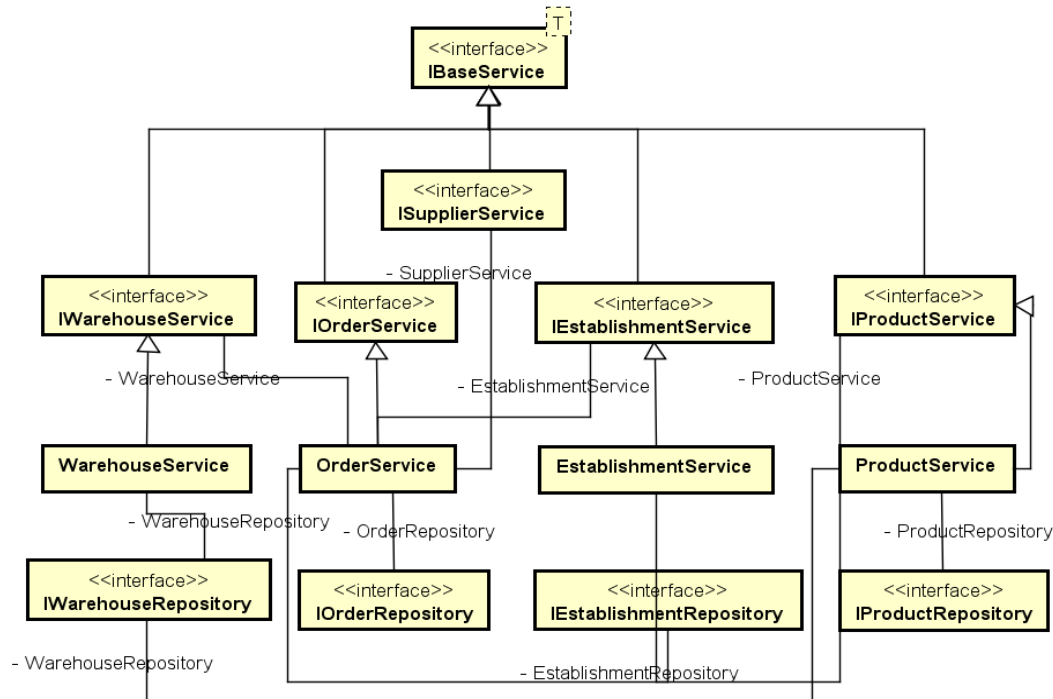
Slika 18 Dijagram razreda pretvarača

## Service

Sloj koji obavlja svu business logiku vezanu za cijeli sustav. Sloj koristi repozitorije za manipulaciju podacima. Service sloj dobiva repozitorije i ostale servise pomoću DI u konstruktoru. Ovaj sloj također koristi Frontend preko svoje reference na Backend projekt kao i sloj Controller. Svaki servis implementira sučelje koje odgovara tom modelu, a svako sučelje nasljeđuje osnovno sučelje **IBaseService** koje propisuje metode koje moraju imati svi servisi.

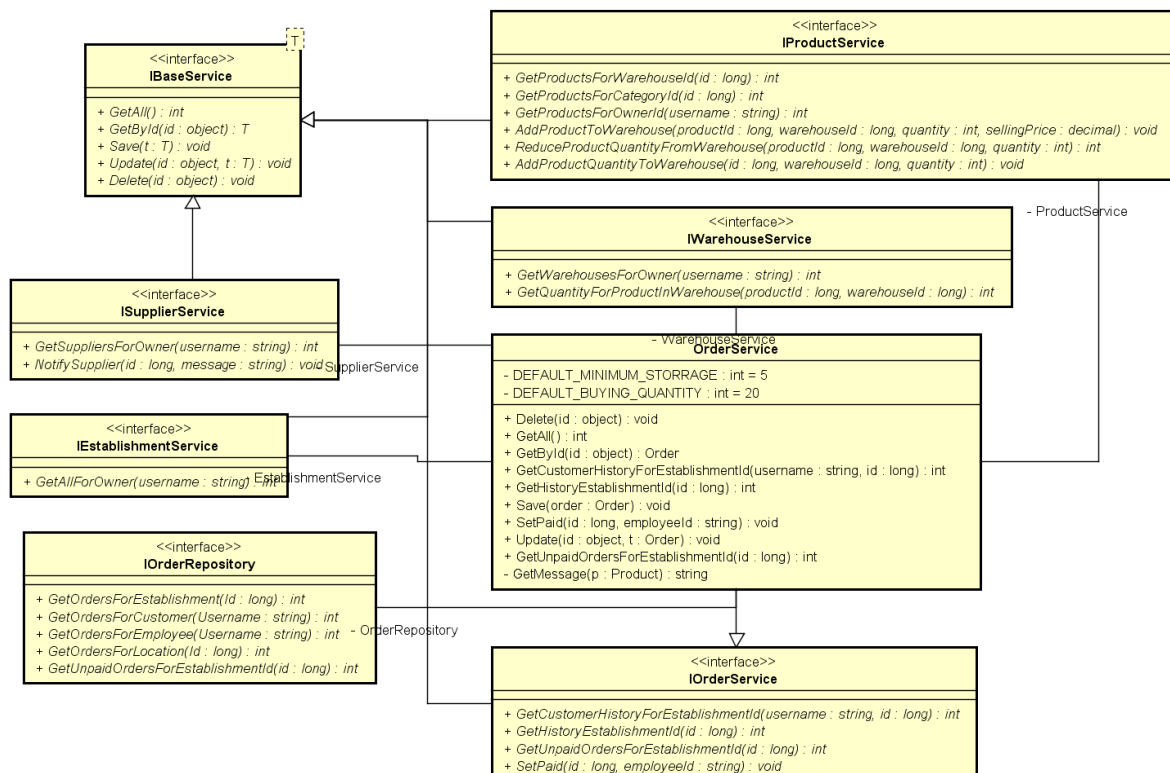


Primjer organizacije razreda za sloj servisa:



Slika 19 Dijagram razreda servisa

Detaljni primjer – OrderService



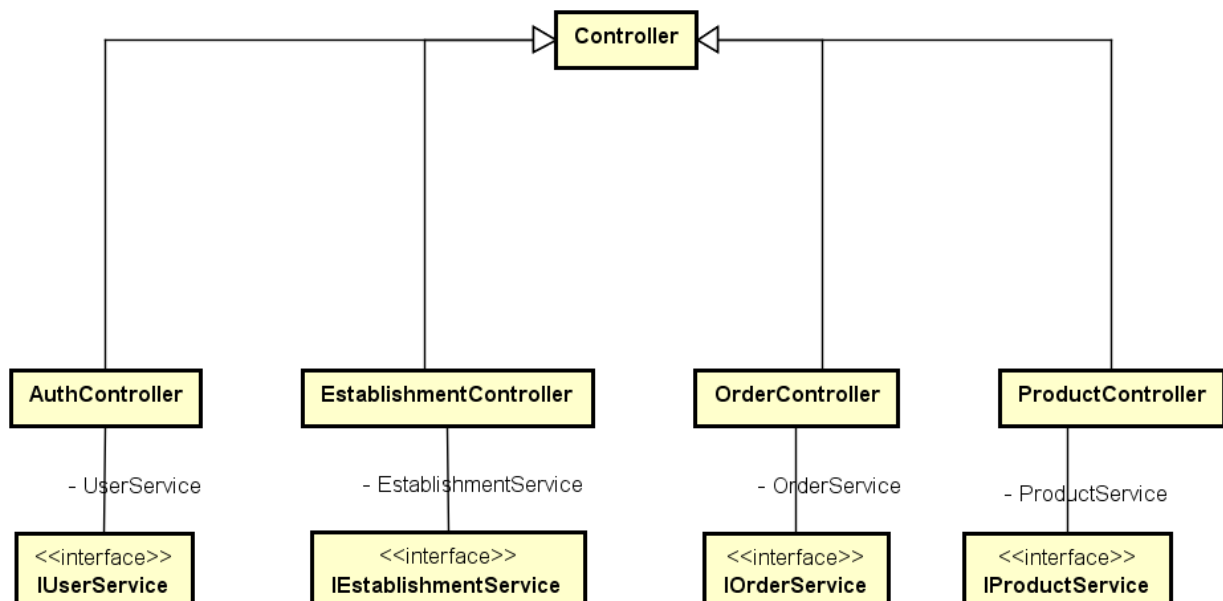
Slika 20 Dijagram razreda OrderService

## Controller

Sloj koji vrši komunikaciju s vanjskim jedinicama preko REST sučelja. Sloj koristi servise za obavljanje svih operacija. Ovaj sloj proslijeđuje sve podatke servisnom

sloju i u sebi ne sadrži nikakvu logiku. Time postaje neovisan na promjene u business logici ili modelima.

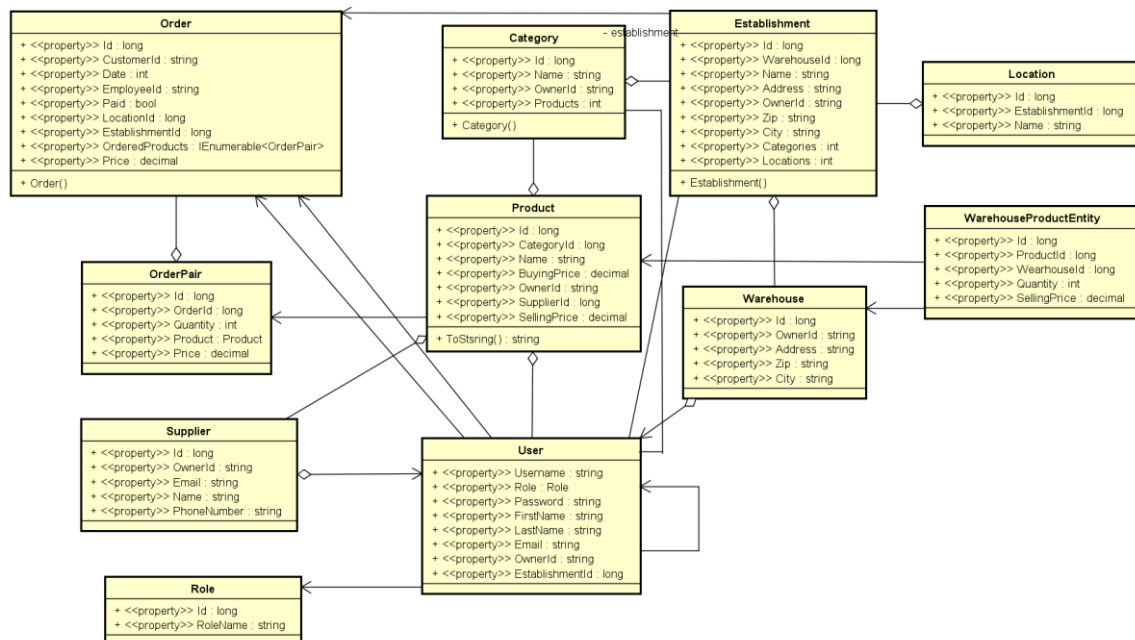
Primjer organizacije razreda za sloj kontrolera:



Slika 21 Dijagram razreda kontrolera

## Business model

Business ili domain modeli kao i njihove veze su prikazani na slijedećoj slici.



Slika 22 Dijagram razreda business modela

## 6.4. Sigurnost

Sigurnost za autorizaciju i autentikaciju riješena je pomoću tehnike tokena. Ideja iza ove tehnike jest da korisnik preko POST metode na REST sučelje kontrolera za sigurnost pošalje svoje podatke, korisničko ime i lozinka, a poslužitelj

nakon potvrde korisničkih podataka vraća generirani token. Token se sastoji od korisničkih podataka i uloge kojoj taj korisnik pripada. S obzirom da token u sebi ima informaciju o ulozi, kontroler može propustiti korisničke zahtjeva tamo gdje taj korisnik ima dopuštenje ili poslati Code 401 (Unauthorized) tamo gdje korisnik nema dopuštenje pristupiti.

Primjer dodavanja sigurnosti u aplikaciju u Startup.cs:

```
services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme =
JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(jwtBearerOptions =>
{
    jwtBearerOptions.TokenValidationParameters = new
TokenValidationParameters()
{
    ValidateActor = true,
    ValidateAudience = true,
    ValidateLifetime = true,
    ValidateIssuerSigningKey = true,
    ValidIssuer = "iOrder.fer.hr",
    ValidAudience = "iOrder.fer.hr",
    IssuerSigningKey =
        new
SymmetricSecurityKey(Encoding.UTF8.GetBytes("System.ArgumentOutOfRangeException"))
    };
});
app.UseAuthentication();
```

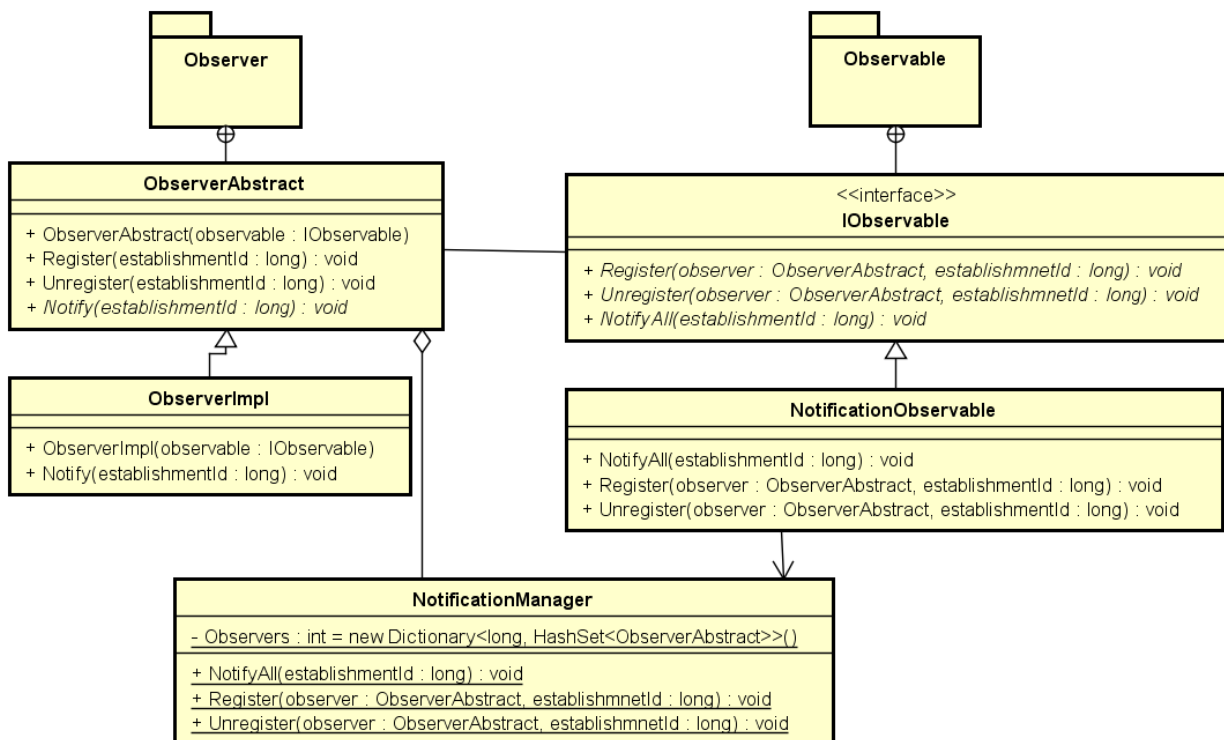
## 6.5. Obavijesti

Obavješćavanje Frontend-a o novoj narudžbi implementirano je oblikovnim obrascem Observable ili promatrač.

Frontend se registrira na promjene za onaj id uslužnog objekta za koji je instanca Frontend-a otvorena, odnosno za onaj establishmentId koji pripada zaposleniku koji se prijavio na Frontend.

Kada korisnik napravi narudžbu, ona se spremi u bazu i nakon toga OrderService pomoću statičnog razreda NotificationManager šalje obavijest svim registriranim instancama Frontend-a koji su pretplaćeni na establishmentId koji se nalazi u narudžbi.

Dijagram razreda implementacije obavijesti:



Slika 23 Dijagram razreda promatrača obavijesti

Frontend će proširiti apstraktni razred ObserverAbstract i implementirati svoju metodu Notify().

Metodu Notify() pozove NotificationManager na svakom promatraču koji se nalazi u Dictionary-u na ključu establishmentId-a na kojoj se napravila narudžba.

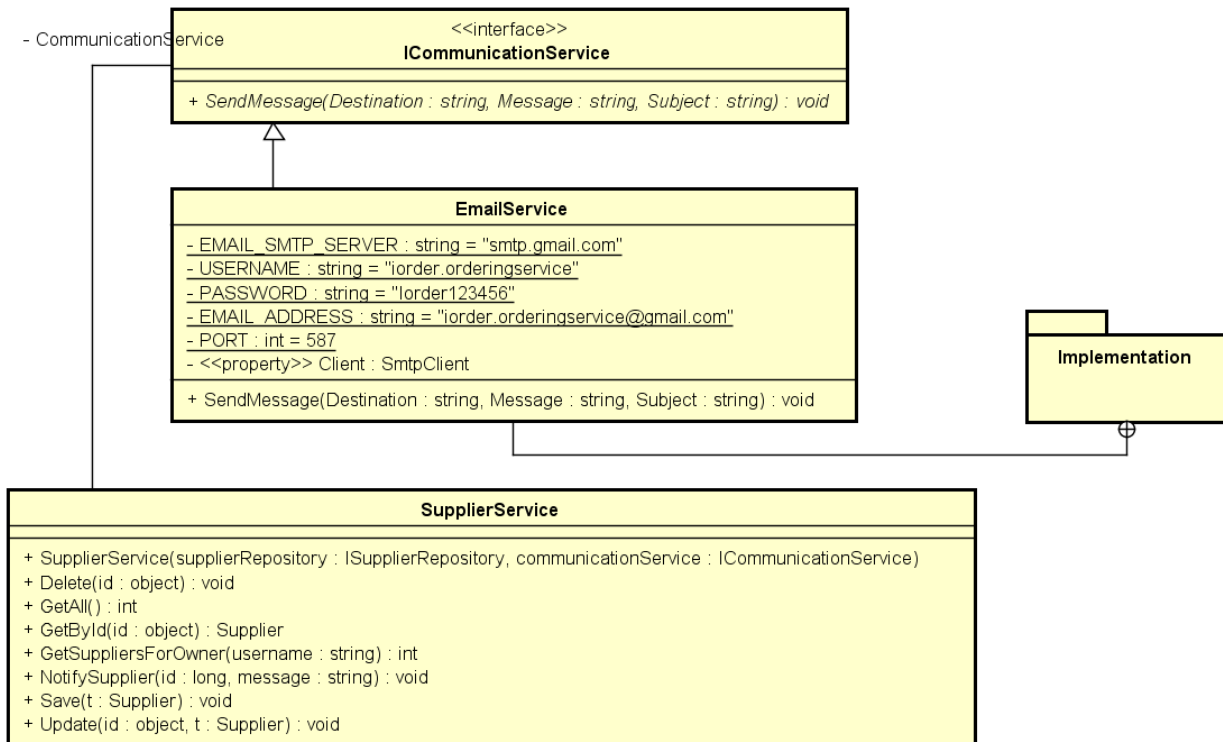
Frontend svojom implementacijom Notify() metode određuje radnje koje će izvršiti nakon dolaska nove narudžbe.

## 6.6. Komunikacijski servis

Komunikacijski servis služi za slanje poruka vanjskim jedinicama (email, SMS). U ovom projektu je implementirano slanje putem email-a i to servisom EmailService. U ovom slučaju email poruke se automatski šalju dobavljačima kada količina proizvoda u skladištu padne ispod minimalne razine. U poruci se navodi proizvod koji je potrebno dobiti, količina te vlasnik skladišta kojem to treba dohvatiti.

U budućnosti moguće je implementirati i slanje poruka dobavljačima putem SMS-a uz minimalne promjene u OrderService-u. Implementacija treba implementirati sučelje ICommunicationService i promijeniti implementaciju za DI u Startup.cs.

Dijagram razreda servisa za komunikacije:



Slika 24 Dijagram razreda servisa za komunikacije

## 6.7. Baza podataka

Baza podataka je napravljena u SQL-u na Microsoft SQL Server-u. Sama baza se nalazi na Azure-u zajedno s Backend i Frontend jedinicama.

### Tablice

U nastavku će biti ukratko opisana svaka tablica te za pojedine pojašnjeni stupci.

**Role** – uloga koju ima svaki korisnik u sustavu

**User** – tablica gdje se spremaju korisnici u sustavu –

ownerId, establishmentId – strani ključevi koje koriste korisnici koji su zaposlenici

**Warehouse** – tablica gdje se spremaju podaci o skladištima

**Establishment** – tablica gdje se spremaju uslužni objekti

**Location** – tablica gdje se spremaju lokacije za pojedini uslužni objekt

**Category** – tablica gdje se spremaju kategorije proizvoda

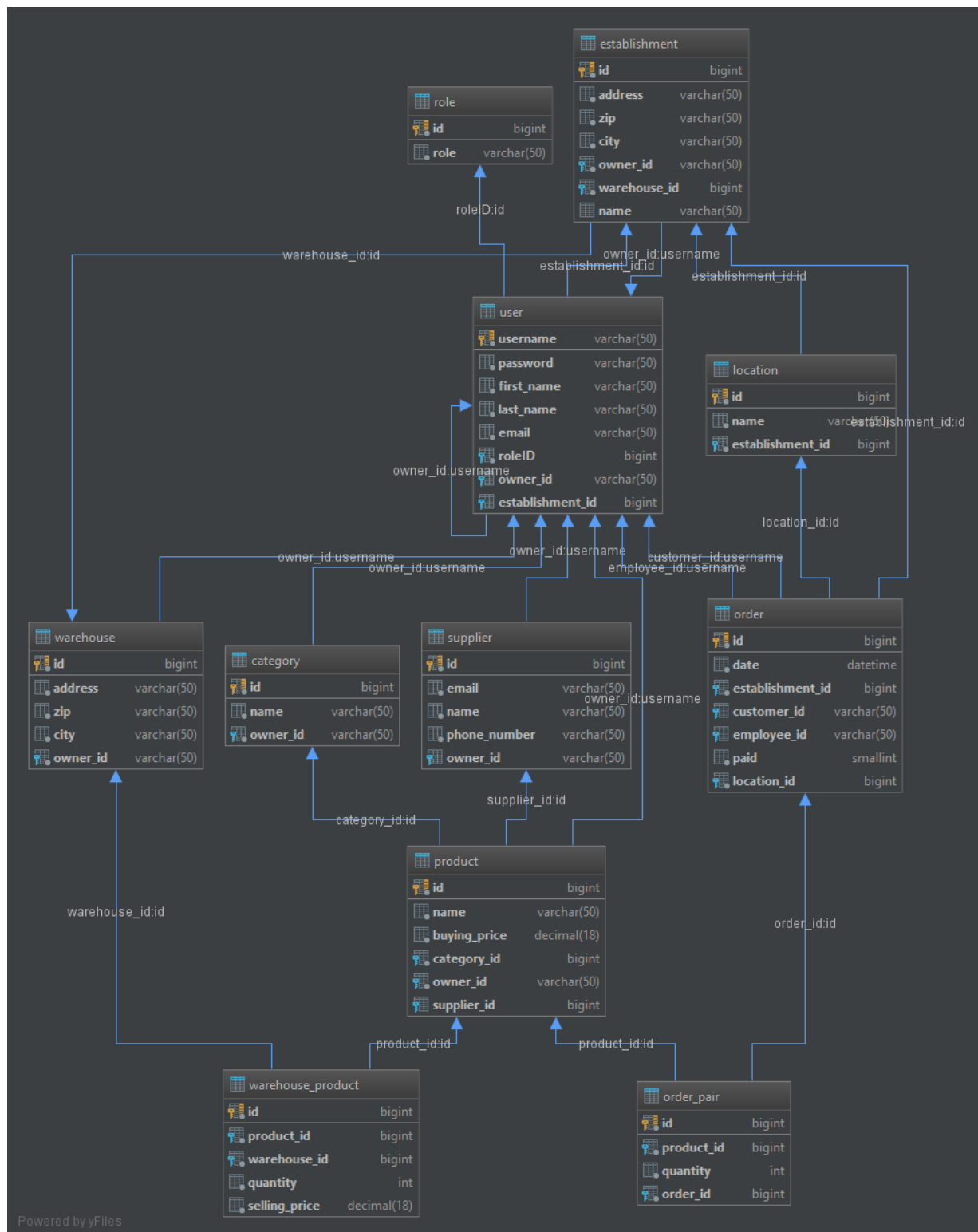
**Product** – tablica gdje se spremaju proizvodi

**Supplier** – tablica gdje se spremaju dobavljači

**Order** – tablica gdje se spremaju narudžbe

**OrderPair** – tablica gdje se sprema par (productId, quantity) za svaku narudžbu odnosno orderId

**WarehouseProduct** – tablica gdje se sprema par (productId, quantity) za svaki warehouseId, odnosno sprema se količina proizvoda koja je dostupna za pojedino skladište



Slika 25 Tablice i relacije u bazi podataka

## 6.8. Arhitektura Desktop aplikacije

Desktop aplikacija napisana je u duhu MVC-a. Aplikacija je sastavljena od Windows forma koje predstavljaju grafiku (View) koju korisnik vidi i pomoću koje unosi podatke u sustav. Drugi dio aplikacije su kontroleri (Controller) koji služe za dohvaćanje podataka sa udaljenog poslužitelja i punjenja podataka u modele. Treći dio aplikacije čine modeli (Model) koji opisuju podatke primljene od kontrolera da bi se mogli prikazati u formama. Ovakvim načinom izrade aplikacije olakšava se nadogradnja same aplikacije i programski kod je lakše čitati.

### **Opis modela razreda**

**Modeli** su domenski objekti koji predstavljaju samu domenu aplikacije. Sadrže attribute koji opisuju domenu aplikacije i služe kao poveznica između objekata koji se dobivaju sa poslužitelja te Windows formi koji čine grafiku aplikacije.

**Kontroleri** nam služe za dohvat podatak sa servera i stvaranje i slanje na poslužitelj novih podataka koje korisnik želi pospremiti u sustav

**Forme** u aplikaciji predstavljaju poglede odnosno grafički dio aplikacije koje korisnik može vidjeti i sa kojima može interaktivno komunicirati kako bi obavio svoj posao.

### **Opis interakcije sa sustavom**

Početni zaslon aplikacije prikazuje formu za korisničku prijavu. Uspješnom korisničkom prijavom poslužitelj vraća **token** koji se šalje u zaglavlju svakog idućeg zahtjeva.

Kada korisnik ima **token** koji je dobio od poslužitelja on može pregledavati, dodavati nove, modificirati i brisati postojeće podatke u sustavu za upravljanje poslovanjem. Korisnik kroz jednostavan i intuitivan **gui** dolazi do željenih podataka te nad njima ima potpunu kontrolu. Poslužitelj korisnika raspoznaje pomoću **token-a** dobivenog prilikom prijave. Tipovi podataka kojima korisnik može upravljati su: skladišta, kategorije proizvoda, proizvodi, lokacije, uslužni objekti, zaposlenici i dobavljači proizvoda.

### 6.9. Arhitektura Web aplikacije

Web aplikacija tj. "Frontend" je napisan pomoću tehnologije ASP.NET MVC za .net core. ASP.NET je već sam po sebi odlično napravljen za MVC paradigmu te je na taj način napravljena i web aplikacija.

Ona se sastoji od dva glavna dijela, a to su prozor za autentikaciju korisnika te prozor za interakciju zaposlenika s aplikacijom koji je tek dostupan nakon uspješne autentikacije. Aplikacija ne koristi REST za dohvat podataka već se direkto integrira s ponuđenim servisima iz "Backend" projekta.

#### **Opis modela razreda**

**Modeli** u web aplikaciji su modeli iz Shared Library-a tj. prakticira se reusability. Postoje 3 dodatna modela ali oni su striktno vezani uz autentikaciju korisnika te domenu same web aplikacije.

**Kontroleri** su podijeljni u funkcionalne skupine. Postoje 4 kontrolera podijeljnih zadaćama a to su kontroleri za povijest narudžbi, za autentikaciju, za generiranje QR kodova te za interakciju s samim narudžbama u stvarnome vremenu.

**Pogledi** su zapravo html stranice koje se dinamički generiraju. Za osvježavanje pogleda se koristi JQuery i AJAX tehnologija.

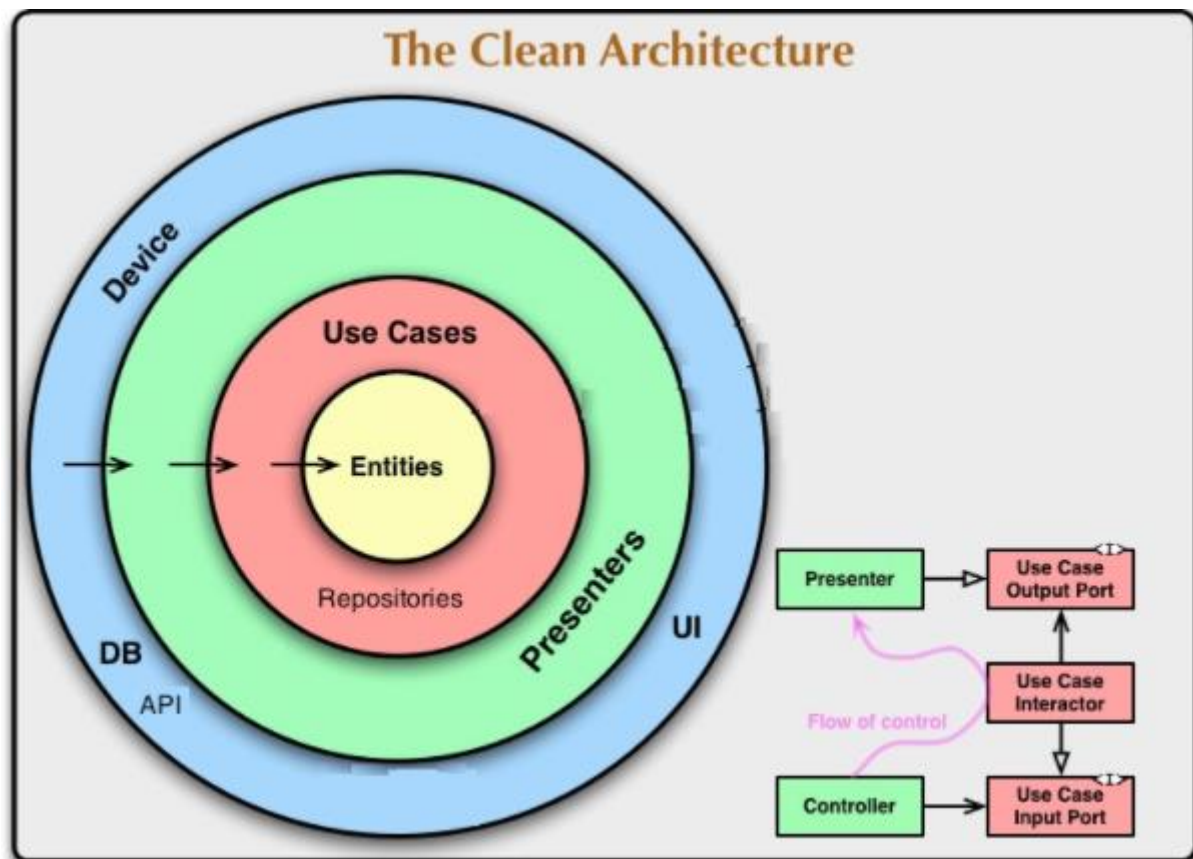
Važno je za spomenuti da se komunikacija između glavnog view-a za pregled trenutnih narudžbi koristi biblioteka SignalR koja obavijesti trenutno autenticiranog korisnika da je pristigla nova narudžba pri čemu se osvježi lista narudžbi.

Korišten je i oblikovni obrazac Observer koji se koristi da "Backend" tj. server obavijesti zaposlenika pri promjeni u njegovoj listi narudžbi. SignalR i obrazac Observer si potpomažu za stvaranje komunikacije u stvarnome vremenu.



### 6.10. Arhitektura Android aplikacije

Srž arhitekture Android aplikacije predstavlja **Clean arhitektura** uz standardni **MVP**. Glavna značajka ove arhitekture je odvojiti odgovornosti i razdvojiti dijelove Android sustava s poslovnom logikom aplikacije. Na taj način svaki dio aplikacije postaje zasebna jedinica koju je lako nadograđivati ili zamijeniti nekom drugom što uvelike olakšava testiranje aplikacije kao i snalaženje u samom kodu.



Clean arhitektura

Iz priložene slike jasno se vidi separacija odgovornosti kao i pravilo ovisnosti. Objekti koji su u plavom krugu ovise o svim objektima zelenog kruga što znači da mogu referencirati bilo što u plavom krugu i unutrašnjim krugovima. S druge strane, objekti zelenog kruga ne mogu referencirati nijedan objekt plavog kruga jer ne zna za njega. Ovakva raspodjela ovisnosti postiže se odvajanjem dijelova aplikacije u module.

#### Opis modela razreda

**Entiteti** su domenski objekti koji čine srce svake Android aplikacije tj. njezine poslovne logike. Najčešće sadrže polja unutar sebe koja pobliže opisuju domenu aplikacije i služe kao vršni sloj između objekata koji se dobivljaju iz nekog izvora

(baza podataka/API) te objekata koji čine ViewModele tj. modele koji se renderiraju na Viewu (Activity ili Fragment).

Oko entiteta se pojavljuju razne vrste objekata poput: **servisa, klijenata, mapera, repozitorija i use caseova**.

**Servisi** su sučelja prema API-u ili bazi podataka koje pozivaju **klijenti**.

**Klijenti** za sobom vuku tzv. API/DbEntity modele koji definiraju pristupne objekte na koje se lijepe podaci s API-a ili baze podataka. Nad tim objektima je potrebno izvršiti konverziju u domenske objekte (**entitete**) korištenjem **mapera**.

**Maperi** imaju niz metoda koje najčešće korištenjem tokova prolaze kroz API modele i vraćaju podatke u obliku koji odgovara **repozitorijima**. Oni su **centralni** dio izvršavanja svih operacija nad podacima i služe kao mjesto gdje se održavaju reference na **klijente, mapere** i ostale dodatne klase koje sudjeluju u ispravnom formatiranju podataka.

**Use caseovi** su zapravo klase koje implementiraju sučelje s jednom metodom **execute()** te održavaju referencu na **repozitorij**. Svojim nazivom potpuno objašnjavaju svoju ulogu u sustavu. Oni se korištenjem injekcije ovisnosti pridružuju kao članske varijable **prezentera**.

**Prezenteri** su dijelovi Android aplikacije koji komuniciraju s onime što se prikazuje na ekranu mobitela te reagiraju na bilo kakve interakcije korisnika s ekranom. Moraju biti izvedeni na način da pokreću use caseove ovisno o događajima na ekranu. Imaju slabu referencu (**weak reference**) na svoj pridruženi **view** (ekran) kako bi se izbjeglo curenje memorije (**memory leak**) pošto se može dogoditi da prezenteri nadžive svoj view u ciklusu života Android aplikacije.

**Pogledi** (view) su aktivnosti i fragmenti koji prikazuju sadržaj na ekranu mobitela i imaju definirane poveznice na svaki dio ekrana.

## **Opis modula**

Cijeli ovaj opisani sustav je **reaktivan** (korištenjem RxJava) i usklađen pomoću **injekcije ovisnosti** (korištenjem Daggera). Sama aplikacija je razdvojena na **app, data i domain** modul.

**Domain** modul čine klase koje se na gore prikazanoj slici nalaze u žutom i crvenom krugu. Te klase su implementirane u običnom Java kodu.

Modul koji je zadužen za perzistenciju i dobavljanje podataka jest **data** modul. Iako bi on sam po sebi trebao biti neovisan o Android sustavu, zbog različitih sučelja kojima se pristupa bazi podataka ili API-u potrebno ga je proglasiti Android knjižnicom.

**App** modul je dio aplikacije u kojem se nalaze sve klase potrebne za inicijalizaciju aplikacije, prikaz podataka na ekranu preko korisničkog sučelja i pokretanje use caseova.

## **Opis interakcije sa sustavom**

Korištenjem Android programskog alata **Retrofit** uvelike je olakšana komunikacija s API-em. Klasa **IOrderService** pruža listu endpointa na koje se aplikacija priključuje pri dobavljanju i slanju podataka. Podatke koje takav servis može primiti moraju biti propisanog JSON oblika kako bi se polja u API modelima mogla ispravno mapirati.

Prvi ekran aplikacije prikazuje formu za korisničku prijavu. Na istom ekranu nalazi se i poveznica do ekrana za registraciju anonimnih korisnika. Uspješnom korisničkom prijavom dobije se **token** koji se šalje u zaglavlju svakog idućeg zahtjeva.

Nepobitna činjenica ove aplikacije je da mora imati pristup kameri mobitela koji se dobije direktno od korisnika. Ukoliko to ne bude slučaj, daljnji rad aplikacije je obustavljen dok god korisnik ne odluči koristiti aplikaciju na pravi način. Kamera je potrebna kako bi se uspješno mogao očitati QR kod koji definira uslužni objekt i lokaciju.

Uspješnim skeniranjem, uslužni objekt dobije zahtjev za dohvaćanjem svog menija (izbornika) koji bi trebao sadržavati popis svih kategorija i proizvoda tog objekta. Usputno se identificira lokacija korisnika unutar objekta tako da prilikom narudžbe ne može doći do zabune.

Narudžba se sastoji od sljedećih polja: lista odabranih produkata, identifikator lokacije i uslužnog objekta. **Backend poslužitelj** iz tokena može raspoznati koji je korisnik poslao zahtjev te sve narudžbe jednog korisnika sprema u bazu podataka. Aplikacija ima i mogućnost pregleda povijesti narudžbi za trenutni uslužni objekt u kojem se korisnik nalazi.

## 7. Korištene tehnologije i alati

**Android** aplikacija programirana je pomoću **Android Studio** u programskom jeziku **Java**.

**Desktop** i **Web** aplikacija te **Backend poslužitelj** programirani su u **Microsoft Visual Studio** u programskom jeziku **C#**.

Za timsku sinkronizaciju i pisanje prvih verzija zahtjeva te stvaranje pojedinih dijelova arhitekture korišten je alat **OneNote**.

Use case te ostali dijagrami stvoreni su u okruženju **Astah Professional** koji čini programsku potporu za crtanje UML dijagrama.

Astah Professional link: <http://astah.net/editions/professional>.

Korišten je **Github** kao open source git repozitorij za čuvanje verzija projekta.

Github link: <https://github.com/>