SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 793

WEB APLIKACIJA ZA E-TRGOVINU U ARHITEKTURI ZASNOVANOJ NA MIKROUSLUGAMA S UKLJUČENIM SUSTAVOM PREPORUČIVANJA

Katarina Bošnjak

Zagreb, srpanj 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 793

WEB APLIKACIJA ZA E-TRGOVINU U ARHITEKTURI ZASNOVANOJ NA MIKROUSLUGAMA S UKLJUČENIM SUSTAVOM PREPORUČIVANJA

Katarina Bošnjak

Zagreb, srpanj 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 3. ožujka 2025.

DIPLOMSKI ZADATAK br. 793

Pristupnica:	**Katarina Bošnjak (0036524178)**
Studij:	Računarstvo
Profil:	Znanost o podacima
Mentor:	izv. prof. dr. sc. Alan Jović

Zadatak: **Web aplikacija za e-trgovinu u arhitekturi zasnovanoj na mikrouslugama s uključenim sustavom preporučivanja**

Opis zadatka:

Suvremeni sustavi za e-trgovinu zasnovani su na više inteligentnih rješenja, uključujući pravilnu raspodjelu usluga po slojevima web aplikacije i preporučivanje novih artikala na temelju ranijih korisničkih preferenci. U ovom diplomskom radu potrebno je osmisliti i razviti web aplikaciju za e-trgovinu koja će biti zasnovana na arhitekturi mikrousluga. Aplikacija treba omogućiti preporučivanje novih artikala korisnicima na temelju njihovih ranijih kupovina, korisničkih profila te preferenci ostalih korisnika. Vezano za same ponuđene usluge, e-trgovina bi nužno uključivala uslugu narudžbe robe, pregled inventara i dostavu, a po mogućnosti i katalog proizvoda. Mikrousluge trebaju komunicirati preko REST API-ja i koristeći Apache Kafku. Korisničko sučelje treba omogućiti izradu narudžbi, pregled proizvoda i administracijski panel s pregledom ranijih kupovina te odgovarajućom vizualizacijom i statistikom. Podaci za učenje sustava preporučivanja mogu se preuzeti iz slobodno dostupnih internetskih repozitorija podataka (npr. Kaggle).

Rok za predaju rada: 4. srpnja 2025.

Iskreno zahvaljujem roditeljima i bratu što su mi bili oslonac kroz cijeli fakultet. Bez njihove podrške, strpljenja i vjere u mene ovaj put ne bih uspjela proći. Ova diploma je i vaša koliko i moja.

Hvala prijateljima koji su me bodrili, nasmijavali i podržavali tijekom svih izazova.

Zahvaljujem mentoru izv. prof. dr. sc. Alanu Joviću na vodstvu, strpljenju i korisnim savjetima.

Sadržaj

1. Uvod 3

2. Arhitektura i oblikovanje sustava 4

2.1. Mikrouslužna arhitektura 4

2.2. Komponente aplikacije 6

2.3. Podaci 7

2.4. Filtriranje na temelju sadržaja 8

2.5. Infrastruktura 10

2.5.1. PostgreSQL 11

2.5.2. Kafka, Zookeeper, KafkaUI 12

2.5.3. Redis 13

2.5.4. Clickhouse, Debeezium, Kafka - ClickHouse Connector 13

3. Korisnička usluga 17

3.1. API dokumentacija 19

3.1.1. Podatkovni modeli 19

3.1.2. API-jeve krajnje točke 20

4. Usluga kataloga 21

4.1. API dokumentacija 22

4.1.1. Podatkovni modeli 22

4.1.2. API-jeve krajnje točke 23

5. Usluga inventara 25

5.1. API dokumentacija 27

5.1.1. Podatkovni modeli 27

5.1.2. API-jeve krajnje točke 28

6. Usluga naručivanja 30

6.1. API dokumentacija 33

6.1.1. Podatkovni modeli 33

6.1.2. API-jeve krajnje točke 34

7. Usluga otpreme 36

7.1. API dokumentacija 38

7.1.1. Podatkovni modeli 38

7.1.2. API-jeve krajnje točke 39

8. Usluga sustava preporučivanja 40

8.1. API dokumentacija 43

8.1.1. Podatkovni modeli 43

8.1.2. API-jeve krajnje točke 45

9. Usluga analitike 46

9.1. API dokumentacija 47

9.1.1. Podatkovni modeli 47

9.1.2. API-jeve krajnje točke 50

10. Korisničko sučelje 52

11. Zaključak 62

Literatura 63

Sažetak 66

Abstract 67

1. Uvod

S obzirom na sadašnje tehnološko doba, e-trgovina se širi vrlo brzim tempom i bitan je dio modernog poslovanja. Osim općih značajki poput otkrivanja proizvoda, upravljanja narudžbama i praćenja dostave, moderne platforme za e-trgovinu sve više obuhvaćaju inteligentne sustave koji kupcima pružaju personalizirano iskustvo kupnje. Sustavi za preporuku proizvoda, možda najznačajnija komponenta ovih sustava, analiziraju povijest korisnika, njihove profile i preferencije drugih korisnika kako bi preporučili odgovarajuće artikle - poboljšavajući zadovoljstvo korisnika i prodajne rezultate.

S rastućim zahtjevima skalabilnosti, održivosti i fleksibilnosti softverskih sustava, arhitektura temeljena na mikrouslugama (engl. *microservice architecture, microservices*) postaje standard kada je u pitanju izgradnja složenih web aplikacija. Mikrousluge omogućuju pisanje i održavanje aplikacija kao pojedinačnih, neovisnih entiteta koji komuniciraju s dobro dokumentiranim sučeljima - prvenstveno putem REST API-ja, ali sve više i s posrednicima poruka kao što je Apache Kafka.

Cilj rada je oblikovati i implementirati aplikaciju za e-trgovinu koristeći arhitekturu mikrousluga s modulom za preporuku proizvoda. Sustav uključuje kreiranje i upravljanje narudžbama, praćenje zaliha i dostave te katalog proizvoda. Preporuke se temelje na povijesti kupnje i sličnosti proizvoda, a podaci za učenje su izvedeni iz javno dostupnih skupova podataka poput onih dostupnih na Kaggleu.

Korisničko iskustvo je intuitivno, sa sekundarnom administratorskom pločom koja predstavlja pregled prošlih transakcija, s ključnim metrikama i vizualizacijama podataka.

2. Arhitektura i oblikovanje sustava

2.1. Mikrouslužna arhitektura

Mikrouslužna arhitektura je arhitektonski stil programske potpore koji strukturira aplikaciju kao skup od dvije ili više usluga koje su:

- Neovisno implementirane
- Labavo povezane

Usluge su obično organizirane oko poslovnih mogućnosti. Svaka usluga često je u vlasništvu jednog, malog tima [1].

Robert C. Martin stvorio je termin načelo jedinstvene odgovornosti koje kaže „okupite one stvari koje se mijenjaju iz istog razloga i odvojite one stvari koje se mijenjaju iz različitih razloga“. Ovo načelo primjenjuje se i u arhitekturi mikrousluga, koja ga dodatno proširuje kroz labavo povezane usluge koje se mogu razvijati, implementirati i održavati neovisno. Svaka od ovih usluga odgovorna je za zaseban zadatak i može komunicirati s drugim uslugama putem jednostavnih API-ja kako bi riješila veći složeni poslovni problem [2].

Glavne karakteristike mikrouslužne arhitekture [3]:

- Komponenta ili usluga je jedinica softvera koja se može neovisno zamijeniti i nadograditi
- Mikrouslužni pristup je podjela na usluge organizirane prema poslovnim mogućnostima

- Usluge teže biti što je moguće odvojenije i kohezivnije, tako da posjeduju vlastitu poslovnu logiku i primaju zahtjev, primjenjuju logiku i proizvode odgovor koristeći RESTful API-je
- Usluge također decentraliziraju odluke o pohrani podataka. Ovaj pristup se zove *Polyglot Persistence* ili *Polyglot Databases*. To znači da se u mikrouslužnoj arhitekturi preferira da svaka usluga upravlja vlastitom bazom podataka, bilo različitim instancama iste tehnologije baze podataka ili potpuno različitim sustavima baza podataka
- Usluge su također oblikovane da budu otporne, što znači da mogu nastaviti s radom čak i ako jedan ili više usluga prestane s radom. Budući da svaka usluga radi neovisno, kvar u jednoj ne bi trebao utjecati na cijelu aplikaciju
- Skalabilnost: Svaka usluga radi neovisno, moguće je skalirati pojedinačne servise prema potrebi, bez utjecaja na ostatak aplikacije. To omogućuje timovima da učinkovitije alociraju resurse i osiguraju da aplikacija može podnijeti povećani promet
- Tehnološka neovisnost: Različite usluge mogu se pisati u različitim programskim jezicima ili koristiti različite tehnološke pakete. To olakšava odabir pravog alata za posao, umjesto da budu vezani za određeni tehnološki paket

Ove karakteristike donose ove prednosti [2]:

- Kod se može lakše ažurirati - nove značajke ili funkcionalnosti mogu se dodati bez mijenjanja cijele aplikacije i bez utjecaja na druge usluge
- Timovi mogu koristiti različite tehnološke pakete i različite programske jezike za različite komponente
- Usluge se mogu skalirati neovisno, što smanjuje troškove povezane sa skaliranjem na cijele aplikacije jer bi jedna značajka mogla biti previše opterećena
- Svaku uslugu je lako održavati i testirati - omogućuje brz i čest razvoj i implementaciju

2.2. Komponente aplikacije

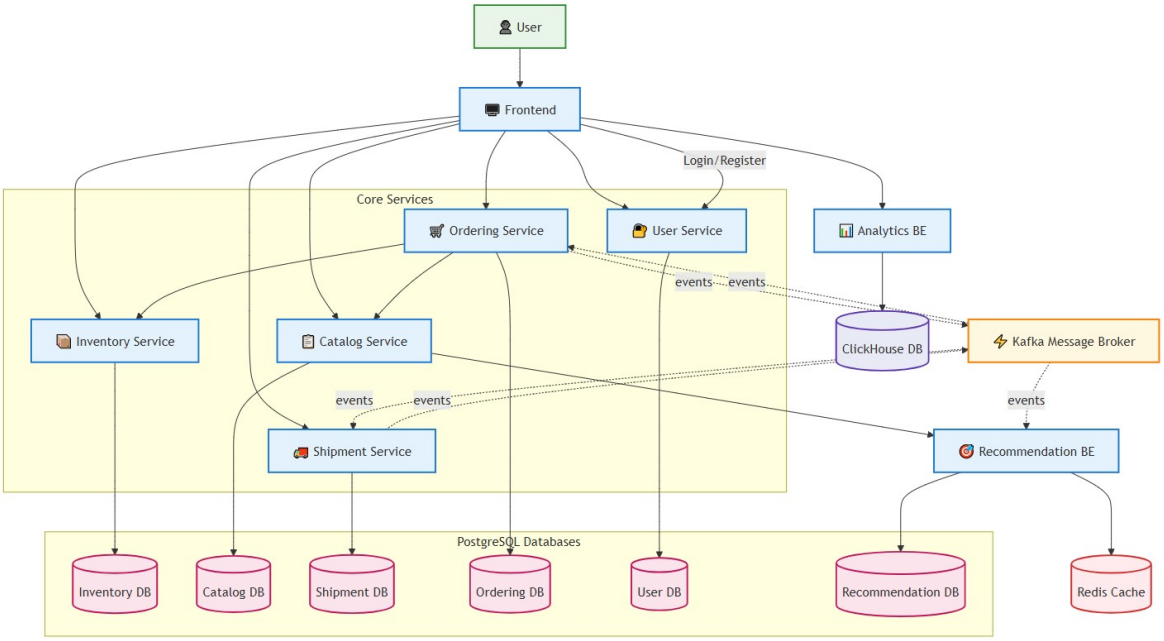
Ova aplikacija je razvijena u mikrouslužnoj arhitekturi.

Ovdje je dan pregled usluga aplikacije:

- Usluga kataloga (engl. *catalog*) - Upravlja katalogom proizvoda. Pruža podatke kao što su nazivi proizvoda, opisi, cijena i kategorije. Drugi servisi mogu pozivati katalog putem REST API-ja.
- Usluga naručivanja (engl. *ordering*) - Zadužena je za kreiranje i obradu narudžbi. Poziva usluge inventara i kataloga koristeći REST klijente kako bi potvrdila dostupnost proizvoda i dobila detalje o proizvodu. Nakon kreiranja narudžbe, objavljuje događaje narudžbi kako bi obavijestila druge usluge, tj. usluge otpreme i analitike.
- Usluga inventara (engl. *inventory*) - Prati zalihe i dostupnost proizvoda. Ima krajnje točke za provjeru i ažuriranje zaliha.
- Usluga otpreme (engl. *shipment*) - Upravlja otpremom i isporukom. Prima događaje narudžbi i ažurira status pošiljke u skladu s tim. Također objavljuje događaje ažuriranja statusa koju mogu koristiti drugi dijelovi, kao što je usluga naručivanja.
- Korisnička usluga (engl. *user-service*) - Obrađuje prijavu i registraciju
- Sustav preporuka (engl. *recommendation-system*) - Usluga koja generira preporuke proizvoda krajnjim korisnicima. Usluga je pretplaćena na događaje o kupnjama i koristi ih za preporuke.
- Usluga analitike (engl. *analytics*) - Obrađuje agregate poslovnih podataka i vizualizacije narudžbi, zaliha i otprema.
- Aplikacija na klijentskoj strani (engl. *frontend*) - Osnovno web sučelje za pregledavanje proizvoda, naručivanje i pregled povijesti narudžbi kupaca. Administratori mogu vidjeti analitičku nadzornu ploču s grafikonima i metrikama, ažurirati statuse pošiljki te podatke o inventaru.

Na sljedećem dijagramu, slika 2.1., prikazan je pregled glavnih komponenti sustava, njihove međusobne interakcije, kao i korištenje baza podataka, sustava pričuvne memo-

rije (engl. *cache*, dalje: keš) i poruka preko posrednika događaja (Kafka).



Slika 2.1. Pregled glavnih komponenti sustava, njihovih međusobnih interakcija te korištenja baza podataka, sustava keširanja i razmjene poruka putem posrednika događaja (Kafka).

2.3. Podaci

Za podatke za punjenje kataloga i inventara te za sustav preporučivanja baziran na sadržaju korišten je skup podataka s platforme Kaggle: Amazon Sales Dataset on Kaggle. Ova zbirka podataka sadrži detaljne informacije o 1351 Amazonovom proizvodu, uglavnom iz odjela elektronike i pribora.

Opis stupaca:

- **product_id** Jedinstveni ID svakog proizvoda.
- **product_name**: Naziv Amazonovog proizvoda.
  - Broj jedinstvenih naziva proizvoda: 1337
- **category**: Put kategorije proizvoda (npr. Elektronika|Nosiva tehnologija|Pametni satovi).
- **discounted_price**: Snižena cijena (npr. 199 Rs, 299 Rs).
- **actual_price**: Stvarna cijena prije popusta.

- **discount_percentage**: Postotak popusta koji se nudi na proizvod (npr. 50 %, 60 %).
- **rating**: Ukupna ocjena kupaca proizvoda (na ljestvici od 1 do 5).
- **rating_count**: Broj korisnika koji su ocijenili.
- **about_product**: Tekstni opis proizvoda, koristan za zadatke obrade prirodnog jezika kao što je izdvajanje značajki za sustave preporuka.
- **user_id**: Anonimni korisnički ID recenzenta.

Koristeći Jupyter bilježnicu proveden je proces čišćenja podataka koji transformira sirove podatke e-trgovine u format spreman za analizu:

- Pretvaranjem numeričkih vrijednosti formatiranih u tekstu u odgovarajuće numeričke tipove
- Ispravljanjem formata valuta i postotaka
- Stvaranjem značajke **specific_category** tako da se uzme zadnji dio puta kategorije: **category**
- Rukovanjem nedostajućim podacima

Također je napravljena skripta za povlačenje slika s besplatnih platformi za slike koristeći njihove REST API-ije: Unsplash, Pixaby i Pexels. Ime slike je dodano u tablicu podataka, a sve slike su ručno postavljene na Github, kako bi se kasnije mogle dohvaćati na korisničkom sučelju.

Ovakvi podaci su spremni za učitavanje u bazu kataloga.

2.4. Filtriranje na temelju sadržaja

Da bi se napravio sustav za preporuke temeljen na sadržaju dodana je značajka **combined_text** koja je napravljena tako da su se konkatenirali nizovi znakova: **product_name**, **category**, **about_product**.

Svaki proizvod je kodiran kao TF-IDF vektor. TF-IDF (engl. *Term Frequency-Inverse*

Document Frequency - Inverse Document Frequency) je statistička mjera koja se koristi u obradi prirodnog jezika i pronalaženju informacija: Procjenjuje važnosti riječi u dokumentu u odnosu na zbirku dokumenata [4].

Kosinusna sličnost se koristi za izračunavanje sličnosti među svim vektorima proizvoda. To stvara simetričnu matricu gdje unos (i, j) predstavlja sličnost proizvoda i i proizvoda j prema njihovim tekstnim opisima.

Kosinusna sličnost [5]:

$$similarity(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||_2 ||\mathbf{b}||_2},$$

gdje je $\mathbf{a} \cdot \mathbf{b}$ skalarni produkt vektora $\mathbf{a}$ i $\mathbf{b}$:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i,$$

a $||\mathbf{a}||_2$ $i|\mathbf{b}||_2$ euklidske norme vektora $\mathbf{a}$ i $\mathbf{b}$:

$$||\mathbf{a}||_2 = \sqrt{\sum_{i=1}^{n} a_i^2}$$

Nakon što je dodana nova značajka **combined_text** pretvorena je u u TF-IDF vektor za sve podatke te je kreirana matrica sličnosti svih proizvoda. Kako bi se omogućila buduća ponovna upotreba bez potrebe za ponovnim učenjem, TF-IDF vektorizator, matrica sličnosti i mapa indeksa ID-a proizvoda spremaju se s pomoću jobliba. Spremljene datoteke se kasnije učitavaju u uslugu sustava za preporuku.

Joblib je Python biblioteka za paralelno izvršavanje računalno zahtjevnih zadataka. Pruža skup funkcija za paralelno izvođenje operacija na velikim skupovima podataka i za predmemoriranje rezultata računalno zahtjevnih funkcija. Joblib je posebno koristan za modele strojnog učenja jer omogućuje spremanje stanja izračuna i nastavak rada kasnije ili na drugom računalu [6].

2.5. Infrastruktura

Da bi se sustav e-trgovine mogao pokrenuti lokalno ili na poslužitelju, nužno je najprije osigurati potrebnu infrastrukturu – baze podataka, poruke (Kafka), alat za praćenje (Kafka UI) itd. Ova se infrastruktura može lako pokrenuti i upravljati s pomoću Dockera i Docker Composea.

Docker je softverska platforma koja omogućuje brzu izgradnju, testiranje i implementaciju aplikacija. Docker pakira softver u standardizirane jedinice zvane kontejneri koji sadrže sve što je softveru potrebno za pokretanje, uključujući biblioteke, sistemske alate, kod i okruženje za izvođenje [7].

Docker funkcionira tako što pruža standardni način pokretanja koda. To je sustav za kontejnere. Slično kao što virtualni stroj virtualizira (uklanja potrebu za izravnim upravljanjem) hardverom poslužitelja, kontejneri virtualiziraju operacijski sustav poslužitelja. Docker se instalira na svaki poslužitelj i pruža jednostavne naredbe koje se mogu koristiti za izgradnju, pokretanje ili zaustavljanje kontejnera [7].

U sustavu e-trgovine koji se koristi u ovom projektu, datoteka docker-compose.yml definira sve ključne servise potrebne za rad:

- PostgreSQL - relacijska baza podataka za proizvode, korisnike, narudžbe itd.
- Kafka i Zookeeper - za slanje događaja (engl. *event streaming*).
- Debezium - alat za praćenje promjena u bazi.
- ClickHouse - analitička baza podataka za izvještaje i vizualizaciju.
- Redis - za brzo spremanje privremenih podataka ili keširanje.
- Kafka UI - web sučelje za upravljanje Kafka temama i porukama.
- Vlastiti konektor - Python usluga koji prebacuje podatke iz Kafke u ClickHouse.

Infrastruktura se pokreće sljedećom naredbom: `docker compose up`

2.5.1. PostgreSQL

PostgreSQL je objektno-relacijski sustav baza podataka otvorenog koda [8].

U arhitekturi e-trgovine koristi se jedna PostgreSQL usluga (`db`) konfigurirana kroz Docker kontejner sa sljedećim ključnim karakteristikama:

Inicijalizacija baza podataka

- Prilikom prvog pokretanja kontejnera automatski se izvršava shell skripta `create-dbs.sh` prikazana na slici 2.2.
- Varijabla okruženja `POSTGRES_MULTIPLE_DATABASES` koja sadrži popis baza podataka koje treba inicijalizirati se postavlja u kontejneru pri njegovom pokretanju
- Pokretanjem skripte kreira se više logički odvojenih baza podataka unutar jedne instance PostgreSQL-a
- Kreirani su sljedeći logički entiteti:
  - Baze podataka: `catalog`, `inventory`, `ordering`, `user_service`, `shipment` i `recommendations`
  - Za svaku bazu kreira se korisnik koji se zove isto kao sama baza sa lozinkom 'changeme'

Sigurnosni model

- Svakom korisniku se dodjeljuju sve potrebne privilegije:
  - Puni pristup nad odgovarajućom bazom podataka
  - Kontrola nad shemom unutar baze (koja se kreira s istim imenom kao baza)
  - Privilegije nad tablicama, sekvencama i funkcijama unutar sheme

U razvojnom okruženju, svi servisi koriste zajednički PostgreSQL kontejner s više baza. Time se pojednostavljuje lokalni razvoj i testiranje te smanjuje opterećenje na lokalni sustav jer se koristi samo jedan kontejner umjesto više njih.

```
function create_user_and_database() {
  local database=$1
  echo "  Creating user and database '$database'"
  psql -v ON_ERROR_STOP=1 --username "$POSTGRES_USER" <<-EOSQL
      CREATE USER $database WITH PASSWORD 'changeme';
      CREATE DATABASE $database;
      GRANT ALL PRIVILEGES ON DATABASE $database TO $database;
      \c $database;
      CREATE SCHEMA $database;
      GRANT ALL ON SCHEMA $database TO $database;
      GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA $database TO $database;
      GRANT ALL ON ALL SEQUENCES IN SCHEMA $database TO $database;
      GRANT ALL ON ALL FUNCTIONS IN SCHEMA $database TO $database;
EOSQL
}


if [ -n "$POSTGRES_MULTIPLE_DATABASES" ]; then
  echo "Multiple database creation requested: $POSTGRES_MULTIPLE_DATABASES"
  for db in $(echo $POSTGRES_MULTIPLE_DATABASES | tr ',' ' '); do
    create_user_and_database $db
  done
  echo "Multiple databases created"
fi
```

Slika 2.2. Shell skripta `create-dbs.sh` koja se automatski izvršava prilikom prvog pokretanja kontejnera

U produkcijskom okruženju bi svaka usluga trebala imati vlastitu bazu podataka i vlastiti servis baze, čime se postiže jača izolacija podataka i odgovornosti, povećava sigurnost i skalabilnost sustava te omogućuje nezavisno skaliranje i održavanje svake baze.

2.5.2. Kafka, Zookeeper, KafkaUI

Apache Kafka je distribuirana platforma otvorenog koda za strujanje (engl. *streaming*) događaja koja se koristi za visokoučinkovite podatkovne kanale, analitiku strujanja (engl. *streaming*), integraciju podataka i aplikacije kritične za poslovanje [9].

Jedna od ključnih prednosti leži u skalabilnosti i toleranciji grešaka, postignutoj putem Kafkinog brokerskog klastera i ovisnosti o Zookeeperu za upravljanje metapodacima i koordinaciju [10].

Neki osnovni pojmovi [10]:

- Kafka topic: Kategorija u koju se šalju zapisi. Topici su podijeljene u particije zbog

skalabilnosti.

- Proizvođač (engl. *producer*): Klijentska aplikacija koja šalje poruke (podatke) na Kafka topic.
- Consumer: Klijentska aplikacija koja čita poruke s Kafka topica.
- Broker: Kafka broker je poslužitelj koji pohranjuje podatke (poruke) u particijama topica. Bavi se pohranom, isporukom i replikacijom poruka.
- Zookeeper: Centralizirana usluga koja upravlja koordinacijom Kafka brokera. Prati koji brokeri pripadaju kojim klasterima, obrađuje izbore vođa i održava metapodatke o Kafka topicima i particijama.

Kafka UI (Provectus Kafka UI) je grafičko korisničko sučelje za rad s Apache Kafkom. Njegova svrha je da omogući praćenje ključnih metrika Kafka klastera - brokera, topica, particija, produkcije i potrošnje [11].

Apache Kafka se u ovoj aplikaciji koristi za razmjenu događaja o stanju narudžbe/pošiljke između usluge otpreme i usluge naručivanja. Također, usluga naručivanja šalje podatke o narudžbama koje kasnije sustav preporuka koristi za preporuke.

2.5.3. Redis

Redis je NoSQL pohrana ključeva/vrijednosti otvorenog koda u memoriji koja se prvenstveno koristi kao predmemorija aplikacije ili baza podataka za brzi odgovor [12].

Redis pohranjuje podatke u memoriju, a ne na disk ili SSD (engl. *solid-state drive*), što pomaže u postizanju brzine, pouzdanosti i performansi [12].

Redis se koristi u sustavu preporuka za pohranjivanje preporuka u svrhu optimizacije.

2.5.4. Clickhouse, Debeezium, Kafka - ClickHouse Connector

ClickHouse je baza podataka otvorenog koda zasnovana na OLAP-u (engl. *Online Analytical Processing*) koja se ističe brzinom i performancama u raznim scenarijima skladištenja i analitike, kao što su analitički podaci i podaci vremenskih serija [13].

Snimanje promjena podataka (engl. *Change Data Capture*) je obrazac integracije podataka koji bilježi promjene napravljene na podacima u izvornom sustavu, kao što su umetanja, ažuriranja i brisanja. Te se promjene, predstavljene kao popis, obično nazivaju *CDC feed* [14].

Postoje različiti načini za bilježenje promjena u bazi podataka, uključujući ispitivanje zadnjih ažuriranih stupaca i korištenje okidača baze podataka, otkrivanje promjena u dnevniku transakcija, itd [15].

Dnevnici transakcija predstavljaju jedan od tih pristupa, pri čemu se sve izmjene koje se naprave u bazi podataka (unosi, ažuriranja i brisanja) automatski bilježe u posebnim datotekama koje se nazivaju dnevnici transakcija (engl. *transaction logs*). Ovi dnevnici sadrže potpuni zapis svih promjena koje su se dogodile u bazi te tako omogućuju sustavima za otkrivanje promjena (engl. *Change Data Capture, CDC*) da iz njih rekonstruiraju tok događaja.

Debezium je distribuirana platforma otvorenog koda za prikupljanje podataka o promjenama [16]. Koristi dnevnike transakcija kako bi kontinuirano pratio i bilježio sve izmjene u bazi podataka, koje zatim pretvara u standardizirane poruke (događaje) te ih prosljeđuje središnjem sustavu za razmjenu poruka, najčešće Apache Kafka [15].

U mikrouslužnoj arhitekturi, svaka usluga ima vlastitu bazu podataka radi labavog povezivanja, skalabilnosti i neovisne implementacije. Na primjer, usluge poput otpreme, zaliha i naručivanja posjeduju vlastite baze podataka i sheme. Ova arhitektura poboljšava modularnost i izolaciju grešaka, ali otežava izvođenje analitike između usluga jer bi bilo potrebno izravno povezivanje s više baza podataka. To nije poželjno zbog potencijalnog gubitka performansi, sigurnosnih problema i uskog povezivanja analitičkih sustava s operativnim pohranama podataka.

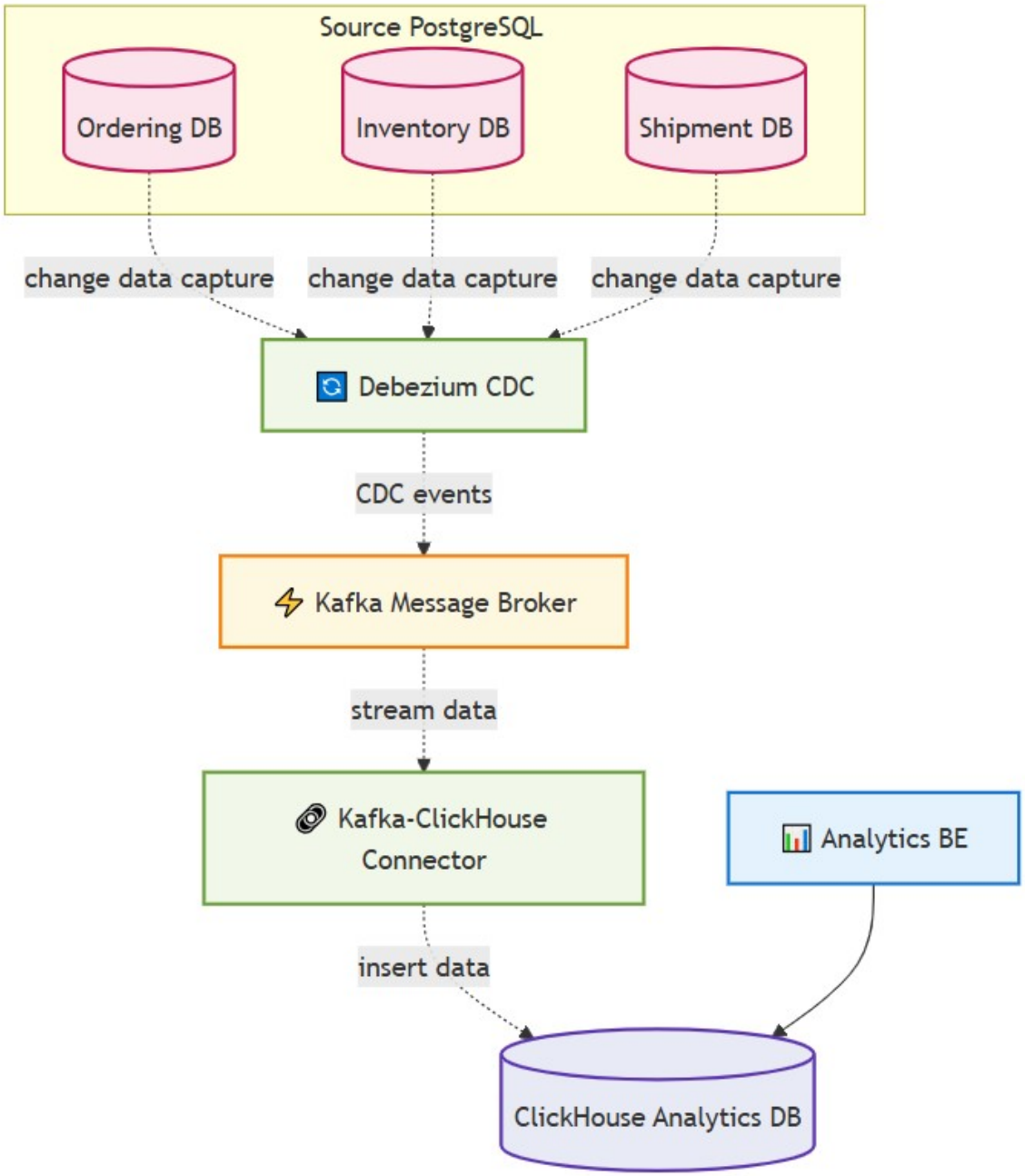
Za kontinuirano strujanje promjena u podacima iz svake mikrouslužne baze podataka u ClickHouse koristi se Debezium. Debezium prenosi promjene na razini redaka (*INSERT, UPDATE, DELETE*) čitajući zapisnike transakcija relacijskih baza podataka. Debezium je konfiguriran pokretanjem skripte koja šalje HTTP POST zahtjev Debezium REST API-ju za registraciju konektora s tijelom zahtjeva koji sadržava: izvornu bazu podataka PostgreSQL (naziv hosta, port, korisničke vjerodajnice, naziv baze podataka),

preslikavanje tablica/topica itd. Debezium objavljuje promjene na Apache Kafka topicima. Na primjer, kada se unese narudžba ili ažurira status pošiljke, Debezium bilježi događaj i objavljuje ga na odgovarajući Kafka topic. Na taj način, sve promjene iz distribuiranih mikrouslužnih baza podataka prenose se na centralnu lokaciju.

Za unos ovih *CDC* događaja u ClickHouse koristi se Kafka-ClickHouse konektor. Konektor se pretplaćuje na Kafka teme i prevodi poruke u odgovarajuće ClickHouse *INSERT* upite. Kafka-ClickHouse konektor implementiran je kao Python usluga koja se pokreće u Docker kontejneru koristeći sliku `python:3.11-slim`.

Prilikom pokretanja okruženja docker-compose, s ClickHouse uslugom, automatski se izvršava skripta za kreiranje analitičkih tablica. Ove tablice zadržavaju istu strukturu kao i operativne tablice usluga za naručivanje, pošiljke i inventar, ali za razliku od njih, sadrže i povijesne podatke zahvaljujući implementiranom mehanizmu *Change Data Capture (CDC)*.

Dijagram prikazan na slici 2.3. prikazuje tok podataka iz operativnih baza podataka mikrousluga prema centraliziranoj analitičkoj bazi ClickHouse pomoću sustava za bilježenje promjena podataka (engl. *Change Data Capture*).



Slika 2.3. Tok podataka iz mikrouslužnih baza podataka u ClickHouse putem Debezium CDC i Apache Kafka sustava.

3. Korisnička usluga

Korisnička usluga razvijena je sa Spring Boot radnim okvirom i modul je za autentifikaciju i rukovanje korisnicima u cjelokupnoj arhitekturi aplikacije e-trgovine. Slijedi široko prihvaćene smjernice o slojevitoj arhitekturi. Razdvaja odgovornosti na slojeve nadglednika, usluga i repozitorija te primjenjuje JWT-baziranu autentifikaciju u kombinaciji s obrascima autorizacije temeljenim na ulogama.

Sustav ima troslojni dizajn:

- Sloj nadglednika: Obrađuje dolazne HTTP zahtjeve i preslikava ih na odgovarajuće operacije servisa.
- Sloj usluga: Izračunava poslovnu logiku, npr. registraciju i autentifikaciju korisnika.
- Sloj repozitorija: Komunicira s bazom podataka putem Spring Data JPA.

Sigurnost je temeljni element u dizajnu usluge. Primarni mehanizmi su:

- Šifriranje lozinke: Lozinke se pohranjuju hashirane koristeći Spring Securityjev PasswordEncoder.
- JWT autentifikacija: Autentifikacija bez stanja s kriptografski potpisanim tokenima.
- Autorizacija temeljena na ulogama: Kontrola pristupa korištenjem uloga integriranih u JWT.
- RSA par ključeva: Koristi se za digitalno potpisivanje i autentifikaciju JWT-ova, osiguravajući integritet i neporecivnost.

- Ograničavanjem trajanja tokena smanjuje se mogućnost napada ponavljanjem.

Korisnička usluga pruža funkcionalnost generiranja i validacije JWT-a korištenjem asimetrične kriptografije temeljene na RSA parovima ključeva. Ovaj pristup poboljšava sigurnost korištenjem privatnog ključa za potpisivanje tokena i odgovarajućeg javnog ključa za provjeru tokena, osiguravajući da se tokeni ne mogu krivotvoriti ili mijenjati.

JWT ili JSON web token je niz znakova koji predstavlja neku informaciju [17].

JWT je kombinacija 3 polja:

1. zaglavlje
2. tijelo
3. potpis

u sljedećem formatu:

zaglavlje.tijelo.potpis [17]

Cilj JWT-a nije sakriti podatke, već dodati autentičnost podacima, tj. dokazati da je poslane podatke kreirao autentični izvor [17].

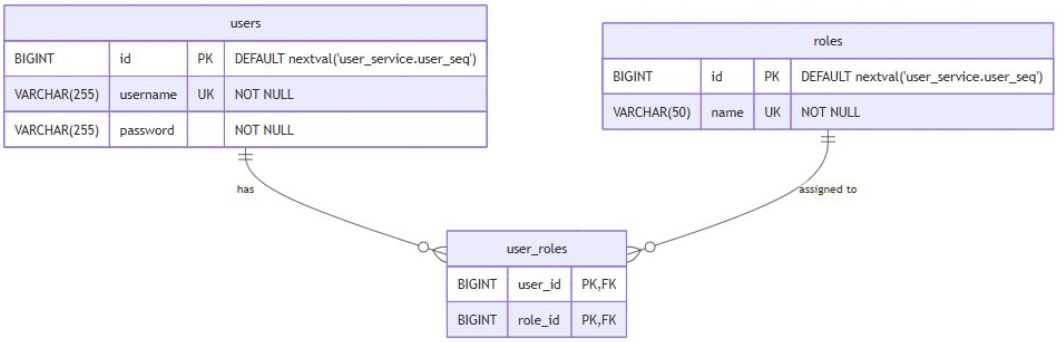
RSA je algoritam za asimetrično šifriranje i digitalni potpis. Asimetrični algoritmi nude mogućnost provjere ili dešifriranja poruke bez mogućnosti stvaranja nove. U asimetričnom algoritmu, dva ključa se koriste za šifriranje i dešifriranje poruka. Jedan ključ (privatni) se koristi za digitalno potpisivanje poruke, a drugi ključ (javni) može se koristiti samo za provjeru autentičnosti potpisa. Dakle, izvor može generirati i javni i privatni ključ, a zatim poslati samo javni ključ drugima kako bi provjerili svoje poruke [17].

Upravljanje shemom baze podataka i početnim podacima za korisničku uslugu ostvareno je kroz Liquibase.

Liquibase je alat za kontrolu verzija baze podataka. Koristi SQL, XML, JSON i YAML datoteke zapisnika promjena za popis promjena baze podataka sekvencijalnim redoslijedom. Promjene su organizirane u skupove promjena (engl. *changesets*) koji sadrže specifične operacije poput dodavanja stupaca, definiranja primarnih ključeva ili drugih

strukturnih modifikacija [18].

U slučaju ove usluge, liquibaseov skup promjena se pokreće kad se aplikacija pokrene. U tom skupu promjena se nalaze naredbe za stvaranje tablice po modelu na slici 3.1.



Slika 3.1. ER dijagram user_service baze

3.1. API dokumentacija

U ovom poglavlju prikazana je dokumentacija programskog sučelja (API) ostvarene korisničke usluge.

3.1.1. Podatkovni modeli

UserRequest

Obavezna polja: username, password

```
{
  "username": "string",
  "password": "string (password format)"
}
```

UserResponse

```
{
  "id": "string (uuid)",
  "username": "string",
```

```
  "roles": "string[]"
}
```

LoginResponse

```
{
  "token": "string (JWT)",
  "expiresIn": "integer (milliseconds)",
  "id": "string (uuid)"
}
```

3.1.2. API-jeve krajnje točke

Korisnici

- `POST /users` – Registriraj novog korisnika

  Tijelo: UserRequest → **201:** UserResponse | **400** | **409:** Korisničko ime već postoji

- `POST /users/login` – Prijava korisnika

  Tijelo: UserRequest → **200:** LoginResponse | **401:** Neispravne vjerodajnice | **400**

4. Usluga kataloga

Usluga kataloga razvijena je na temelju radnog okvira Spring Boot i ključni je modul za smještaj kataloga proizvoda unutar sustava. Objavljuje API-je za dohvaćanje informacija o proizvodima i preporuka, nudeći siguran pristup putem mehanizma za autentifikaciju i autorizaciju koristeći JSON web tokena (JWT). Usluga koristi autentifikaciju bez stanja s JWT-ovima koje pruža središnja usluga za korisnike. Koristi javni ključ korisničke usluge za autentifikaciju dolaznih zahtjeva, što joj omogućuje sigurnu validaciju JWT tokena bez dijeljenja korisničke sjednice.

Funkcionalnosti:

- Podržava pretraživanje imena bez razlikovanja velikih i malih slova, s validacijom upita i paginacijom
- Nudi personalizirane preporuke proizvoda na temelju ID-a autentificiranog korisnika. U slučajevima kada podaci o preporuci nedostaju ili nisu uspjeli, usluga se vraća na ručno odabrani skup najpopularnijih proizvoda. Preporuke dobiva tako što komunicira s REST API-jem vanjske usluge preporuka putem web klijenta.
- Vraća paginirane popise popularnih proizvoda
- Omogućuje skupno učitavanje naziva proizvoda i brze pretrage na temelju ID-a

U slučaju usluge kataloga, liquibaseov skup promjena se pokreće kad se aplikacija pokrene. U tom skupu promjena se nalaze naredbe za stvaranje tablice po modelu na slici 4.1. Također postoji i liquibaseov skup promjena kojime se baza puni očišćenim podacima.

4.1. API dokumentacija

U ovom poglavlju prikazana je dokumentacija programskog sučelja (API) ostvarene usluge kataloga.

4.1.1. Podatkovni modeli

ProductDTO

```
{
  "id": "integer (int32)",
  "name": "string",
  "description": "string",
  "price": "float",
  "rating": "float",
  "ratingCount": "integer (int32)",
  "category": "string",
  "imageUrl": "string"
}
```

ProductNameDTO

```
{
  "id": "integer (int32)",
  "name": "string"
}
```

PageInfo

```
{
  "totalItems": "integer",
  "totalPages": "integer",
  "currentPage": "integer",
  "pageSize": "integer"
}
```

ProductPage

```
{
  "content": "ProductDTO[]",
```

```
  "pageInfo": "PageInfo"
}
```

4.1.2. API-jeve krajnje točke

Proizvodi

- `GET /products/{productId}` – Dohvati proizvod po ID-u
  → **200:** ProductDTO | **404**
- `GET /products/search?q={query}&page={0}&size={10}` – Pretraži proizvode
  → **200:** ProductPage | **400**
- `GET /products/recommendations?page={0}&size={10}` – Personalizirane preporuke (Auth)
  → **200:** ProductPage | **401**
- `GET /products/names?ids={1,2,3}` – Dohvati nazive po ID-ovima
  → **200:** ProductNameDTO[] | **400**
- `GET /products/popular?page={0}&size={10}` – Popularni proizvodi
  → **200:** ProductPage

Parametri

- **page** – Broj stranice (zadano: 0)
- **size** – Stavki po stranici (zadano: 10)
- **q** – Upit za pretraživanje (obavezno)
- **ids** – Lista ID-ova odvojena zarezima (obavezno)

Autentifikacija

Bearer token potreban za `GET /products/recommendations`

products			
SERIAL	id	PK	NOT NULL
VARCHAR(500)	name		NOT NULL
VARCHAR(400)	category		
VARCHAR(60)	specific_category		
DECIMAL	discounted_price		
DECIMAL	actual_price		NOT NULL
DECIMAL	discount_percentage		
DECIMAL	rating		INDEX
INT	rating_count		
TEXT	about_product		
TEXT	img_link		
TEXT	product_link		
TEXT	combined_text		
TEXT	image_name		
TEXT	image_path		

Slika 4.1. ER dijagram baze catalog (products)

5. Usluga inventara

Usluga inventara razvijena je na temelju radnog okvira Spring Boot i modul je za je upravljanje razinama zaliha proizvoda, podršku rezervacijama narudžbi i osiguravanje dosljednosti podataka o zalihama u različitim interakcijama sustava. Objavljuje API za ove funkcionalnosti nudeći siguran pristup putem mehanizma za autentifikaciju i autorizaciju koristeći JSON web tokena (JWT). Koristi javni ključ korisničke usluge za autentifikaciju dolaznih zahtjeva, što joj omogućuje sigurnu validaciju JWT tokena. Implementira kontrolu pristupa temeljenu na ulogama (RBAC) za upravljanje korisničkim privilegijama.

Osnovne funkcionalnosti

- Održava detaljne zapise o količinama proizvoda
- Podržava oznake statusa kao što su ***IN_STOCK***, ***LOW_STOCK*** i ***OUT_OF_STOCK***, dinamički određene na temelju količinskih pragova.
- Kako bi se podržao životni ciklus narudžbe, usluga uključuje podsustav rezervacija koji omogućuje privremeno zadržavanje zaliha:
  - Rezervacija: Privremeno oduzima dostupne zalihe bez finaliziranja narudžbe
  - Otpuštanje: Otkazuje rezervaciju, vraćajući zalihe na raspoloživost
  - Potvrda: Finalizira transakciju i smanjuje stvarne razine zaliha.
- Omogućuje sinkronu provjeru dostupnosti proizvoda.
- Implementira pesimistično zaključavanje kako bi se osigurala sigurnost transakcija tijekom izmjena zaliha.

Usluga inventara implementira kontrolu konkurentnosti korištenjem pesimističnog zaključavanja kako bi se osigurala konzistentnost podataka, spriječili *race conditions* i održale točne razine inventara pod istodobnim pristupom. Usluga koristi pesimistično zaključavanje putem JPA `@Lock` anotacije s `LockModeType.PESSIMISTIC_WRITE`, listing 5..1 To osigurava da nakon što transakcija pročita određeni zapis inventara, druge transakcije su blokirane od mijenjanja dok se trenutačna transakcija ne završi [19].

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT␣i␣FROM␣Inventory␣i␣WHERE␣i.productId␣=␣:productId
    ")
Optional<Inventory> findByProductIdWithLock(Long productId);
```

Listing 5..1: Zaključavanje na razini repozitorija

Uz ovo, primjenjuje se i anotacija `@Transactional` na razini servisnog sloja, čime se osigurava da sve operacije koje mijenjaju stanje zaliha budu izvršene atomarno, uz upotrebu pesimističkog zaključavanja zbog sprječavanja konflikata [20].

Status zaliha se dinamički preračunava nakon svake operacije, listing 5..2:

```
public void updateStatus() {
    int availableForSale = quantityAvailable - reservedQuantity;
    if (availableForSale <= 0) {
        this.status = InventoryStatus.OUT_OF_STOCK;
    } else if (availableForSale <= reorderThreshold) {
        this.status = InventoryStatus.LOW_STOCK;
    } else {
        this.status = InventoryStatus.IN_STOCK;
    }
}
```

Listing 5..2: Logika ažuriranja statusa zaliha

U slučaju usluge inventara, liquibaseov skup promjena se pokreće kad se aplikacija pokrene. U tom skupu promjena se nalaze naredbe za stvaranje tablice po modelu na slici 5.1. Također postoji i liquibaseov skup promjena kojime se baza puni podacima. Podaci su kreirani tako da su se u datoteku csv formata kopirali product_id-jevi iz kataloga te se

ostali podaci potrebni za inventar ručno generirali.

inventory			
BIGINT	id	PK	AUTO_INCREMENT
BIGINT	product_id	UK	NOT NULL
VARCHAR(255)	sku	UK	NOT NULL
INT	quantity_available		NOT NULL
INT	reserved_quantity		NOT NULL
INT	reorder_threshold		NOT NULL
VARCHAR(255)	warehouse_location		
VARCHAR(255)	shelf_location		
TIMESTAMP	last_updated		NOT NULL
VARCHAR(50)	status		

Slika 5.1. ER dijagram baze inventory

5.1. API dokumentacija

U ovom poglavlju prikazana je dokumentacija programskog sučelja (API) ostvarene usluge inventara.

5.1.1. Podatkovni modeli

InventoryDTO

```
{
  "id": "integer (int64)",
  "productId": "integer (int64)",
  "productName": "string",
```

```
  "sku": "string",
  "quantityAvailable": "integer",
  "reservedQuantity": "integer",
  "reorderThreshold": "integer",
  "warehouseLocation": "string",
  "shelfLocation": "string",
  "status": "string (IN_STOCK|LOW_STOCK|OUT_OF_STOCK)",
  "lastUpdated": "string (date-time)"
}
```

InventoryRequest

```
{
  "quantity": "integer (min: 1)"
}
```

5.1.2. API-jeve krajnje točke

Inventar

- `GET /inventory?page={0}&size={20}&status={IN_STOCK}` – Dohvati sve stavke (Auth: ADMIN)
  → **200:** InventoryDTO[] | **401** | **403**

- `PUT /inventory/{id}` – Ažuriraj stavku (Auth: ADMIN)
  Body: InventoryDTO
  → **200:** InventoryDTO | **400** | **401** | **403** | **404**

- `GET /inventory/product/{productId}` – Dohvati inventar po proizvodu (Auth)
  → **200:** InventoryDTO | **401** | **404**

- `GET /inventory/{productId}/availability?quantity={1}` – Provjeri dostupnost
  → **200:** boolean | **401** | **404**

- `POST /inventory/{productId}/reservations` – Rezerviraj inventar
  Body: InventoryRequest
  → **200** | **400** | **401** | **404**

- `DELETE /inventory/{productId}/reservations` – Otkaži rezervaciju
  Body: InventoryRequest
  → **200** | **400** | **401** | **404**

- `POST /inventory/{productId}/commit` – Potvrdi rezervaciju
  Body: InventoryRequest
  → **200** | **400** | **401** | **404**

- `PUT /inventory/{productId}/restock` – Dopuni inventar (Auth: ADMIN)
  Body: InventoryRequest
  → **200:** InventoryDTO | **400** | **401** | **403** | **404**

Parametri

- **page** – Broj stranice (zadano: 0)
- **size** – Stavki po stranici (zadano: 20)
- **status** – Filter po statusu (IN_STOCK, LOW_STOCK, OUT_OF_STOCK)
- **quantity** – Količina za operaciju (minimum: 1)
- **id** – ID stavke inventara (int64)
- **productId** – ID proizvoda (int64)

Autentifikacija

- *Bearer* token potreban za sve operacije
- ADMIN uloga potrebna za: GET /inventory, PUT /inventory/{id}, PUT /inventory/{productId}/restock

6. Usluga naručivanja

Usluga naručivanja razvijena je na temelju radnog okvira Spring Boot i modul je za je upravljanje narudžbama i košaricama. Objavljuje API-je za ove funkcionalnosti nudeći siguran pristup putem mehanizma za autentifikaciju i autorizaciju koristeći JSON web token (JWT). Koristi javni ključ korisničke usluge za autentifikaciju dolaznih zahtjeva, što joj omogućuje sigurnu validaciju JWT tokena.

Funkcionalnosti:

- Upravljanje košaricom za kupnju
  - Pregled košarice: Vraća sve proizvode u košarici korisnika s metapodacima proizvoda kao što su naziv, cijena i količina.
  - Izmjena košarice: Korisnici mogu dodavati, ažurirati ili uklanjati proizvode. Sve operacije potvrđuju dostupnost zaliha.
  - Brisanje košarice: Uklanja sve artikle i oslobađa prethodno rezervirane zalihe.
- Upravljanje narudžbama
  - Izrada narudžbi: Prenosi artikle košarice u trajni zapis narudžbe, izračunava ukupni trošak i povezuje podatke o dostavi i naplati.
  - Praćenje narudžbi: Izlaže krajnje točke za dohvaćanje detalja narudžbe

Usluga naručivanja komunicira s uslugama kataloga i inventara

- Usluga zaliha: Potvrđuje količine zaliha, zaključava zalihe u operacijama košarice i oslobađa ih nakon uklanjanja ili neuspješne naplate.

- Usluga kataloga: Dohvaća detalje o proizvodu, osiguravajući točnost cijene prije kreiranja narudžbe.

Komunikacija se obavlja putem OpenFeign klijenata. Feign je deklarativni klijent web servisa. Olakšava pisanje klijenata web servisa. Za korištenje Feigna potrebno je napraviti sučelje za zahtjeve i dodati određene anotacije [21].

Sustav naručivanja komunicira sa sustavima pošiljke i sustavom preporuka preko Apache Kafke.

Kada korisnik kreira narudžbu šalje se poruka sljedećeg formata na Kafku (listing 6..1):

Listing 6..1: JSON Schema Order

```
{
    "orderId": "Long",
    "shippingAddress": "String",
    "billingAddress": "String",
    "createdAt": "LocalDateTime",
    "updatedAt": "LocalDateTime",
    "status": "OrderStatus enum"
}
```

Također se šalju i poruke ovog formata (listing 6..2):

Listing 6..2: JSON Schema UserProduct

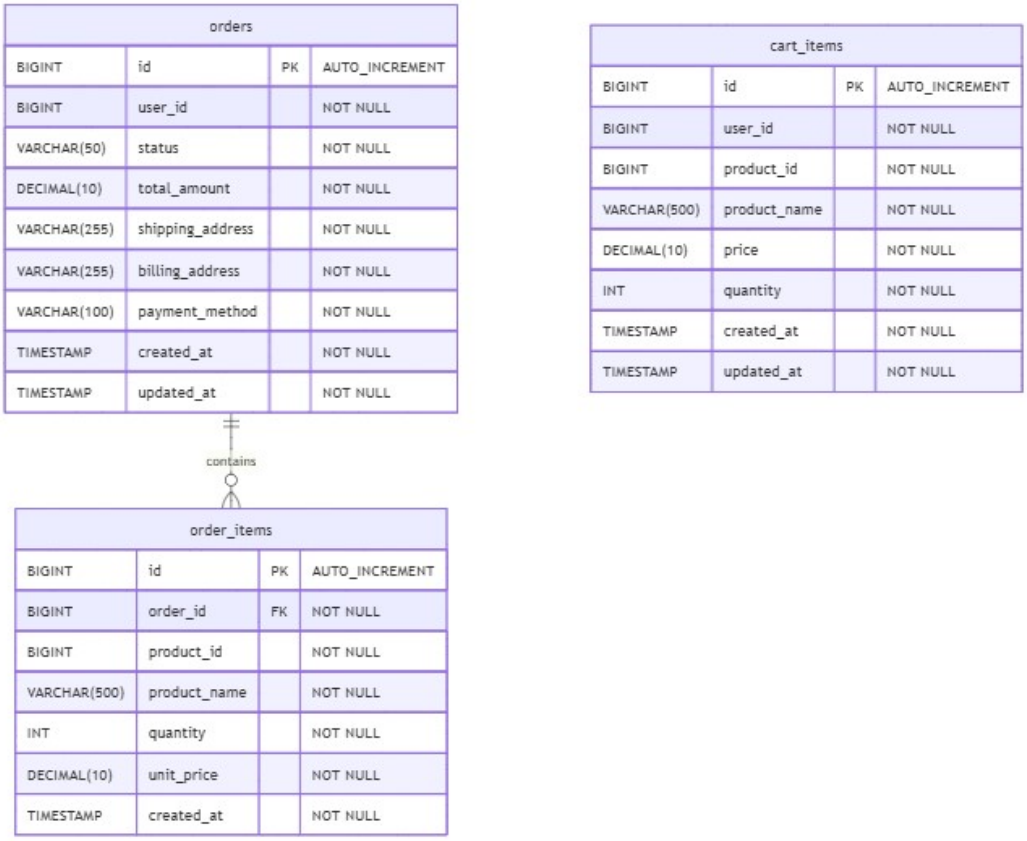
```
{
    "userId": "Long",
    "productId": "Long",
    "quantity": "int",
    "timestamp": "long"
}
```

Događaj narudžbe šalje se sinkrono na Kafku. Budući da se usluga otpreme oslanja na ovaj događaj za pokretanje ispunjenja narudžbe, ključno je da je poruka poslana. Zbog toga je proizvođač konfiguriran s visokom pouzdanošću (*acks=all*, *retries* i *idempotence*),

Nasuprot tome, događaji korisnik-proizvod šalju se asinkrono u temu događaja preporuka. Ovi događaji prate interakcije korisnika i proizvoda (npr. kupnje) za izgradnju modela preporuka. Budući da logika preporuka nije kritična i tolerantna je na povremeni gubitak poruka, sustav koristi brzog proizvođača po principu *fire and forget* (*acks*=0, bez *retries*) kako bi se smanjila latencija i izbjeglo blokiranje cjevovoda obrade narudžbi.

Consumer u usluzi za naručivanje osigurava da statusi narudžbi ostanu sinkronizirani s napretkom pošiljke konzumiranjem događaja iz Kafka topica ažuriranja statusa pošiljke na koji poruke šalje usluga otpreme.

U slučaju usluge naručivanja, liquibaseov skup promjena se pokreće kad se aplikacija pokrene. U tom skupu promjena se nalaze naredbe za stvaranje tablica po modelu na slici 6.1.



Slika 6.1. ER dijagram baze ordering

6.1. API dokumentacija

U ovom poglavlju prikazana je dokumentacija programskog sučelja (API) ostvarene usluge naručivanja.

6.1.1. Podatkovni modeli

OrderDTO

```
{
  "id": "integer (int64)",
  "userId": "integer (int64)",
  "status": "PENDING|PAID|SHIPPED|DELIVERED|CANCELLED",
  "totalAmount": "float",
  "shippingAddress": "string",
  "billingAddress": "string",
  "paymentMethod": "string",
  "items": "OrderItemDTO[]",
  "createdAt": "datetime",
  "updatedAt": "datetime"
}
```

OrderItemDTO

```
{
  "productId": "integer (int64)",
  "productName": "string",
  "quantity": "integer",
  "unitPrice": "float"
}
```

OrderRequest

```
{
  "shippingAddress": "string",
  "billingAddress": "string",
  "paymentMethod": "string",
  "paymentDetails": "object",
  "status": "string"
}
```

CartItemDTO

```
{
  "id": "integer (int64)",
  "productId": "integer (int64)",
  "productName": "string",
  "quantity": "integer",
  "price": "float"
}
```

6.1.2. API-jeve krajnje točke

Narudžbe

- `POST /orders` – Stvori narudžbu iz košarice

  Tijelo: OrderRequest → **201:** OrderDTO

- `GET /orders` – Dohvati sve narudžbe (samo admin)

  → **200:** OrderDTO[]

- `GET /orders/{orderId}` – Dohvati narudžbu po ID-u

  → **200:** OrderDTO | **404**

- `GET /orders/user/{userId}` – Dohvati korisničke narudžbe

  → **200:** OrderDTO[]

Košarica

- `GET /cart` – Dohvati stavke košarice

  → **200:** CartItemDTO[]

- `POST /cart` – Dodaj stavku

  Tijelo: CartItemDTO → **201:** CartItemDTO

- `PUT /cart/{cartItemId}` – Ažuriraj stavku

  Tijelo: CartItemDTO → **200:** CartItemDTO

- `DELETE /cart/{cartItemId}` – Ukloni stavku

  → **200**

- `DELETE /cart` – Očisti košaricu

  → **200**

Autentifikacija

- *Bearer* token potreban za zaštićene krajnje točke
- Admin uloga potrebna za `GET /orders`

7. Usluga otpreme

Usluga otpreme razvijena je na temelju radnog okvira Spring Boot i modul je za je za upravljanje životnim ciklusom otpreme narudžbi kupaca. Objavljuje API za ove funkcionalnosti nudeći siguran pristup putem mehanizma za autentifikaciju i autorizaciju koristeći JSON web tokena (JWT). Koristi javni ključ korisničke usluge za autentifikaciju dolaznih zahtjeva, što joj omogućuje sigurnu validaciju JWT tokena.

Funkcionalnosti:

- Stvaranje pošiljki prilikom prijema događaja stvaranja narudžbe.
- Ažuriranje statusa pošiljki na temelju poslovne logike i administratorskih operacija te slanje događaja o promjeni statusa
- Dohvaćanje informacija o pošiljci, kako za redovne korisnike tako i za administratore.
- Popis svih pošiljki na administratorskoj razini, s mogućnostima paginacije i sortiranja.

Usluga otpreme šalje poruke ovog formata na Kafka topic (listing 7..1):

Listing 7..1: JSON Schema Shipment

```
{
    "id": "Long",
    "createdAt": "LocalDateTime",
    "updatedAt": "LocalDateTime",
    "status": "ShipmentStatus enum",
    "order": {
```

```
        "orderId": "Long",
        "shippingAddress": "String",
        "billingAddress": "String",
        "createdAt": "LocalDateTime",
        "updatedAt": "LocalDateTime",
        "status": "OrderStatus enum"
    }
}
```

U slučaju usluge otpreme, liquibaseov skup promjena se pokreće kad se aplikacija pokrene. U tom skupu promjena se nalaze naredbe za stvaranje tablice po modelu na slici 7.1.

shipment			
BIGSERIAL	id	PK	NOT NULL
VARCHAR(255)	shipping_address		NOT NULL
VARCHAR(255)	billing_address		NOT NULL
TIMESTAMP	created_at		
TIMESTAMP	updated_at		
VARCHAR(255)	status		NOT NULL
BIGINT	order_id		

Slika 7.1. ER dijagram baze shipment

7.1. API dokumentacija

U ovom poglavlju prikazana je dokumentacija programskog sučelja (API) ostvarene usluge otpreme.

7.1.1. Podatkovni modeli

ShipmentDTO

```
{
  "id": "integer (int64)",
  "createdAt": "string (date-time)",
  "updatedAt": "string (date-time)",
  "status": "string (CREATED|IN_DELIVERY|DELIVERED|REJECTED)",
  "order": "OrderEvent"
}
```

OrderEvent

```
{
  "orderId": "integer (int64)",
  "shippingAddress": "string",
  "billingAddress": "string",
  "createdAt": "string (date-time)",
  "updatedAt": "string (date-time)"
}
```

ShipmentStatusUpdateDTO

```
{
  "status": "string (CREATED|IN_DELIVERY|DELIVERED|REJECTED)"
}
```

PageInfo

```
{
  "totalItems": "integer",
  "totalPages": "integer",
  "currentPage": "integer",
  "pageSize": "integer"
}
```

ShipmentPage

```
{
  "content": "ShipmentDTO[]",
  "pageInfo": "PageInfo"
}
```

7.1.2. API-jeve krajnje točke

Pošiljke

- `GET /shipments?page={0}&size={20}` – Dohvati sve pošiljke (Auth: ADMIN)
  → **200:** ShipmentPage | **401** | **403**
- `GET /shipments/{id}` – Dohvati pošiljku po ID-u
  → **200:** ShipmentDTO | **404**
- `PATCH /shipments/{id}/status` – Ažuriraj status pošiljke (Auth: ADMIN)
  Body: ShipmentStatusUpdateDTO
  → **200:** ShipmentDTO | **400** | **401** | **403** | **404**

Parametri

- **page** – Broj stranice (zadano: 0)
- **size** – Stavki po stranici (zadano: 20)
- **id** – ID pošiljke (int64)

Autentifikacija

- *Bearer* token potreban za: GET /shipments, PATCH /shipments/{id}/status
- ADMIN uloga potrebna za: GET /shipments, PATCH /shipments/{id}/status

8. Usluga sustava preporučivanja

Sustav preporuka razvijen je u programskom jeziku Python koristeći biblioteke i alate za obradu podataka, razvoj web servisa, obradu događaja u stvarnom vremenu, te implementaciju modela strojnog učenja. Koristi javni ključ korisničke usluge za autentifikaciju dolaznih zahtjeva, što omogućuje sigurnu validaciju JWT tokena.

Za izradu REST API servisa korišten je FastAPI. Aplikacija se pokreće s pomoću Uvicorn, asinkronog ASGI servera. Za serijalizaciju i validaciju podataka koristi se Pydantic v2. Modeli za preporuke implementirani su s pomoću scikit-learn, uz podršku za numeričke operacije putem NumPy i obradu podataka putem Pandas. Modeli su serijalizirani i učitavani korištenjem biblioteke joblib.

Za spremanje podataka koristi se PostgreSQL, kojem se pristupa putem SQLAlchemy ORM-a. Tablice u SQLAlchemy se generiraju na osnovu definiranih modela. Modeli prate ER dijagram sa slike 8.1.

Prije pokretanja aplikacije učitavaju se prethodno naučene komponente serijalizirane koristeći joblib:

- TF-IDF vektorizator (`tfidf_vectorizer.joblib`)
- Matrica sličnosti (`similarity_matrix.joblib`)
- Mapa indeksa proizvoda (`product_index_map.joblib`): Rječnik koji preslikava ID-ove proizvoda na njihove odgovarajuće indekse u matrici sličnosti, osiguravajući dosljedno pretraživanje tijekom generiranja preporuka.

Sustav preporuka je hibridni, kombinirajući filtriranje na temelju sadržaja i kolaborativno filtriranje kako bi pružio personalizirane prijedloge proizvoda.

interactions			
BIGINT	id	PK	AUTO_INCREMENT
INT	user_id		NOT NULL
INT	product_id		NOT NULL
VARCHAR(50)	interaction_type		NOT NULL
FLOAT	rating		OPTIONAL
INT	quantity		DEFAULT 1
TIMESTAMP	created_at		NOT NULL

Slika 8.1. ER dijagram tablice interactions baze recommendation

Filtriranje na temelju sadržaja koristi značajke stavki kako bi preporučilo druge stavke slične onima koje korisnik voli, na temelju njegovih prethodnih radnji ili eksplicitnih povratnih informacija [22].

Kod kolaborativnog filtriranja koristi se matrica R u kojoj su zapisane ocjene svih korisnika za objekte. Ova matrica može biti jako slabo popunjena, to jest rijetka, jer korisnici najčešće ne daju ocjene svim objektima. U ovom slučaju matrica nije popunjena ocjenama koje su korisnici dali već se računaju implicitne ocjene.

Funkcija računanja implicite ocjene uzima kao ulaz niz korisničkih interakcija na određenom proizvodu, u obliku DataFramea gdje je jedan redak događaj kupnje. Postoje dva primarna razmatranja koja utječu na rezultat:

- Ukupna kupljena količina (`total_quantity`): Ovo je iznos količina koje je korisnik kupio u svim dokumentiranim interakcijama za proizvod. Izračunava kumulativnu razinu interakcije s proizvodom, pretpostavljajući da je kupnja u velikim količinama veći interes ili preferencija.

- Učestalost interakcije (`frequency`): Ovo je broj različitih događaja kupnje koje je korisnik napravio za proizvod. Procjenjuje ponavljanje interakcije pod pretpostavkom da česte kupnje znače kontinuirani interes.

Korisnik koji kupi mnogo jedinica proizvoda u nekoliko transakcija ostvarit će veći rezultat po količini, ali niži po učestalosti. Suprotno tome, korisnik koji više puta kupuje manje količine ostvarit će veći rezultat po učestalosti. Kombinirani rezultat stoga obuhvaća i intenzitet i dosljednost angažmana korisnika.

Postoje dvije vrste kolaborativnog filtriranja [23]:

- Tehnike temeljene na memoriji uvelike se oslanjaju na jednostavne mjere sličnosti (kosinusna sličnost, Pearsonova korelacija, Jaccardov koeficijent itd.) kako bi spojile slične ljude ili stavke.

- Tehnike temeljene na modelu pokušavaju dodatno popuniti matricu ocjena kupaca za stavke. One se bave zadatkom "pogađanja" koliko će se korisniku svidjeti stavka koju prije nije susreo. Za to koriste algoritme strojnog učenja.

SVD (engl. *Singular Value Decomposition*) je matematička tehnika koja nam omogućuje rastavljanje matrice na svojstvene vrijednosti i svojestvene vektore. Te vrijednosti i vektori zatim se mogu koristiti za predviđanje izvorne matrice. U slučaju sustava preporuka, matrica je obično matrica korisnik-proizvod, gdje svaki redak predstavlja korisnika, svaki stupac predstavlja proizvod, a ćelije sadrže ocjene ili interakcije između korisnika i proizvoda (u ovom slučaju implicitne ocjene) [24].

Cilj SVD-a je pronaći dvije nove matrice, jednu koja predstavlja korisnike, a drugu koja predstavlja proizvode, koje kada se pomnože, aproksimiraju izvornu matricu što je moguće bliže. Ove nove matrice nazivaju se latentne reprezentacije i sadrže važne informacije o korisnicima i proizvodima koje se mogu koristiti za davanje preporuka [24].

Truncated SVD, koji je korišten u ovom sustavu, uzima k svojstvenih vrijednosti za aproksimaciju izvorne matrice. Taj broj k definiran je u konfiguraciji aplikacije. Ovo stvara komprimirane prikaze: svaki korisnik i proizvod preslikavaju se u k-dimenzionalni latentni prostor. Za predviđanje nedostajuće ocjene, izračunava se skalarni produkt latentnog vektora korisnika i latentnog vektora stavke [25].

Kada dobije zahtjev za preporukom, sustav prvo dohvaća preporuke na temelju sadržaja koristeći sličnost proizvoda. To radi tako da koristi nedavne interakcije i pronalazi slične proizvode matrice. Nakon toga, predviđa proizvode koje korisnik još nije ocijenio koristeći SVD faktorizaciju matrice. Ove vrijednosti se normaliziraju u interval [0,1] te se onda kombiniraju oba izvora stvaranjem rječnika s content_score i collaborative_score po proizvodu. Proizvodi koji postoje u oba izvora dobivaju ocjene iz oba, drugi dobivaju jednu ocjenu i 0 za drugu. Svaki proizvod dobiva hibridnu ocjenu, izračunatu kao suma ocjena sadržaja i kolaborativne ocjene. Ocjene se kontroliraju konfiguracijom (content_weight, collaborative_weight). Sortira se prema hybrid_score te se vraća prvih N najboljih rezultata.

Usluga sustava preporučivanja je pretplaćena na topic korisnik-proizvod i kada dobije poruku sa shemom prikazanom listingom 6..2 zapisuje je u tablicu interakcija.

Kako bi modele preporuka održao ažurnima s dolaznim podacima, sustav koristi AsyncIOScheduler za pokretanje periodičnog učenja modela. Period ponovnog pokretanja učenja modela definiran je u konfiguraciji. Periodično ponovno učenje, umjesto kontinuiranog ažuriranja odabrano je zato što se narudžbe najčešće ne događaju jako često (svake minute, svakog sata..).

Nakon generiranja preporuka za korisnika, sustav pohranjuje rezultat u Redis. Preporuke se serijaliziraju u JSON. Vrijeme isteka je 1 sat (3600 sekundi). Ovo sprječava ponovljeno izračunavanje kada korisnik osvježi ili ponovno zatraži iste preporuke unutar tog razdoblja. Kada se dobije zahtjev za preporukom, prvo se pokuša dohvatiti podatke iz Redisa prije generiranja novih preporuka.

8.1. API dokumentacija

U ovom poglavlju prikazana je dokumentacija programskog sučelja (API) ostvarene usluge sustava preporučivanja.

8.1.1. Podatkovni modeli

RecommendationRequest

```
1 {
```

```
  "userId": "integer (int64)",
  "page": "integer (min: 0, default: 0)",
  "pageSize": "integer (min: 1, max: 100, default: 10)"
}
```

PaginatedRecommendationResponse

```
{
  "recommendations": "integer[] (int64)",
  "pagination": "Pagination",
  "metadata": "Metadata"
}
```

Pagination

```
{
  "currentPage": "integer",
  "pageSize": "integer",
  "totalItems": "integer",
  "totalPages": "integer",
  "hasNext": "boolean",
  "hasPrevious": "boolean",
  "nextPage": "integer (nullable)",
  "previousPage": "integer (nullable)"
}
```

Metadata

```
{
  "userId": "integer (int64)",
  "algorithm": "string (hybrid)",
  "generatedAt": "string (date-time)",
  "cacheSizeUsed": "integer",
  "actualRecommendations": "integer",
  "pageItems": "integer"
}
```

8.1.2. API-jeve krajnje točke

Preporuke

- `POST /api/v1/recommendations` – Dohvati personalizirane preporuke (Auth)
  Body: RecommendationRequest
  → **200:** PaginatedRecommendationResponse | **400** | **401** | **404**

Parametri

- **userId** – ID korisnika (int64, obavezno)
- **page** – Broj stranice (zadano: 0, minimum: 0)
- **pageSize** – Stavki po stranici (zadano: 10, 1-100)

Autentifikacija

Bearer token potreban za sve operacije

9. Usluga analitike

Usluga analitike razvijena je na temelju radnog okvira Spring Boot i modul je koji pruža sveobuhvatne poslovne metrike u ključnim domenama e-trgovine, uključujući narudžbe, zalihe, pošiljke i košarice za kupnju. Objavljuje API za ovu funkcionalnost nudeći siguran pristup putem mehanizma za autentifikaciju i autorizaciju koristeći JSON web token (JWT). Koristi javni ključ korisničke usluge za autentifikaciju dolaznih zahtjeva, što omogućuje sigurnu validaciju JWT tokena. Za pristup ClickHouse bazi koristi se JdbcTemplate, kojem se prosljeđuju SQL upiti koji se zatim izvršavaju nad bazom.

Funkcionalnosti:

- Pruža 30-dnevni sažetak metrike narudžbi tako što agregira broj statusa narudžbi, ukupne prihode i stope konverzije (postotke isporuke/otkazivanja) tijekom razdoblja od 30 dana koristeći uvjetne zbrojeve (`sumIf()`) i brojeve (`countIf()`).
- Analizira distribuciju načina plaćanja putem broja narudžbi, doprinosa prihodima i postotnih udjela
- Izračunava trenutačne metrike zaliha (broj na zalihi/nema na zalihi, rezervirane količine) koristeći `row_number()` za filtriranje najnovijih zapisa o zalihama.
- Grupira podatke o zalihama prema lokaciji, agregira dostupne/rezervirane količine i izračunava postotke stanja zaliha po skladištu.
- Primjenjuje se `toStartOfWeek()` segmentiranje za analizu 4-tjednih trendova u razinama zaliha i pojavama nestašice zaliha.
- Izračunava stope uspješnosti dostave i vremena tranzita koristeći uvjetne agregacije (`avgIf()`) na zapisima pošiljki filtriranim prema statusu.

- Kvantificira razdoblja neaktivnosti košarice koristeći `dateDiff()` za izračun stopa napuštanja na pragovima od 24 sata/72 sata/1 tjedan s povezanim gubitkom prihoda.
- Računa trajanje narudžbi u pojedinim statusima

9.1. API dokumentacija

U ovom poglavlju prikazana je dokumentacija programskog sučelja (API) ostvarene usluge analitike.

9.1.1. Podatkovni modeli

OrderAnalyticsDTO

```
{
  "totalOrders": "integer (int64)",
  "pendingOrders": "integer (int64)",
  "paidOrders": "integer (int64)",
  "processingOrders": "integer (int64)",
  "shippedOrders": "integer (int64)",
  "deliveredOrders": "integer (int64)",
  "cancelledOrders": "integer (int64)",
  "refundedOrders": "integer (int64)",
  "totalOrderValue": "number (double)",
  "deliveredRevenue": "number (double)",
  "lostRevenue": "number (double)",
  "avgOrderValue": "number (double)",
  "avgDeliveredOrderValue": "number (double)",
  "deliveryRatePct": "number (double)",
  "cancellationRatePct": "number (double)"
}
```

OrderStatusDurationDTO

```
{
  "status": "string",
  "statusOccurrences": "integer (int64)",
  "avgHoursInStatus": "number (double)",
```

```
  "medianHoursInStatus": "number (double)",
  "maxHoursInStatus": "integer (int64)"
}
```

PaymentMethodAnalyticsDto

```
{
  "paymentMethod": "string",
  "orderCount": "integer (int64)",
  "totalRevenue": "number (double)",
  "avgOrderValue": "number (double)",
  "percentageOfOrders": "number (double)",
  "percentageOfRevenue": "number (double)"
}
```

InventoryHealthDTO

```
{
  "inStockProducts": "integer (int64)",
  "lowStockProducts": "integer (int64)",
  "outOfStockProducts": "integer (int64)",
  "totalAvailableQty": "integer (int64)",
  "totalReservedQty": "integer (int64)",
  "avgAvailablePerProduct": "number (double)",
  "inStockRatePct": "number (double)",
  "outOfStockRatePct": "number (double)",
  "productsNeedReorder": "integer (int64)"
}
```

WarehouseInventoryStatsDTO

```
{
  "warehouseLocation": "string",
  "productsCount": "integer (int64)",
  "totalAvailableQty": "integer (int64)",
  "totalReservedQty": "integer (int64)",
  "avgQtyPerProduct": "number (double)",
  "inStockCount": "integer (int64)",
  "outOfStockCount": "integer (int64)",
  "stockHealthPct": "number (double)"
```

```
}
```

WeeklyInventoryStatsDTO

```
{
  "weekStart": "string (date)",
  "totalAvailable": "integer (int64)",
  "totalReserved": "integer (int64)",
  "avgAvailablePerProduct": "number (double)",
  "outOfStockCount": "integer (int64)",
  "productsTracked": "integer (int64)"
}
```

ShipmentAnalyticsDto

```
{
  "totalShipments": "integer",
  "deliveredShipments": "integer",
  "inTransitShipments": "integer",
  "rejectedShipments": "integer",
  "pendingShipments": "integer",
  "deliverySuccessRatePct": "number (double)",
  "rejectionRatePct": "number (double)",
  "avgDeliveryTimeHours": "number (double)",
  "medianDeliveryTimeHours": "number (double)"
}
```

CartAnalyticsDTO

```
{
  "totalActiveCarts": "integer (int64)",
  "totalItemsInCarts": "integer (int64)",
  "totalCartValue": "number (double)",
  "avgCartValue": "number (double)",
  "avgItemsPerCart": "number (double)",
  "cartsAbandoned24h": "integer (int64)",
  "cartsAbandoned72h": "integer (int64)",
  "cartsAbandoned1week": "integer (int64)",
  "abandonmentRate24hPct": "number (double)",
  "abandonmentRate72hPct": "number (double)",
```

```
  "abandonedCartValue24h": "number (double)",
  "abandonedCartValue72h": "number (double)",
  "usersWithCartNoRecentOrder": "integer (int64)",
  "noRecentOrderRatePct": "number (double)"
}
```

9.1.2. API-jeve krajnje točke

Narudžbe

- `GET /order/summary` – Dohvati analitiku narudžbi za zadnjih 30 dana
  → **200:** OrderAnalyticsDTO
- `GET /order/status` – Dohvati analitiku trajanja statusa narudžbi
  → **200:** OrderStatusDurationDTO[]
- `GET /order/payment` – Dohvati statistike načina plaćanja
  → **200:** PaymentMethodAnalyticsDto[]

Inventar

- `GET /inventory/summary` – Dohvati snimak zdravlja inventara
  → **200:** InventoryHealthDTO
- `GET /inventory/warehouse` – Dohvati statistike inventara po skladištu
  → **200:** WarehouseInventoryStatsDTO[]
- `GET /inventory/weekly` – Dohvati tjedne trendove inventara
  → **200:** WeeklyInventoryStatsDTO[]

Pošiljke

- `GET /shipment/summary` – Dohvati analitiku pošiljaka
  → **200:** ShipmentAnalyticsDto

Košarice

- `GET /cart/summary` – Dohvati sažetak analitike košarica

  → **200:** CartAnalyticsDTO

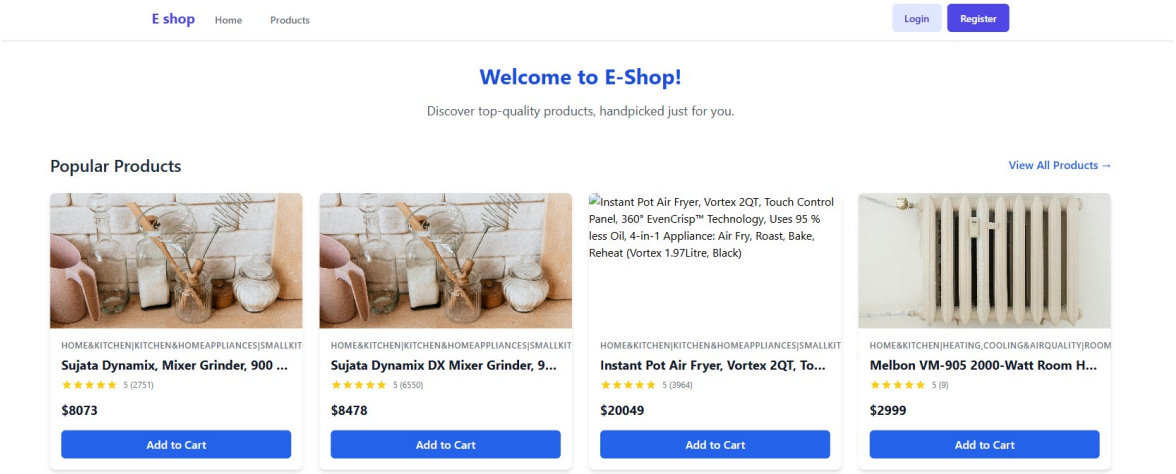
10. Korisničko sučelje

Grafičko korisničko sučelje (GUI) za e-trgovinu je aplikacija koja koristi React i Tailwind CSS (responzivno i prilagodljivo stiliziranje). React omogućuje izgradnju dinamičkog i komponentno utemeljenog korisničkog sučelja, što olakšava upravljanje stanjima aplikacije i ponovno korištenje koda. Usmjeravanje na strani klijenta postiže se putem React Routera, a Axios omogućuje API komunikaciju. Korisničko sučelje uključuje Recharts za vizualizaciju podataka. Arhitektura podržava osnovne funkcionalnosti e-trgovine poput pregledavanja proizvoda, funkcionalnosti košarice, prijave korisnika i administratorskih nadzornih ploča s analitikom.

Funkcionalnosti:

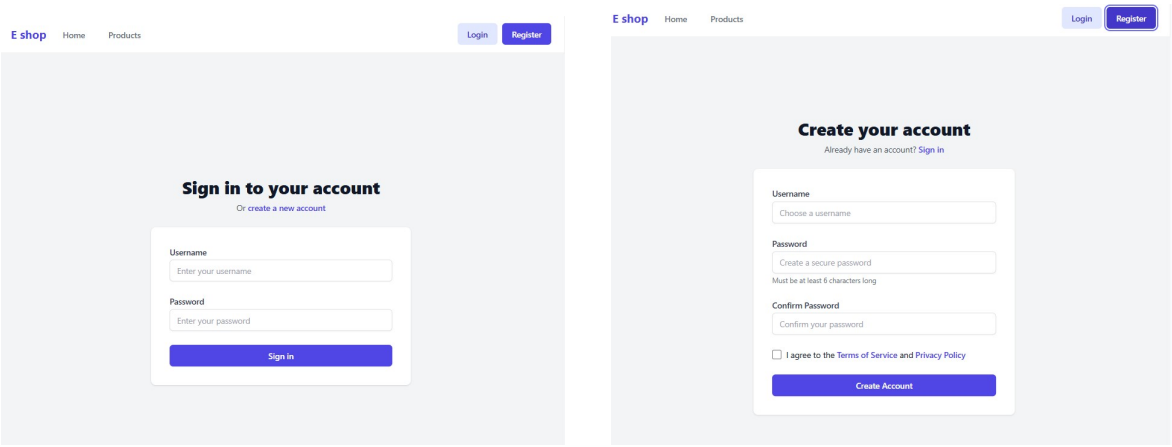
- Značajke usmjerene na korisnika
  - Autentifikacija i autorizacija: JWT prijava, pristup temeljen na ulogama (korisnik/administrator)
  - Pregledavanje proizvoda: Pretraživanje, filtriranje, paginacija
  - Košarica za kupnju: pregled, dodavanje i uklanjanje proizvoda i mijenjanje količina proizvoda
  - Pregled povijesti narudžbi i praćenje statusa
- Administratorska nadzorna ploča
  - Ažuriranje dostupne količine, lokacije proizvoda i ostalog
  - Ažuriranje statusa narudžbe
  - Vizualizacija podataka

Na slici 10.1. prikazana je početna stranica koju vidi korisnik koji nije prijavljen.



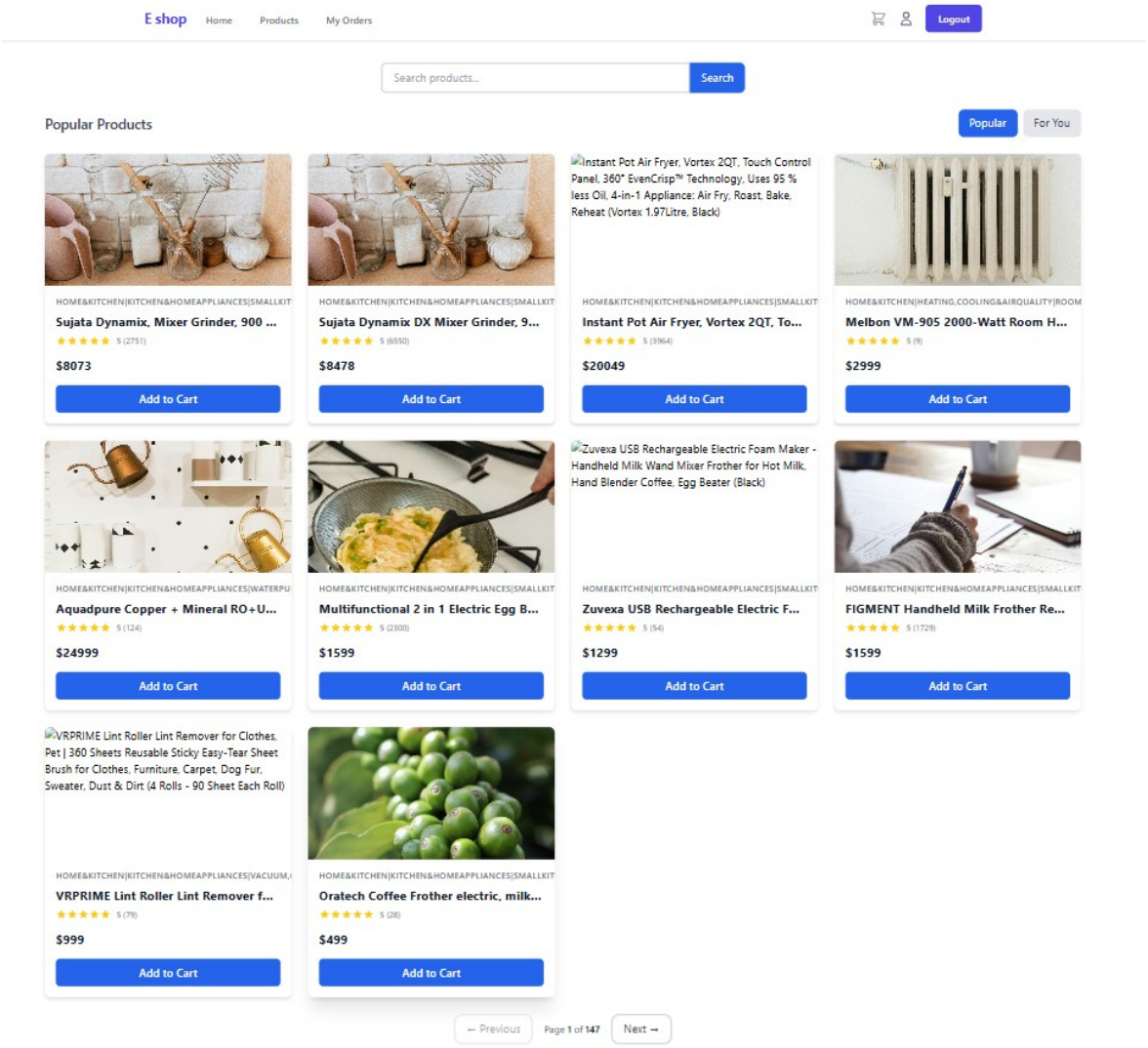
Slika 10.1. Početna stranica

Na slici 10.2. prikazane su stranice za prijavu i registraciju.



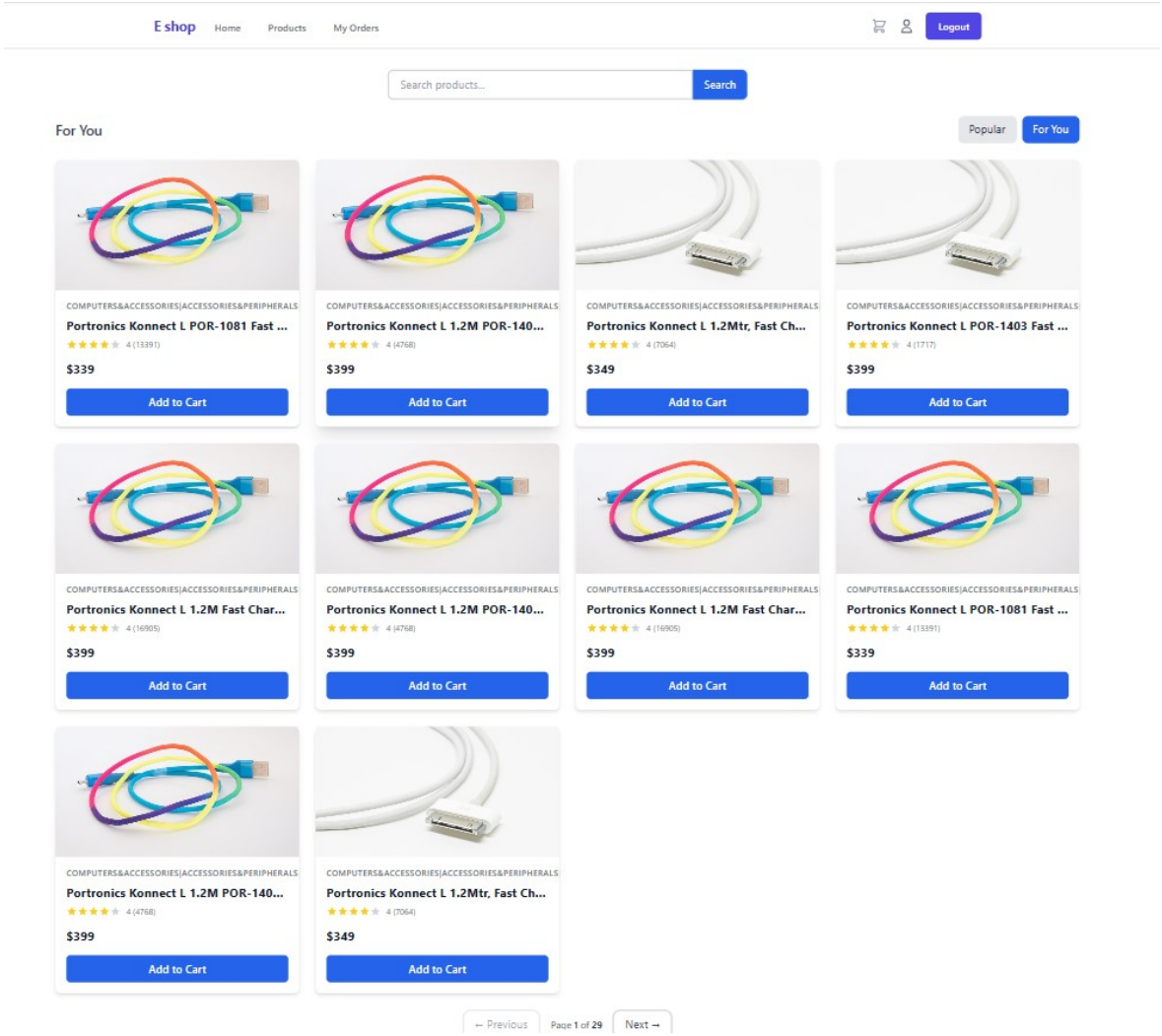
Slika 10.2. Stranice za prijavu i registraciju

Na slici 10.3. prikazana je stranica sa popularnim proizvodima.



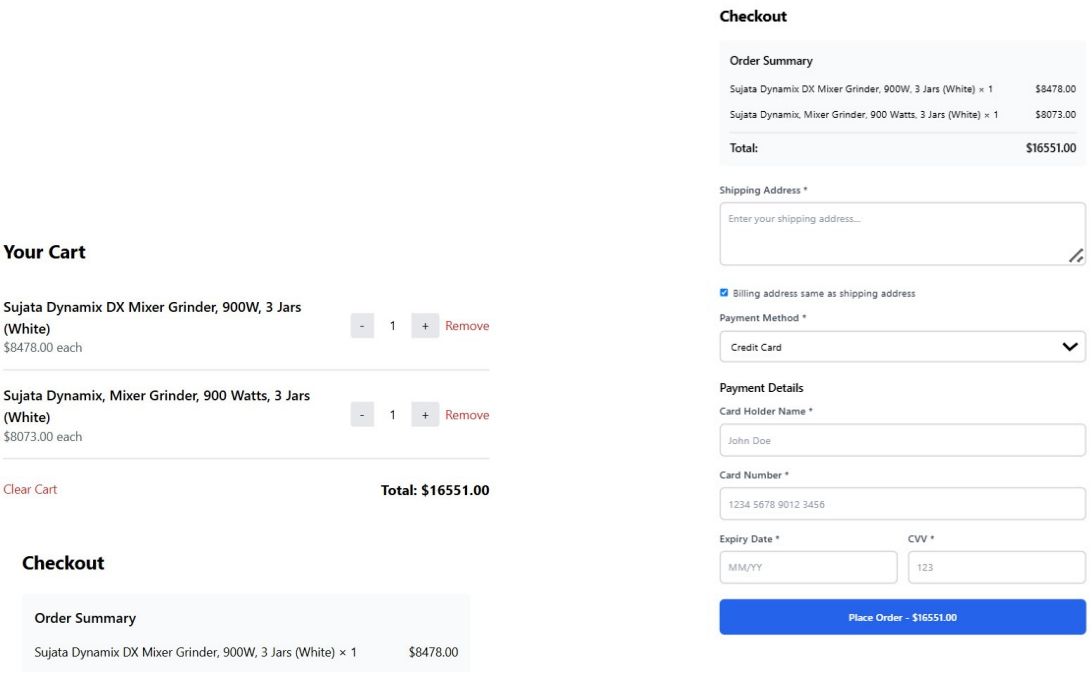
Slika 10.3. Stranica sa popularnim proizvodima

Na slici 10.4. prikazana je stranica sa preporučenim proizvodima (prikazuje se samo korisnicima koji su prijavljeni). Ovaj korisnik je prije kupio sličan kabel.



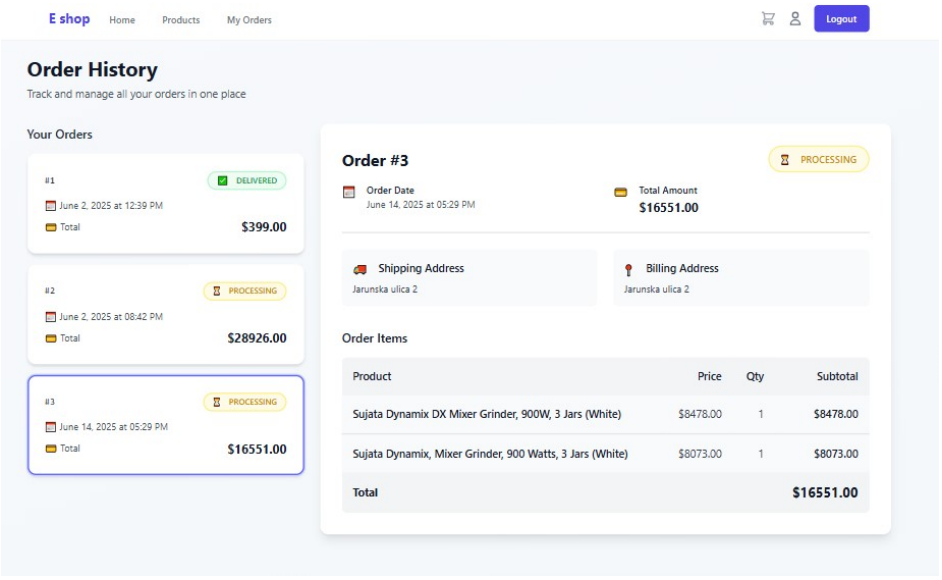
Slika 10.4. Stranica sa preporučenim proizvodima

Na slici 10.5. prikazana je košarica kad korisnik doda proizvod.



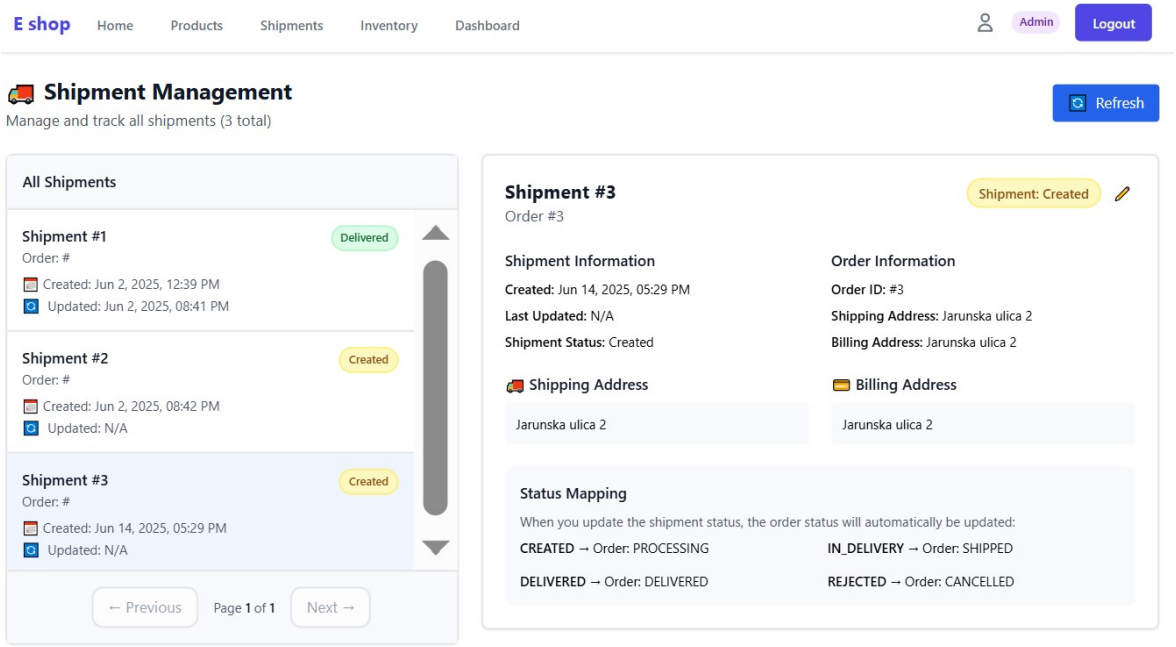
Slika 10.5. Košarica i završavanje narudžbe

Na slici 10.6. prikazane su prošle narudžbe.



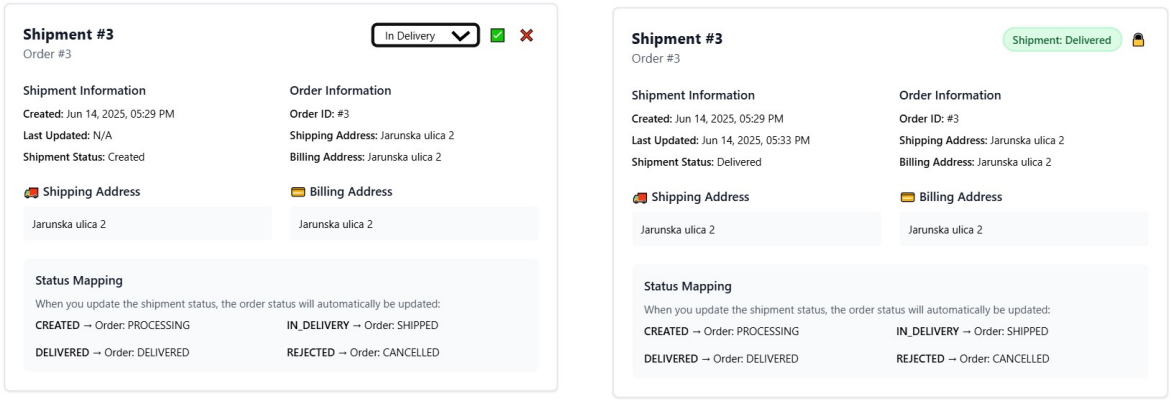
Slika 10.6. Prošle narudžbe

Na slici 10.7. prikazano je administratorsko sučelje za pregled pošiljki.



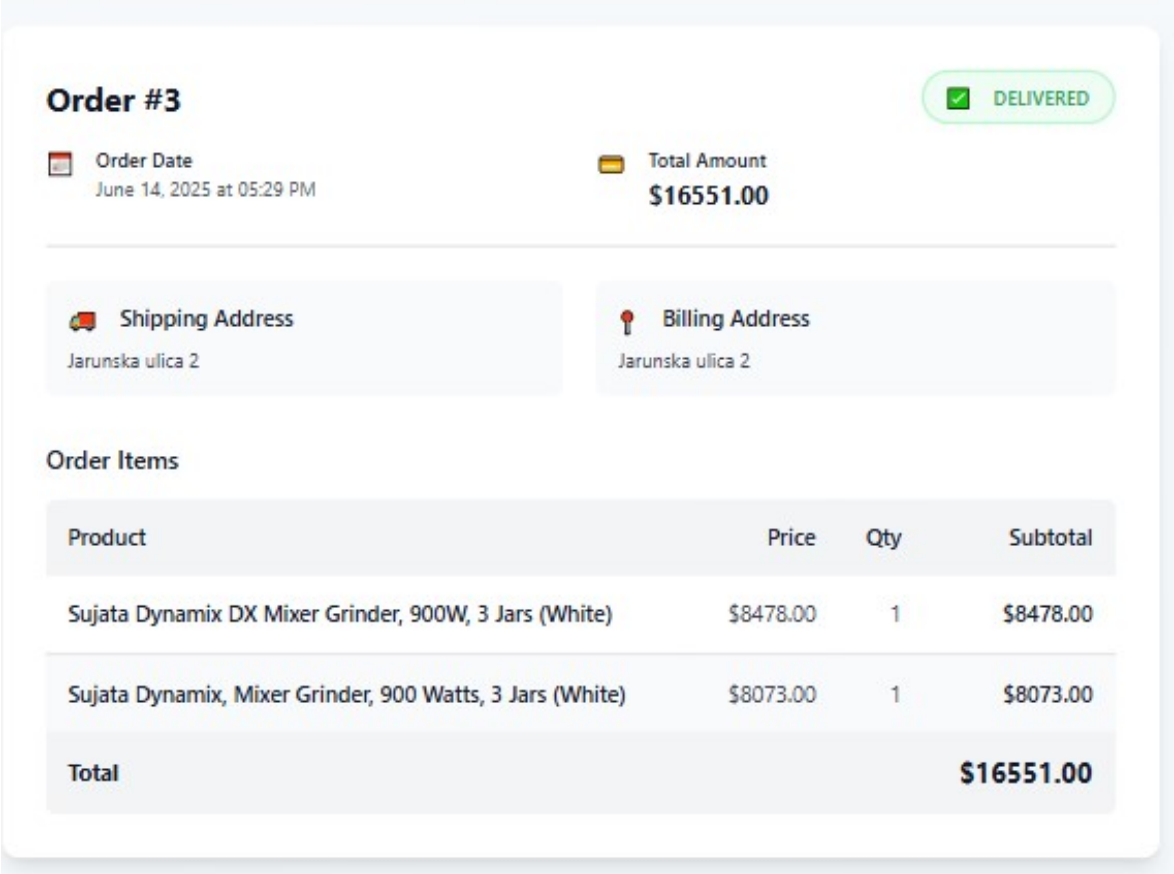
Slika 10.7. Pregled pošiljki

Na slici 10.8. prikazano je administratorsko sučelje za uređivanje pošiljki i pošiljka kad je uređena sa statusom ’dostavljeno’ koji se ne može dalje mijenjati.



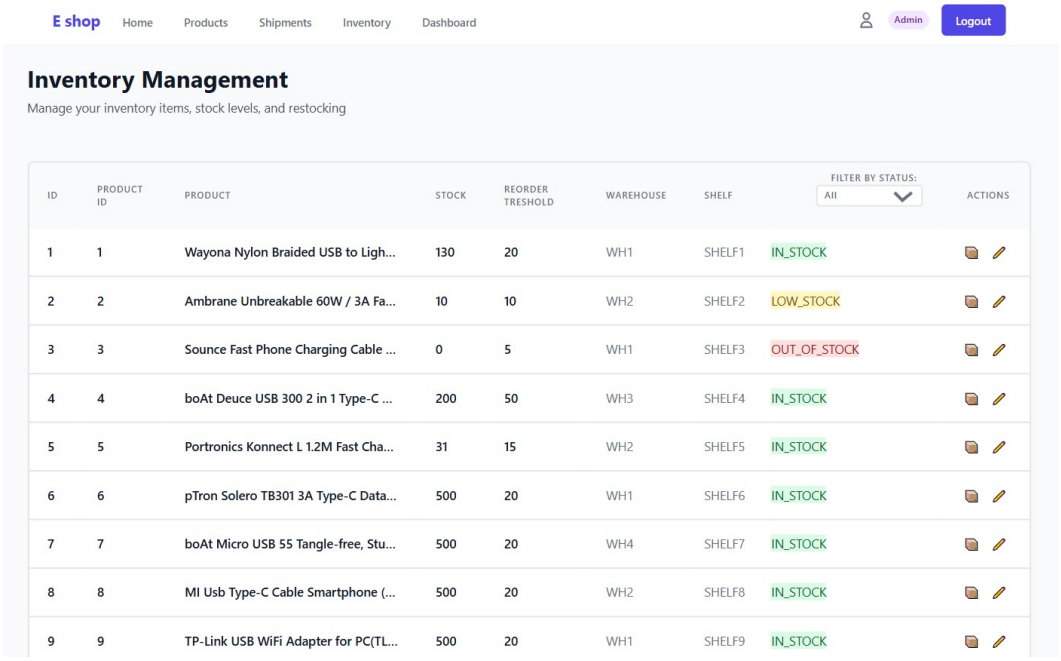
Slika 10.8. Uređivanje pošiljke

Na slici 10.9. prikazani su detalji narudžbe koje korisnik vidi, status je dostavljeno nakon što je administrator stavio taj status.



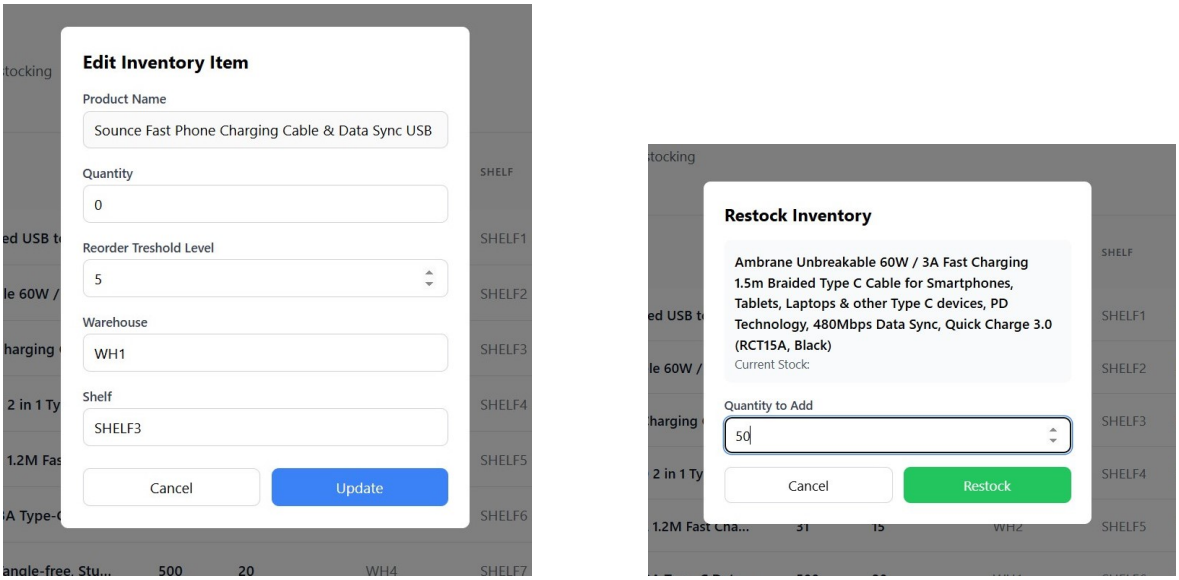
Slika 10.9. Dostavljena narudžba na stranici povijesti narudžbi kod korisnika

Na slici 10.10. se vidi administratorski pregled inventara.



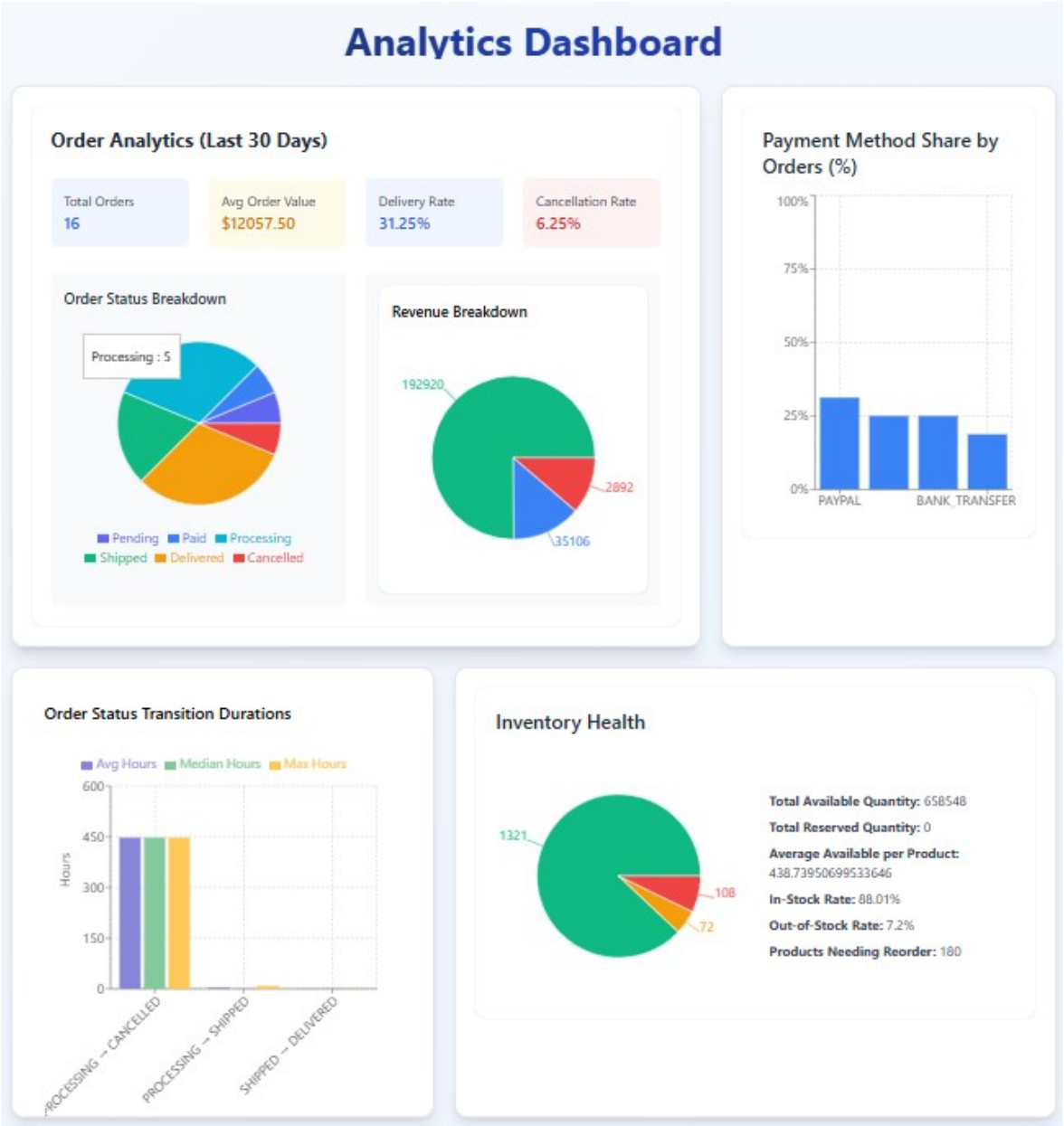
Slika 10.10. Pregled inventara

Na slici 10.11. vidi se mogućnost uređivanja inventara - dodavanje količine ili detaljnije uređivanje.

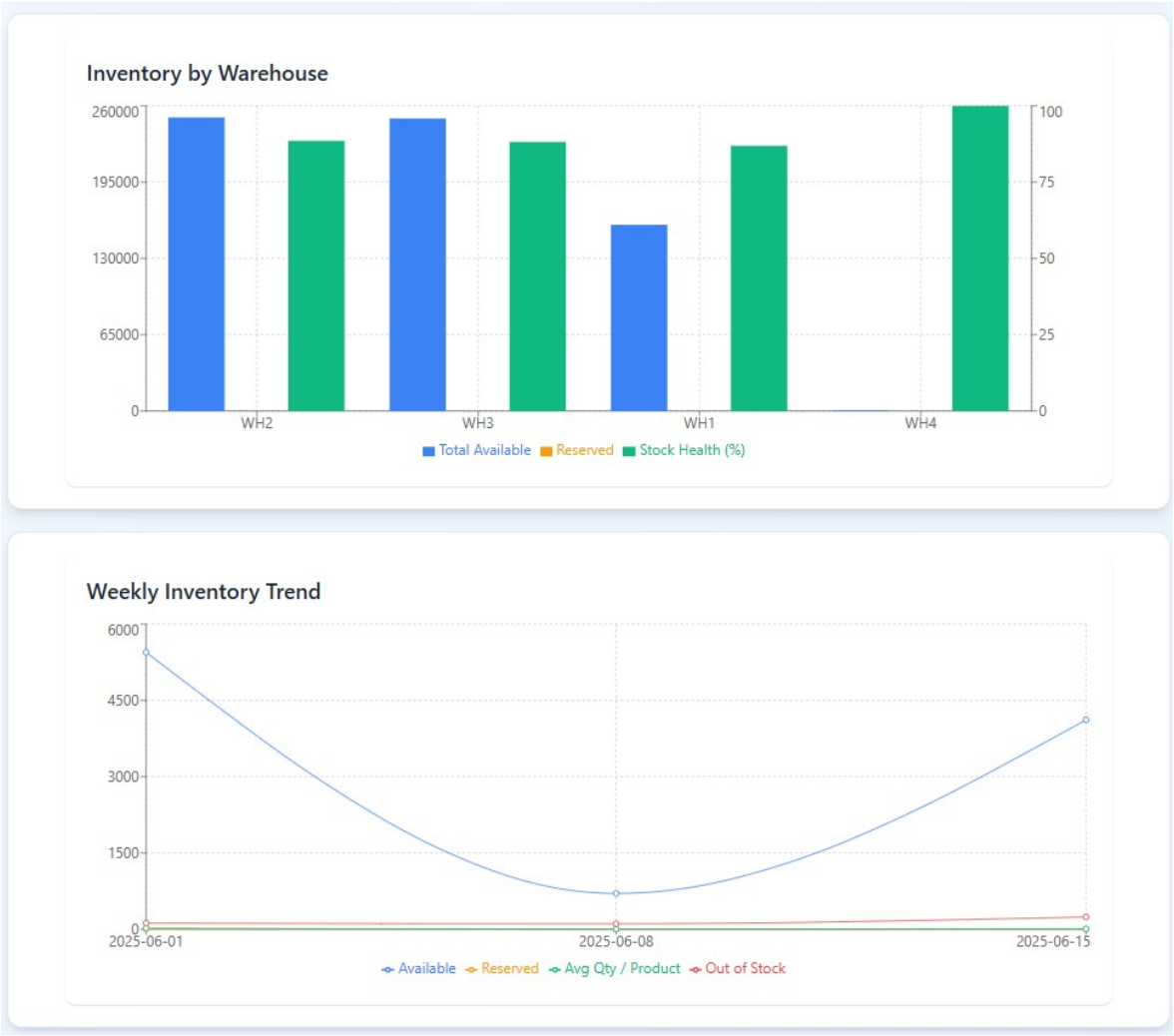


Slika 10.11. Uređivanje inventara

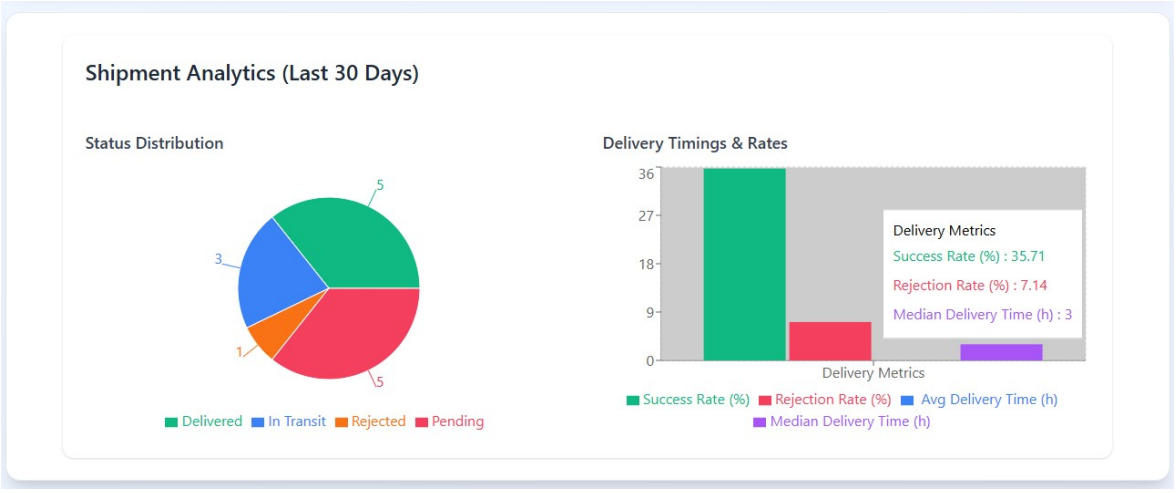
Na slikama 10.12., 10.13., 10.14. prikazana je administratorska nadzorna ploča s analitikom.



Slika 10.12. Pregled analitike



Slika 10.13. Pregled analitike



Slika 10.14. Pregled analitike

11. Zaključak

Brzi rast e-trgovine zahtijeva razvoj skalabilnih i prilagodljivih softverskih rješenja koja mogu pružiti personalizirana korisnička iskustva. Obrazac mikrousluga prikladan je izbor za izgradnju složenih web-mjesta za e-trgovinu jer poboljšava modularnost, a aplikacija je lakša za razumijevanje, razvoj i testiranje.

Kroz ovaj diplomski rad izrađen je sustav koji obuhvaća usluge za upravljanje narudžbama, pošiljkama, inventarom, katalogom proizvoda, analitikom i preporukama. Sustav za preporučivanje koristi povijest kupnji i sličnosti između proizvoda kako bi krajnjem korisniku pružio prijedloge, čime se poboljšava korisničko iskustvo i potencijalno povećava prodaja. Za učenje modela preporuka korišteni su javno dostupni skupovi podataka.

Također, izrađeno je intuitivno korisničko sučelje zajedno s administratorskim dijelom koji nudi pregled i ažuriranje pošiljki i inventara te nadzornom pločom koja ima vizualizacije glavnih performansi što doprinosi transparentnosti poslovanja i podrški odlučivanju.

Moguća poboljšanja ove aplikacije bi bila dodavanje mogućnosti unosa novih proizvoda kroz administratorsko sučelje, isprobavanje više vrsta algoritama za sustav preporučivanja, mogućnost da korisnik prilikom registracije izabere najdraže kategorije te uvođenje analitike u stvarnom vremenu korištenjem obrade tokova podataka (engl. *data streams*). Nadalje, preporuke bi se mogle dodatno poboljšati implementacijom sustava ocjenjivanja i praćenjem korisničkih klikova što bi omogućilo bolji uvid u korisničke interese i dovelo do učinkovitijeg sustava preporučivanja.

Projekt pokazuje praktičnu primjenu trenutačnih praksi programsog inženjerstva zajedno s principima strojnog učenja u stvaranju sustava e-trgovine.

Literatura

[1] Microservices.io, https://microservices.io/, [mrežno; stranica posjećena: svibanj 2025.].

[2] J. Singh, https://medium.com/hashmapinc/the-what-why-and-how-of-a-microservices-architecture-4179579423a9, [mrežno; stranica posjećena: svibanj 2025.].

[3] M. Ozkaya, https://medium.com/design-microservices-architecture-with-patterns/top-10-characteristics-of-microservices-12b046a59bfc, [mrežno; stranica posjećena: svibanj 2025.].

[4] GeeksForGeeks, https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/, [mrežno; stranica posjećena: travanj 2025.].

[5] F. Karabiber, https://www.learndatasci.com/glossary/cosine-similarity/, [mrežno; stranica posjećena: svibanj 2025.].

[6] P. Sharma, "How to save and load machine learning models in python using joblib library?" https://www.analyticsvidhya.com/blog/2023/02/how-to-save-and-load-machine-learning-models-in-python-using-joblib-library/, [mrežno; stranica posjećena: lipanj 2025.].

[7] Amazon, https://aws.amazon.com/docker/, [mrežno; stranica posjećena: svibanj 2025.].

[8] PostgreSQL, https://www.postgresql.org/, [mrežno; stranica posjećena: travanj 2025.].

[9] Apache Kafka, https://kafka.apache.org/, [mrežno; stranica posjećena: travanj 2025.].

[10] M. Jaiswal, https://manishankarjaiswal.medium.com/understanding-broker-cluster-and-zookeeper-in-kafka-a-beginners-guide-part-4-2a28640c8470, [mrežno; stranica posjećena: svibanj 2025.].

[11] Provectus, https://github.com/provectus/kafka-ui, [mrežno; stranica posjećena: svibanj 2025.].

[12] IBM, https://www.ibm.com/think/topics/redis, [mrežno; stranica posjećena: svibanj 2025.].

[13] S. Asad, https://medium.com/@suffyan.asad1/beginners-guide-to-clickhouse-introduction-features-and-getting-started-55315107399a, [mrežno; stranica posjećena: svibanj 2025.].

[14] Databricks, https://docs.databricks.com/aws/en/dlt/what-is-change-data-capture, [mrežno; stranica posjećena: svibanj 2025.].

[15] D. Snapshots, https://medium.com/@jay.gokani/change-data-capture-cdc-with-debezium-a8a6a77195df, [mrežno; stranica posjećena: svibanj 2025.].

[16] Debezium.io, https://debezium.io/, [mrežno; stranica posjećena: svibanj 2025.].

[17] S. Raina, https://medium.com/@swayamraina/symmetric-vs-asymmetric-jwts-bd5d1a9567f6, [mrežno; stranica posjećena: svibanj 2025.].

[18] Liquibase, https://docs.liquibase.com/concepts/introduction-to-liquibase.html#:~:text=Liquibase%20is%20a%20database%20schema,your%20migration%20scripts%20in%20SQL., [mrežno; stranica posjećena: svibanj 2025.].

[19] P. Kucharz, “Pessimistic locking in jpa”, https://www.baeldung.com/jpa-pessimistic-locking, [mrežno; stranica posjećena: travanj 2025.].

[20] Spring Docs, https://docs.spring.io/spring-framework/reference/data-access/transaction/declarative/annotations.html, [mrežno; stranica posjećena: travanj 2025.].

[21] Spring Cloud OpenFeign, https://docs.spring.io/spring-cloud-openfeign/docs/current/reference/html/, [mrežno; stranica posjećena: svibanj 2025.].

[22] developers.google.com, “Content-based filtering”, https://developers.google.com/machine-learning/recommendation/content-based/basics, [mrežno; stranica posjećena: svibanj 2025.].

[23] Y. Ebrahim, https://yasserebrahim.wordpress.com/2012/10/13/, [mrežno; stranica posjećena: svibanj 2025.].

[24] R. Gupta, “How singular value decomposition (svd) is used in recommendation systems, “clearly explained”.” https://medium.com/@ritik_gupta/how-singular-value-decomposition-svd-is-used-in-recommendation-systems-clearly-explained-201b24e175db, [mrežno; stranica posjećena: lipanj 2025.].

[25] milvus.io, “How does singular value decomposition (svd) work in recommender systems?” https://milvus.io/ai-quick-reference/how-does-singular-value-decomposition-svd-work-in-recommender-systems, [mrežno; stranica posjećena: lipanj 2025.].

Sažetak

Ovaj rad prikazuje oblikovanje i razvoj skalabilne aplikacije za e-trgovinu temeljene na mikrouslužnoj arhitekturi sa sustavom za preporuke proizvoda. Detaljno je opisana potrebna infrastruktura, uključujući baze podataka, sustav za razmjenu poruka Apache Kafka i ostale ključne komponente te njihovo pokretanje kroz Docker. Opisan je i razvijen sustav koji obuhvaća funkcionalnosti kao što su upravljanje narudžbama, zalihama, pošiljkama i katalogom proizvoda. Sustav za preporuke temelji se na analizi povijesti kupnji korisnika i sličnosti među proizvodima, s ciljem generiranja personaliziranih prijedloga. Također je razvijeno korisničko sučelje koje omogućuje pregled proizvoda, izradu narudžbi te administraciju. Administratorski dio uključuje funkcionalnosti za pregled i ažuriranje pošiljki i inventara, kao i nadzornu ploču s vizualizacijama ključnih performansi sustava. Rad demonstrira primjenu suvremenih principa programsog inženjerstva i strojnog učenja u izradi održivog i proširivog sustava za elektroničku trgovinu.

Ključne riječi: e-trgovina; sustav prepručivanja; mikrouslužna arhitektura; Apache Kafka; Docker; Clickhouse; Redis; Spring Boot; Python; SVD; filtriranje temeljeno na sadržaju; kolaborativno filtriranje

Abstract

This paper presents the design and development of a scalable e-commerce application based on a microservice architecture with a product recommendation system. The required infrastructure, including databases, the Apache Kafka messaging system and other key components, and their launch through Docker are described in detail. The system is described and developed, which includes functionalities such as order, inventory, shipment and product catalog management. The recommendation system is based on the analysis of user purchase history and similarities between products, with the aim of generating personalized suggestions. A user interface has also been developed that allows for product review, order creation and administration. The administrative part includes functionalities for reviewing and updating shipments and inventory, as well as a dashboard with visualizations of key system performances. The paper demonstrates the application of modern principles of software engineering and machine learning in the development of a sustainable and scalable e-commerce system.

Keywords: e-commerce; recommendation system; microservices architecture; Apache Kafka; Docker; Clickhouse; Redis; Spring Boot; Python; SVD; collaborative filtering; content based filtering