

*Prolećni semestar, 2024/25*

*PREDMET: CS230 Distribuirani sistemi*

Projektni zadatak

***E-commerce aplikacija bazirana na mikroservisima***

Ime i prezime: **Stevan Stojanović**

Broj indeksa: **5607**

Datum izrade: **30.08.2025.**

Asistent: **Anđela Grujić**

Profesor: **Nemanja Zdravković**

Sadržaj

[Uvod i teoretska postavka izabrane teme 3](#_Toc169127152)

[Primer/studija slučaja primene izabrane teme 5](#_Toc169127153)

[Zaključak 15](#_Toc169127154)

[Literatura 16](#_Toc169127155)

# Uvod i teoretska postavka izabrane teme

E-commerce sistemi predstavljaju glavni vid savremenog poslovanja jer omogućavaju kupovinu i prodaju proizvoda i usluga preko interneta. Ovakvi sistemi zahtevaju visok nivo dostupnosti, skalabilnosti i bezbednosti kako bi korisnicima obezbedili odgovarajuće iskustvo. U osnovi samog sistema nalaze se procesi poput registracije i autentifikacije korisnika, upravljanja samim proizvodima, kreiranje porudžbina i ažuriranja podataka u realnom vremenu.

Tradicionalno su se ovi sistemi implementirali kao monolitne aplikacije. Monolitni pristup znači da ceo sistem (od samog korisničkog interfejsa do rada sa bazom) funkcioniše kao jedinstvena celina. Iako ovaj pristup može biti jednostavniji u početku razvoja vremenom postaje ograničavajući zbog problema sa skalabilnošću i održavanjem.

Rešenje za ovaj problem predstavlja mikroservisna arhitektura. Mikroservisi predstavljaju način projektovanja aplikacija kroz niz manjih nezavisnih komponenti, odnosno servisa. Svaki od tih servisa ima jasno definisanu ulogu. Ovaj pristup projektovanja nam omogućava da se pojedini servisi razvijaju, testiraju i distribuiraju nezavisno od drugih i time obezbeđujemo otpornost i fleksibilnost sistema. Ako jedan servis prestane da radi ostatak aplikacije može slobodno da funkcioniše nezavisno od njega.

**Arhitektura distribuiranog sistema**

Arhitektura e-commerce aplikacije zasnovane na mikroservisima razvijene u okviru ovog projekta obuhvata sledeće komponente:

* **API Gateway (Flask):** Gateway služi kao ulazna tačka u sistem i obavlja ulogu posrednika između frontenda i ostalih servisa. On upravlja autentifikacijom i autorizacijom korisnika, prosleđuje zahteve ka potrebnim servisima i osigurava logovanje i kontrolu pristupa.
* **User Service (Flask + SQLite):** Ovaj servis je zadužen za registraciju i login korisnika. Svaki korisnik se registruje pomoću imena, emaila i lozinke, dok admin ima unapred definisan nalog i može samo da se uloguje na sistem. Servis koristi SQLite bazu podataka za trajno čuvanje korisničkih informacija.
* **Product Service (Flask + SQLite):** Ovaj servis je zadužen za upravljanje proizvodima i obezbeđuje sve neophodne CRUD funkcionalnosti (kreiranje, čitanje, ažuriranje i brisanje). Administrator može dodavati nove proizvode, menjati postojeće i upravljati stanjem zaliha.
* **Order Service (Flask + SQLite + Redis):** Ovaj servis omogućava korisnicima kreiranje porudžbina. Pre svake porudžbine proverava se stanje zaliha proizvoda i ukoliko su svi uslovi ispunjeni smanjuju se zalihe. Ovaj servis takođe komunicira sa Redis brokerom i šalje događaje o kreiranim porudžbinama čime se obezbeđuje mogućnost da se u budućnosti sistem proširi, recimo slanje notifikacija.
* **Frontend aplikacija (React):** Klijentska aplikacija je razvijena u React-u. Korisnicima je omogućeno da se registruju, prijave pregledaju listu proizvoda i kreiraju porudžbine. Sa druge strane administrator ima poseban interfejs preko kog može da upravlja proizvodima i porudžbinama korisnika. Može da kreira nove proizvode, može da menja zalihe o postojećim proizvodima, da ih briše ako želi, i ima pregled svih porudžbina koje su korisnici napravili koje takođe može da briše.

Ovakav pristup obezbeđuje da sistem može da funkcioniše i pri većem broju korisnika, da ostane stabilan i pouzdan čak i u slučaju kvara jednog od servisa. Tehnologije koje sam koristio u razvoju ovog projekta su: Flask, React, Redis i Docker. Odabrao sam ovaj set tehnologija zato što obezveđuju visok nivo fleksibilnosti i jednostavnu povezanost, dok SQLite baze podataka omogućavaju pregledno i trajno čuvanje podataka u okviru Docker kontejnera.

# Primer/studija slučaja primene izabrane teme

**Kratak opis razvijenog sistema**

Razvijeni sistem je web aplikacija za elektronsku trgovinu koja korisnicima pruža mogućnost da naprave nalog, uloguju se, pregledaju postojeće proizvode i kreiraju svoju porudžbinu, dok administrator ima ulogu da upravlja ponudom proizvoda i prati aktivnosti korisnika. Ceo sistem je zamišljen tako da simulira osnovne funkcionalnosti pravih e-commerce platformi ali na jednostavniji način, pojednostavljen je i pregledan i sve to uz korišćenje mikroservisne arhitekture.

Iz ugla običnog korisnika rad sa aplikacijom izgleda jako intuitivno. Korisnik prvo kreira svoj nalog ukoliko ga nema a zatim je potrebno da se uloguje na sistem. Nakon toga može da pregleda listu proizvoda koje je administrator prethodno dodao, da vidi njihove cene i količine na stanju. Ako mu se neki proizvod dopada može da ga izabere i da kreira porudžbinu. Sistem odmah proverava da li proizvoda ima dovoljno na stanju i ako je uslov ispunjen porudžbina se čuva a stanje proizvoda se automatski smanjuje. Korisnik može i da pregleda listu svojih ranijih porudžbina i time dobija osećaj personalizovanog iskustva.

Iz ugla administratora rad sa aplikacijom izgleda takođe vrlo jednostavno i intuitivno. Pre svega neophodno je da se administrator uloguje. Zatim mu sistem nudi kontrolni panel u kome je moguće dodavati nove proizvode, ažurirati postojeće ili ih brisati po potrebi. Pored toga administrator ima uvid u porudžbine svih korisnika i može ih po potrebi brisati, što pokazuje mogućnost centralizovanog upravljanja. Na ovaj način administrator praktično simulira ulogu prodavca koji upravlja sopstvenom online prodavnicom.

Ono što čini ovaj sistem interesantnim je to na koji način sve funkcioniše u pozadini. Kada se napravi nova porudžbina, backend servis automatski komunicira sa bazom podataka i ažurira stanje proizvoda. Paralelno sa tim događaj o porudžbini objavljuje se preko Redis brokera što nam omogućava da u budućnosti jednostavno dodajemo nove funkcionalnosti (slanje notifikacija, integracija sa nekim platnim sistemom...). Zahvaljujući mikroservisnom pristupu svaki deo sistema ima svoju jasnu ulogu i radi potpuno nezavisno ali svi zajedno čine jednu celinu.

**Funkcionalnosti sistema uključuju:**

1. Registraciju i autentifikaciju korisnika
   1. Sistem omogućava kreiranje novih korisničkih naloga sa osnovnim podacima (ime, email, lozinka)
   2. Nakon registracije korisnici se mogu prijaviti na sistem i dobiti svoj token za pristup, dok je administratorski nalog unapred definisan i služi za upravljanje celim sistemom
2. Upravljanje proizvodima putem administratorskog panela
   1. Administrator ima mogućnost da dodaje nove proizvode, menja postojeće podatke (naziv, cenu, valutu i stanje zaliha) kao i da briše proizvode iz sistema
   2. Svi proizvodi se trajno čuvaju u bazi podatala i dostupni su korisnicima odmah nakon kreiranja
3. Pregled i naručivanje proizvoda od strane korisnika
   1. Korisnici koji su prijavljeni na sistem mogu da vide listu svih dostupnih proizvoda sa trenutnim stanjem zaliha
   2. Kada žele da naorave porudžbinu biraju proizvod i količinu. Sistem proverava da li postoji dovoljno proizvoda na stanju, kreira porudžbinu i automatski ažurira preostale zalihe proizvoda
4. Praćenje porudžbina
   1. Svaki korisnik ima uvid u listu svojih ranije kreiranih porudžbina sa osnovim informacijama o proizvodima, kolilini i statusu porudžbine
   2. Administrator može da pregleda porudžbine svih korisnika i po potrebi ih briše čime simulira proces centralitovanog upravljanja e-commerce platformom
5. Centralizovana komunikacija putem API Gatewaya
   1. Ceo saobraćaj između forntend aplikacije i backend servisa prolazi kroz AOI Gateway
   2. Gateway je zadužen za prosleđivanje zahteva odgovarajućem servisu, kontrolu autentifikacije i centralizovano logovanje aktivnosti
6. Skladištenje podataka i događaja
   1. Podaci o korisnicima, proizvodima i porudžbinama čuvaju se u SQLite bazama, po principu da svaki servis ima svoju bazu
   2. Order Service pored rada sa bazom šalje i događaje ka Redis brokeru što omogućava proširenje sistema

**Implementacija sistema**

A diagram of a computer application

AI-generated content may be incorrect.

**Backend (Flask mikroservisi)**

Backend sistema razvijen je u Flask-u i organizovan u obliku više manjih servisa umesto jedne monolitne aplikacije. Svaki servis ima svoju jasnu ulogu i sopstvenu bazu podataka, a komunikacija se odvija preko REST API-ja.

* API Gateway je ulazna tačka sistema. Prima sve zahteve sa fronta i proverava autentifikaciju, prosleđuje ih ka odgovarajućim servisima. Na taj način frontend komunicira samo sa jednim mestom, dok Gateway u pozadini raspoređuje poslove.
* User Service se bavi registracijom i loginom korisnika. U bazi čuva podatke o nalozima i razlikuje korisnike od administratora
* Product Service omogućava administratoru da kreira, menja i briše proizvode, dok korisnici mogu samo da ih pregledaju. Podaci o proizvodima i zalihama se čuvaju u SQLite bazi
* Order Service je zadužen za kreiranje porudžbina i smanjenje stanja zaliha proizvoda. Pored baze koristi i Redis broker za slanje događaja što omogućava lako proširenje sistema u buduće.

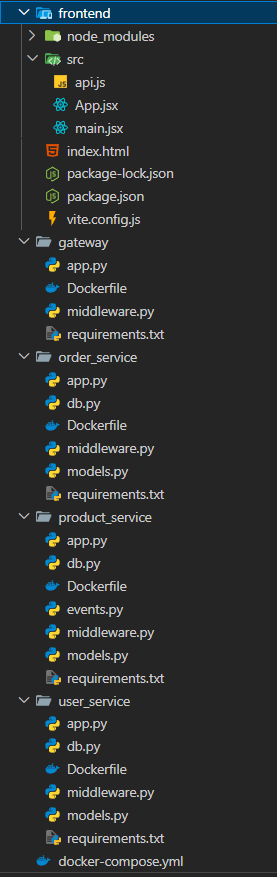
**Baza podataka (SQLite)**

Za čuvanje podataka se koristi SQLite. Svaki servis ima svoju zasebnu bazu (app.db) što znači da su podaci nezavisni i da se ne deli jedna zajednička baza.

* User Service čuva korisničke podatke
* Product Service čuva proizvode i zalihe
* Order Service čuva porudžbine

Na ovaj način se poštuje princip mikroservisa da svaki servis upravlja svojim podacima i ne zavisi direktno od drugih.

**Struktura aplikacije:**



**REST API rute:**

**Auth Controller**

* **[POST] /auth/register**  
  Kreiranje novog korisnika (role = user)
* **[POST] /auth/login**  
  Prijava korisnika, vraća token i podatke o korisniku

**Products Controller**

* **[GET] /products**Vraćanje liste proizvoda
* **[GET] /products/{id}**Vraćanje detalja jednog proizvoda
* **(ADMIN) [POST] /products**Kreiranje novog proizvoda (name, price, currency, stock)
* **(ADMIN) [PUT] /products/{id}**Ažuriranje postojećeg proizvoda
* **(ADMIN) [DELETE] /products/{id}**Brisanje proizvoda

**Orders Controller**

* **[GET] /orders**

Ako je user vraća samo njegove porudžbine

Ako je admin podržan je query parametar user\_id za filtriranje po korisniku

* **[POST] /orders**Kreiranje porudžbinu, user\_id se uzima iz tokena, proverava se stock, vraćanje greške ako nema dovoljno
* **(ADMIN) [DELETE] /orders/{id}**Brisanje porudžbine

**Users Controller (admin panel)**

* **(ADMIN) [GET] /users**Vraćanje liste korisnika

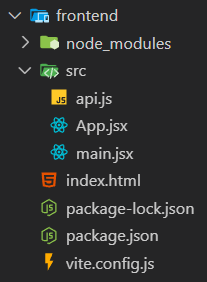
**Klijentska aplikacija (React)**

Klijentska aplikacija je napravljena u React-u i omogućava dva tipa korisnika:

* Korisnik (user): registracija, login, pregled proizvoda i kreiranje porudžbina, uvid u listu svojih porudžbina
* Administrator: login bez registracije, kreiranje, izmena i brisanje proizvoda, pregled i brisanje porudžbina bilo kog korisnika

Aplikacija komunicira preko API Gateway-a (<http://localhost:8000/api>), koristeći REST pozive i Bearer token nakon logina.

Struktura aplikacije:



**Rute i ekrani**

* /login – za prijavu korisnika, nakon logovanja čuva se korisnik i token u localStorage a UI preusmerava:
  + Admin –> /admin
  + User –> /user
* /register – registracija korisnika
* /admin – admin panel
  + Forma za kreiranje proizvoda
  + Tabela proizvoda sa edit i delete
  + Dropdown korisnika i tabela sa porudžbinama, delete order dugme
* /user – korisnicki panel
  + Lista proizvoda
  + Forma za kreiranje porudžbine
  + Tabela sa korisnikovim porudžbinama

**Login strana:**

**A screenshot of a login form

AI-generated content may be incorrect.**

**Register strana:**

**A screenshot of a login form

AI-generated content may be incorrect.**

**User strana:**

**A screenshot of a user dashboard

AI-generated content may be incorrect.**

**Admin strana:**

**Screens screenshot of a computer screen

AI-generated content may be incorrect.**

**Logika ekrana:**

Admin strana:

* Učitavanje proizvoda i korisnika (GET /products, GET /users)
* Kreira proizvod (POST /products)
* Uređuje i briše proizvod (PUT i DELETE /products/{id})
* Bira korisnika iz dropdown menija i učitava njegove porudžbine (GET /orders?user\_id=...)
* Briše porudžbinu (DELETE /orders/{id})

User strana:

* Učitava listu proizvoda (GET /products)
* Pravi porudžbinu (POST /orders), a onda se:
  + Refrešuje lista porudžbina (GET /orders – gateway vraća samo njegove)
  + Osvežava lista proizvoda (zalihe se smanjuju)

Navbar i stanje sesije:

* Ako niko nije ulogovan prikazuje se Login i Register
* Ako je ulogovan: prikazuje Hello {admin/ime korisnika} i Logout
* Posle refreša stranice čita se currentUser i authToken iz localStorage da bi ostao ulogovan i ostao na istoj ruti

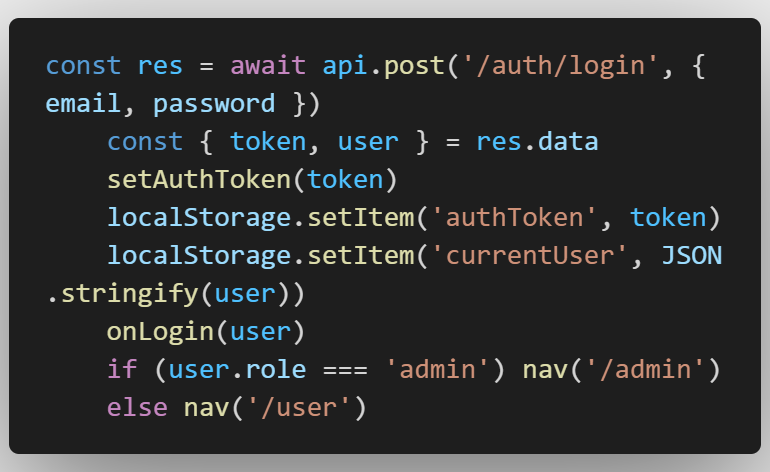
**Komunikacija sa backendom**

1. Konfiguracija API klijenta (src/api.js)

A screen shot of a computer screen

AI-generated content may be incorrect.

1. Login i čuvanje sesije (iz App.jsx)



# Zaključak

Razvoj distribuiranog sistema za upravljanje e-commerce procesima pokazuje kako se uz upotrebu modernih tehnologija može postići modularno i jasno razdvojeno rešenje. Podelom na više servisa – korisnički servis, servis za proizvode, servis za porudžbine i API gateway – obezbeđena je fleksibilnost, skalabilnost i lakše održavanje. Gateway centralizuje autentikaciju i autorizaciju, dok svaki mikroservis ima jasno definisanu ulogu, što doprinosi boljoj organizaciji i preglednosti sistema.

Frontend aplikacija razvijena u React-u omogućava jednostavnu interakciju korisnika i administratora sa sistemom, dok SQLite baza obezbeđuje pouzdano skladištenje podataka. Ugrađena je i osnovna bezbednosna logika kroz role-based pristup, pa su korisnicima dostupne samo njihove porudžbine, dok administratori imaju mogućnost pregleda i upravljanja svim podacima.

Implementacija ovog projekta pokazuje praktičnu primenu distribuiranih sistema u domenu elektronske trgovine i daje dobar osnov za dalje unapređenje – kao što su dodavanje plaćanja, naprednije analitike ili proširenje na veće baze podataka i cloud okruženja. Sistem je jednostavan za upotrebu, pregledan i dovoljno fleksibilan da može da raste zajedno sa potrebama korisnika.

# Literatura

<https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices>

<https://flask.palletsprojects.com/en/latest/>

<https://www.sqlite.org/docs.html>

<https://docs.docker.com/get-started/>

<https://microservices.io/patterns/apigateway.html>

<https://redis.io/docs/latest/develop/pubsub/>

Nastavni materijal iz predmeta CS230 – Distribuirani sistemi