

Sveučilište u Zagrebu
Prirodoslovno-matematički fakultet
Matematički odsjek

Tena Albina Kao, Petar Pavlović, Magdalena Potočnjak i Jurica Preksavec

SUSTAV ZA UPRAVLJANJE TRGOVINAMA INFORMATIČKE OPREME

Seminarski rad

Mentor:
dr. sc. Ognjen Orel

Zagreb, 2024.

Sadržaj

1.	Uvod	1
2.	Baza podataka sustava	2
2.1.	Relacijska baza sustava	2
2.2.	Dokument baza sustava	4
3.	Backend aplikacije	6
4.	Frontend aplikacije	9
4.1.	Opis poslova	9
4.2.	Poteškoće	13
5.	Zaključak	15
6.	Samoevaluacije	16
6.1.	Tena Albina Kao	16
6.2.	Petar Pavlović	16
6.3.	Magdalena Potočnjak	16
6.4.	Jurica Preksavec	17
7.	Literatura	18

1. Uvod

Svaka tvrtka da bi dobro poslovala mora imati uvid u svoje poslovanje i trenutno stanje na tržištu. U trgovačkim djelatnostima važno je održavati proizvode dostupnima svojim potrošačima, odnosno upravljati zalihama na način da potrebe kupaca budu zadovoljene. Obično trgovine imaju više poslovnica na raznim lokacijama pa je potrebno i distribuirati proizvode u pravilnom omjeru sukladno potražnji.

U ovom projektu želimo izraditi sustav koji omogućava jednom takvom lancu trgovina da vodi evidenciju u svim potrebnim aspektima prodaje. Konkretno, za primjer smo uzeli trgovinu informatičkom opremom s više poslovnica i popratnim asortimanom laptopa.

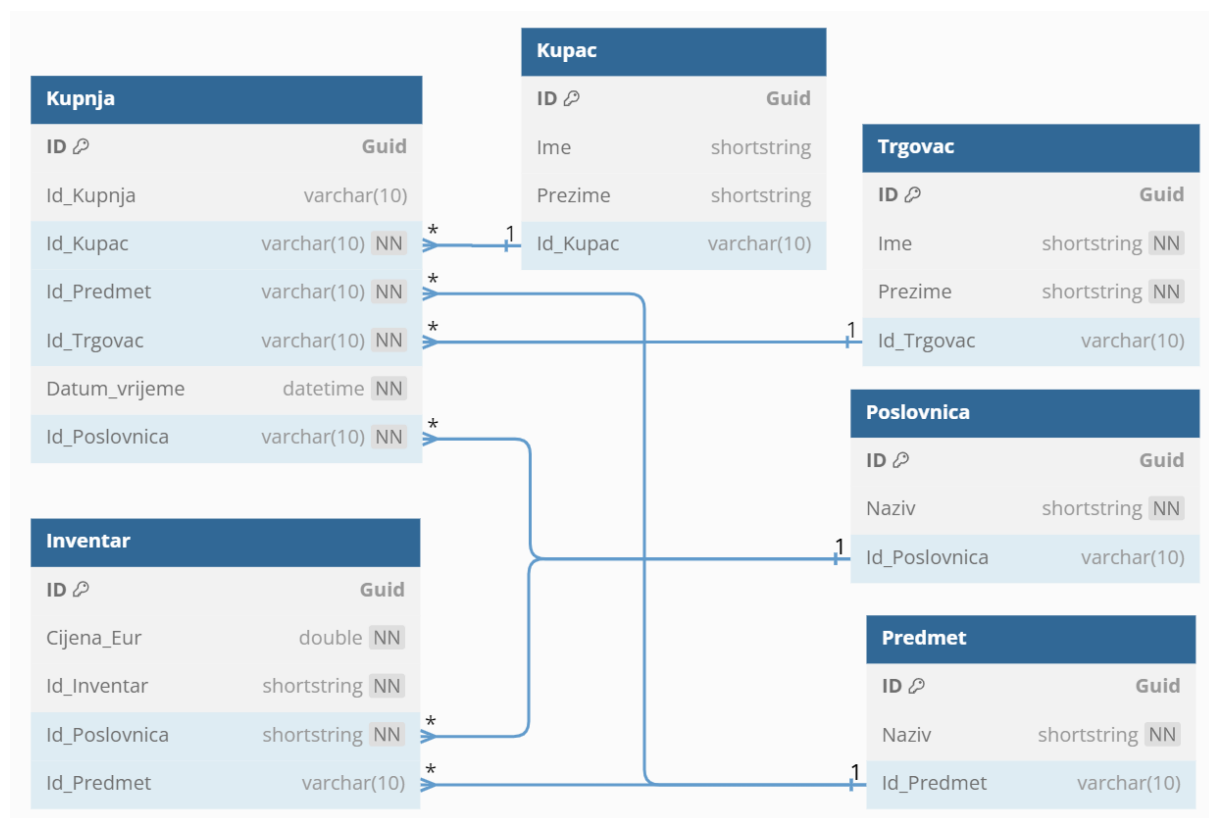
Sustav se sastoji od relacijske baze podataka, dokument baze podataka i frontend sučelja koje sve povezuje i čini sustav lako upotrebljivim korisnicima. Relacijska baza podataka sadrži podatke o poslovnicama, trgovcima, predmetima, kupcima, a dokument baza podataka sadrži opise proizvoda. Cilj nam je prepoznati i omogućiti funkcionalnosti za razne korisnike.

2. Baza podataka sustava

Trgovina za uspješno upravljanje zalihama mora imati uvid u kompletan asortiman koji nudi na tržištu, ukupnu količinu pojedinog proizvoda kojom raspolaže te dostupnost tih proizvoda u svojim poslovnica. Kako bismo pravovremeno reagirali na potrebe kupaca i dostavili proizvode u poslovnice na vrijeme, bitno je i predvidjeti buduću potražnju. To možemo na temelju podataka o prošlim kupnjama u našim trgovinama. Stoga, želimo imati uvid u ostvarene kupnje kao i informacije o svojim kupcima.

Ovdje se prirodno nameće struktura relacijske baze podataka u kojoj ćemo pamti naš inventar i njegovu raspodjelu po poslovnica te kupnje i pripadne podatke koji su nam važni za nju. Kako bismo lakše predvidjeli potrebe kupaca, zanimaju nas i informacije o kupljenim proizvodima što ćemo pamti u dokument bazi koja je pogodna za pohranu veće količine podataka po elementu tablice.

2.1. Relacijska baza sustava



Slika 1: Shema relacijske baze sustava

Tip Guid je zadan od strane SQLa, a NN označava *not null* odnosno obavezno ispunjeno polje. Sve veze su 1:više.

Potrebni su nam sljedeći entiteti:

- Kupac – pohranjujemo ime i prezime osobe koja je nekad kupila naš proizvod
- Trgovac – pohranjujemo ime i prezime osobe zaposlene u našoj trgovini
- Poslovnica – pohranjujemo naziv poslovnice koji sadrži njegovu lokaciju
- Predmet – pohranjujemo naziv proizvoda koji će u dokument bazi biti model
- Inventar – pohranjujemo asortiman trgovine uz cijenu i lokaciju poslovnice u kojoj se nalazi (u raznim poslovnicama moguća je različita cijena)
- Kupnja – pohranjujemo tko je kupio, što je kupio, kod koga je kupio, kada je kupio i u kojoj poslovnici (jedna kupovina istog kupca može sadržavati više proizvoda te će tada postojati više redaka u tablici)

Za unos podataka u bazu smo ručno napisali skriptu u kojoj su pozivane pripadne SQL naredbe. Podaci o osobama i lokaciji poslovnica su izmišljeni, a podaci o proizvodima su rezultat pretrage tržišta laptopa.

Primjer dijela unosa podataka u relacijsku bazu možemo vidjeti ovdje:

```

14 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Trgovac] (ID,Ime,Prezime,Id_Trgovac) VALUES (NEWID(), 'Ana', 'Anić', 'Trg_001')
15 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Trgovac] (ID,Ime,Prezime,Id_Trgovac) VALUES (NEWID(), 'Ena', 'Enić', 'Trg_002')
16 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Trgovac] (ID,Ime,Prezime,Id_Trgovac) VALUES (NEWID(), 'Ivo', 'Ivić', 'Trg_003')
17 SELECT * FROM [NBP_Project_Store].[NBP_project_Store].[Trgovac]
18
19 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Kupac] (ID,Ime,Prezime,Id_Kupac) VALUES (NEWID(), 'Tena Alhina', 'Kao', 'Kupac_001')
20 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Kupac] (ID,Ime,Prezime,Id_Kupac) VALUES (NEWID(), 'Magdalena', 'Potočnjak', 'Kupac_002')
21 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Kupac] (ID,Ime,Prezime,Id_Kupac) VALUES (NEWID(), 'Petar', 'Potočnjak', 'Kupac_003')
22 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Kupac] (ID,Ime,Prezime,Id_Kupac) VALUES (NEWID(), 'Jurica', 'Pekavac', 'Kupac_004')
23 SELECT * FROM [NBP_Project_Store].[NBP_project_Store].[Kupac]
24
25 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Predmet] (ID,Naziv,Id_Predmet) VALUES (NEWID(), 'Acer Aspire 5 A515-55-50W', 'Lap_001')
26 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Predmet] (ID,Naziv,Id_Predmet) VALUES (NEWID(), 'Dell XPS 13 9310', 'Lap_002')
27 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Predmet] (ID,Naziv,Id_Predmet) VALUES (NEWID(), 'HP Envy x360 15-eu0023dx', 'Lap_003')
28 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Predmet] (ID,Naziv,Id_Predmet) VALUES (NEWID(), 'Lenovo ThinkPad X1 Carbon Gen 9', 'Lap_004')
29 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Predmet] (ID,Naziv,Id_Predmet) VALUES (NEWID(), 'Asus ROG Zephyrus G14 GA502R', 'Lap_005')
30 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Predmet] (ID,Naziv,Id_Predmet) VALUES (NEWID(), 'Microsoft Surface Laptop 5', 'Lap_006')
31 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Predmet] (ID,Naziv,Id_Predmet) VALUES (NEWID(), 'Apple MacBook Pro 14-inch M1 Pro', 'Lap_007')
32 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Predmet] (ID,Naziv,Id_Predmet) VALUES (NEWID(), 'Asus Zenbook Duo 14 UX802', 'Lap_008')
33 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Predmet] (ID,Naziv,Id_Predmet) VALUES (NEWID(), 'Razer Blade 15 Advanced Model', 'Lap_009')
34 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Predmet] (ID,Naziv,Id_Predmet) VALUES (NEWID(), 'HP Spectre x360 14-sa1013dx', 'Lap_010')
35 SELECT * FROM [NBP_Project_Store].[NBP_project_Store].[Predmet]
36
37 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '699,00', 'Inv_001', 'Pos_001', 'Lap_001')
38 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '699,00', 'Inv_002', 'Pos_002', 'Lap_001')
39 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '1599,00', 'Inv_003', 'Pos_001', 'Lap_002')
40 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '1599,00', 'Inv_004', 'Pos_002', 'Lap_002')
41 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '1049,00', 'Inv_005', 'Pos_001', 'Lap_003')
42 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '1049,00', 'Inv_006', 'Pos_002', 'Lap_003')
43 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '1079,00', 'Inv_007', 'Pos_001', 'Lap_004')
44 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '1079,00', 'Inv_008', 'Pos_002', 'Lap_004')
45 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '2299,00', 'Inv_009', 'Pos_001', 'Lap_005')
46 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '2299,00', 'Inv_010', 'Pos_002', 'Lap_005')
47 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '2299,00', 'Inv_011', 'Pos_001', 'Lap_006')
48 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '2299,00', 'Inv_012', 'Pos_002', 'Lap_006')
49 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '2299,00', 'Inv_013', 'Pos_001', 'Lap_007')
50 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '1499,00', 'Inv_014', 'Pos_002', 'Lap_008')
51 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '2899,00', 'Inv_015', 'Pos_001', 'Lap_009')
52 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Inventar] (ID,Cijena_Eur,Id_Inventar,Id_Poslovnica,Id_Predmet) VALUES (NEWID(), '1549,00', 'Inv_016', 'Pos_002', 'Lap_010')
53 SELECT * FROM [NBP_Project_Store].[NBP_project_Store].[Inventar]
54
55 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Kupnja] (ID,Id_Kupnja,Id_Kupac,Id_Predmet,Id_Trgovac,Datum_vrijeme,Id_Poslovnica) VALUES (NEWID(), 'Kupnja_001', 'Kupac_001', 'Lap_001', 'Trg_001', GETDATE(), 'Pos_001')
56 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Kupnja] (ID,Id_Kupnja,Id_Kupac,Id_Predmet,Id_Trgovac,Datum_vrijeme,Id_Poslovnica) VALUES (NEWID(), 'Kupnja_002', 'Kupac_002', 'Lap_001', 'Trg_001', GETDATE(), 'Pos_001')
57 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Kupnja] (ID,Id_Kupnja,Id_Kupac,Id_Predmet,Id_Trgovac,Datum_vrijeme,Id_Poslovnica) VALUES (NEWID(), 'Kupnja_003', 'Kupac_002', 'Lap_002', 'Trg_002', GETDATE(), 'Pos_001')
58 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Kupnja] (ID,Id_Kupnja,Id_Kupac,Id_Predmet,Id_Trgovac,Datum_vrijeme,Id_Poslovnica) VALUES (NEWID(), 'Kupnja_004', 'Kupac_003', 'Lap_003', 'Trg_002', GETDATE(), 'Pos_001')
59 INSERT INTO [NBP_Project_Store].[NBP_project_Store].[Kupnja] (ID,Id_Kupnja,Id_Kupac,Id_Predmet,Id_Trgovac,Datum_vrijeme,Id_Poslovnica) VALUES (NEWID(), 'Kupnja_005', 'Kupac_004', 'Lap_004', 'Trg_003', GETDATE(), 'Pos_002')
60 SELECT * FROM [NBP_Project_Store].[NBP_project_Store].[Kupnja]
61

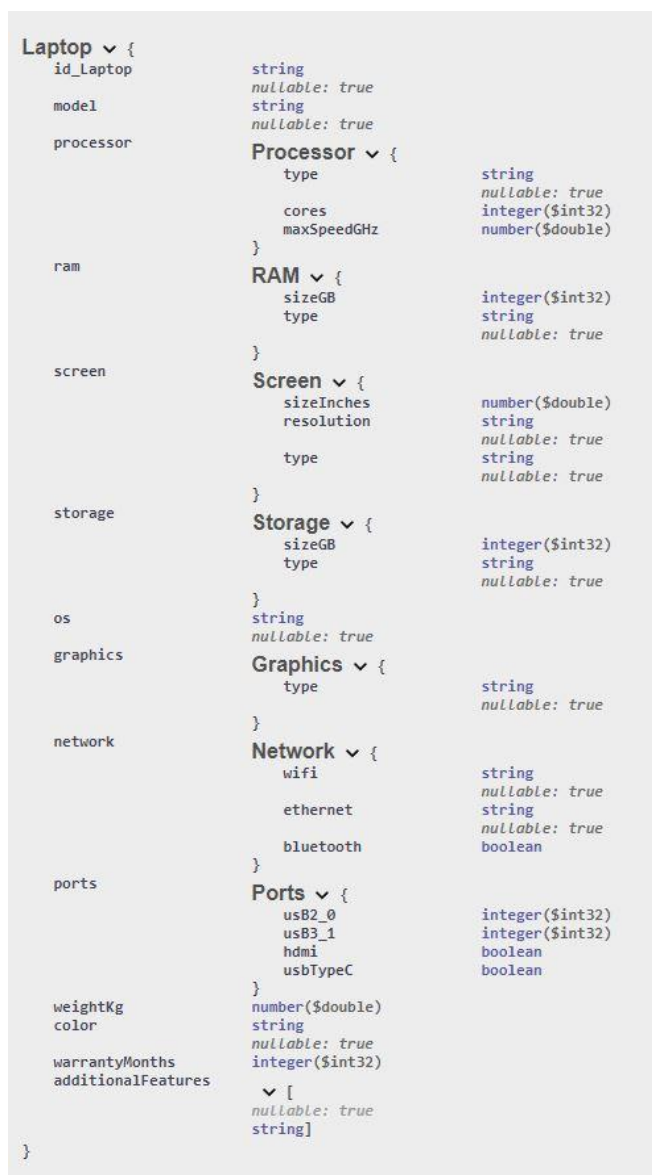
```

Slika 2: Skripta za unos podataka u relacijsku bazu sustava

2.2. Dokument baza sustava

Za lakšu pohranu podataka o raspoloživom asortimanu i njegovim detaljima potrebnima prodajnom odjelu, koristit ćemo dokument bazu. U njoj ćemo pohraniti sve naše proizvode i njihove pripadne karakteristike. Također, bit će povezana i s prethodno opisanom relacijskom bazom na način da *Id_Predmet* u tablici Predmet iz relacijske baze odgovara polju *<Id_proizvod>* u dokument bazi.

Konkretno, za naš primjer trgovine informatičkom opremom uzeli smo laptose za proizvode pa tako svaki redak iz relacijske tablice Predmet odgovara točno jednom dokumentu u dokument bazi te je *Id_Predmet* iz tog retka jednak polju *Id_Laptop* u pripadnom dokumentu.



Slika 3: Shema dokument baze sustava

Kao što je već navedeno, podaci o laptopima su preuzeti s jedne internet trgovine te su temeljem pronađenih podataka napravljeni atributi unutar dokumenta čije tipove možemo vidjeti na Slici 3.

Nadalje, na sljedećim slikama možemo vidjeti primjere dokumenata iz dokument baze koji su također uneseni pomoću skripte i uvezeni u MongoDB.

	ObjectID	ObjectID	ObjectID
1	_id: ObjectId('6668e85b05e9c66f51d0d59e')	1	_id: ObjectId('6668e85b05e9c66f51d0d5a4')
2	Id_Laptop: "Lap_001"	2	Id_Laptop: "Lap_007"
3	model: "Acer Aspire 5 A515-55-56VKG"	3	model: "Apple MacBook Pro 14-inch M1 Pro"
4	processor: Object	4	processor: Object
5	type: "Intel Core i5-1035G1"	5	type: "Apple M1 Pro"
6	cores: 4	6	cores: 10
7	max_speed_GHz: 3.6	7	max_speed_GHz: 3.2
8	ram: Object	8	ram: Object
9	size_GB: 8	9	size_GB: 16
10	type: "DDR4"	10	type: "Unified"
11	screen: Object	11	screen: Object
12	size_inches: 15.6	12	size_inches: 14.2
13	resolution: "1920x1080"	13	resolution: "3024x1964"
14	type: "FHD"	14	type: "Liquid Retina XDR"
15	storage: Object	15	storage: Object
16	size_GB: 256	16	size_GB: 512
17	type: "SSD"	17	type: "SSD"
18	os: "Windows 10 Home"	18	os: "macOS Monterey"
19	graphics: Object	19	graphics: Object
20	type: "Intel UHD Graphics"	20	type: "Apple Integrated"
21	network: Object	21	network: Object
22	wifi: "802.11ac"	22	wifi: "802.11ax"
23	ethernet: "None"	23	ethernet: "None"
24	bluetooth: true	24	bluetooth: true
25	ports: Object	25	ports: Object
26	usb_2_0: 1	26	usb_2_0: 0
27	usb_3_1: 2	27	usb_3_1: 0
28	hdmi: true	28	hdmi: true
29	usb_type_c: true	29	usb_type_c: true
30	weight_kg: 1.8	30	weight_kg: 1.6
31	color: "Silver"	31	color: "Space Gray"
32	warranty_months: 12	32	warranty_months: 12
33	additional_features: Array (2)	33	additional_features: Array (2)
34	0: "Backlit Keyboard"	34	0: "Touch ID"
35	1: "Fingerprint Reader"	35	1: "ProMotion"

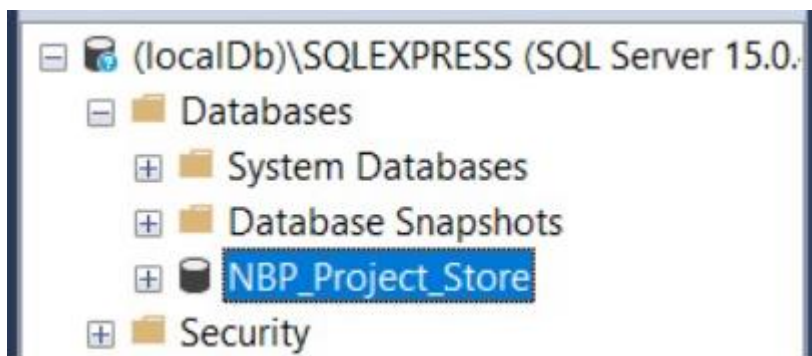
Slika 4a: Dokument iz baze za laptop tvrtke Acer

Slika 4b: Dokument iz baze za laptop tvrtke Apple

Primjetimo da je Slika 4a vezana za liniju 25 na Slici 2, a Slika 4b ima poveznicu s linijom 31.

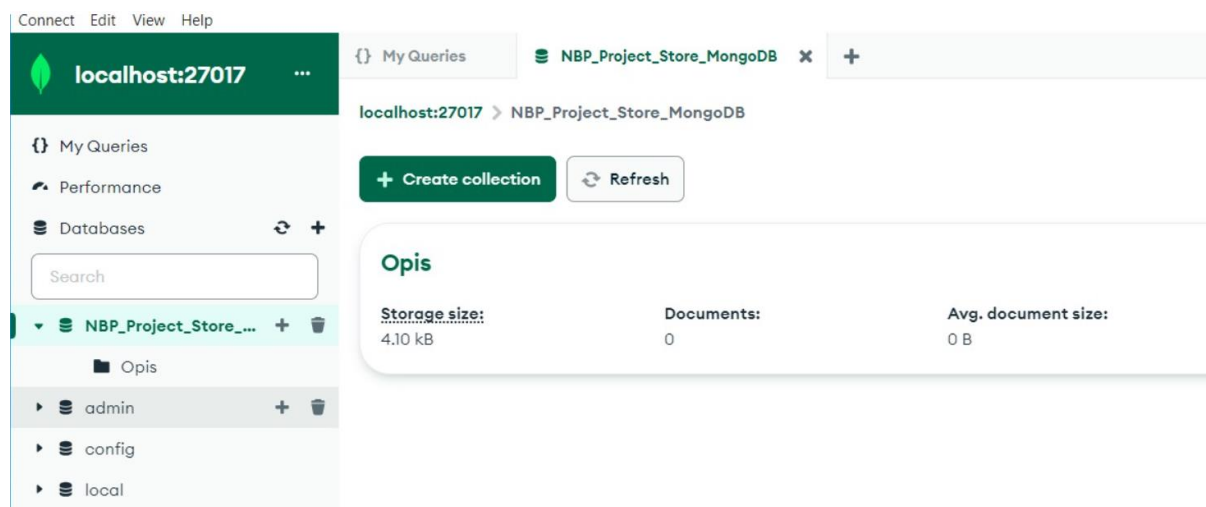
3. Backend aplikacije

Za relacijsku bazu podataka korišten je Microsoft SQL Server Management Studio u kojem je kreirana baza opisana u prethodnom poglavlju naziva „NBP_Project_Store“.



Slika 5: Relacijska baza sustava u SSMS-u

MongoDB Compass nam je poslužio za izradu dokument baze podataka pod nazivom „NBP_Project_Store_MongoDB“ u kojoj je kolekcija „Opis“.



Slika 6: Dokumentaska baza sustava u MongoDB Compass-u

Backend aplikacije je pisan u jeziku C# pa je stoga u Microsoft Visual Studiu kreiran novi C# projekt. Slijedile su osnovne radnje povezivanja s bazama i instalacija potrebnih NuGet paketa. Korišten je Rhotos framework za kreiranje skripte *NBP_Project_Store.rhe* u kojoj su definirani entiteti SQL baze. Nakon toga su definirani objekti i izrađeni kontroleri za API.

Za entitete se prilikom stvaranja instanci generira Guid. Obzirom da je riječ o 14-znamenkastom heksadecimalnom ključu, kreirano je dodatno polje za svaki entitet kojeg bismo normalno koristili za primarni ključ (Id_Kupac, Id_Trgovac itd.). Time je olakšano međusobno povezivanje i čitljivost te pojednostavljeno korištenje unutar baze, a korisnicima prepušteno proizvoljno korištenje stringova za ključ. Sva pretraživanja, kreiranja i brisanja su trenutno postavljena tako da koriste naše pojednostavljene id-eve. No, treba biti pažljiv pri odabiru vrijednosti ključeva za svaku od instanci kreiranih entiteta. Koristili smo i predlažemo „standard“ na način da prvih par slova opisuju entitet, zatim ide znak _ pa onda troznamenkasti redni broj retka u tablici. Primjerice, Pos_008 (za Poslovnici) ili Kupac_004.

To je napravljeno ponajprije zbog testiranja API-ja, jer je izrada frontenda tek slijedila, a htjeli smo testirati backend i uvjeriti se da sve radi ispravno. Generiranje i manipulacije s Guid-ima su nam uzimale dosta vremena u tom procesu pa je to bio glavni razlog ovakvog izbora rješenja. Kvalitetnim frontendom bi se ova prepreka riješila i sve manipulacije bi se tada u backendu prebacile na Guid-e, te time osigurala željena konzistentnost.

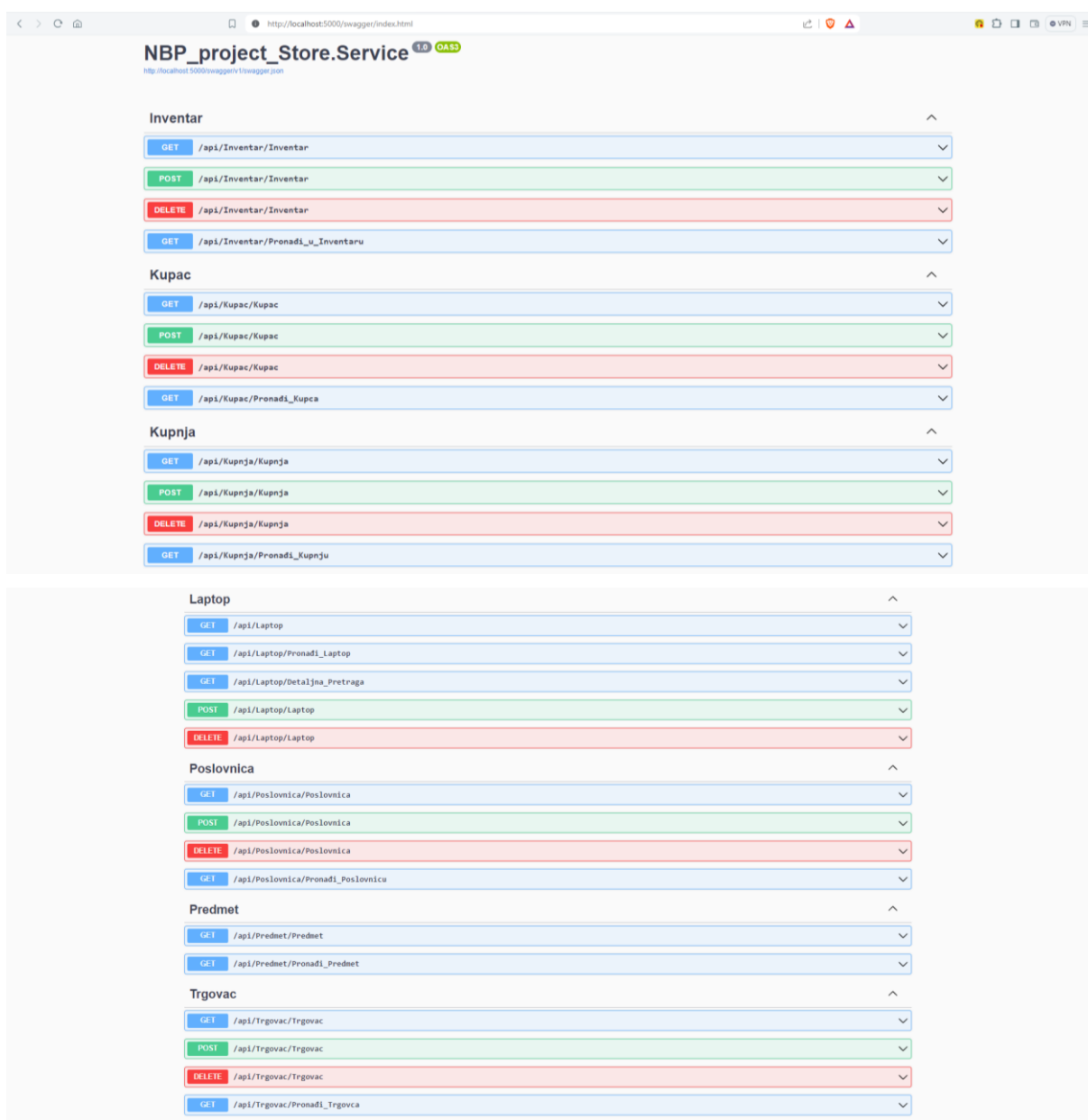
Određenu prepreku i dosta potrošenog vremena prouzročili su pokušaji u kojima smo htjeli (i na kraju uspjeli u tome) da se kreiranjem instance „Laptopa“ istovremeno popune i SQL i MongoDB baze. Cilj je bio da se kreiranjem laptopa u Mongo bazi informacija o vrijednosti ključa i nazivu laptopa prepiše u SQL bazu u entitet „Predmet“. U kontroler „Laptopa“ je iz tog razloga ugrađena povratna informacija o uspješnosti tog postupka. Bilo bi dobro doraditi taj dio koda na način da se u slučaju negativne informacije izbriše laptop iz Mongo baze ili da se automatizira višekratni pokušaj upisa u SQL bazu, odnosno neka smisljena kombinacija toga. Zasad je taj dio ostao samo na razini informacije o uspjehu odnosno neuspjehu postupka.

Ishod je da se sada u „Predmetu“ u SQL-u nalaze ključ (Id_Predmet) i Naziv (u samom nazivu je zapravo kratki opis laptopa) što pruža mogućnost stvaranja relacija u kojima brzo možemo pronaći tko je gdje i kada prodao laptop kome, a zatim sve detalje tog laptopa možemo potražiti u dokument bazi.

Dodane su i kontrole za detaljnu pretragu po entitetima. Rade tako da pronalaze i zapise koji su „slični“ onome što tražimo, a ne samo strogo precizne upite, tj. ako tražimo tko je sve kupio laptop marke Acer, možemo u pretragu napisati i „ace“ – dakle detaljne pretrage „vide“ podstringove, dok je istovremeno dostupan i „obični GET“ koji „vidi“ samo strogo precizne upite poput točno napisanog ključa.

Po istom principu se može za Kupca, Trgovca, Poslovnicu ili bilo koji entitet SQL baze napraviti kolekcija u dokument bazi, ali to nismo radili jer smo htjeli ovim projektom stvoriti nekakav „template“ koji će nam svima u budućnosti biti osnova za izradu aplikacija koje koriste grafičko sučelje da bi komunicirale s bazom (relacijskom i dokument) preko API-ja. Iz tog razloga smo željeli stvoriti što veću funkcionalnost a što manju robusnost, jer olakšava prijelaz iz „Trgovine laptopa“ u bilo što drugo te znatno olakšava i ubrzava izradu kompleksnih aplikacija.

Na kraju ove faze razvoja aplikacije moglo se pristupiti bazičnom frontend-u uz pomoć Swaggera koji izgleda ovako:



Slika 7: Prikaz frontenda Swagger

4. Frontend aplikacije

Na ovome projektu radili smo i na razvoju korisničkog sučelja (frontenda), što uključuje prikazivanje podataka iz baze, pripremu podataka za spremanje u bazu te komunikaciju sa serverom na kojem je pokrenuta C# backend aplikacija. Za razvoj smo koristili programski jezik Angular za koji je bilo potrebno instalirati Node.js jer je sadržavao sve potrebne pakete. Kao razvojno okruženje koristili smo Visual Studio Code. Server na kojem se vrtjela aplikacija pokretali smo kroz Command Prompt sljedećom naredbom: ``ng serve --open.``

4.1. Opis poslova

Svaka pojedina tablica u bazi ima svoju zasebnu stranicu u aplikaciji, a do pojedine stranice dolazimo preko navigacijske trake. Također, svaka pojedina stranica unutar aplikacije kreira se u Angularu kao komponenta sljedećom naredbom: ``ng g c <ime-stranice>``, pri čemu `<ime-stranice>` zamijenimo s nekim logičnim imenom koje bi odgovaralo korištenju komponente ili stranice. Primjerice, ako želimo prikazivati podatke iz tablice Laptop, ime komponente bi bilo *laptop-list*.

```
src > app > features > tables > services > TS laptop.service.ts > LaptopService
1  import { Injectable } from '@angular/core';
2  import { Observable } from 'rxjs';
3  import { HttpClient } from '@angular/common/http';
4  import { HttpParams } from '@angular/common/http';
5  import { GetLaptopRequest } from '../models/get-laptop-request.model';
6  import { GetOneLaptopRequest } from '../models/get-one-laptop-request.model';
7  import { GetLaptopDetailsRequest } from '../models/get-laptop-details-request.model';
8
9  @Injectable({
10    providedIn: 'root'
11  })
12  export class LaptopService {
13
14    constructor(private http: HttpClient) { }
15
16    addLaptop(model: GetLaptopRequest): Observable<GetLaptopRequest> {
17      const headers = {
18        'Content-Type': 'application/json'
19      };
20
21      return this.http.post<GetLaptopRequest>("https://localhost:44393/api/Laptop/Laptop", model);
22    }
23  }
```

Slika 8: Prikaz koda za LaptopService

Unutar projekta kreirali smo i *services* folder koji nam služi za spremanje klasa i unutar klasa funkcija za komunikaciju s backendom. Kako bismo kreirali jedan takav *service*, koristimo sljedeću naredbu: ``ng g s poslovnica``. Tipične funkcije za komunikaciju s backendom su GET, POST i DELETE, a one nam redom predstavljaju dohvat iz baze, spremanje u bazu i brisanje iz baze. Iako u našem projektu nismo morali za sve tablice raditi baš sve metode, za neke tablice u bazi morali smo napraviti više GET metoda za pristup backendu.

```

24 deleteLaptop(model: GetOneLaptopRequest): Observable<GetOneLaptopRequest> {
25   return this.http.delete<GetOneLaptopRequest>("https://localhost:44393/api/Laptop/Laptop?id_Laptop=" + model.id_Laptop);
26 }
27
28 getAllLaptops(): Observable<GetLaptopRequest[]> {
29   return this.http.get<GetLaptopRequest[]>("https://localhost:44393/api/Laptop");
30 }
31
32 getLaptop(model: GetOneLaptopRequest): Observable<GetLaptopRequest[]> {
33   const params = new HttpParams()
34     .set('id_laptop', model.id_Laptop);
35   return this.http.get<GetLaptopRequest[]>("https://localhost:44393/api/Laptop/Pronađi_Laptop", { params });
36 }
37
38 getLaptopDetails(model: GetLaptopDetailsRequest): Observable<GetLaptopRequest[]> {
39   const params = new HttpParams()
40     .set('keyword', model.keyword);
41   return this.http.get<GetLaptopRequest[]>("https://localhost:44393/api/Laptop/Detaljna_Pretraga", { params });
42 }
43 }
44

```

Slika 9: Prikaz funkcija za DELETE i GET metode

Funkcije u backendu se osim po operaciji na bazi razlikuju i po svojoj definiciji, odnosno ulaznim parametrima funkcije, kao i URL-u na kojem im pristupamo. Dakle, jednom kada pokrenemo server na kojem se vrti backend controller, on nam daje osnovni URL koji je u našem slučaju bio `https://localhost:44393/`.

Kako bismo pristupili svakoj od funkcija i komunicirali s bazom, morali smo dodati dio URL-a koji je specifičan za svaku od tih funkcija, primjerice za funkciju *Pronađi_u_Inventaru* pripadni URL glasi `/api/Inventar/Pronađi_u_Inventaru`. Ovaj dio URL-a dobili smo preko Swagger korisničkog sučelja koji se pokrene kada pokrenemo i backend server.

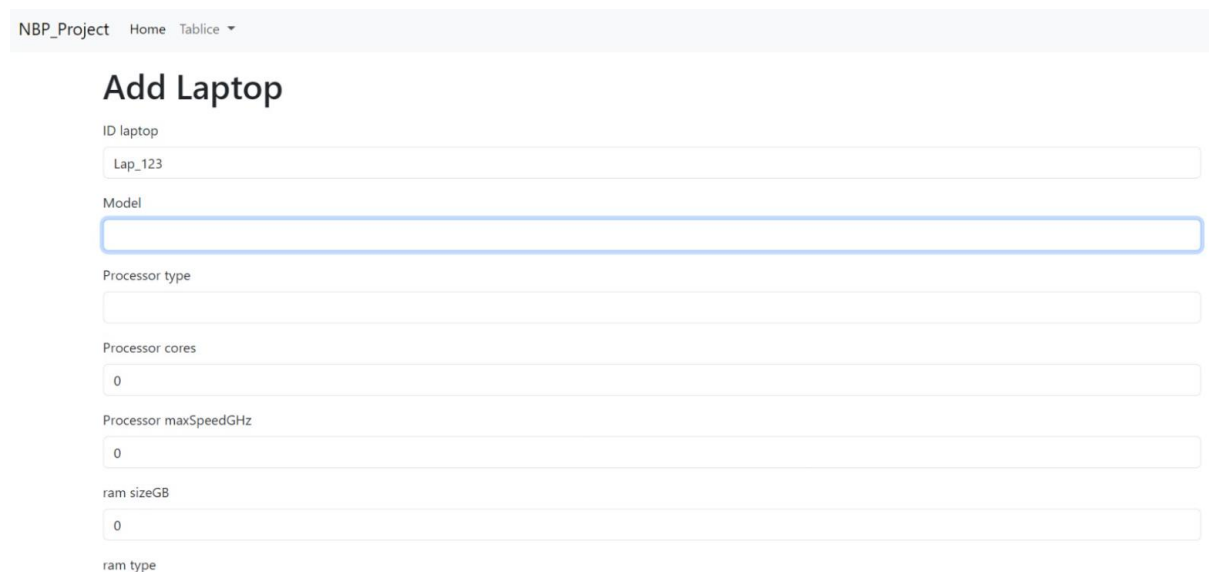
```

src > app > features > tables > delete-laptop > delete-laptop.component.html > div.container
1  <div class="container">
2    <h1 class="mt-3">Remove Laptop</h1>
3
4    <form #form="ngForm" (ngSubmit)="onFormSubmit()">
5      <div class="mt-2">
6        <div class="mt-3">
7          <label class="form-label">Id laptop</label>
8          <input type="" id="id_Laptop" class="form-control" name="id_Laptop" [(ngModel)]="model.id_Laptop">
9        </div>
10
11        <div class="mt-3">
12          <button type="submit" class="btn btn-outline-danger">Delete</button>
13        </div>
14      </div>
15    </form>
16  </div>
17

```

Slika 10: Primjer stranice za brisanje laptopa iz baze

Važno je napomenuti da se podaci šalju ovisno o pojedinoj operaciji, odnosno načinu na koji je implementirana njezina funkcija u backendu (definiciji ulaznih argumenata u funkciju). Primjerice, funkcija može primiti samo parametar *id*, te bismo tada pripremili *HttpParams* u koji bismo naveli ime ulaznog argumenta te njegovu vrijednost dobivenu s korisničkog sučelja.



The screenshot shows a web application interface for adding a laptop. At the top, there is a navigation bar with 'NBP_Project', 'Home', and a dropdown menu 'Tablice'. The main heading is 'Add Laptop'. Below it, there are several input fields: 'ID laptop' with the value 'Lap_123', 'Model' (empty), 'Processor type' (empty), 'Processor cores' with the value '0', 'Processor maxSpeedGHz' with the value '0', 'ram sizeGB' with the value '0', and 'ram type' (empty).

Slika 11: Primjer frontenda za dodavanje laptopa u bazu

Kreirali smo i posebnu mapu za modele koji nam predstavljaju podatke u onom formatu i definiciji kakvi se nalaze u bazi. Također, možemo modelirati neku funkciju koju koristimo za poziv na backend pa smo tako za neke kontrolere pisali odvojene modele za GET, POST i DELETE jer nam definicija tablice u bazi nije odgovarala definiciji ulaznih argumenata u pojedinu funkciju.

Važna datoteka za Angular projekt koji smo također često mijenjali je ``app-routing.module.ts`` u kojem postavljamo definiciju za *routing*, odnosno kažemo kompajleru koju komponentu mora pozvati, odnosno koju stranicu učitati kada kliknemo na neki link unutar stranice. Još jedna bitna datoteka za projekt je ``app.module.ts`` u kojem smo definirali module (u nekim drugim programskim jezicima to se može zvati bibliotekama) koje kompajler treba učitati prilikom pokretanja servera, a koji su nam nužni jer ih koristimo u projektu.

Primjer jednog eksternog modula koji smo morali instalirati unutar Node.js-a i koji koristimo u projektu za prikaz podataka koje nam backend server vrati na korisničko sučelje je *MatTableModule*. Modul za komunikaciju s backendom i slanje upita na backend je *HttpClientModule*. Isto tako u navedenu datoteku smo preko naredbi za kreiranje stavljali i deklaracije komponenti.

Cijena (€)	ID Inventar	ID Poslovnica	ID Predmet	ID
699,00	Inv_001	Pos_001	Lap_001	72de7a21-b0a1-4ffd-82d1-740dc8525597
699,00	Inv_002	Pos_002	Lap_001	b6c53a95-64b9-4f93-b2a2-2e0db725edc6
1599,00	Inv_003	Pos_001	Lap_002	8ba727c7-1a25-41d7-ba7c-50d0a69e4f14
1599,00	Inv_004	Pos_002	Lap_002	5e5abfcd-ca9d-4665-8015-78e9ce533acb
1049,00	Inv_005	Pos_001	Lap_003	227a88ea-a410-4b08-b30d-58c14ce43f16
1049,00	Inv_006	Pos_002	Lap_003	b485cb76-5443-4698-8e06-712a8c1e63c2
1079,00	Inv_007	Pos_001	Lap_004	f6f8c4b9-4e0e-429a-9ee4-5173a8e03168
1079,00	Inv_008	Pos_002	Lap_004	146e34a3-0d1c-43c1-b743-a99c3c8535bf
2299,00	Inv_009	Pos_001	Lap_005	71d92efa-2d78-468d-91d4-c78b944342b6
2299,00	Inv_010	Pos_002	Lap_005	fb719632-bb68-40b6-a2e0-27b6c7f5cb1d

Slika 12: Primjer početne stranice za Inventar

Svaki pojedini entitet u bazi ima svoju "početnu" stranicu na kojoj su unutar tablice prikazani zapisi koji se nalaze u bazi za taj entitet. Sastoji se od gumba za GET, POST i DELETE operacije. Klikom na jedan od tih gumba odlazimo na pripadajuću stranicu koje smo kreirali kao komponente. Na tim stranicama ispunjavamo vrijednosti u forme te klikom na gumb se okida funkcija koja šalje upit na backend koji onda odradi željenu operaciju. U slučaju GET stranica, backend će nam vratiti zapise (jedan ili više njih) koje je pronašao u bazi pa ih stoga još dodatno prikazuje u tablici ispod forme.

Get one poslovnica

Naziv:

ID poslovnica:

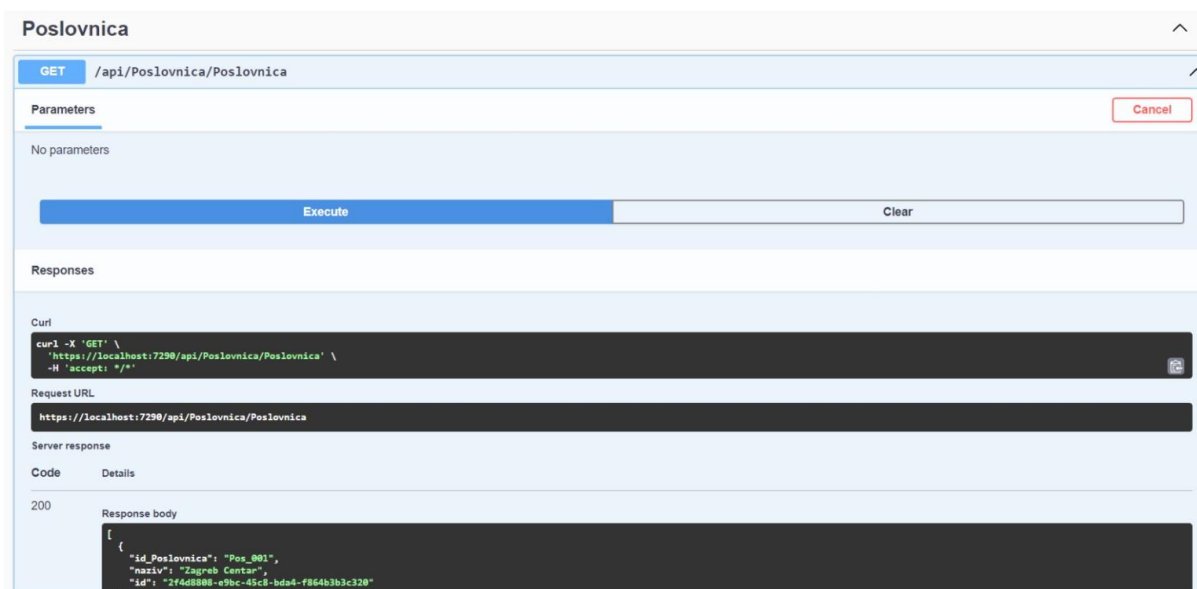
Naziv	ID poslovnica
Zagreb Centar	Pos_001
Split Riva	Pos_002

Slika 13: Primjer frontenda za dohvaćanje poslovnica iz baze

4.2. Poteškoće

Jedan primjer s kojim smo se dosta namučili bila je prva POST metoda koju smo razvijali jer tada još nismo shvaćali razliku između poziva backenda preko parametara i preko objekta. Primjer poziva preko objekta je POST metoda za tablicu Laptop jer funkcija na backendu prima objekt laptop. Ovo je jedina takva funkcija u backendu; dok sve ostale koriste parametarski poziv gdje preko *HttpParams* definiramo rječnik `{"ime_argumenta": "vrijednost"}`. Rješenje smo našli na GitHub forumu sličnom Stack Overflow-u gdje je u jednom komentaru pisalo da se mora paziti prima li funkcija u backendu objekt ili pojedine parametre, te smo stoga od toga trenutka više koristili i iščitavali definicije funkcija iz backenda.

Također, koristili smo backend za debugiranje tako što smo stavljali *breakpoint-ove* u funkcije kojima pokušavamo uputiti upit preko korisničkog sučelja. To je uvelike pomoglo da vidimo što funkcija zapravo primi kao vrijednost argumenta. Na ovakav način debugiranja potaklo nas je i to što su neki upiti za upis u bazu vraćali greške, a razlog je bio što je neki atribut bio *null* vrijednost. Greška je bila u tome što smo krivo napisali ime ulaznog argumenta u upitu unutar *HttpParams* rječnika, što bi rezultiralo time da ga backend ne može pročitati, odnosno tumačio bi to kao da nismo poslali vrijednost za taj argument.



Slika 14: Primjer iz Swaggera za metodu GET

Slična stvar se događala i kada bismo koristili GET upit koji bi nam vraćao zapise iz tablice, samo što je greška bila u tome što unutar modela nismo dobro definirali ime varijable te Angular nije mogao ispravno spojiti vrijednosti za prikaz. Ovo smo otkrili preko *developer mode-a* koji možemo otvoriti unutar svakoga internetskog preglednika.

Koristili smo Microsoft Edge te bismo ga otvorili tako da kliknemo na desni klik i pritisnemo gumb "Inspect", potom otvorimo Console sekciju u kojoj se onda ispisuju greške i primjerice `console.log()` outputi. Greška ovdje je bila da se ne može spojiti `id` s "`imenom_varijable`". Istim putem smo također otkrivali greške povezane s tipovima varijabli jer Angular nije mogao spojiti nešto što je u Angularu tipa *number*, a u bazi je tipa *boolean*.

Također neke greške smo otkrili preko CMD-a gdje je bio pokrenut Angular server. Tamo su uglavnom bile ispisane greške koje uzrokuju rušenje servera ili nemogućnost podizanja servera, a povezane su s definicijom i učitavanjem modela, komponenti ili servisa, kao npr. nešto nije definirano, varijabla nije ispravnog tipa za ono što vraća pojedina funkcija i slično.

5. Zaključak

Izgrađen je sustav za upravljanje trgovačkim lancem informatičke opreme koji se sastoji od relacijske i dokument baze podataka te frontend sučelja. Izrađena aplikacija administratorima omogućava unos, izmjenu i brisanje pojedinih podataka, a korisnicima odabir laptopa i dostupnost po poslovnicama.

Naš cilj je bio stvoriti template za kompletnu apstraktnu aplikaciju koja će omogućiti uvid u poslovanje jedne trgovine s više poslovnica. Informacije kojima najčešće želimo raspolagati su asortiman koji nudimo i njihov opis, ukupna količina pojedinog proizvoda kojom raspolažemo, distribucija pojedinih proizvoda po poslovnicama te sama prodaja odnosno pojedine transakcije kupnje.

Aplikacija je dostupna za slobodnu upotrebu i korištenje svim kolegicama i kolegama na linku https://github.com/jurprek/NBP_project_Store

6. Samoevaluacije

6.1. Tena Albina Kao

S obzirom na to da do sada nisam imala puno iskustva u razvoju korisničkog sučelja, odlučila sam se upravo za tu komponentu projekta. U Angularu do sada nikada nisam radila pa sam stoga morala krenuti iz nule te proučavati tutorijale preko YouTube-a, W3Schools-a i sličnih portala. Također je ogromna pomoć u razvoju bilo Google-anje specifične greške ili opisivanje problema na Google-u ako greška nije bila jasna. To bi uglavnom dovelo do Stack Overflow stranice na kojoj su u većini slučajeva jako dobro objašnjeni uzroci i rješenja za problem. Moj zadatak bio je kreirati stranice za tablice 'Inventar', 'Kupac', 'Kupnja', 'Laptop' i 'Poslovnica'.

6.2. Petar Pavlović

Budući da ovaj projekt koristi brojne programe, te imam samo malo iskustva sa C#-om i nikakvog iskustva sa Angularom, zahtjevalo je dosta vremena i razgovora s kolegama da bi se sve instaliralo, spojilo i pokrenulo. Iako sam pokušao uskladiti verzije svih paketa u Visual Studiu, pokretanje na predviđeni način nikada mi nije proradilo pa sam backend morao pokretati pomoću https opcije što je rezultiralo time da Swagger nije na localhost:44393 već na localhost:7290. To je značilo da sam morao u frontendu prilagoditi web adrese kako bi proradilo. Sa svime funkcionalnim mogao sam konačno izraditi dio frontenda, konkretno tablice Trgovac sa funkcionalnostima get, add, delete i get sa pretragom, te tablice Predmet sa ta dva get-a.

6.3. Magdalena Potočnjak

U sklopu ovog projekta radila sam na konstruiranju sheme relacijske i dokument baze te sam bila zadužena za njihovo popunjavanje testnim podacima. Obzirom da je to praktički prvi korak pri konstrukciji projekta, odrađen je kostur baze kako bi se kolege mogli nastaviti dalje baviti razvojem aplikacije. Nažalost, zbog nedostatka vremena je taj kostur doživio i sami kraj. Svjesna sam da ima mnogo mjesta za napredak u tom dijelu te da nisu obuhvaćeni svi atributi za pojedine entitete koji bi bili potrebni u stvarnosti za jednu takvu aplikaciju. Primjerice, važna je informacija o količini proizvoda dostupnih u pojedinoj poslovnici.

Iako sam imala poteškoća oko same instalacije svega potrebnoga za pokretanje aplikacije, zahvaljujući instrukcijama kolega sve su otklonjene te sam pomogla pri izradi backenda koliko sam znala. Također, kolege su popisali svoje obavljene zadatke te sam ih potom sve ukomponirala u ovaj seminarski rad i prezentaciju.

6.4. Jurica Preksavec

Bilo je korisno prisjetiti se izrade backend dijela aplikacije kroz Rhetos framework, ali kako do sad nisam radio s više bazi različitog tipa izazov je bio povezati Mongo u odabranom okruženju. S obzirom na to da je odabrani framework kreacija hrvatskih programera izvori informacija nisu toliko dostupni kao u slučaju npr. Microsoft-ovih proizvoda, pa je bilo potrebno malo više vremena provesti u testiranju. Iz tog razloga radi jednostavnosti je rađen dodatni kratki string ključ. Swagger, koji se inače rutinski dodaje, je tako u testiranju odigrao ključnu ulogu. Problem kreiranja entiteta zbog kojeg imamo dokument bazu se pokazao zahtjevnim zbog toga što se mogu desiti situacije da se kreira u jednoj ali ne i u drugoj bazi. Taj dio je riješen upozorenjem da se takvo što dogodilo, pa se ručno jednostavno ispravlja, ali tu mjesta za poboljšanja ima na pretek. Ipak, osobno sam zadovoljan postignutim, upravo zbog činjenice da nam ova aplikacija može poslužiti kao podloga u širokom spektru srodnih, a kako je već pronašla i svog prvog stvarnog korisnika – zadovoljstvo zbog kreiranja iste je potpunije.

7. Literatura

- Alat za kreiranje sheme relacijske baze podataka (<https://dbdiagram.io/d>)
- Rhetos okruženje (<https://github.com/Rhetos/Rhetos-wiki/blob/master/Data-model-and-relationships.md>)
- Angular tutoriali:
 - <https://v17.angular.io/docs>
 - <https://www.w3schools.com/angular/default.asp>
 - <https://www.youtube.com/watch?v=eNVbiIsoEUw>
 - <https://material.angular.io/components/table/overview>
- Rješenje problema u Angularu (<https://stackoverflow.com/questions/74974668/how-to-transfer-data-via-query-params-of-a-post-request-using-angular>)