Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli



**TIM 4: Webshop informatičke opreme**

Loren Bažon

Morena Martan

Bruno Rebić

Fran Barba

Josip Milković

Leo Hrvojić

Smjer : Informatika

Kolegij : Baze podataka II

Mentor : doc. Dr. sc. Goran Oreški

Asistent : mag. inf. Romeo Šajina

Pula, studeni, 2024. godina

Sadržaj

[Opis poslovnog procesa 4](#_Toc188217092)

[ER dijagram 5](#_Toc188217093)

[Opis ER dijagrama 6](#_Toc188217094)

[Kombinacija shema 7](#_Toc188217095)

[EER DIJAGRAM (MYSQL WORKBENCH) 8](#_Toc188217096)

[Tablice 8](#_Toc188217097)

[tablice Loren 8](#_Toc188217098)

[tablice Morena 11](#_Toc188217099)

[tablice Bruno 13](#_Toc188217100)

[tablice Fran 15](#_Toc188217101)

[tablice Josip 17](#_Toc188217102)

[tablice Leo 21](#_Toc188217103)

[Upiti i pogledi 26](#_Toc188217104)

[Upiti i pogledi Loren 26](#_Toc188217105)

[Upiti i pogledi Morena 32](#_Toc188217106)

[Upiti i pogledi Bruno 36](#_Toc188217107)

[Upiti i pogledi Fran 38](#_Toc188217108)

[Upiti i pogledi Josip 40](#_Toc188217109)

[Upiti i pogledi Leo 42](#_Toc188217110)

[Funkcije 48](#_Toc188217111)

[Funkcije Loren 48](#_Toc188217112)

[Funkcije Morena 51](#_Toc188217113)

[Funkcije Bruno 53](#_Toc188217114)

[Funkcije Fran 55](#_Toc188217115)

[Funkcije Josip 55](#_Toc188217116)

[Funkcije Leo 57](#_Toc188217117)

[Procedure 58](#_Toc188217118)

[Procedure Loren 58](#_Toc188217119)

[Procedure Morena 68](#_Toc188217120)

[Procedure Bruno 76](#_Toc188217121)

[Procedure Fran 80](#_Toc188217122)

[Procedure Josip 82](#_Toc188217123)

[Procedure Leo 82](#_Toc188217124)

[Okidači 89](#_Toc188217125)

[Okidači Loren 89](#_Toc188217126)

[Okidači Morena 91](#_Toc188217127)

[Okidači Bruno 94](#_Toc188217128)

[Okidači Fran 95](#_Toc188217129)

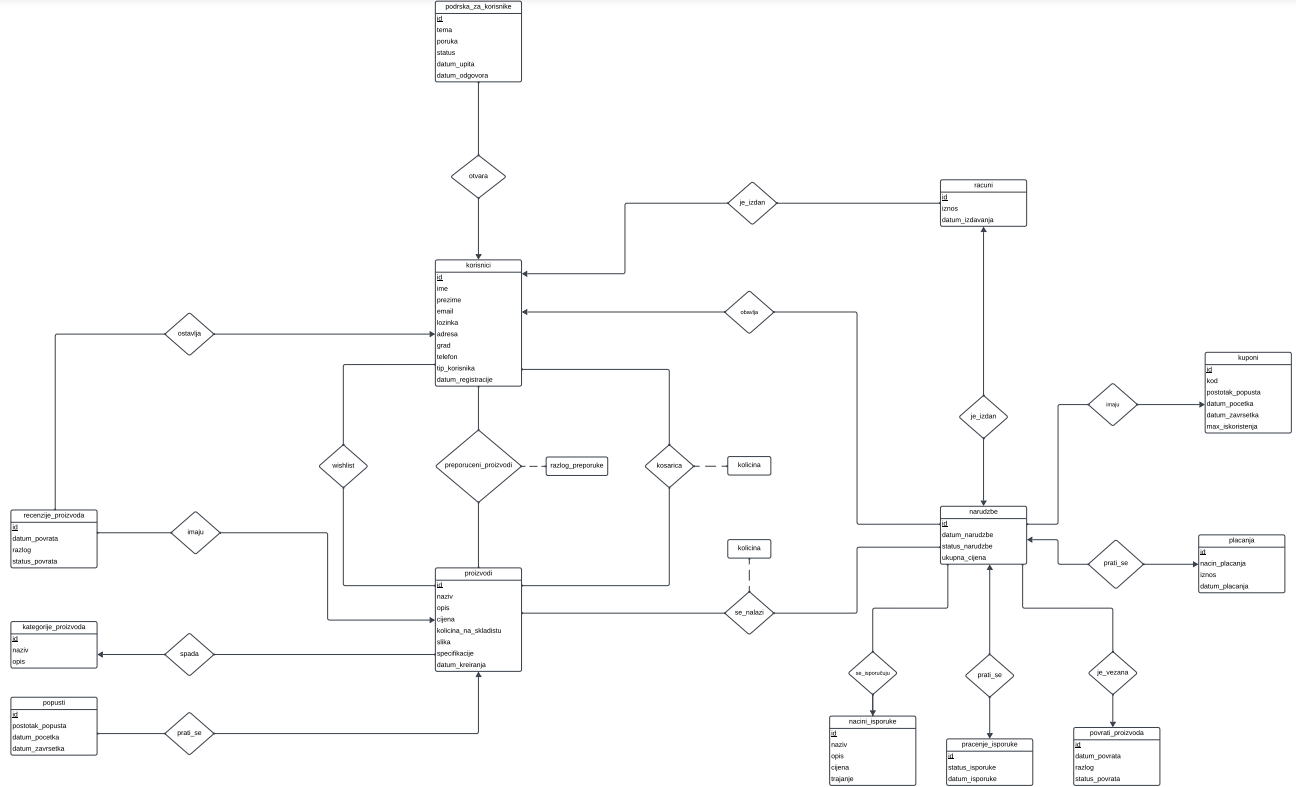
[Okidači Josip 96](#_Toc188217130)

[Okidači Leo 97](#_Toc188217131)

# Opis poslovnog procesa

Naš webshop informatičke opreme omogućava korisnicima kupnju proizvoda te upravljanje narudžbama kroz jednostavan i intuitivan proces. Kada korisnik dođe na početnu stranicu webshopa, prikazuju mu se promotivni proizvodi, kategorije i preporuke. Proizvodi su rangirani prema popularnosti, popustima ili novitetima. Za *proizvode* se prati naziv, opis, cijena, kategorija\_id, kolicina\_na\_skladistu, slika, specifikacije, datum\_kreiranja. Prijavljeni korisnici vide preporučene proizvode (tablica *preporuceni\_proizvodi*) na temelju njihovih preferencija. Za prijavljene *korisnike* se prati ime, prezime, email, lozinka, adresa, grad, telefon, tip\_korisnika(admin ili kupac) te datum\_registracije. Korisnici mogu pregledavati proizvode kroz kategorije (*kategorije\_proizvoda*) za koje se prati naziv i opis. Klikom na određeni proizvod otvara se detaljna stranica sa već spomenutim informacijama o proizvodu, recenzijama, popustima te promotivnim kuponima. Za *recenzije* se prati ocjena, komentar i datum\_recenzije. Za *popuste* se prati postotak, datum početka i datum završetka te se za *kupone* prati kod, postotak popusta, datum početka, datum završetka te maksimalni broj korištenja. Sustav implementira listu želja, odnosno *wishlist*, koji omogućava korisniku da spremi omiljene proizvode koje planira kupiti. Kad korisnik želi kupiti određeni proizvod, klikom na „Dodaj u košaricu“ sustav provjerava dostupnost proizvoda (tablica *proizvodi,* atribut kolicina\_na\_skladistu). Ako je proizvod dostupan, dodaje se u *košaricu* za koju se prati količina. Ako je korisnik prijavljen, u košarici se veže njegov id sa proizvodom koji je dodao u košaricu. Ako korisnik nije prijavljen, kreira se privremeni *sesija\_id* koji se koristi za pohranu proizvoda u košaricu. Korisnik prije potvrde narudžbe može pregledati sve proizvode u košarici, promijeniti količine ili ukloniti stavke. Na kraju se prikazuje ukupna cijena (sa ili bez popusta) te troškovi dostave koji se računaju iz tablice *nacin\_isporuke*. Za način isporukese prati naziv, opis, cijena, trajanje. Da bi korisnik naručio proizvode, mora biti prijavljen. Ako nije prijavljen ili nema račun, mora se registrirati. Kada je to sve obavljeno, klikom na „Potvrdi narudžbu“ sustav kreira zapis u tablici *narudzbe* za koju se prati datum\_narudzbe, status\_narudzbe i ukupna cijena. Proizvodi iz košarice se prenose u tablicu *stavke\_narudzbe* za koju se prati količina. Korisnik bira način plaćanja koje sustav validira te se zapisuje u tablicu *placanja* za koju se prati iznos i datum plaćanja. Nakon uspješnog plaćanja, kreira se *račun* za koji se prati iznos i datum izdavanja. Korisnik može vidjeti status svoje narudžbe putem webshopa kroz tablicu *pracenje\_isporuke* za koju se prati status\_isporuke („priprema“, „poslano“ i „dostavljeno“) i datum\_isporuke. Ako nekim slučajem korisnik želi napraviti povrat narudžbe, podnosi zahtjev webshopu te se kreira zapis u tablici *povrati\_proizvoda* za koju se prati datum\_povrata, razlog i status\_povrata. Za bilo kakve nejasnoće ili probleme na koje korisnik naiđe za vrijeme korištenja nađeg webshopa, korisnik se obraća službi za korisnike. Povijest razgovora korisnika za službom se sprema u tablicu *podrska\_za\_korisnike* za koju se prati tema, poruka, status, datum\_upita i datum\_odgovora.

# ER dijagram



# Opis ER dijagrama

**Proizvod** spada pod **kategoriju** (Jedan proizvod može biti iz jedne kategorije, dok iz jedne kategorije može biti više proizvoda). Jedan na više.

**Korisnik** obavlja **narudžbe** (Jedan korisnik može obavljati više narudzbi, dok je jedna narudžba obavljena od strane samo jednog korisnika) jedan na više.

**Narudžbe** se isporučuju **načinom isporuke** (Jedna narudžba se može isporučiti samo jednim načinom isporuke, dok jednim načinom isporuke se može isporučiti više narudžbi) jedan na više.

**Narudžbe** imaju **kupone** (Jedna narudžba ima jedan kupon, dok jedan kupon može biti na više narudžbi) jedan na više.

**Proizvodi** se nalaze na **narudžbama** (jedan proizvod može biti na više narudžbi, dok na jednoj narudžbi može biti više proizvoda) više na više

Stvara se dodatni entitet **stavke\_narudzbe** gdje se prati količina proizvoda na narudžbama, a primarni ključevi entiteta proizvodi i narudzbe se referenciraju kao strani ključ u stavke\_narudzbe.

**Narudžbe** se prati **plaćanje** (Jednoj narudžbi se prati jedno plaćanje, dok jedno plaćanje je praćena za jednu narudžbu) jedan na jedan.

**Korisniku** je izdan **račun** (Jedan račun je izdan za jednog korisnika, dok jedan korisnik može imati više izdanih računa) jedan na više.

**Račun** je izdan za **narudžbu** (Za jednu narudžbu je izdan samo jedan račun, dok je jedan račun vezan samo za jednu narudžbu) jedan na jedan

**Popusti** se prati za **proizvode** (Jedan proizvod može imati više različitih popusta, dok jedan popust koji se prati je vezan samo za jedan proizvod) jedan na više.

**Narudžba** može imati **povrat** (Jedan povrat proizvoda može biti vezan za više narudžbi, dok jedna narudžba ima jedan povrat.) jedan na više

**Proizvod** ima **recenziju** (Jedna recenzija je vezana za jedan proizvod, dok jedan proizvod može imati više recenzija) jedan na više

**Korisnik** ostavlja **recenziju** (Jedan korisnik može ostaviti više recenzija, dok jedna recenzija je napisana od strane jednog korisnika) jedan na više

**Korisnik** sprema wishlistu **proizvoda** (Jedan korisnik može spremiti više proizvoda, dok jedan proizvod može biti spremljen kod više korisnika) više na više.

Stvara se dodatni entitet **wishlist** gdje primarni ključevi entiteta proizvodi i korisnici se referenciraju kao strani ključ u wishlist.

Za **narudžbu** se prati **isporuka** (Jedna narudzba ima jednu isporuku i jedna isporuka prati jednu narudžbu) jedan na jedan

**Korisniku** se preporučuje **proizvod** (Jedan korisnik može imati više preporuka, dok jedna preporuka može biti kod više korisnika) više na više.

Stvara se dodatni entitet **preporuceni\_proizvodi** gdje se prati razlog\_preporuke, a primarni ključevi entiteta proizvodi i korisnici se referenciraju kao strani ključ u preporuceni\_proizvodi.

**Korisnik** otvara poruku za **podršku** (Jedan korisnik može otvoriti više poruka za podršku, dok jedna poruka je otvorena od samo jednog korisnika) jedan na više

**Korisnik** sprema **proizvode** u košaricu (Jedan korisnik može spremiti više proizvoda u košaricu, dok jedan proizvod može biti spremljen kod više korisnika) više na više

Stvara se dodatni entitet **kosarica** gdje je prati količina, a primarni ključevi entiteta proizvodi i korisnici se referenciraju kao strani ključ u kosarica.

# Kombinacija shema

**kategorije\_proizvoda**(id, naziv, opis)

**proizvodi**(id, naziv, opis, cijena, kategorija\_id, kolicina\_na\_skladistu, slika, specifikacije, datum\_kreiranja)

**korisnici**(id, ime, prezime, email, lozinka, adresa, grad, telefon, tip\_korisnika, datum\_registracije)

**nacini\_isporuke**(id, naziv, opis, cijena, trajanje) PROMIJENITI TIP ZA trajanje u INTEGER

**kuponi**(id, kod, postotak\_popusta, datum\_pocetka, datum\_zavrsetka, max\_iskoristenja)

**narudzbe**(id, korisnik\_id, datum\_narudzbe, status\_narudzbe, ukupna\_cijena, nacin\_isporuke\_id, kupon\_id)

**stavke\_narudzbe**(id, narudzba\_id, proizvod\_id, kolicina)

**placanja**(id, narudzba\_id, nacin\_placanja, iznos, datum\_placanja)

**racuni**(id, korisnik\_id, narudzba\_id, iznos, datum\_izdavanja)

**povrati\_proizvoda**(id, narudzba\_id, datum\_povrata, razlog, status\_povrata)

**popusti**(id, proizvod\_id, postotak\_popusta, datum\_pocetka, datum\_zavrsetka)

**recenzije\_proizvoda**(id, proizvod\_id, korisnik\_id, ocjena, komentar, datum\_recenzije)

**wishlist**(id, korisnik\_id, proizvod\_id)

**pracenje\_isporuka**(id, narudzba\_id, status\_isporuke, datum\_isporuke)

**preporuceni\_proizvodi**(id, korisnik\_id, proizvod\_id, razlog\_preporuke)

**podrska\_za\_korisnike**(id, korisnik\_id, tema, poruka, status, datum\_upita, datum\_odgovora)

**kosarica**(id, korisnik\_id, proizvod\_id, kolicina)

# EER DIJAGRAM (MYSQL WORKBENCH)



# Tablice

## tablice Loren

**CREATE TABLE** kategorije\_proizvoda (

id **INT AUTO\_INCREMENT PRIMARY KEY,**

naziv **VARCHAR(255) NOT NULL UNIQUE,**

opis **TEXT NOT NULL**

);

Ova tablica se naziva „kategorije\_proizvoda“ ima atribut „id “ koji je cijeli broj i postavljen je za primarni ključ ove relacije. Pomoću dodatne naredbe AUTO\_INCREMENT osiguravamo jedinstveni broj koji se automatski povećava za jedan unosom svakog novog retka u tablicu. U atribut „naziv“ spremamo puni naziv kategorije VARCHAR tipa do maksimalne duljine od 255 znakova. Ograničenjem NOT NULL smo osigurali da u ovaj stupac mora biti unesena neka vrijednost također, ograničenjem UNIQUE smo osigurali da ona bude jedinstvena za svaki unos novog retka. Posljednji stupac u ovoj relaciji je „opis“ i on je (TEXT) tekstualnog tipa kako bismo mogli unijeti više informacija i detaljnije opisati svaku kategoriju radi lakšeg razumijevanja od strane korisnika.

**CREATE TABLE** proizvodi (

id **INT AUTO\_INCREMENT PRIMARY KEY,**

naziv **VARCHAR(255) NOT NULL,**

opis **TEXT NOT NULL,**

cijena **DECIMAL(10, 2) NOT NULL,**

kategorija\_id **INT NOT NULL,**

kolicina\_na\_skladistu **INT NOT NULL,**

slika **VARCHAR(255),**

specifikacije **TEXT,**

datum\_kreiranja **TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**CONSTRAINT** chk\_cijena\_proizvodi **CHECK (cijena > 0),**

**CONSTRAINT** chk\_kolicina\_proizvodi **CHECK (kolicina\_na\_skladistu >= 0),**

**CONSTRAINT** fk\_kategorija\_proizvoda\_proizvodi **FOREIGN KEY** (kategorija\_id) **REFERENCES** kategorije\_proizvoda(id)

);

Tablica „proizvodi“ sadrži jedinstveni identifikacijski broj cjelobrojnog tipa koji smo ograničili AUTO\_INCREMENTom tako da se „id“ jedinstveno povećava za svaki unos u tablicu. Stupac „naziv“ je VARCHAR tipa ograničen na 255 znakova a ograničenje NOT NULL nam osigurava nužan unos podatka u ovaj stupac. „opis“ nam je sljedeći atribut tekstualnog tipa. „cijena“ pojedinog proizvoda decimalnog je tipa ograničena na 10 znamenaka od kojih su dvije iza decimalnog zareza. Check constraintom smo osigurali da unos cijene mora biti pozitivan broj, u suprotnom će se javiti greška. Atribut „ kategorija\_id“ nam je referencirani strani ključ na tablicu „kategorije\_proizvoda“ koja također mora biti unesena kako bi tablice ostale povezane u suprotnom nam se opet javlja greška prilikom unosa. Stupac „kolicina\_na\_skladistu“ je ograničena na unos pozitivne cjelobrojne vrijednosti. „slika“ je tipa VASCHAR ograničena na 255 znakova, a „specifikacija“ je atribut tekstualnog tipa (TEXT). Posljednji atribut u ovoj relaciji je „datum\_kreiranja“ koji automatski popunjava prilikom kreiranja ovog retka u tablici sa vrijednošću trenutnog vremena (CURRENT\_TIMESTAMP).

**CREATE TABLE** recenzije\_proizvoda (

id **INT AUTO\_INCREMENT PRIMARY KEY,**

proizvod\_id **INT NOT NULL,**

korisnik\_id **INT NOT NULL,**

ocjena **INT NOT NULL,**

komentar **TEXT,**

datum\_recenzije **TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**,

**CONSTRAINT** chk\_ocjena\_recenzije\_proizvoda **CHECK** (ocjena **BETWEEN** 1 **AND** 5),

**CONSTRAINT** fk\_proizvod\_recenzija\_recenzije\_proizvoda **FOREIGN KEY** (proizvod\_id) **REFERENCES** proizvodi(id),

**CONSTRAINT** fk\_korisnik\_recenzija\_recenzije\_proizvoda **FOREIGN KEY** (korisnik\_id) **REFERENCES** korisnici(id) );

Ova tablica se naziva „recenzije\_proizvoda“ i sadrži atribut „id“ koji je cjelobrojnog tipa postavljen na AUTO\_INCREMENT tako da se automatski povećava svaki put prilikom novog unosa u tablicu. Služi kao primarni ključ i osigurava jedinstvenu identifikaciju svake recenzije. „proizvod\_id“ je cjelobrojni atribut koji se odnosi na primarni ključ u tablici „proizvodi“. Označava proizvod na koji se odnosi recenzija. Atributom „korisnik\_id“ referenciramo se na tablicu „korisnik “ preko stranog ključa ove tablice koji je također cjelobrojnog tipa. Stupac „ocjena“ je također cijeli broj ograničen na brojeve od 1 do 5 koje unosi korisnik. U suprotnom mu vraća grešku prilikom unosa podatka u ovaj stupac. Atribut „komentar“ je tekstualnog oblika (TEXT) kako bi korisniku bilo omogućeno unijeti veću količinu znakova. Zadnji atribut je „datum\_recenzije“ koji se popunjava automatski sukladno vremenu napisane recenzije.

## tablice Morena

CREATE TABLE kosarica (

id INT AUTO\_INCREMENT PRIMARY KEY,

korisnik\_id INT NOT NULL,

proizvod\_id INT NOT NULL,

kolicina INT NOT NULL,

FOREIGN KEY (korisnik\_id) REFERENCES korisnik(id),

FOREIGN KEY (proizvod\_id) REFERENCES proizvodi(id)

);

Kreirali smo tablicu naziva „kosarica“ pomoću koje ćemo pratiti odabrane proizvode od strane naših korisnika kako bi ih mogli kasnije u bazi podataka evidentirati. Tablica se sastoji od atributa: id koji nam je primarni ključ ove tablice i jedinstveni broj pomoću kojeg ćemo određene proizvode pratiti od narudžbe do isporuke. Odabrane proizvode u košarici moramo povezati sa stranom tablicom „korisnici“ preko stranog ključa „korisnik\_id“ koji je cjelobrojnog tipa i jedinstven za svakog korisnika. U tablicu dodajemo atribut „proizvod\_id“ preko kojeg bilježimo odabrani proizvod također putem cjelobrojnog jedinstvenog broja jer nam je i on ovdje strani ključ. Na kraju, dodajemo i „kolicina“ pomoću koje bilježimo količinu odabranog proizvoda koju smo također ograničili na cjelobrojnu pozitivnu vrijednost jer u suprotnom ova tablica ne bi imala smisla.

CREATE TABLE narudzbe (

id INT AUTO\_INCREMENT PRIMARY KEY,

korisnik\_id INT NOT NULL,

datum\_narudzbe DATE NOT NULL,

status\_narudzbe ENUM('u obradi', 'poslano', 'dostavljeno') NOT NULL,

ukupna\_cijena DECIMAL(10, 2) NOT NULL,

nacin\_isporuke\_id INT,

kupon\_id INT NULL,

FOREIGN KEY (korisnik\_id) REFERENCES korisnici(id),

FOREIGN KEY (nacin\_isporuke\_id) REFERENCES nacini\_isporuke(id),

FOREIGN KEY (kupon\_id) REFERENCES kuponi(id)

);

Ova SQL tablica, nazvana narudžbe, koristi se za pohranjivanje informacija o narudžbama u sustavu. U nju korisnik sam kreira podatke preko GUI našeg Web-shopa. Atribut „id“ sadrži jedinstveni cjelobrojni identifikator narudžbe koji se automatski povećava i koji nam je ovdje primarni ključ. Kako bismo znali tko nam je kreirao ovu tablicu, koristimo atribut „korisnik\_id“ koji je cjelobrojni pozitivan broj te strani ključ, iz tablice koja je generirana unosom podataka i registracijom korisnika „korisnik“. Atributom „datum\_narudzbe“ bilježimo kada je naudžba kreirana pomoću DATE tipa podatka koji se generira sukladno datumu kada je korisnik kreirao narudžbu. Samo vrijeme nam nije bitno (pa nismo stavili DATETIME) ali nam je potencijalno bitno koliko dana je prošlo od zahtjeva za narudžbom. Obzirom da su datumi zapravo tri odvojena podatka (dan, mjesec i godina), lakše nam je manevrirati ovim tipom podatka nego da smo ga okarakterizirali kao integer, pa je tako ovdje tip podatka DATE sretnije rješenje. Da bismo pratili u kojoj fazi između narudžbe i dostave je naš paket, koristimo atribut „status\_narudžbe“, ograničen tipom ENUM sa mogućim vrijednostima „u obradi“, „poslano“ ili „dostavljeno“ radi lakše obrade narudžbe i u slučaju kašnjenja ili potencijalno nedospjelog paketa kako bismo mu lakše detektirali lokaciju. Dodajemo također atribut „ukupna\_cijena“ u kojem nam je zbrojena suma naručenih proizvoda ovisno o količini. Cijenu smo ograničili decimalnim tipom podataka sa dvije znamenke iza decimalne točke i osam znamenaka ispred i koja glasi: DECIMAL(10,2) , također mora biti pozitivna vrijednost. Vrstu tranzicije naručenih proizvoda pratimo preko atributa „nacin\_isporuke\_id“ jer u našoj ponudi postoje četiri opcije dostave od kojih jednu korisnik odabire istovremeno kad i naručuje proizvod. Atribut „kupon\_id“ povezuje tablicu „kuponi“ sa našom tablicom, međutim, nije nužno da korisnik ima neki od kupona pa ovdje postoji opcija NULL u slučaju da upisani kupon ne postoji ili da kupon ni nije upisan.

CREATE TABLE stavke\_narudzbe (

id INT AUTO\_INCREMENT PRIMARY KEY,

narudzba\_id INT NOT NULL,

proizvod\_id INT NOT NULL,

kolicina INT NOT NULL,

FOREIGN KEY (narudzba\_id) REFERENCES narudzbe(id),

FOREIGN KEY (proizvod\_id) REFERENCES proizvodi(id)

);

Tablica „stavke\_narudzbe“, koristi se za praćenje stavki unutar narudžbe. Ova relacija sadrži jedinstveni identifikacijski broj koji se automatski povećava za svaku novu stavku preko „id“ atributa. Obzirom da tablica spaja tablice „narudzbe“ i „proizvodi“ jer su one povezane relacijom više-na-više, zahtijevamo unos stranih ključeva „narudzba\_id“ i „proizvod\_id“ koji moraju biti unešeni pa smo ih ograničili sa NOT NULL. U suprotnom, ova relacija gubi svoj smisao. Jedino što se ovdje dodaje je količina odabranog proizvoda koja se zahtjeva od strane kupca. Količinu smo također ograničili na cjelobrojni pozitivan broj.

## tablice Bruno

CREATE TABLE nacini\_isporuke (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    naziv VARCHAR(255) NOT NULL,

    opis TEXT,

    cijena DECIMAL(10, 2) NOT NULL,

    trajanje INT,

    CONSTRAINT chk\_trajanje\_nacin\_isporuke CHECK (trajanje > -1)

);

Ova tablica se naziva **nacini\_isporuke** i služi za pohranjivanje informacije o različitim načinima isporuke narudžbi. Sadrži sljedeće atribute:

* „id“ - cjelobrojni atribut koji se automatski povećava svaki put kad se napravi novi unos u tablicu. Osigurava da svaki način isporuke ima jedinstvenu identifikaciju.
* „naziv“ - tekstualni atribut (VARCHAR) s maksimalnom duljinom znakova od 255. Sadrži naziv načina isporuke.
* „opis“ - tekstualni atribut (TEXT) koji sadrži detaljniji opis načina isporuke.
* „cijena“ - decimalni atribut (DECIMAL(10,2)) koji sadržava cijenu načina isporuke na 2 decimale, do maksimalno 10 znamenki.
* „trajanje“ - cjelobrojni atribut koji sadržava maksimalno trajanje isporuke.

Također, tablica sadrži sljedeća ograničenja:

* NOT NULL – ograničenje za atribute „naziv“ i „cijena“, osigurava da atributi ne smiju biti prazni.
* CHECK – ograničenje za atribut „trajanje“, osigurava da trajanje isporuke ne smije biti negativno.

CREATE TABLE popusti (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    proizvod\_id INT NOT NULL,

    postotak\_popusta DECIMAL(5, 2) NOT NULL,

    datum\_pocetka DATE NOT NULL,

    datum\_zavrsetka DATE NOT NULL,

    CONSTRAINT chk\_postotak\_popusta\_popusti CHECK (postotak\_popusta BETWEEN 0 AND 100),

    CONSTRAINT chk\_datum\_pocetka\_and\_datum\_zavrsetka\_popusti CHECK (datum\_pocetka <= datum\_zavrsetka),

    CONSTRAINT fk\_proizvod\_popusti FOREIGN KEY (proizvod\_id) REFERENCES proizvodi(id)

);

Ova tablica se naziva **popusti** i služi za pohranjivanje informacije o popustima za određene proizvode, kao i njihove datume dostupnosti.

Sadrži sljedeće atribute:

* „id“ – cjelobrojni atribut koji osigurava jedinstvenu identifikaciju popusta, te se automatski povećava svakim novim unosom u tablicu.
* „proizvod\_id“ – strani ključ koji referencira na koji proizvod se popust primjenjuje.
* „postotak\_popusta“ – decimalni atribut koji sadrži podatak o iznosu popusta u postotcima, izražava se na dvije decimale, a maksimalno može imati 5 znamenki.
* „datum\_pocetka“ – podatak tipa DATE, koji sadrži datum početka primjenjivanja popusta.
* „datum\_zavrsetka“ – podatak tipa DATE, koji sadrži datum završetka primjenjivanja popusta.

Također, tablica sadrži sljedeća ograničenja:

* CHECK – ograničenje za atribut „postotak\_popusta“ koje osigurava da je postotak između 0 i 100. Također, isto ograničenje za atribute „datum\_pocetka“ i „datum\_zavrsetka“ koje osigurava da je datum početka manji ili jednak datumu završetka.
* FOREIGN KEY – ograničenje pomoću kojeg atribut „proizvod\_id“ referenciramo na atribut „id“ iz tablice **proizvodi*.***

## tablice Fran

A computer code with text

Description automatically generated

Tablica pod nazivom "povrati\_proizvoda" služi za pohranjivanje informacija o povratima proizvoda unutar sustava za upravljanje narudžbama.

Sadrži sljedeće atribute:

* „id“ – cjelobrojni atribut koji osigurava jedinstvenu identifikaciju svakog povrata proizvoda. Vrijednost se automatski povećava za svaki novi unos u tablicu.
* „stavka\_id“ - strani ključ koji referencira na stavku narudžbe iz tablice "stavke\_narudzbe", čime se povezuje povrat s konkretnom stavkom narudžbe.
* „datum\_povrata“ - atribut tipa DATE koji pamti datum kada je povrat proizvoda zabilježen u sustavu.
* „razlog“ – atribut tipa TEXT koji sadrži opis razloga povrata proizvoda.
* „status\_povrata“ – atribut tipa ENUM koji omogućava izbor jedne od tri unaprijed definirane vrijednosti:
  + „u obradi“ – označava da je povrat u procesu obrade.
  + „odbijeno“ – označava da je povrat odbijen.
  + „odobreno“ – označava da je povrat odobren.

Ograničenja tablice:

* FOREIGN KEY – ograničenje na atributu „stavka\_id“ osigurava referencijalni integritet tako što povezuje ovaj atribut s atributom „id“ iz tablice "stavke\_narudzbe", čime se garantira da se povrat može odnositi samo na postojeće stavke narudžbe.

A computer code with text

Description automatically generated with medium confidence

Tablica pod nazivom "pracenje\_isporuka" služi za pohranjivanje informacija o statusima isporuke pojedinih narudžbi, čime omogućuje praćenje tijeka isporuke unutar sustava.

Sadrži sljedeće atribute:

* „id“ – cjelobrojni atribut koji osigurava jedinstvenu identifikaciju svakog zapisa o praćenju isporuke. Vrijednost se automatski povećava sa svakim novim unosom u tablicu.
* „narudzba\_id“ – strani ključ koji povezuje zapis o isporuci s određenom narudžbom iz tablice "narudzbe", čime se uspostavlja veza između narudžbe i njezinog statusa isporuke.
* „status\_isporuke“ – atribut tipa ENUM koji omogućuje izbor jedne od tri moguće vrijednosti:
  + „u pripremi“ – označava da je narudžba u fazi pripreme za isporuku.
  + „poslano“ – označava da je narudžba napustila skladište ili distributivni centar.
  + „dostavljeno“ – označava da je narudžba uspješno dostavljena kupcu.
* „datum\_isporuke“ – atribut tipa DATE koji bilježi datum kada je isporuka dovršena. Ovaj atribut može ostati prazan (NULL) dok isporuka nije dovršena.

Ograničenja tablice:

* FOREIGN KEY – ograničenje na atributu „narudzba\_id“ osigurava referencijalni integritet tako što povezuje ovaj atribut s atributom „id“ iz tablice "narudzbe", čime se garantira da se statusi isporuke mogu odnositi samo na postojeće narudžbe.

A computer code with text

Description automatically generated

Tablica pod nazivom "podrska\_za\_korisnike" služi za pohranjivanje informacija o korisničkim upitima prema službi za podršku, čime omogućuje praćenje i upravljanje komunikacijom između korisnika i podrške.

Sadrži sljedeće atribute:

* „id“ – cjelobrojni atribut koji osigurava jedinstvenu identifikaciju svakog korisničkog upita. Vrijednost se automatski povećava sa svakim novim unosom u tablicu.
* „korisnik\_id“ – strani ključ koji povezuje upit s korisnikom iz tablice "korisnici", čime se uspostavlja veza između korisnika i njegovog zahtjeva za podrškom.
* „tema“ – atribut tipa VARCHAR(255) koji sadrži kratki opis ili naslov korisničkog upita, npr. "Problem s narudžbom" ili "Pitanje o povratu novca".
* „poruka“ – atribut tipa TEXT koji sadrži detaljan opis korisničkog upita ili problema.
* „status“ – atribut tipa ENUM koji omogućuje izbor jedne od dvije moguće vrijednosti:
  + „otvoreno“ – označava da korisnički upit još nije riješen.
  + „riješeno“ – označava da je korisnički upit uspješno riješen.
* „datum\_upita“ – atribut tipa DATE koji bilježi datum kada je korisnički upit zabilježen u sustavu.
* „datum\_odgovora“ – atribut tipa DATE koji bilježi datum kada je korisnički upit odgovoreno. Ovaj atribut može ostati prazan (NULL) dok odgovor nije zabilježen.

Ograničenja tablice:

* FOREIGN KEY – ograničenje na atributu „korisnik\_id“ osigurava referencijalni integritet tako što povezuje ovaj atribut s atributom „id“ iz tablice "korisnici", čime se jamči da se korisnički upit može povezati samo s postojećim korisnikom u sustavu.

## tablice Josip

CREATE TABLE kuponi (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    kod VARCHAR(50) UNIQUE NOT NULL,

    postotak\_popusta DECIMAL(5, 2) NOT NULL,

    datum\_pocetka DATE NOT NULL,

    datum\_zavrsetka DATE NOT NULL,

    max\_iskoristenja INT NOT NULL

);

Tablica „**kuponi**“ koristi se za pohranjivanje informacija o kuponima koji omogućuju popuste unutar sustava. Kuponi se mogu koristiti za smanjenje ukupnog iznosa narudžbe.

**Atributi tablice:**

* **id**
  + Cjelobrojni atribut koji se automatski povećava svaki put kada se unese novi redak u tablicu (AUTO\_INCREMENT).
  + Služi kao primarni ključ i osigurava jedinstvenu identifikaciju svakog kupona.
* **kod**
  + Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 50 znakova.
  + Mora biti jedinstven (UNIQUE) i ne može ostati prazan (NOT NULL).
  + Sadrži jedinstveni kod kupona koji korisnici unose prilikom korištenja.
* **postotak\_popusta**
  + Decimalni atribut (DECIMAL(5, 2)) koji označava iznos popusta u postotcima.
  + Mora biti unesen (NOT NULL).
  + Omogućava precizne vrijednosti popusta do dvije decimalne znamenke.
* **datum\_pocetka**
  + Atribut tipa DATE koji bilježi datum od kada je kupon važeći.
  + Mora biti unesen (NOT NULL).
* **datum\_zavrsetka**
  + Atribut tipa DATE koji bilježi datum do kada je kupon važeći.
  + Mora biti unesen (NOT NULL).
* **max\_iskoristenja**
  + Cjelobrojni atribut koji određuje maksimalan broj puta koliko se kupon može iskoristiti.
  + Mora biti unesen (NOT NULL).
* Ograničenja poput UNIQUE i NOT NULL osiguravaju da se unose samo valjani i jedinstveni podaci, dok atribut AUTO\_INCREMENT pojednostavljuje kreiranje novih redaka.
* Kombinacija datuma početka i završetka omogućava precizno definiranje razdoblja valjanosti kupona.

CREATE TABLE placanja (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    narudzba\_id INT NOT NULL,

    nacin\_placanja ENUM('kartica', 'pouzeće') NOT NULL,

    iznos DECIMAL(10, 2) NOT NULL,

    datum\_placanja DATE NOT NULL,

    FOREIGN KEY (narudzba\_id) REFERENCES narudzbe(id)

);

Tablica **„placanja“** koristi se za evidentiranje informacija o plaćanjima vezanima uz narudžbe u sustavu. Njena glavna svrha je pohranjivanje detalja o izvršenim plaćanjima, uključujući način plaćanja, iznos, te datum transakcije.

**Atributi tablice:**

* **id**
  + Cjelobrojni atribut koji se automatski povećava (AUTO\_INCREMENT) svaki put kada se unese novi red u tablicu.
  + Služi kao primarni ključ i osigurava jedinstvenu identifikaciju svakog plaćanja.
* **narudzba\_id**
  + Cjelobrojni atribut koji je strani ključ povezan s atributom „id“ u tablici **„narudzbe“**.
  + Ovaj atribut povezuje svako plaćanje s odgovarajućom narudžbom.
  + Mora biti unesen (NOT NULL), jer svako plaćanje mora biti vezano uz neku narudžbu.
* **nacin\_placanja**
  + Enumerirani atribut (ENUM) koji definira način na koji je plaćanje izvršeno.
  + Moguće vrijednosti su:
    - **'kartica'** – plaćanje kreditnom ili debitnom karticom.
    - **'pouzeće'** – plaćanje gotovinom prilikom preuzimanja.
  + Mora biti unesen (NOT NULL) kako bi se odredio način transakcije.
* **iznos**
  + Decimalni atribut (DECIMAL(10, 2)) koji prikazuje ukupni iznos plaćanja.
  + Mora biti unesen (NOT NULL).
  + Vrijednost je ograničena na 10 znamenki (uključujući dvije decimalne znamenke za cent ili lipu).
* **datum\_placanja**
  + Atribut tipa DATE koji bilježi datum kada je plaćanje izvršeno.
  + Mora biti unesen (NOT NULL) kako bi se osigurala evidencija o vremenu transakcije.
* Strani ključ **(FOREIGN KEY)** osigurava referencijalni integritet između tablica „placanja“ i „narudzbe“, čime se osigurava da plaćanje može postojati samo za postojeće narudžbe.
* Enumerirani tip podataka **(ENUM)** za atribut „nacin\_placanja“ pruža preciznost i ograničava unos na unaprijed definirane vrijednosti, čime se minimizira rizik od grešaka.

CREATE TABLE racuni (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    korisnik\_id INT NOT NULL,

    narudzba\_id INT NOT NULL,

    iznos DECIMAL(10, 2) NOT NULL,

    datum\_izdavanja DATE NOT NULL,

    FOREIGN KEY (korisnik\_id) REFERENCES korisnici(id),

    FOREIGN KEY (narudzba\_id) REFERENCES narudzbe(id)

);

Tablica **„racuni“** koristi se za pohranu podataka o izdanim računima korisnicima u sustavu. Svrha tablice je evidentiranje financijskih transakcija i povezanih narudžbi te osiguravanje transparentnog praćenja izdanih računa.

**Atributi tablice:**

* **id**
  + Cjelobrojni atribut koji se automatski povećava (AUTO\_INCREMENT) sa svakim novim unosom.
  + Služi kao primarni ključ i osigurava jedinstvenu identifikaciju svakog računa.
* **korisnik\_id**
  + Cjelobrojni atribut koji je strani ključ povezan s atributom **„id“** u tablici **„korisnici“**.
  + Povezuje račun s korisnikom kojem je izdan.
  + Mora biti unesen (NOT NULL), čime se osigurava da svaki račun ima pridruženog korisnika.
* **narudzba\_id**
  + Cjelobrojni atribut koji je strani ključ povezan s atributom **„id“** u tablici **„narudzbe“**.
  + Povezuje račun s pripadajućom narudžbom.
  + Mora biti unesen (NOT NULL), jer svaki račun mora biti vezan uz postojeću narudžbu.
* **iznos**
  + Decimalni atribut (DECIMAL(10, 2)) koji označava ukupni iznos računa.
  + Mora biti unesen (NOT NULL).
  + Vrijednost može imati do 10 znamenki (uključujući dvije decimalne znamenke za cent ili lipu).
* **datum\_izdavanja**
  + Atribut tipa DATE koji bilježi datum kada je račun izdan.
  + Mora biti unesen (NOT NULL), čime se osigurava evidencija vremena izdavanja računa.
* **korisnik\_id** osigurava da se račun može izdati samo postojećim korisnicima.
* **narudzba\_id** osigurava da je svaki račun povezan s valjanom narudžbom.

## tablice Leo

CREATE TABLE korisnici (

id INT AUTO\_INCREMENT PRIMARY KEY,

ime VARCHAR(255) NOT NULL,

prezime VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

lozinka VARCHAR(255) NOT NULL,

adresa TEXT,

grad VARCHAR(255),

telefon VARCHAR(20),

tip\_korisnika ENUM('kupac', 'admin') NOT NULL,

datum\_registracije DATE NOT NULL

);

Ova tablica se naziva „korisnici“ i sadrži sljedeće atribute:

* **„id“** Cjelobrojni atribut koji se automatski povećava (AUTO\_INCREMENT) svaki put kada se unese novi red u tablicu. Služi kao primarni ključ i osigurava jedinstvenu identifikaciju svakog korisnika.
* **„ime“** Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova. Sadrži ime korisnika.
* **„prezime“** Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova. Sadrži prezime korisnika.
* **„email“** Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova. Mora biti jedinstven (UNIQUE) i služi za identifikaciju korisnika u sustavu.
* **„lozinka“** Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova. Pohranjuje lozinku korisnika (obično hashiranu radi sigurnosti).
* **„adresa“** Tekstualni atribut (TEXT). Sadrži fizičku adresu korisnika (ulica, broj i slično).
* **„grad“** Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova. Sadrži naziv grada korisnika.
* **„telefon“** Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 20 znakova. Pohranjuje telefonski broj korisnika.
* **„tip\_korisnika“** Enumerirani atribut (ENUM) koji se dodaje u tablicu korisnika. Ima moguće vrijednosti „kupac“ i „admin“, te označava vrstu korisnika (npr. obični kupac ili administrator sustava). Ovaj atribut se automatski dodaje s vrijednošću „kupac“ prilikom unosa novog korisnika i ne mora biti ručno postavljen prilikom unosa.
* **„datum\_registracije“** Atribut tipa DATE. Pohranjuje datum kada je korisnik kreirao račun u sustavu.

**Ograničenja tablice:**

* **Primarni ključ:** Atribut „id“ postavljen je kao primarni ključ tablice, osiguravajući jedinstvenost svakog korisnika.
* **Jedinstveni email:** Atribut „email“ mora biti jedinstven kako bi se izbjeglo dupliciranje korisnika s istim mailom.
* **Obavezni atributi:** Atributi „ime“, „prezime“, „email“, „lozinka“, i „datum\_registracije“ moraju biti popunjeni prilikom unosa (NOT NULL). Atribut „tip\_korisnika“ se automatski postavlja na „kupac“ i ne mora biti eksplicitno unesen.

CREATE TABLE wishlist (

id INT AUTO\_INCREMENT PRIMARY KEY,

korisnik\_id INT NOT NULL,

proizvod\_id INT NOT NULL,

FOREIGN KEY (korisnik\_id) REFERENCES korisnici(id),

FOREIGN KEY (proizvod\_id) REFERENCES proizvodi(id)

);

Ova tablica se naziva „wishlist“ i sadrži sljedeće atribute:

* **„id“**Cjelobrojni atribut koji se automatski povećava (AUTO\_INCREMENT) svaki put kada se unese novi red u tablicu. Služi kao primarni ključ i osigurava jedinstvenu identifikaciju svakog unosa u „wishlist“.
* **„korisnik\_id“**Cjelobrojni atribut koji se odnosi na primarni ključ u tablici „korisnici“. Predstavlja referencu na korisnika koji je dodao proizvod u svoju „wishlist“.
* **„proizvod\_id“**Cjelobrojni atribut koji se odnosi na primarni ključ u tablici „proizvodi“. Predstavlja referencu na proizvod koji je korisnik dodao u svoju „wishlist“.
* **„grupa“**Tekstualni atribut (VARCHAR) koji se koristi za označavanje kategorije u kojoj se proizvod nalazi unutar „wishlist“-a. Dodan je kako bi korisnici mogli organizirati proizvode u različite grupe (npr. „Gaming oprema“, „Za posao“, „Hobi“). Ovo polje ima zadanu vrijednost „Bez grupe“, ukoliko korisnik ne specificira grupu prilikom dodavanja proizvoda u „wishlist“.

**Ograničenja tablice:**

* **Primarni ključ:** Atribut „id“ postavljen je kao primarni ključ, osiguravajući jedinstvenost svakog unosa u „wishlist“.
* **Vanjski ključ „korisnik\_id“:** Atribut „korisnik\_id“ povezan je s atributom „id“ iz tablice „korisnici“, čime se osigurava referencijalna integracija između korisnika i njihove „wishlist“.
* **Vanjski ključ „proizvod\_id“:** Atribut „proizvod\_id“ povezan je s atributom „id“ iz tablice „proizvodi“, čime se osigurava referencijalna integracija između proizvoda i „wishlist“.
* **Obavezno polje za grupu:** Atribut „grupa“ je dodan s zadanim vrijednostima, omogućujući korisnicima da grupiraju proizvode. Ukoliko korisnik ne odabere grupu, ona se automatski postavlja na „Bez grupe“.

**Opis funkcionalnosti:**

* Tablica „wishlist“ omogućava povezivanje korisnika s proizvodima koje su dodali u svoju „wishlist“. Svaki unos u tablicu predstavlja proizvod koji je određeni korisnik odabrao za pohranu u svoju „wishlist“. Ova tablica omogućuje korisnicima da organiziraju proizvode u različite grupe (kao što su „Gaming oprema“, „Za posao“, „Hobi“) za lakšu pretragu i upravljanje.
* Na primjer:  
  Korisnik s „korisnik\_id“ može imati više proizvoda u svojoj „wishlist“, dok svaki proizvod može biti dodan u „wishlist“ više korisnika. Relacije između korisnika i proizvoda omogućene su pomoću vanjskih ključeva.

**Napomena:**

* Korištenje vanjskih ključeva u tablicama "korisnici" i "proizvodi" osigurava referencijalnu integraciju, što znači da proizvod ne može biti dodan u "wishlist" ako ne postoji u tablici "proizvodi", a "wishlist" unos se ne može kreirati za korisnika koji ne postoji u tablici "korisnici". Također, atribut „grupa“ omogućava korisnicima da grupiraju proizvode u skladu s njihovim potrebama, a ukoliko grupa nije specificirana, koristi se zadana vrijednost „Bez grupe“.

CREATE TABLE preporuceni\_proizvodi (

id INT AUTO\_INCREMENT PRIMARY KEY,

korisnik\_id INT NOT NULL,

proizvod\_id INT NOT NULL,

razlog\_preporuke TEXT,

FOREIGN KEY (korisnik\_id) REFERENCES korisnici(id),

FOREIGN KEY (proizvod\_id) REFERENCES proizvodi(id)

);

Ova tablica se naziva „preporučeni\_proizvodi“ i sadrži sljedeće atribute:

**„id“**Cjelobrojni atribut koji se automatski povećava (AUTO\_INCREMENT) svaki put kada se unese novi red u tablicu. Služi kao primarni ključ i osigurava jedinstvenu identifikaciju svake preporuke proizvoda.

**„korisnik\_id“**Cjelobrojni atribut koji se odnosi na primarni ključ u tablici „korisnici“. Predstavlja korisnika koji je preporučio određeni proizvod.

**„proizvod\_id“**Cjelobrojni atribut koji se odnosi na primarni ključ u tablici „proizvodi“. Predstavlja proizvod koji je preporučen.

**„razlog\_preporuke“**Tekstualni atribut (TEXT) koji pohranjuje opis ili objašnjenje zašto je korisnik preporučio određeni proizvod.

**Ograničenja tablice:**

* **Primarni ključ:**Atribut „id“ postavljen je kao primarni ključ, osiguravajući jedinstvenost svake preporuke.
* **Vanjski ključ „korisnik\_id“:**Atribut „korisnik\_id“ povezan je s atributom „id“ iz tablice „korisnici“, čime se osigurava referencijalna integracija između korisnika i njihovih preporuka.
* **Vanjski ključ „proizvod\_id“:**Atribut „proizvod\_id“ povezan je s atributom „id“ iz tablice „proizvodi“, čime se osigurava referencijalna integracija između preporuka i proizvoda.

**Opis funkcionalnosti:**

Tablica „preporučeni\_proizvodi“ omogućava evidentiranje preporuka proizvoda koje su korisnici predložili, uključujući dodatne razloge zašto je određeni proizvod preporučen.

Na primjer:

* Jedan korisnik može preporučiti više proizvoda.
* Više korisnika može preporučiti isti proizvod, svaki s različitim razlozima.

**Primjeri upotrebe:**

* **Dodavanje preporuke proizvoda:**Kada korisnik preporuči proizvod, unosi se novi red s njegovim korisničkim ID-jem, ID-jem proizvoda i opisom razloga preporuke.
* **Prikaz preporuka:**Pomoću relacija između tablica moguće je prikazati popis preporučenih proizvoda za korisnika ili prikazati proizvode koje su korisnici najčešće preporučivali.

**Napomena:**

* Ograničenja stranih ključeva osiguravaju referencijalnu integraciju. Na primjer, preporuka ne može biti kreirana za korisnika ili proizvod koji ne postoji u bazi podataka.
* Polje „razlog\_preporuke“ omogućava fleksibilno objašnjenje razloga za preporuku. Ako nije potrebno, može ostati prazno.

# Upiti i pogledi

## Upiti i pogledi Loren

**Pregled svih proizvoda s kategorijama**

**CREATE VIEW** svi\_proizvodi\_s\_kategorijama **AS**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod,

p.cijena,

k.naziv **AS** kategorija,

p.kolicina\_na\_skladistu

**FROM** proizvodi p

**JOIN** kategorije\_proizvoda k **ON** p.kategorija\_id = k.id;

Pogled „svi\_proizvodi\_s\_kategorijama“ kombinira podatke iz tablica „proizvodi“ i „kategorije\_proizvoda“ na temelju zajedničkog atributa „kategorija\_id“. Ovaj pogled pruža način za pregled svih proizvoda zajedno s pripadajućim kategorijama. Stupci koji se prikazuju za ovaj pogled su „proizvod\_id“ koji prikazuje jedinstveni identifikator proizvoda iz tablice „proizvodi“, „proizvod“ koji sadrži naziv proizvoda, „cijena“ sa cijenom proizvoda, „kategorija“ sa nazivom kategorije kojoj proizvod pripada i „kolicina\_na\_skladistu“ koja ima trenutno dostupnu količinu proizvoda na skladištu. Pogled omogućava korisnicima pregled informacija o proizvodima i njihovim kategorijama kao da je riječ o jednoj tablici.

**Pregled proizvoda koji su trenutno na skladištu**

**CREATE VIEW** dostupni\_proizvodi **AS**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod,

p.kolicina\_na\_skladistu

**FROM** proizvodi p

**WHERE** p.kolicina\_na\_skladistu > 0;

Ovaj pogled pod imenom „dostupni\_proizvodi“ filtrira proizvode iz tablice „proizvodi“ koji imaju pozitivnu količinu na skladištu, odnosno proizvode koji su trenutno dostupni za prodaju. Atributi koji se prikazuju u ovom pogledu su „proizvod\_id“ koji sadrži jedinstveni identifikator proizvoda, „proizvod“ s nazivom proizvoda, „kolicina\_na\_skladistu“ s dostupnom količinom proizvoda na skladištu. Pogled se koristi za dohvat samo onih proizvoda koji su odmah dostupni za kupnju.

**Pregled svih recenzija s podacima o proizvodima i korisnicima**

**CREATE VIEW** recenzije\_s\_proizvodima\_i\_korisnicima **AS**

**SELECT**

r.id **AS** recenzija\_id,

p.naziv **AS** proizvod,

k.ime **AS** korisnik\_ime,

r.ocjena,

r.komentar,

r.datum\_recenzije

**FROM** recenzije\_proizvoda r

**JOIN** proizvodi p **ON** r.proizvod\_id = p.id

**JOIN** korisnici k **ON** r.korisnik\_id = k.id;

Pogled s nazivom „recenzije\_s\_proizvodima\_i\_korisnicima“ povezuje podatke iz tablica „recenzije\_proizvoda“, „proizvodi“ i „korisnici“ kako bi omogućio pregled recenzija zajedno s informacijama o proizvodima i korisnicima koji su ih napisali. Atributi koji se prikazuju u pogledu su „recenzija\_id“ s jedinstvenim identifikatorom recenzije, „proizvod“ sa nazivom proizvoda koji je recenziran, „ime“ s imenom korisnika koji je napisao recenziju, „ocjena“ s ocjenom proizvoda (vrijednost od 1 do 5), „komentar“ koji sadrži komentar korisnika o proizvodu i „datum\_recenzije“ koji prikazuje datum kada je recenzija napisana. Pogled se koristi za dohvat podataka o recenzijama proizvoda s detaljnim informacijama o proizvodima i korisnicima.

**Pregled najpopularnijih proizvoda (najveći broj recenzija)**

**CREATE VIEW** najpopularniji\_proizvodi **AS**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod,

**COUNT**(r.id) AS broj\_recenzija

**FROM** proizvodi p

**LEFT JOIN** recenzije\_proizvoda r **ON** p.id = r.proizvod\_id

**GROUP BY** p.id, p.naziv

**ORDER BY** broj\_recenzija **DESC**;

Pogled „najpopularniji\_proizvodi“ i prikazuje proizvode rangirane prema broju recenzija. Kombinira podatke iz tablica „proizvodi“ i „recenzije\_proizvoda“ koristeći LEFT JOIN kako bi uključio sve proizvode, čak i one bez recenzije. Stupci koji se prikazuju u pogledu su „proizvod\_id“ sa jedinstvenim identifikatorom proizvoda, „proizvod“ sa nazivom proizvoda, „broj\_recenzija“ koji sadži broj recenzija za svaki proizvod. Proizvodi su sortirani prema broju recenzija u opadajućem redoslijedu, tako da budu najpopularnijih proizvodi na samom vrhu pogleda.

**Pogled za proizvode koji su na popustu**

**CREATE VIEW** proizvodi\_na\_popustu **AS**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod\_naziv,

p.opis **AS** proizvod\_opis,

p.cijena **AS** originalna\_cijena,

(p.cijena - (p.cijena \* pop.postotak\_popusta / 100)) **AS** cijena\_sa\_popustom,

pop.postotak\_popusta,

pop.datum\_pocetka,

pop.datum\_zavrsetka

**FROM**

proizvodi p

**JOIN**

popusti pop ON p.id = pop.proizvod\_id

**WHERE**

**CURRENT\_DATE BETWEEN** pop.datum\_pocetka **AND** pop.datum\_zavrsetka

**ORDER BY**

pop.postotak\_popusta **DESC**;

Pogled „proizvodi\_na\_popustu“ prikazuje trenutačno aktivne popuste na proizvode, rangirane prema visini popusta. Pogled kombinira podatke iz tablica „proizvodi“ i „popusti“ koristeći „INNER JOIN“, čime uključuje samo one proizvode koji imaju aktivan popust u danom trenutku. Stupci koji se prikazuju u pogledu su „proizvod\_id“ sa jedinstvenim identifikatorom proizvoda, „proizvod“ sa nazivom proizvoda, „proizvod\_opis“ koji sadrži opis za svaki proizvod, „originalna\_cijena“ koja prikazuje cijenu bez popusta, „cijena\_sa\_popustom“ koja prikazuje cijenu s popustom, „postotak\_popusta“ s postotkom popusta i datumio za početak i završetak aktivnosti popusta. Filtrirano je tako da se prikazuju samo proizvodi koji su u datumu stvaranja između datuma početka i završetka popusta a zatim su sortirani u silaznom nizu po postotku.

**Upit proizvoda koji su dostupni na skladištu, imaju više od 10 recenzija i prosječnu ocjenu iznad 4**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod,

p.kolicina\_na\_skladistu,

**COUNT**(r.id) **AS** broj\_recenzija,

**AVG**(r.ocjena) **AS** prosjecna\_ocjena

**FROM** proizvodi p

**LEFT JOIN** recenzije\_proizvoda r **ON** p.id = r.proizvod\_id

**WHERE** p.kolicina\_na\_skladistu > 0

**GROUP BY** p.id, p.naziv, p.kolicina\_na\_skladistu

**HAVING** broj\_recenzija > 10 **AND** prosjecna\_ocjena > 4;

Ovaj upit dohvaća proizvode koji su dostupni na skladištu i koji imaju više od 10 recenzija s prosječnom ocjenom većom od 4. Podaci se kombiniraju iz tablica „proizvodi“ i „recenzije\_proizvoda“. Podaci se grupiraju prema svakom proizvodu, a rezultati su filtrirani s „HAVING“, gdje su uključeni samo proizvodi s više od 10 recenzija i prosječnom ocjenom većom od 4.

**Upit proizvoda s najvećom i najmanjom cijenom po kategoriji**

**SELECT**

k.naziv **AS** kategorija,

**MAX**(p.cijena) **AS** najskuplji\_proizvod,

**MIN**(p.cijena) **AS** najjeftiniji\_proizvod

**FROM** proizvodi p

**JOIN** kategorije\_proizvoda k **ON** p.kategorija\_id = k.id

**GROUP BY** k.naziv;

Ovaj upit pronalazi proizvode s najvećom i najmanjom cijenom unutar svake kategorije. Tablice „proizvodi“ i „kategorije\_proizvoda“ povezane su preko atributa „kategorija\_id“. Podaci se grupiraju po kategorijama, a funkcije agregacije „MAX“ i „MIN“ koriste se za izračunavanje maksimalnih i minimalnih cijena.

**Upit za korisnike koji su napisali najviše recenzija i njihove prosječne ocjene**

**SELECT**

k.id **AS** korisnik\_id,

k.ime,

k.prezime,

**COUNT**(r.id) **AS** broj\_recenzija,

**AVG**(r.ocjena) **AS** prosjecna\_ocjena

**FROM** korisnici k

**JOIN** recenzije\_proizvoda r **ON** k.id = r.korisnik\_id

**GROUP BY** k.id, k.ime, k.prezime

**ORDER BY** broj\_recenzija **DESC**

**LIMIT** 5;

Ovaj upit pronalazi korisnike koji su napisali najviše recenzija, zajedno s prosječnim ocjenama koje su dali. Podaci se kombiniraju iz tablica „korisnici“ i „recenzije\_proizvoda“. Podaci se grupiraju prema korisnicima, a rezultati su sortirani prema broju recenzija u silaznom redoslijedu. Prikazuju se podaci za najviše 5 korisnika pomoću ograničenja „LIMIT 5“.

**Upit za prosječne ocjene proizvoda na temelju recenzija**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod\_naziv,

p.opis **AS** proizvod\_opis,

**AVG**(r.ocjena) **AS** prosjecna\_ocjena,

**COUNT**(r.id) **AS** broj\_recenzija

**FROM**

proizvodi p

**LEFT JOIN**

recenzije\_proizvoda r **ON** p.id = r.proizvod\_id

**GROUP BY**

p.id, p.naziv, p.opis

**ORDER BY**

prosjecna\_ocjena **DESC**;

Ovaj upit pronalazi prosječnu ocjenu za sve proizvode obzirom na recenzije koje su korisnici napisali. Podaci se kombiniraju iz tablica „korisnici“ i „recenzije\_proizvoda“. „LEFT JOIN“ osigurava da se u rezultat uključe svi proizvodi, čak i oni koji nemaju nijednu recenziju. Podaci se grupiraju prema „p.id“ , „p.naziv“, „p.opis“, kako bi se za svaki proizvod odvojeno izračunala prosječna ocjena i broj recenzija. Prikazuju se podaci koji su sortirani silazno tako da imamo prosječno najbolje ocjenjene proizvode prikazane prve.

## Upiti i pogledi Morena

**Upit koji vraća sve narudžbe određenog korisnika, uključujući podatke o proizvodima i količinama u narudžbi**

SELECT

n.id AS narudzba\_id,

n.datum\_narudzbe,

n.status\_narudzbe,

n.ukupna\_cijena,

sn.proizvod\_id,

p.naziv AS proizvod\_naziv,

sn.kolicina AS proizvod\_kolicina

FROM narudzbe n

JOIN stavke\_narudzbe sn ON n.id = sn.narudzba\_id

JOIN proizvodi p ON sn.proizvod\_id = p.id

WHERE n.korisnik\_id = 1;

Spajanjem tablica „narudzbe“, „stavke\_narudzbe“ i „proizvodi“ preko join naredbe. Join naredba nam na temelju logičke povezanosti spaja tablice. Za povezivanje tablica „narudzbe“ i „stavke\_narudzbe“ smo odabrali zajednički atribut „narudzba\_id“ . Dobili smo tablicu sa redovima koji imaju zajednicku vrijednost id-a narudžbe. Nakon toga smo također preko joina spojili novonastalu tablicu sa tablicom „proizvodi “ preko zajedničkog atributa „proizvod\_id“. Kada smo spojili sve tri tablice, pretražujemo sve narudžbe zadanog korisnika (za primjer smo uzeli korisnika sa id=1) te na kraju ispisujemo samo atribute: narudzba\_id, datum\_narudzbe, status\_narudzbe, ukupna\_cijena, proizvodi, te naziv i kolicinu.

**Upit koji prikazuje sve proizvode u košarici određenog korisnika s količinama.**

SELECT

k.id AS kosarica\_id,

k.proizvod\_id,

p.naziv AS proizvod\_naziv,

k.kolicina AS proizvod\_kolicina,

p.cijena \* k.kolicina AS ukupna\_cijena

FROM kosarica k

JOIN proizvodi p ON k.proizvod\_id = p.id

WHERE k.korisnik\_id = 1;

Upit nam prikazuje odabrane kolicine i odabrane proizvode u košarici od korisnika. Ovaj upit je jednostavniji od prethodnog i u konačnici nam daj manje podataka nakon što se provede. Spajanjem pomoću naredbe join tablica kosarica i proizvod preko zajedničkog atributa „proizvod\_id“ ispisuje sve kolicine i vrste proizvoda odabrane od korisnika čiji je id, i našem primjeru 1.

**Upita za provjeru trenutnog statusa narudžbe i isporuke na temelju ID-a narudžbe**

SELECT n.id AS narudzba\_id,

n.status\_narudzbe,

p.status\_isporuke,

p.datum\_isporuke,

n.datum\_narudzbe,

n.ukupna\_cijena,

n.nacin\_isporuke\_id

FROM narudzbe n

LEFT JOIN pracenje\_isporuka p

ON n.id = p.narudzba\_id

WHERE n.id =1 ;

Ovim upitom želimo pratiti stanje pošiljke. Spajanjem tablica „narudzbe“ i „pracenje\_isporuka“ po atributu „narudzba\_id“ upišemo id tražene narudžbe i kao rješenje dobijemo uvid u status i datum narudžbe, status i datum isporuke te ukupno cijenu i jedan od četiri načina isporuke koji je korisnik odabrao prilikom kreiranja same narudžbe. Koristili smo naredbu left join za spajanje tablica jer nam je bitno da se ne vraćaju samo oni redovi iz obje tablice koji imaju odgovarajući zapis prema uvjetu za spajanje već da nam se kao rješenje u tablici prikažu svi redovi iz lijeve tablice (prve tablice u upitu), čak i ako nema podudaranja u desnoj tablici.Ako nema podudaranja u desnoj tablici, za te će retke vrijednosti iz desne tablice biti NULL tj. ako nema podudaranja, taj redak se ne uključuje u rezultat.

**Upit za prikaz najprodavanijih proizvoda koji se nalaze u stavke\_narudzbe**

SELECT p.id AS proizvod\_id,

p.naziv AS naziv\_proizvoda,

SUM(sn.kolicina) AS ukupno\_prodano

FROM stavke\_narudzbe sn

JOIN proizvodi p ON sn.proizvod\_id = p.id

GROUP BY p.id, p.naziv

ORDER BY ukupno\_prodano DESC

LIMIT 5;

Upit za prikaz najprodavanijih proizvoda postigli smo spajanjem tablica „stavke\_narudzbe“ i „proizvodi“ naredbom join preko zajedničkog atributa „proizvod\_id“. Potom smo grupirali id od proizvoda i naziv te smo poredali silazno obzirom na sumiranje prema prodanoj kolicini. Sumirano po kolicini smo aliasom nazvali „ukupno\_prodano“ tako da će nam to biti naziv novonastalog atributa u tablici. Selekcijom smo novi stupac izdvojili zajedno sa „proizvod\_id“ i „naziv\_proizvoda“ koji će nam se prikazati u tablici rješenja. Limitirali smo tablicu rješenja na, u ovom primjeru, 5 najprodavanijih proizvoda.

**Pogled koji prikazuje stavke u košarici korsinika i ukupnu cijenu svih odabranih artikala:**

CREATE VIEW pogled\_kosarica\_korisnika AS

SELECT

k.id AS kosarica\_id,

k.korisnik\_id,

k.proizvod\_id,

p.naziv AS proizvod\_naziv,

p.cijena AS proizvod\_cijena,

k.kolicina AS proizvod\_kolicina,

(k.kolicina \* p.cijena) AS ukupna\_cijena

FROM

kosarica k

JOIN

proizvodi p ON k.proizvod\_id = p.id;

Pogled „pogled\_kosarica\_korisnika“ nam prikazuje podatke o sadržaju košarice zajednos a informacijama o proizvodima, ukupnim troškom svakog proizvoda te podatke o korisniku. Prvo preko join naredbe spajamo tablice „kosarica“ i „proizvodi“ po zajednickom atributu „proizvod\_id“ i onda selektiramo podatke koje želimo ispisati nakon prosljeđenog zahtjeva za ovim upitom, a to su: id kosarice, korisnika i proizvoda, naziv proizvoda te cijena i kolicina, te kao dodatni stupac pomnožimo količinu i cijenu određenog artikla. Kako nam je to novonastali stupac, preko aliasa mu dodjeljujemo ime: „ukupna\_cijena“.

**Pogled prikazuje sve narudžbe određenog korisnika zajedno s detaljima o stavkama narudžbi, uključujući količinu proizvoda i ukupnu cijenu.**

CREATE VIEW pogled\_narudzbi\_korisnika AS

SELECT

n.id AS narudzba\_id,

n.datum\_narudzbe,

n.status\_narudzbe,

n.ukupna\_cijena,

n.nacin\_isporuke\_id,

n.kupon\_id,

s.proizvod\_id,

p.naziv AS proizvod\_naziv,

s.kolicina AS proizvod\_kolicina,

p.cijena AS proizvod\_cijena,

(s.kolicina \* p.cijena) AS ukupna\_cijena\_stavke

FROM

narudzbe n

JOIN

stavke\_narudzbe s ON n.id = s.narudzba\_id

JOIN

proizvodi p ON s.proizvod\_id = p.id;

Ovaj pogled smo kreirali kako bismo vidjeli sve narudžbe jednog korisnika koje je ikad napravio. Naredbom join smo spojili tablice „narudzbe“ sa tablicom „stavke\_narudzbe“ preko zajedničkog id „narudzba\_id“ te smo im još dodali i treću tablicu „proizvodi“ s kojom smo ih spojili preko zajedničkog „proizvod\_id“. Kad smo povezali željene tablice, selektirali smo željene stupce: id, status i datum narudžbi, korištene kupone i ukupnu cijenu, naziv proizvoda, id i pojedinačnu cijenu, te način isporuke i ukupnu cijenu svih stavki. Ukupnu cijenu svih stavki smo dobili množenjem količine određenog proizvoda sa njegovom cijenom.

## Upiti i pogledi Bruno

Upiti

Prikaži sve narudžbe gdje se narudžba isporučila u manje od 3 dana (uključujući i preuzimanje)

SELECT \*

  FROM narudzbe

    INNER JOIN (SELECT \*

         FROM nacini\_isporuke

                 WHERE trajanje < 3) AS temp\_nacini\_isporuke

    ON narudzbe.nacin\_isporuke\_id = temp\_nacini\_isporuke.id

    HAVING narudzbe.status\_narudzbe = "dostavljeno";

Prikaži sve proizvode koji su bili sniženi 20% ili više u zadnjih godinu dana

SELECT proizvodi.\*, temp\_popusti.postotak\_popusta, temp\_popusti.datum\_pocetka, temp\_popusti.datum\_zavrsetka

  FROM proizvodi

    INNER JOIN (SELECT \*

          FROM popusti

          WHERE (postotak\_popusta >= 20 AND datum\_zavrsetka > CURDATE() - INTERVAL 1 YEAR)) AS temp\_popusti

    ON proizvodi.id = temp\_popusti.proizvod\_id;

Prikaži broj kupaca koji su preuzeli svoju narudžbu u trgovini

SELECT COUNT(\*) AS broj\_kupaca

  FROM narudzbe

    INNER JOIN (SELECT id

          FROM nacini\_isporuke

                    WHERE trajanje = 0) AS temp\_nacini\_isporuke

        ON narudzbe.nacin\_isporuke\_id = temp\_nacini\_isporuke.id

                GROUP BY temp\_nacini\_isporuke.id;

Pogledi

Napravi pogled koji prikazuje sve proizvode sa sniženom cijenom kad se primjeni popust

CREATE VIEW proizvodi\_sa\_snizenom\_cijenom AS

SELECT proizvodi.\*, ROUND((cijena \* (1 - popusti.postotak\_popusta / 100)), 2) AS snizena\_cijena

  FROM popusti

    INNER JOIN proizvodi ON popusti.proizvod\_id = proizvodi.id;

SELECT \* FROM proizvodi\_sa\_snizenom\_cijenom;

Napravi pogled koji prikazuje trenutno aktivne popuste na proizvodima

CREATE VIEW proizvodi\_sa\_aktivnim\_popustom AS

SELECT proizvodi.\*, popusti.postotak\_popusta, popusti.datum\_pocetka, popusti.datum\_zavrsetka

  FROM popusti

    INNER JOIN proizvodi ON proizvodi.id = popusti.proizvod\_id

    HAVING datum\_pocetka <= CURDATE() AND datum\_zavrsetka >> CURDATE();

SELECT \* FROM proizvodi\_sa\_aktivnim\_popustom;

## Upiti i pogledi Fran

A computer code with text

Description automatically generated

Pogled pod nazivom "proizvodi\_sa\_statusom\_popusta" služi za generiranje prikaza proizvoda zajedno s informacijama o popustima, uključujući trenutni status popusta i izračunatu cijenu s uključenim popustom (ako je primjenjiv).

Detalji pogleda:

* „proizvod\_id“ – prikazuje jedinstveni identifikator proizvoda iz tablice "proizvodi".
* „proizvod\_naziv“ – prikazuje naziv proizvoda iz tablice "proizvodi".
* „originalna\_cijena“ – prikazuje izvornu cijenu proizvoda bez popusta.
* „cijena\_sa\_popustom“ – izračunava cijenu proizvoda nakon primjene popusta. Ako proizvod nema važeći popust, prikazuje se izvorna cijena. Funkcija COALESCE koristi se kako bi se osiguralo da cijena ostane originalna ako popust nije definiran.
* „postotak\_popusta“ – prikazuje postotak popusta iz tablice "popusti". Ako popust nije definiran, vrijednost je NULL.
* „status\_popusta“ – koristi CASE izraz za određivanje statusa popusta:
  + „Aktivan“ – popust je trenutno važeći, tj. trenutni datum i vrijeme (CURRENT\_TIMESTAMP) nalaze se unutar raspona definiranih atributima „datum\_pocetka“ i „datum\_zavrsetka“ iz tablice "popusti".
  + „Neaktivan“ – popust nije trenutno važeći (bilo da je istekao ili još nije počeo).

A screenshot of a computer code

Description automatically generated

Pogled pod nazivom "narudzbe\_sa\_statusom\_isporuke" omogućuje pregled narudžbi zajedno s njihovim statusima isporuke, čime se pruža bolji uvid u proces narudžbi i isporuka.

Detalji pogleda:

* „narudzba\_id“ – prikazuje jedinstveni identifikator narudžbe iz tablice "narudzbe".
* „datum\_narudzbe“ – prikazuje datum kada je narudžba kreirana.
* „status\_narudzbe“ – prikazuje trenutni status narudžbe iz tablice "narudzbe" (npr. "otkazano", "potvrđeno" ili "plaćeno").
* „status\_isporuke“ – prikazuje status isporuke narudžbe iz tablice "pracenje\_isporuka", s vrijednostima poput:
  + „u pripremi“ – narudžba je u procesu pripreme.
  + „poslano“ – narudžba je otpremljena.
  + „dostavljeno“ – narudžba je dostavljena.
* „datum\_statusa“ – prikazuje datum koji odgovara statusu isporuke iz tablice "pracenje\_isporuka" (atribut „datum\_isporuke“). Ako isporuka još nije evidentirana, vrijednost može biti NULL.

A screenshot of a computer code

Description automatically generated

Pogled pod nazivom "proizvodi\_po\_kategorijama\_sa\_ocjenama" omogućuje prikaz proizvoda grupiranih prema kategorijama, uz dodatne informacije o prosječnoj ocjeni i broju recenzija za svaki proizvod.

Detalji pogleda:

* „kategorija“ – prikazuje naziv kategorije kojoj proizvod pripada, dohvaćen iz tablice "kategorije\_proizvoda".
* „proizvod\_id“ – prikazuje jedinstveni identifikator proizvoda iz tablice "proizvodi".
* „proizvod\_naziv“ – prikazuje naziv proizvoda iz tablice "proizvodi".
* „prosjecna\_ocjena“ – prikazuje prosječnu ocjenu proizvoda, izračunatu na temelju svih recenzija povezanih s tim proizvodom u tablici "recenzije\_proizvoda". Ako proizvod nema recenzija, vrijednost će biti NULL.
* „broj\_recenzija“ – prikazuje ukupan broj recenzija koje su napisane za određeni proizvod, temeljem podataka iz tablice "recenzije\_proizvoda".

## Upiti i pogledi Josip

A computer screen shot of a computer code

Description automatically generated

* Prikazuje informacije o narudžbama koje su ostvarile popust korištenjem kupona.
* Dohvaća ID narudžbe, ID računa, kod kupona, postotak popusta, originalni iznos računa, izračunati iznos popusta, i konačni iznos nakon popusta.
* Sortira podatke prema ID-u narudžbe.

A computer screen shot of a code

Description automatically generated

* Prikazuje detalje uplata za svaku narudžbu.
* Dohvaća ID narudžbe, broj uplata, ukupni iznos uplata, originalni iznos računa, i preostali iznos za plaćanje.
* Grupira podatke po narudžbama i računa agregate poput broja uplata i ukupno plaćenog iznosa.

A computer screen with text and images

Description automatically generated

* Prikazuje račune povezane s kuponima.
* Dohvaća ID računa, ukupni iznos računa, kuponski kod, postotak popusta, i iznos računa nakon primjene popusta.
* Sortira račune prema datumu izdavanja.

A computer screen shot of white text

Description automatically generated

* Prikazuje agregirane podatke o plaćanjima grupirane prema načinu plaćanja.
* Dohvaća način plaćanja, broj uplata, i ukupan iznos uplata za svaki način plaćanja.
* Sortira rezultate prema ukupnom iznosu uplata (od najvećeg prema najmanjem).

## Upiti i pogledi Leo

CREATE VIEW profil\_korisnika AS

SELECT

id AS korisnik\_id,

ime,

prezime,

email,

adresa,

grad,

telefon,

datum\_registracije,

tip\_korisnika

FROM korisnici;

**Pogled: Profil korisnika**

* Pogled „profil\_korisnika“ koristi se za dohvaćanje osnovnih informacija o korisnicima iz tablice „korisnici“. Služi za pojednostavljenje upita koji prikazuju korisničke profile, omogućujući aplikaciji da lako prikaže informacije poput imena, prezimena, email adrese, adrese stanovanja, grada, telefona i datuma registracije.
* **Zašto se koristi ovaj pogled?**Umjesto ponavljanja složenih upita u različitim dijelovima aplikacije, pogled pruža strukturiran i uvijek ažuran prikaz korisničkih podataka. To čini aplikaciju efikasnijom i čitljivijom.
* **Specifične prednosti:**Podaci se preuzimaju direktno u obliku koji je potreban aplikaciji, smanjujući potrebu za dodatnim manipulacijama na strani aplikacijskog koda.

-- Pogled: Preporuke drugih (Leo)

CREATE VIEW preporuke\_drugih AS

SELECT

k.id AS korisnik\_id,

k.ime,

k.prezime,

p.naziv AS proizvod\_naziv,

pp.razlog\_preporuke

FROM preporuceni\_proizvodi pp

JOIN korisnici k ON pp.korisnik\_id = k.id

JOIN proizvodi p ON pp.proizvod\_id = p.id;

**Pogled: Preporuke drugih**Pogled „preporuke\_drugih“ koristi se za dohvaćanje informacija o proizvodima koje su preporučili drugi korisnici u aplikaciji. Pomoću ovog pogleda aplikacija može jednostavno i efikasno prikazati proizvode koje su drugi korisnici preporučili, zajedno s osnovnim informacijama o korisniku koji je dao preporuku, kao što su ime, prezime i razlog preporuke.

**Zašto se koristi ovaj pogled?**Korištenje ovog pogleda omogućava jednostavno i brzo dohvaćanje svih potrebnih podataka o preporukama drugih korisnika bez potrebe za ponavljanjem složenih upita u različitim dijelovima aplikacije. Umjesto da svaki put radimo join između tablica korisnici, preporučeni\_proizvodi i proizvodi u aplikaciji, ovaj pogled centralizira upit i omogućava jednostavno preuzimanje relevantnih podataka u obliku koji je direktno potreban.

**Specifične prednosti:**

* **Pojednostavljenje upita:** Ovaj pogled omogućava aplikaciji da dobije sve potrebne informacije o preporukama drugih korisnika u samo jednom upitu, što poboljšava efikasnost i smanjuje dupliciranje koda.
* **Brzi pristup podacima:** Podaci o korisnicima i njihovim preporukama dostupni su odmah, što omogućava brže učitavanje stranica i bolji korisnički doživljaj.
* **Ažurnost podataka:** Pogled je uvijek ažuran jer se podaci iz baze povlače u trenutnom stanju, što znači da preporuke drugih korisnika neće biti zastarjele.

-- Pogled: Popularni proizvodi (najviše puta dodati u wishlist) (Leo)

CREATE OR REPLACE VIEW popularni\_proizvodi AS

SELECT

p.id AS proizvod\_id,

p.naziv,

COUNT(w.proizvod\_id) AS broj\_dodavanja

FROM proizvodi p

LEFT JOIN wishlist w ON p.id = w.proizvod\_id

GROUP BY p.id, p.naziv

ORDER BY broj\_dodavanja DESC;

**Pogled: Popularni proizvodi**

* Pogled „popularni\_proizvodi“ koristi se za identifikaciju proizvoda koji su najpopularniji među korisnicima prema broju dodavanja na wishlist. Kombinira podatke iz tablica „proizvodi“ i „wishlist“ pomoću LEFT JOIN.
* **Zašto koristiti LEFT JOIN?**LEFT JOIN osigurava da svi proizvodi iz tablice „proizvodi“ budu uključeni u rezultat, čak i ako nijedan korisnik nije dodao taj proizvod na wishlist. Ovo je korisno za pružanje sveobuhvatne analize popularnosti proizvoda.
* **Što omogućava ovaj pogled?**Funkcija COUNT broji koliko puta je svaki proizvod dodat na wishlist, dok GROUP BY grupira rezultate po proizvodima. ORDER BY sortira proizvode po popularnosti u opadajućem redoslijedu, tako da se najpopularniji proizvodi nalaze na vrhu.
* **Prednosti:**Pogled omogućava brzi uvid u popularnost proizvoda i olakšava donošenje odluka, kao što su kreiranje marketinških kampanja ili ažuriranje ponude na temelju popularnosti.

-- Pogled: Narudžbe Korisnika (Leo)

CREATE VIEW narudzbe\_korisnika AS

SELECT

n.id AS narudzba\_id,

n.datum\_narudzbe,

n.status\_narudzbe AS status,

s.proizvod\_id,

p.naziv AS proizvod\_naziv,

s.kolicina,

s.kolicina \* p.cijena AS ukupna\_cijena\_proizvoda,

n.korisnik\_id

FROM narudzbe n

JOIN stavke\_narudzbe s ON n.id = s.narudzba\_id

JOIN proizvodi p ON s.proizvod\_id = p.id;

**Pogled: Narudžbe korisnika**

* Pogled „narudzbe\_korisnika“ omogućava pregled narudžbi korisnika s svim relevantnim detaljima, uključujući povezane proizvode i njihove troškove. Povezuje tablice „narudzbe“, „stavke\_narudzbe“ i „proizvodi“ pomoću INNER JOIN.
* **Zašto koristiti INNER JOIN?**INNER JOIN osigurava da se u rezultate uključuju samo narudžbe koje imaju validne stavke i proizvode, čime se izbjegava prikazivanje nepotpunih ili nevažećih podataka.
* **Kako funkcionira ovaj pogled?**Informacije o narudžbama, kao što su datum, status i ukupni troškovi, dolaze iz tablice „narudzbe“.  
  Pojedinačne stavke narudžbi povezane su preko tablice „stavke\_narudzbe“, dok su podaci o proizvodima dohvaćeni iz tablice „proizvodi“.  
  Ukupna cijena svake stavke računa se dinamički množenjem količine proizvoda s njegovom cijenom.
* **Prednosti:**Pogled pruža sveobuhvatne informacije o narudžbama korisnika, što omogućava transparentan pregled narudžbi unutar korisničkog profila.

-- Upit: Provjera popularnosti proizvoda (koji proizvodi su najčešće dodati u wishlist) (Leo)

SELECT

p.id AS proizvod\_id,

p.naziv AS proizvod\_naziv,

COUNT(w.proizvod\_id) AS broj\_dodavanja

FROM proizvodi p

LEFT JOIN wishlist w ON p.id = w.proizvod\_id

GROUP BY p.id, p.naziv

ORDER BY broj\_dodavanja DESC

LIMIT 5;

**Upit: Provjera popularnosti proizvoda (koji proizvodi su najčešće dodani u wishlist)**

* + **Opis:**Ovaj upit koristi se za identifikaciju proizvoda koji su najpopularniji među korisnicima na temelju broja dodavanja u wishlist. Upit kombinira podatke iz tablica „proizvodi“ i „wishlist“ pomoću LEFT JOIN kako bi uključio sve proizvode, čak i one koji nisu dodani na wishlist.
  + **Detalji implementacije:**LEFT JOIN omogućava da svi proizvodi budu uključeni u rezultat, čak i oni koji nisu dodani u wishlist.  
    Funkcija COUNT računa broj puta koliko je svaki proizvod dodan u wishlist.  
    GROUP BY osigurava grupiranje rezultata prema jedinstvenim proizvodima.  
    ORDER BY sortira proizvode prema broju dodavanja u opadajućem redoslijedu.  
    LIMIT 5 ograničava rezultat na prvih 5 najpopularnijih proizvoda.

-- Upit: Provjera korisnika sa najviše narudžbi (Leo)

SELECT

k.id AS korisnik\_id,

k.ime,

k.prezime,

COUNT(n.id) AS broj\_narudzbi

FROM korisnici k

JOIN narudzbe n ON k.id = n.korisnik\_id

GROUP BY k.id, k.ime, k.prezime

ORDER BY broj\_narudzbi DESC

LIMIT 5; -- Prikazuje prvih 5 korisnika sa najviše narudžbi

**Upit: Provjera korisnika sa najviše narudžbi**

**Opis:**Ovaj upit omogućava identifikaciju korisnika koji su kreirali najviše narudžbi. Kombinuje podatke iz tablica **„korisnici“** i **„narudzbe“** pomoću **INNER JOIN** kako bi obezbedio da se uključe samo korisnici sa valjanim narudžbama.

* **Detalji implementacije:**
  + **INNER JOIN** osigurava povezivanje samo onih korisnika koji imaju najmanje jednu narudžbu.
  + Funkcija **COUNT** broji broj narudžbi za svakog korisnika.
  + **GROUP BY** grupiše rezultate prema jedinstvenim korisnicima.
  + **ORDER BY** sortira rezultate prema broju narudžbi u padajućem redoslijedu.
  + **LIMIT 5** ograničava prikaz na prvih 5 korisnika sa najviše narudžbi.

-- Upit: Ukupna zarada po korisnicima na osnovu narudžbi(Leo)

SELECT

k.id AS korisnik\_id,

k.ime,

k.prezime,

SUM(n.ukupna\_cijena) AS ukupna\_zarada

FROM korisnici k

JOIN narudzbe n ON k.id = n.korisnik\_id

GROUP BY k.id, k.ime, k.prezime

ORDER BY ukupna\_zarada DESC;

**Upit: Ukupna zarada po korisnicima na temelju narudžbi**

* + **Opis:**Ovaj upit koristi se za izračunavanje ukupne zarade koju je svaki korisnik generirao na temelju svojih narudžbi. Kombinira podatke iz tablica „korisnici“ i „narudzbe“ pomoću INNER JOIN.
  + **Detalji implementacije:**INNER JOIN osigurava da se uzimaju u obzir samo korisnici koji imaju valjane narudžbe.  
    Funkcija SUM računa ukupnu vrijednost svih narudžbi za svakog korisnika.  
    GROUP BY grupira rezultate prema jedinstvenim korisnicima.  
    ORDER BY sortira rezultate prema ukupnoj zaradi u opadajućem redoslijedu.

# 

# Funkcije

## Funkcije Loren

**Funkcija koja vraća ukupnu vrijednost svih proizvoda na skladištu**

DELIMITER //

**CREATE FUNCTION** UkupnaVrijednostSkladista ()

**RETURNS DECIMAL**(15,2)

**DETERMINISTIC**

**BEGIN**

**DECLARE** ukupna\_vrijednost **DECIMAL**(15,2);

**SELECT** **SUM**(kolicina\_na\_skladistu \* cijena)

**INTO** ukupna\_vrijednost

**FROM** proizvodi;

**RETURN** ukupna\_vrijednost;

**END** //

DELIMITER ;

Funkcija „UkupnaVrijednostSkladista“ izračunava ukupnu vrijednost svih proizvoda na skladištu tako da množi cijenu svakog proizvoda s količinom na skladištu i zbroji sve rezultate. Prvo funkcija deklarira varijablu „ukupna\_vrijednost“ za spremanje ukupne vrijednosti. Zatim izračunava zbroj vrijednosti svih proizvoda koristeći „SUM“. I na kraju vraća rezultat tipa „DECIMAL“ s dvije decimale.

**Funkcija koja provjerava dostupnost proizvoda na skladištu**

DELIMITER //

**CREATE FUNCTION** provjeri\_dostupnost(

p\_proizvod\_id **INT**,

p\_kolicina **INT**

)

**RETURNS BOOLEAN**

**DETERMINISTIC**

**BEGIN**

**DECLARE** v\_kolicina\_na\_skladistu **INT**;

**SELECT** kolicina\_na\_skladistu

**INTO** v\_kolicina\_na\_skladistu

**FROM** proizvodi

**WHERE** id = p\_proizvod\_id;

**IF** v\_kolicina\_na\_skladistu **IS NULL THEN**

**RETURN FALSE**;

**END IF**;

**IF** p\_kolicina > v\_kolicina\_na\_skladistu **THEN**

**RETURN FALSE;**

**END IF**;

**RETURN TRUE**;

**END** //

DELIMITER ;

Funkcija „provjeri\_dostupnost“ služi za provjeru dostupnosti određenog proizvoda na skladištu. Prima dva parametra, „p\_proizvod\_id“, što je jedinstveni identifikator proizvoda, i „p\_kolicina“, što predstavlja željenu količinu tog proizvoda. Funkcija koristi tablicu **„**proizvodi“ kako bi dohvatila trenutnu količinu proizvoda na skladištu putem stupca „kolicina\_na\_skladistu“. Nakon što se podaci preuzmu, funkcija prvo provjerava postoji li proizvod s tim „id“. Ako proizvod ne postoji, funkcija vraća vrijednost „FALSE“, što označava da proizvod nije dostupan. Zatim funkcija uspoređuje traženu količinu s dostupnom količinom na skladištu. Ako tražena količina premašuje dostupnu količinu, funkcija također vraća „FALSE“, inače vraća „TRUE“, što znači da je tražena količina proizvoda dostupna na skladištu.

**Funkcija koja izračunava prosječne ocjene proizvoda na temelju recenzija.**

DELIMITER //

**CREATE FUNCTION** prosjecna\_ocjena\_proizvoda(

p\_proizvod\_id **INT**

)

**RETURNS DECIMAL**(3, 2)

**DETERMINISTIC**

**BEGIN**

**DECLARE** v\_prosjecna\_ocjena **DECIMAL**(3, 2);

**SELECT** **AVG**(ocjena)

**INTO** v\_prosjecna\_ocjena

**FROM** recenzije\_proizvoda

**WHERE** proizvod\_id = p\_proizvod\_id;

**RETURN** v\_prosjecna\_ocjena

**END** //

DELIMITER ;

Funkcija „prosjecna\_ocjena\_proizvoda“ izračunava prosječnu ocjenu proizvoda na temelju recenzija iz tablice „recenzije\_proizvoda“. Funkcija prima „p\_proizvod\_id“, koji predstavlja jedinstveni identifikator proizvoda čije recenzije se analiziraju. Unutar funkcije se deklarira varijabla „v\_prosjecna\_ocjena“ tipa „DECIMAL(3, 2)“, u koju se sprema rezultat izračuna prosječne ocjene, dobiven iz stupca „ocjena“ iz tablice „recenzije\_proizvoda“**.** Računa se prosjek svih ocjena za proizvod čiji identifikator odgovara „p\_proizvod\_id“. Nakon izvršenja upita, funkcija vraća izračunatu vrijednost, a ukoliko ne postoji nijedna recenzija vraća „NULL“.

## Funkcije Morena

**Funkcija koja vraća broj proizvoda u košarici određenog korisnika. Broj proizvoda je zbroj svih količina proizvoda u košarici**

DELIMITER //

CREATE FUNCTION brojProizvodaUKosarici(korisnikId INT)

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE brojProizvoda INT DEFAULT 0;

SELECT SUM(kolicina) INTO brojProizvoda

FROM kosarica

WHERE korisnik\_id = korisnikId;

RETURN brojProizvoda;

END//

DELIMITER ;

SELECT brojProizvodaUKosarici(1)

Funkcija „brojProizvodaUKosarici“ služi nam za izračun ukupnog broja proizvoda u košarici temeljem id korisnika. Kao ulaznu vrijednost prima id traženog korisnika a kao output vraća cjelobrojnu vrijednost koja vraća ukupan broj stavki u košarici. Deklariramo lokalnu varijablu „brojProizvoda“ i postavimo ga na nulu. Iz tablice kosarica gdje nam se podudaraju id-evi korisnika selektiramo kolicinu odnosno, pomoću naredbe SUM sumiramo sve kolicine oznacenih proizvoda u košarici. Kao što je prethodno navedeno, rješenje ove funkcije je cijelobrojna vrijednost kolicine svih elemenata. Za primjer smo naredbom SELECT pozvali funkciju za korisnika sa id-em 1.

**Funkcija izračunava ukupnu cijenu narudžbe na temelju stavki narudžbe. Uzimajući u obzir količinu proizvoda i cijenu svakog proizvoda, funkcija vraća ukupnu cijenu za zadanu narudžbu.**

DELIMITER //

CREATE FUNCTION izracunajUkupnuCijenuNarudzbe(narudzbaId INT)

RETURNS DECIMAL(10, 2)

DETERMINISTIC

BEGIN

DECLARE ukupnaCijena DECIMAL(10, 2) DEFAULT 0.00;

SELECT SUM(p.cijena \* sn.kolicina) INTO ukupnaCijena

FROM stavke\_narudzbe sn

JOIN proizvodi p ON sn.proizvod\_id = p.id

WHERE sn.narudzba\_id = narudzbaId;

RETURN ukupnaCijena;

END//

DELIMITER ;

SELECT izracunajUkupnuCijenuNarudzbe(1);

Funkcija „izracunajUkupnuCijenuNarudzbe“ nam izračunava ukupan iznos navedenih stavki. Kao input prima vrijednos id-a narudzbe a kao output vraća decimalni broj sa ukupno osam znamenaka od kojih su dvije predodređene za desetinku i stotinku decimalnog zapisa. U početku funkcije deklariramo lokalnu varijablu „ukuonaCijena“ sa deklaracijom broja ekvivalentnom outputu funkcije. Za početak postavimo početnu vrijednost na nula. Tablicu stavke\_narudzbe naredbom join spojimo sa tablicom „proizvodi“ preko zajedničkog atributa „proizvod\_id“. Dodatni uvjet na spojene tablice nam je i jednaka vrijednost „narudzba\_id“. Sada u SELECT dijelu pomoću funkcije agregacije sumiramo sve vrijednosti cijene i kolicine odabranih proizvoda te ih pomoću aliasa stavljamo u novi stupac pod nazivom „ukupnaCijena“

## Funkcije Bruno

Funkcija koja računa konačnu cijenu proizvoda nakon popusta

DELIMITER //

CREATE FUNCTION cijena\_s\_popustom(cijena DECIMAL(10,2), postotak\_popusta DECIMAL(5,2)) RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

  RETURN cijena \* (1 - postotak\_popusta / 100);

END //

DELIMITER ;

Opis: Ova funkcija ima ulazne parametre cijena i postotak\_popusta, te vraća cijenu proizvoda sa primjenjenim popustom.

Poziv funkcije:

SELECT proizvodi.\*, cijena\_s\_popustom(cijena, postotak\_popusta) AS snizena\_cijena

  FROM proizvodi

    INNER JOIN popusti ON proizvodi.id = popusti.proizvod\_id;

Funkcija koja dohvaća naziv načina isporuke po unesenom ID-u

DELIMITER //

CREATE FUNCTION naziv\_isporuke(isporuka\_id INTEGER) RETURNS VARCHAR(255)

DETERMINISTIC

BEGIN

  DECLARE naziv\_p VARCHAR(255);

    SELECT naziv INTO naziv\_p

    FROM nacini\_isporuke

        WHERE id = isporuka\_id;

    RETURN naziv\_p;

END //

DELIMITER ;

Opis: Ova funkcija ima ulazni parametar isporuka\_id. Prvo se deklarira parametar unutar funkcije naziv\_p i u njega se kroz SELECT naredbu sprema naziv traženog načina isporuke. Funkcija na kraju vraća naziv\_p.

Poziv funkcije:

SELECT \*, naziv\_isporuke(nacin\_isporuke\_id) AS naziv\_isporuke

  FROM narudzbe;

Funkcija koja računa koliko je pojedinih načina isporuke narudžbi

DELIMITER //

CREATE FUNCTION broj\_nacina\_isporuke(isporuka\_id INTEGER) RETURNS INTEGER

DETERMINISTIC

BEGIN

  DECLARE broj INTEGER;

  SELECT COUNT(\*) INTO broj

    FROM narudzbe

        WHERE nacin\_isporuke\_id = isporuka\_id

        GROUP BY nacin\_isporuke\_id;

    RETURN broj;

END //

DELIMITER ;

Opis: Ova funkcija ima ulazni parametar isporuka\_id pomoću kojeg se pretražuju načini isporuke. Deklarira se pomoćna varijabla broj u koju se sprema broj koliko je traženih načina isporuke.

Poziv funkcije:

SELECT \*, broj\_nacina\_isporuke(id) AS broj\_nacina\_isporuke

  FROM nacini\_isporuke;

## Funkcije Fran

A computer screen shot of a code

Description automatically generated

Vraća tekstualni popis povrata u kojem su navedeni datumi povrata i statusi povrata, ili poruku ako povrati ne postoje.

Detalji funkcije:

* Ulazni parametar:
  + narudzbaId INT: Identifikator narudžbe za koju se traže povrati proizvoda.
* Povratna vrijednost:
  + Funkcija vraća vrijednost tipa TEXT, koja sadrži formatiran popis informacija o povratima za zadanu narudžbu. Ako nema povrata, vraća tekst "Nema povrata za odabranu narudžbu."
* Glavna logika:
  + Koristi GROUP\_CONCAT za spajanje informacija o povratima u jedan string. Za svaki povrat prikazuje:
    - Datum povrata formatiran kao YYYY-MM-DD (koristi DATE\_FORMAT).
    - Status povrata, npr. "u obradi", "odbijeno" ili "odobreno".
  + Povrati su odvojeni znakom za novi redak za bolju čitljivost.
  + Ako ne postoje povrati za zadanu narudžbu, koristi IFNULL da vrati poruku "Nema povrata za odabranu narudžbu.“

## Funkcije Josip

A computer screen shot of a code

Description automatically generated

* Funkcija izračunava koliko puta se određeni kupon može još iskoristiti.
* **kupon\_id**: ID kupona za koji se provjerava preostalo iskorištenje.
* **k.max\_iskoristenja**: Maksimalan broj dopuštenih iskorištenja kupona.
* **COUNT(n.id)**: Broj narudžbi koje su koristile taj kupon.
* Ako kupon nije iskorišten, funkcija vraća maksimalan broj iskorištenja definiran u tablici **kuponi**.

A screen shot of a computer code

Description automatically generated

* Funkcija izračunava ukupan prihod od računa koji pripadaju određenom korisniku.
* **korisnik\_id**: ID korisnika za kojeg se računa ukupan prihod.
* **SUM(iznos)**: Zbraja sve iznose računa za tog korisnika.
* Ako korisnik nema račune, vraća **0.00**.

## Funkcije Leo

-- Funkcija: Formatiranje datuma (Leo)

DELIMITER //

CREATE FUNCTION formatiraj\_datum(input\_date DATE)

RETURNS VARCHAR(10)

DETERMINISTIC

BEGIN

RETURN DATE\_FORMAT(input\_date, '%d.%m.%Y');

END//

DELIMITER ;

**Funkcija: Formatiranje datuma**

**Opis:**Ova funkcija omogućava formatiranje datuma u željenom formatu „dan.mjesec.godina“. Namijenjena je za upotrebu u raznim SQL upitima ili pogledima kako bi datumi bili prikazani u čitljivom formatu za korisnike.

**Detalji implementacije:**  
Funkcija koristi DATE\_FORMAT, ugrađenu MySQL funkciju, za prilagodbu formata prikaza datuma.  
Ulazni parametar input\_date očekuje vrijednost tipa DATE.  
Vraća vrijednost tipa VARCHAR(10) s formatom „dd.mm.yyyy“, gdje:

* dd predstavlja dan u mjesecu,
* mm predstavlja mjesec u godini,
* yyyy predstavlja četverocifrenu godinu.

Funkcija je deklarirana kao DETERMINISTIC, što znači da će uvijek vratiti isti izlaz za isti ulaz.

# Procedure

## Procedure Loren

**Procedura za dodavanje proizvoda s provjerom grešaka**

DELIMITER //

**CREATE PROCEDURE** DodajProizvod (

**IN** p\_naziv **VARCHAR**(255),

**IN** p\_opis **TEXT**,

**IN** p\_cijena **DECIMAL**(10, 2),

**IN** p\_kategorija\_id **INT**,

**IN** p\_kolicina **INT**,

**IN** p\_slika **VARCHAR**(255),

**IN** p\_specifikacije **TEXT**

)

**BEGIN**

**IF** p\_cijena <= 0 **THEN**

**SIGNAL SQLSTATE '45501'**

**SET MESSAGE\_TEXT** = 'Cijena mora biti veća od 0.';

**ELSEIF** p\_kolicina < 0 **THEN**

**SIGNAL SQLSTATE '45502'**

**SET MESSAGE\_TEXT** = 'Količina na skladištu ne može biti negativna.';

**ELSE**

**INSERT INTO** proizvodi (naziv, opis, cijena, kategorija\_id, kolicina\_na\_skladistu, slika, specifikacije)

**VALUES** (p\_naziv, p\_opis, p\_cijena, p\_kategorija\_id, p\_kolicina, p\_slika, p\_specifikacije);

**END IF**;

**END** //

DELIMITER ;

Procedura „DodajProizvod“ omogućava dodavanje proizvoda u tablicu „proizvodi“ uz provjeru valjanosti unosa. Prvo provjerava valjanost unosa tj. cijena mora biti veća od 0 i količina ne smije biti negativna. Ako je unos ispravan, proizvod se unosi u tablicu „proizvodi“. U slučaju neispravnog unosa, generira se greška pomoću „SIGNAL“.

**Procedura za brisanje proizvoda s provjerom povezanih podataka**

DELIMITER //

**CREATE PROCEDURE** ObrisiProizvod (

**IN** p\_proizvod\_id **INT**

)

**BEGIN**

**IF EXISTS** (

**SELECT** \*

**FROM** stavke\_narudzbe

**WHERE** proizvod\_id = p\_proizvod\_id

) **THEN**

**SIGNAL SQLSTATE '45503'**

**SET MESSAGE\_TEXT** = 'Proizvod ne može biti obrisan jer postoje povezane narudžbe.';

**ELSE**

**DELETE FROM** popusti

**WHERE** proizvod\_id = p\_proizvod\_id;

**DELETE FROM** recenzije\_proizvoda

**WHERE** proizvod\_id = p\_proizvod\_id;

**DELETE FROM** wishlist

**WHERE** proizvod\_id = p\_proizvod\_id;

**DELETE FROM** proizvodi

**WHERE** id = p\_proizvod\_id;

**END IF**;

**END** //

DELIMITER ;

Procedura „ObrisiProizvod“ briše proizvod i sve povezane zapise iz tablica „popusti“, „recenzije\_proizvoda“, i „wishlist“. Međutim, ne dopušta brisanje ako postoje zapisi u tablici „stavke\_narudzbe“. Najprije provjerava postoji li zapis u tablici „stavke\_narudzbe“ za neki proizvod i ako postoji, generira grešku. Ako nema povezanih podaka, briše sve povezane n-torke i sam proizvod iz tablice „proizvodi“.

**Procedura za ažuriranje svih podataka vezanih za neki proizvod**

DELIMITER //

**CREATE PROCEDURE** azuriraj\_proizvod(

**IN** p\_proizvod\_id **INT**,

**IN** p\_naziv **VARCHAR**(255),

**IN** p\_opis **TEXT**,

**IN** p\_cijena **DECIMAL**(10, 2),

**IN** p\_kategorija\_id **INT**,

**IN** p\_kolicina\_na\_skladistu **INT**,

**IN** p\_slika **VARCHAR**(255),

**IN** p\_specifikacije **TEXT**

)

**BEGIN**

**IF** p\_cijena <= 0 **THEN**

**SIGNAL SQLSTATE '45501'**

**SET MESSAGE\_TEXT** = 'Cijena mora biti veća od 0.';

**END IF**;

**IF** p\_kolicina\_na\_skladistu < 0 **THEN**

**SIGNAL SQLSTATE '45502'**

**SET MESSAGE\_TEXT** = 'Količina na skladištu ne može biti negativna.';

**END IF**;

**IF NOT EXISTS** (**SELECT** \* **FROM** kategorije\_proizvoda **WHERE** id = p\_kategorija\_id) **THEN**

**SIGNAL SQLSTATE '45509'**

**SET MESSAGE\_TEXT** = 'Kategorija s unesenim ID ne postoji.';

**END IF**;

**IF NOT EXISTS** (**SELECT** \* **FROM** proizvodi **WHERE** id = p\_proizvod\_id) **THEN**

**SIGNAL SQLSTATE '45510'**

**SET MESSAGE\_TEXT** = 'Proizvod s unesenim ID ne postoji.';

**END IF**;

**UPDATE** proizvodi

**SET** naziv = p\_naziv,

opis = p\_opis,

cijena = p\_cijena,

kategorija\_id = p\_kategorija\_id,

kolicina\_na\_skladistu = p\_kolicina\_na\_skladistu,

slika = p\_slika,

specifikacije = p\_specifikacije

**WHERE** id = p\_proizvod\_id;

**END** //

DELIMITER ;

Procedura „azuriraj\_proizvod“ služi za ažuriranje podataka o proizvodu u tablici „proizvodi“. Procedura prima 8 ulaznih parametara: jedinstveni identifikator proizvoda, naziv, opis, cijenu, ID kategorije, količinu na skladištu, naziv slike i specifikacije. Prvo, procedura provodi nekoliko provjera kako bi osigurala ispravnost unesenih podataka. Provjerava se je li cijena veća od 0 te se u slučaju da nije, ispisuje grešku „Cijena mora biti veća od 0.“. Provjerava se je količina na skladištu pozitivna te se u slučaju da nije, ispisuje grešku „Količina na skladištu ne može biti negativna.“. Sljedeća provjera je da u tablici „kategorije\_proizvoda“ postoji zapis s unesenim „kategorija\_id“, ako nije ispisuje se „Kategorija s unesenim ID ne postoji.“. Na kraju provjerava ako postoji id koji je unesen u tablici „proizvodi“, ukoliko nije i za to ispisuje grešku „Kategorija s unesenim ID ne postoji.“. Ako nema nikakve greške procedura ažurira postojeći zapis iz tablice „proizvodi“ s novim vrijednostima za sve osim „id “ i „datum\_kreiranja“.

**Procedura za pretragu proizvode s ključnom riječi**

DELIMITER //

**CREATE PROCEDURE** PronadjiProizvode (

**IN** kljucna\_rijec **VARCHAR**(255)

)

**BEGIN**

**SELECT** \*

**FROM** proizvodi

**WHERE** naziv **LIKE** **CONCAT**('%', kljucna\_rijec, '%')

**OR** opis **LIKE CONCAT**('%', kljucna\_rijec, '%');

**END** //

DELIMITER ;

Procedura „PronadjiProizvode“ omogućava pretraživanje proizvoda na temelju ključne riječi u nazivima i opisima proizvoda. Koristi se „LIKE“ operator za pretraživanje proizvoda prema ključnim riječima u atributima „naziv“ i „opis“. Na kraju vraća rezultat kao tablicu sa svim n-torkama koje se podudaraju s ključnom riječi.

**Procedura za dodavanje kategorije proizvoda**

DELIMITER //

**CREATE PROCEDURE** dodaj\_kategoriju\_proizvoda(

**IN** p\_naziv VARCHAR(255),

**IN** p\_opis **TEXT**

)

**BEGIN**

**IF EXISTS** (**SELECT** \* **FROM** kategorije\_proizvoda **WHERE** naziv = p\_naziv) **THEN**

**SIGNAL SQLSTATE '45504'**

**SET MESSAGE\_TEXT** = 'Kategorija s unesenim nazivom već postoji.';

**END IF**;

**INSERT INTO** kategorije\_proizvoda (naziv, opis)

**VALUES** (p\_naziv, p\_opis);

**END** //

DELIMITER ;

Procedura „dodaj\_kategoriju\_proizvoda“ se koristi za dodavanje nove kategorije u tablicu „kategorije\_proizvoda“. Prima „naziv“ i „opis“ za kategoriju koju dodajemo u tablicu „kategorije\_proizvoda“. Prvo se provjerava postoji li već kategorija s istim nazivom u tablici. Ukoliko postoji, ispisuje „Kategorija s unesenim nazivom već postoji.“. Ako kategorija ne postoji, procedura nastavlja izvršavanje i umetne novi zapis u tablicu „kategorije\_proizvoda“.

**Procedura za ažuriranje kategorije proizvoda**

DELIMITER //

**CREATE PROCEDURE** azuriraj\_kategoriju\_proizvoda(

**IN** p\_kategorija\_id **INT**,

**IN** p\_naziv **VARCHAR**(255),

**IN** p\_opis **TEXT**

)

**BEGIN**

**IF NOT EXISTS** (**SELECT** \* **FROM** kategorije\_proizvoda **WHERE** id = p\_kategorija\_id) **THEN**

**SIGNAL SQLSTATE '45509'**

**SET MESSAGE\_TEXT** = 'Kategorija s unesenim ID ne postoji.';

**END IF;**

**IF EXISTS** (

**SELECT** \* **FROM** kategorije\_proizvoda

**WHERE** naziv = p\_naziv **AND** id != p\_kategorija\_id

) **THEN**

**SIGNAL SQLSTATE '45504'**

**SET MESSAGE\_TEXT** = 'Kategorija s unesenim nazivom već postoji.';

**END IF;**

**UPDATE** kategorije\_proizvoda

**SET** naziv = p\_naziv,

opis = p\_opis

**WHERE** id = p\_kategorija\_id;

**END** //

DELIMITER ;

Procedura „azuriraj\_kategoriju\_proizvoda“ omogućava ažuriranje podataka o postojećoj kategoriji u tablici „kategorije\_proizvoda“. Prima 3 ulazna parametra, id kategorije koju mjenjamo, novi naziv kategorije i novi opis kategorije. Prvo provjerav postoji li kategorija s danim id u tablici. Ako ne postoji, ispisuje grešku „Kategorija s unesenim ID ne postoji.“. Nakon toga, procedura provjerava postoji li već neka druga kategorija s istim novim nazivom, kako bi se spriječilo dupliciranje naziva. Ako takva kategorija postoji, ispisuje se greška „Kategorija s unesenim nazivom već postoji.“. Kad obje provjere prođu bez problema, procedura nastavlja izvršavati naredbu „UPDATE“, koja u tablici „kategorije\_proizvoda“ postavlja novi „naziv“ i „opis“ za zapis čiji je „id“ jednak „p\_kategorija\_id“.

**Procedura za brisanje kategorije proizvoda**

DELIMITER //

**CREATE PROCEDURE** obrisi\_kategoriju\_proizvoda(

**IN** p\_kategorija\_id **INT**

)

**BEGIN**

**IF NOT EXISTS** (**SELECT** \* **FROM** kategorije\_proizvoda **WHERE** id = p\_kategorija\_id) **THEN**

**SIGNAL SQLSTATE '45509'**

**SET MESSAGE\_TEXT** = 'Kategorija s unesenim ID ne postoji.';

**END IF;**

**IF EXISTS** (**SELECT** \* **FROM** proizvodi **WHERE** kategorija\_id = p\_kategorija\_id) **THEN**

**SIGNAL SQLSTATE '45507'**

**SET MESSAGE\_TEXT** = 'Nije moguće obrisati kategoriju jer postoje proizvodi povezani s njom.';

**END IF**;

**DELETE FROM** kategorije\_proizvoda

**WHERE** id = p\_kategorija\_id;

**END** //

DELIMITER ;

Procedura „obrisi\_kategoriju\_proizvoda“ omogućava brisanje kategorije proizvoda iz tablice „kategorije\_proizvoda“. Procedura prima jedan ulazni parametar, „p\_kategorija\_id“, koji predstavlja jedinstveni identifikator kategorije koju želimo obrisati. Prvo se provjerava postoji li kategorija s tim id u tablici „kategorije\_proizvoda“, ako ne postoji ispisuje se greška „Kategorija s unesenim ID ne postoji.“. Zatim se provjerava postoji li u tablici „proizvodi“ bilo koji proizvod koji koristi ovu kategoriju, odnosno ima vrijednost „kategorija\_id“ jednaku id kojeg unesemo. Ako takav proizvod postoji, ispisuje se greška „'Nije moguće obrisati kategoriju jer postoje proizvodi povezani s njom.“. Ako obje provjere prođu procedura izvršava naredbu da s iz tablice „kategorije\_proizvoda“ izbriše unos s unesenim id i izvršava se brisanje n-torke.

**Procedura za unos recenzije proizvoda**

DELIMITER //

**CREATE PROCEDURE** dodaj\_recenziju\_proizvoda(

**IN** p\_proizvod\_id **INT**,

**IN** p\_korisnik\_id **INT**,

**IN** p\_ocjena **INT**,

**IN** p\_komentar **TEXT**

)

**BEGIN**

**IF NOT EXISTS** (**SELECT** \* **FROM** proizvodi **WHERE** id = p\_proizvod\_id) **THEN**

**SIGNAL SQLSTATE '45510'**

**SET MESSAGE\_TEXT** = 'Proizvod s unesenim ID ne postoji.';

**END IF**;

**IF NOT EXISTS** (**SELECT** \* **FROM** korisnici **WHERE** id = p\_korisnik\_id) **THEN**

**SIGNAL SQLSTATE '45508'**

**SET MESSAGE\_TEXT** = 'Korisnik s unesenim ID ne postoji.';

**END IF**;

**IF** p\_ocjena < 1 **OR** p\_ocjena > 5 **THEN**

**SIGNAL SQLSTATE '45511'**

**SET MESSAGE\_TEXT** = 'Ocjena mora biti između 1 i 5.';

**END IF**;

**IF EXISTS** (

**SELECT** \* **FROM** recenzije\_proizvoda

**WHERE** proizvod\_id = p\_proizvod\_id **AND** korisnik\_id = p\_korisnik\_id

) **THEN**

**SIGNAL SQLSTATE '45506'**

**SET MESSAGE\_TEXT** = 'Korisnik je već ostavio recenziju za ovaj proizvod.';

**END IF**;

**INSERT INTO** recenzije\_proizvoda (proizvod\_id, korisnik\_id, ocjena, komentar)

**VALUES** (p\_proizvod\_id, p\_korisnik\_id, p\_ocjena, p\_komentar);

**END** //

DELIMITER ;

Procedura „dodaj\_recenziju\_proizvoda“ omogućava za dodavanje nove recenzije proizvoda u tablicu recenzije\_proizvoda. Prima 4 parametra. Identifikator proizvoda ,identifikator korisnika ,ocjenu i tekstualni komentar. Prvo se provjerava ako proizvod s unesenim id postoji u tablici proizvodi. Ako ne postoji ispisuje se greška „Proizvod s unesenim ID ne postoji.“. Druga provjera je ako postoji korisnik s upisanim id u tablici „korisnik“. Ako ne ispisuje se greška „Korisnik s unesenim ID ne postoji.“. Zatim je provjera da unesena ocjena mora biti između 1 i 5 koja isto tako ukoliko ne ispoštujemo to ispisuje grešku „Ocjena mora biti između 1 i 5.“. I zadnja provjera je ta da se provjerava ako je korisnik već napisao recenziju, ukoliko nije procedura prolazi i izvršava unos novog zapisa u tablicu „recenzije\_proizvoda“ pomoću naredbe „INSERT“, čime se dodaje recenzija s navedenim podacima.

**Procedura za ažuriranje recenzije proizvoda**

DELIMITER //

**CREATE PROCEDURE** azuriraj\_recenziju\_proizvoda(

**IN** p\_recenzija\_id **INT**,

**IN** p\_ocjena **INT**,

**IN** p\_komentar **TEXT**

)

**BEGIN**

**IF NOT EXISTS** (**SELECT** \* **FROM** recenzije\_proizvoda **WHERE** id = p\_recenzija\_id) **THEN**

**SIGNAL SQLSTATE '45512'**

**SET MESSAGE\_TEXT** = 'Recenzija s unesenim ID ne postoji.';

**END IF**;

**IF** p\_ocjena < 1 **OR** p\_ocjena > 5 **THEN**

**SIGNAL SQLSTATE '45511'**

**SET MESSAGE\_TEXT** = 'Ocjena mora biti između 1 i 5.';

**END IF**;

**UPDATE** recenzije\_proizvoda

**SET** ocjena = p\_ocjena,

komentar = p\_komentar,

datum\_recenzije = **CURRENT\_TIMESTAMP**

**WHERE** id = p\_recenzija\_id;

**END** //

DELIMITER ;

Procedura „azuriraj\_recenziju\_proizvoda“ omogućava ažuriranje podataka o recenziji proizvoda u tablici „recenzije\_proizvoda“. Prima 3 parametra a to su, identifikator recenzije, novu ocjenu i novi komentar. Provjerava ako postoji recenzija u tablici „recenzije\_proizvoda“ koja se poklapa s id kojeg smo unijeli, ukoliko ne postoji ispisuje se greška „Recenzija s unesenim ID ne postoji.“. Nakon te provjere, druga provjera provjerava ako je nova ocjena koja je unesena između 1 i 5, ukoliko nije ispisuje se greška „Ocjena mora biti između 1 i 5.“. Kad prođu sve provjere ažuriraja se zapis putem naredbe „UPDATE“ i automatski, uz unesene podatke, postavlja se „datum\_recenzije“ na trenutni datum.

**Procedura za brisanje recenzije proizvoda**

DELIMITER //

**CREATE PROCEDURE** obrisi\_recenziju\_proizvoda(

**IN** p\_recenzija\_id **INT**

)

**BEGIN**

**IF NOT EXISTS** (**SELECT** \* **FROM** recenzije\_proizvoda **WHERE** id = p\_recenzija\_id) **THEN**

**SIGNAL SQLSTATE '45512'**

**SET MESSAGE\_TEXT** = 'Recenzija s unesenim ID ne postoji.';

**END IF**;

**DELETE FROM** recenzije\_proizvoda

**WHERE** id = p\_recenzija\_id;

**END** //

DELIMITER ;

Procedura „obrisi\_recenziju\_proizvoda“ omogućava brisanje recenzije proizvoda iz tablice „recenzije\_proizvoda“**.** Procedura prima jedan ulazni parametar, to je identifikator „recenzije\_proizvoda“ koji želimo obrisati. Prije nego što pokuša obrisati zapis, procedura provjerava postoji li recenzija s danim ID-jem u tablici. Ako recenzija ne postoji ispisuje se greška „Recenzija s unesenim ID ne postoji.“. Kad prođe provjeru procedura izvršava naredbu koja briše recenziju na temelju unesenog id.

## Procedure Morena

**Procedura omogućuje korisnicima da stvore novu narudžbu. Unosi korisnika, odabrani način isporuke, i kupon (ako postoji), a zatim automatski dodaje stavke iz košarice u narudžbu.**

SET @narudzba\_id=NULL;

DELIMITER //

CREATE PROCEDURE kreiraj\_narudzbu(

IN p\_korisnik\_id INT,

IN p\_nacin\_isporuke\_id INT,

IN p\_kupon\_id INT,

OUT p\_narudzba\_id INT

)

BEGIN

DECLARE v\_ukupna\_cijena DECIMAL(10, 2) DEFAULT 0.00;

SELECT SUM(p.cijena \* k.kolicina) INTO v\_ukupna\_cijena

FROM kosarica k

JOIN proizvodi p ON k.proizvod\_id = p.id

WHERE k.korisnik\_id = p\_korisnik\_id;

INSERT INTO narudzbe (korisnik\_id, datum\_narudzbe, status\_narudzbe, ukupna\_cijena, nacin\_isporuke\_id, kupon\_id)

VALUES (p\_korisnik\_id, CURRENT\_DATE, 'u obradi', v\_ukupna\_cijena, p\_nacin\_isporuke\_id, p\_kupon\_id);

SET p\_narudzba\_id = LAST\_INSERT\_ID();

INSERT INTO stavke\_narudzbe (narudzba\_id, proizvod\_id, kolicina)

SELECT p\_narudzba\_id, k.proizvod\_id, k.kolicina

FROM kosarica k

WHERE k.korisnik\_id = p\_korisnik\_id;

DELETE FROM kosarica WHERE korisnik\_id = p\_korisnik\_id;

END //

DELIMITER ;

Procedura „kreiraj\_narudzbu“ nam olakšava kreiranje narudžbe i objedinjuje više upita. Svi koraci se izvršavaju pozivom jedne procedure i tako smanjuje potreba za slanjem više zahtjeva prema bazi podataka.

Na početku, prije poziva procedure, osiguramo da izlazni parametar „narudzba\_id“ bude jedinstven za svakog korisnika tj. Svaki put kad netko posjeti naš web-shop, da dobije novi id. To ćemo napraviti tako da izlaznu varijablu definiramo na razini sesije tj. Naša varijabla je dostupna samo tijekom trenutne sesije baze podataka i može se koristiti za pohranu privremenih podataka.

Definirali smo ulazne parametre: „korisnik\_id“(automatski se popunjava logiranjem korisnika), „nacin\_isporuke“ i „kupon\_id“ koji se dobivaju unosom korisnika. Izlazni parametar iz ove procedure je „narudzba\_id“ koji se automatski dodaje u bazu nakon završenog procesa naručivanja, te smo na kraju naredbom “DELETE FROM kosarica..“ osigurali pražnjenje košarice nakon što je narudžba primljena u obradu kako ne bi došlo do kolizije među narudžbama.

Na početku definiramo novu varijablu v\_ukupna\_cijena decimalnog tipa. Nakon toga računamo ukupnu cijenu na temelju elemenata u košarici. Naredbom join povežemo tablice „kosarica“ i „proizvod“ preko zajednickog atributa „proizvod\_id“ samo za retke gdje nam se poklapaju i „korisnik\_id“ te na kraju preko select naredbe selektiramo sumu cijene i količine proizvoda koju smo preko aliasa stavili u novi stupac i nazvali ga „v\_ukupna\_cijena“ tj rezultat smo spremili u varijablu koju smo definirali na početku procedure kako bismo je dalje mogli koristiti. Ovime smo izračunali ukupnu cijenu na temelju košarice

Novu narudžbu unosimo u tablicu „narudzbe“ sa ulaznima vrijednostima koje je unio korisnik na početku, id, trenutnim vremenom, odnosno CURRENT\_DATE, statusom narudžbe „u obradi“ dok ne završimo sa procedurom, prethodno izračunatu ukupnu cijenu artikala u košarici, načinom isporuke koji je također ulazna varijabla te eventualnim unosom koda za kupon ako ga korisnik posjeduje, u suprotnom jue ova varijabla postavljena na NULL.

Za dohvaćanje vrijednosti id-a posljednjeg unosa koristimo funkciju LAST\_INSERT\_ID() koja vraća vrijednost upravo generiranog id-a pomoću AUTO\_INCREMENT-a i vraćenu vrijednost spremamo u varijablu „p\_narudzba\_id“ koja je definirana kao izlazni parametar ove procedure.

Slijedi nam kopiranje stavki iz košarice u stavke narudžbe i to radimo insertanjem u tablicu stavke\_narudžbe atribute iz košarice za koje vrijedi da im korisnici imaju identičan id.

Za kraj moramo biti sigurni da nakon prepisa podataka u tablicu stavke\_narudzbe, u tablici košarica nije ostalo ništa pa naredbom DELETE brišemo sve retke u tablici „kosarica“.

**Procedura koja omogućuje administraciji da ažurira status narudžbe (npr. 'u obradi', 'poslano', 'dostavljeno') prema ID-u narudžbe.**

DELIMITER //

CREATE PROCEDURE azuriraj\_status\_narudzbe(

IN p\_narudzba\_id INT,

IN p\_status\_narudzbe ENUM('u obradi', 'poslano', 'dostavljeno')

)

BEGIN

-- Ažuriraj status narudžbe

UPDATE narudzbe

SET status\_narudzbe = p\_status\_narudzbe

WHERE id = p\_narudzba\_id;

END //

DELIMITER ;

Kako bismo lakše ažurirali status narudžbe, kreirali smo proceduru kroz koju to možemo lakše napraviti i sa manje mogućnosti za pogrešku. Kao ulazni parametar od administratora tražimo id narudžbe te njen trenutni status u tranzitu do korisnika. Moguće varijable su nam postavljene na „u obradi“, „poslano“ i „dostavljeno“. Odabirom jedne od tri mogućnosti kreiramo drugu ulaznu vrijednost od strane administratora. Proceduru započinjemo ažuriranjem tablice narudžbe tako da varijabla „p\_status\_narudzbe“ poprimi vrijednost ulaznog parametra (jednog od tri moguća), za redak u kojem se id-evi narudžbe poklapaju (onaj u postojećoj tablici sa ulaznim id-em unešenim od strane administratora). Nakon toga završavamo sa procedurom

**Procedura omogućuje korisnicima dodavanje stavke u svoju košaricu, povećavajući količinu proizvoda u košarici ako je proizvod već prisutan.**

DELIMITER //

CREATE PROCEDURE dodaj\_u\_kosaricu(

IN p\_korisnik\_id INT,

IN p\_proizvod\_id INT,

IN p\_kolicina INT

)

BEGIN

DECLARE v\_kolicina INT;

SELECT kolicina INTO v\_kolicina

FROM kosarica

WHERE korisnik\_id = p\_korisnik\_id AND proizvod\_id = p\_proizvod\_id;

IF v\_kolicina IS NOT NULL THEN

UPDATE kosarica

SET kolicina = kolicina + p\_kolicina

WHERE korisnik\_id = p\_korisnik\_id AND proizvod\_id = p\_proizvod\_id;

ELSE

INSERT INTO kosarica (korisnik\_id, proizvod\_id, kolicina)

VALUES (p\_korisnik\_id, p\_proizvod\_id, p\_kolicina);

END IF;

END //

DELIMITER ;

Ova procedura kao ulazne parametre zahtjeva id od korisnika i odabranog proizvoda te količinu. Deklariramo privremenu varijablu za korištenje unutar procedure cjelobrojnog tipa i nazvali smo ju „v\_kolicina“. U tijelu procedure prvo provjeravamo postoji li već taj proizvod u košarici korisnika koji je trenutno aktivan. U tablici „kosarica“ provjeravamo podudaraju li se ulazni id korisnika sa id-em korisnika koji već ima kreiranu košaricu, isto tako uspoređujemo i id-eve proizvoda. Ako količina odabranog proizvoda u našoj tablici nije NULL tj ako je artikl već bio odabran, tada preko update vrijednost količine iz tablice zamijenimo ulaznom varijablom količine na početku procedure. Ako odabrani proizvod nije u košarici tj. ako je njegova količina postavljena na NULL vrijednost, onda dodajemo novi redak u tablici „kosarica“ unosom svih ulaznih podataka u taj novi redak. Izlazimo iz if/else petlje i završavamo proceduru.

**Procedura za dodavanje proizvoda u košaricu**

DELIMITER //

CREATE PROCEDURE dodaj\_proizvod\_u\_kosaricu( IN p\_korisnik\_id INT,

IN p\_proizvod\_id INT,

IN p\_kolicina INT )

BEGIN DECLARE v\_kolicina\_na\_skladistu INT;

DECLARE v\_postojeca\_kolicina INT DEFAULT 0;

SELECT kolicina\_na\_skladistu INTO v\_kolicina\_na\_skladistu

FROM proizvodi

WHERE id = p\_proizvod\_id;

IF v\_kolicina\_na\_skladistu IS NULL

THEN SIGNAL SQLSTATE '45300' SET MESSAGE\_TEXT = 'Proizvod s navedenim ID-em ne postoji.';

ELSEIF v\_kolicina\_na\_skladistu < p\_kolicina

THEN SIGNAL SQLSTATE '45300' SET MESSAGE\_TEXT = 'Nedovoljna količina proizvoda na skladištu.';

ELSE

SELECT kolicina INTO v\_postojeca\_kolicina

FROM kosarica

WHERE korisnik\_id = p\_korisnik\_id AND proizvod\_id = p\_proizvod\_id;

IF v\_postojeca\_kolicina > 0

THEN UPDATE kosarica SET kolicina = kolicina + p\_kolicina

WHERE korisnik\_id = p\_korisnik\_id AND proizvod\_id = p\_proizvod\_id;

ELSE

INSERT INTO kosarica (korisnik\_id, proizvod\_id, kolicina) VALUES (p\_korisnik\_id, p\_proizvod\_id, p\_kolicina);

END IF;

UPDATE proizvodi SET kolicina\_na\_skladistu = kolicina\_na\_skladistu - p\_kolicina

WHERE id = p\_proizvod\_id;

END IF;

END; //

DELIMITER ;

Kreiranjem procedure „dodaj\_proizvod\_u\_kosaricu“ stvaramo efikasnije ažuriranje dodanih proizvoda u procesu potencijalne kupnje. Kao ulazne parametre od korisnika spremamo id, uzimamo id odabranog proizvoda te njegovu količinu. Kao lokalnu varijablu na samom pocetku procedure postavljamo „v\_kolicina\_na\_skladistu“ cjelobrojnog tipa i „v\_postojeca\_kolicina“ koju smo postavili na nulu, ne na NULL vrijednost. Nakon postavljanja parametara, moramo dohvatiti trenutnu kolicinu proizvoda na skladištu. To mozemo izvuci iz tablice proizvodi gdje nam je ulazna varijabla „proizvod\_id“ jednaka id-ju u tablici „proizvodi“. Izdvojimo proizvod određen id-jem i u selekt djelu naredbom INTO spremimo njegovu količinu na skladištu u pomoćnu varijablu „v\_kolicina\_na\_skladistu“. Sada provjeravamo ima li trežene količine odabranog proizvoda u skladištu preko for petlje. Ako nam je trazena kolicina jednaka NULL, Baza će nam vratiti grešku „45300“ sa porukom da naveden proizvod ne postoji. Kroz elseif naredbu ulazimo u drugi dio petlje koji nam vraća grešku i poruku u slučaju da je tražena količina veća od one na skladištu. Tada ulazimo u zadnji dio petlje gdje prvo provjeravamo postoji li već proizvod u košarici. From dijelu ulazimo u tablicu „kosarica“ i kroz WHERE uvjet uspoređujemo ulazni id proizvoda sa id-evima u košarici. Ako se oni podudaraju, moramo ažurirati kolicinu naredbom UPDATE tako da ulaznu vrijednost količine zamjenimo sa postojećom. Ako proizvod ne postoji u košarici, naredbom INSERT INTO dodajemo novi redak sa tim proizvodom i ulaznom vrijednošću količine. Za kraj, oduzmemo unešenu količinu proizvoda u košaricu od količine proizvoda na skladištu.

**Procedura za ažuriranje proizvoda u košarici**

DELIMITER //

CREATE PROCEDURE azuriraj\_proizvod\_u\_kosarici( IN p\_korisnik\_id INT,

IN p\_proizvod\_id INT,

IN p\_nova\_kolicina INT )

BEGIN

DECLARE v\_trenutna\_kolicina INT DEFAULT 0;

DECLARE v\_kolicina\_na\_skladistu INT DEFAULT 0;

DECLARE v\_razlika INT;

SELECT kolicina INTO v\_trenutna\_kolicina

FROM kosarica

WHERE korisnik\_id = p\_korisnik\_id AND proizvod\_id = p\_proizvod\_id;

IF v\_trenutna\_kolicina IS NULL

THEN SIGNAL SQLSTATE '45310' SET MESSAGE\_TEXT = 'Proizvod nije pronađen u košarici.';

END IF;

SELECT kolicina\_na\_skladistu INTO v\_kolicina\_na\_skladistu

FROM proizvodi

WHERE id = p\_proizvod\_id;

SET v\_razlika = p\_nova\_kolicina - v\_trenutna\_kolicina;

IF v\_razlika > 0

THEN IF v\_razlika > v\_kolicina\_na\_skladistu

THEN SIGNAL SQLSTATE '45311' SET MESSAGE\_TEXT = 'Nedovoljna količina proizvoda na skladištu za ažuriranje.';

END IF;

UPDATE proizvodi

SET kolicina\_na\_skladistu = kolicina\_na\_skladistu - v\_razlika

WHERE id = p\_proizvod\_id;

ELSEIF v\_razlika < 0

THEN UPDATE proizvodi

SET kolicina\_na\_skladistu = kolicina\_na\_skladistu + ABS(v\_razlika) WHERE id = p\_proizvod\_id;

END IF;

UPDATE kosarica

SET kolicina = p\_nova\_kolicina

WHERE korisnik\_id = p\_korisnik\_id AND proizvod\_id = p\_proizvod\_id;

END; //

DELIMITER ;

Procedura „azuriraj\_proizvod\_u\_kosarici“ kao ulazne podatke traži id korisnika, id proizvoda te količinu koju želimo ažurirati. Deklariramo lokalne varijable trenutna količina, količina na skladištu i razlika između ulazne varijable količine i trenutne količine proizvoda na skladištu (s tim podatkom ćemo manipulirati prije usporedbe sa ulaznom varijablom količine). Sada preko SELECT naredbe dohvaćamo količinu proizvoda u košarici i spremamo ga u lokalnu varijablu „v\_trenutna\_kolicina“ . Preko If/else petlje provjerit ćemo postoji li proizvod u košarici. Ako je njegova vrijednost NULL odnosno, nije definirana, vratit će nam grešku broj „45310“ sa porukom da proizvod nije pronađen u košarici. Time završavamo if petlju. Sada dohvaćamo količinu proizvoda sa istim id-jem na skladištu i spremamo u lokalnu varijablu „v\_kolicina\_na\_skladistu“. Moramo izračunati razliku između nove i trenutne količine te provjeravamo ima li nove količine odabranog proizvoda na skladištu. Preko if/else petlje prolazimo prvo slučaj kada je razlika veća od nule i kada bi trebali dodatnu količinu sa skladišta, još treba provjerili postoji li ta količina na stanju. U slučaju da ne postoji, vraća nam se greška br.45311 i poruka o nedovoljnoj količini proizvoda na stanju u skladištu. U suprotnom, oduzimamo razliku od količine na skladištu i dodajemo ju u košaricu. Ako nam je tražena količina manja od prvotno dodane u košaricu, tada oduzimamo višak iz košarice i vraćamo na stanju u skladištu preko naredbe UPDATE.

**Procedura za brisanje proizvoda iz košarice**

DELIMITER //

CREATE PROCEDURE obrisi\_proizvod\_iz\_kosarice( IN p\_korisnik\_id INT,

IN p\_proizvod\_id INT )

BEGIN DECLARE v\_kolicina\_u\_kosarici INT DEFAULT 0;

SELECT kolicina INTO v\_kolicina\_u\_kosarici

FROM kosarica

WHERE korisnik\_id = p\_korisnik\_id AND proizvod\_id = p\_proizvod\_id;

IF v\_kolicina\_u\_kosarici

IS NULL THEN SIGNAL SQLSTATE '45312' SET MESSAGE\_TEXT = 'Proizvod nije pronađen u košarici.';

END IF;

UPDATE proizvodi

SET kolicina\_na\_skladistu = kolicina\_na\_skladistu + v\_kolicina\_u\_kosarici

WHERE id = p\_proizvod\_id;

DELETE FROM kosarica

WHERE korisnik\_id = p\_korisnik\_id AND proizvod\_id = p\_proizvod\_id;

END; //

DELIMITER ;

Ovom procedurom brišemo proizvode iz košarice. Kao ulazne vrijednosti tražimo id od korisnika i proizvoda kojeg bi obrisali. Stvaramo novu lokalnu varijablu „v\_kolicina\_u\_kosarici“ koju postavljamo na nulu. Prvo preko SELECT naredbe tražimo proizvod unesenog id-ja u tablici košarica, ako proizvod nije nađen, javlja nam se greška br.45312 sa porukom da taj proizvod nije pronađen u košarici. U suprotnom nastavljamo sa sljedećom naredbom u proceduri koja vraća količinu natrag na skladište naredbom UPDATE preko koje dodajemo količinu iz košarice u količinu proizvoda na skladištu. Novu vrijednost spremamo u varijablu „kolicina\_na\_skladistu“ te ju mijenjamo s postojećom u tablici „proizvodi“. Za kraj, moramo obrisati taj proizvod iz košarice a to radimo naredbom DELETE iz tablice košarica u redu gdje nam se podudaraju id-evi proizvoda.

## Procedure Bruno

Procedura za dodavanje načina isporuke

DELIMITER //

CREATE PROCEDURE dodaj\_nacin\_isporuke(naziv\_p VARCHAR(255), opis\_p TEXT, cijena\_p DECIMAL(10,2), trajanje\_p INTEGER)

BEGIN

  DECLARE brojac INTEGER DEFAULT 0;

  SELECT COUNT(\*) INTO brojac

    FROM nacini\_isporuke

        WHERE naziv = naziv\_p;

    IF brojac > 0 THEN

    SIGNAL SQLSTATE "45102" SET MESSAGE\_TEXT = "Greška u unosu, način isporuke već postoji!";

  ELSE

    INSERT INTO nacini\_isporuke (naziv, opis, cijena, trajanje) VALUES (naziv\_p, opis\_p, cijena\_p, trajanje\_p);

  END IF;

END //

DELIMITER ;

Opis: Procedura koja služi za dodavanje novog načina isporuke. Prvo se provjerava da li taj način isporuke već postoji tako da se provjerava je li novi naziv već postoji u tablici. Ako postoji, baca grešku.

Poziv procedure:

CALL dodaj\_nacin\_isporuke("Express dostava", "Dostava u roku 3 sata", 150, 0);

Procedura za brisanje načina isporuke po ID-u

DELIMITER //

CREATE PROCEDURE obrisi\_nacin\_isporuke(id\_p INTEGER)

BEGIN

  DECLARE brojac INTEGER DEFAULT 0;

  SELECT COUNT(\*) INTO brojac

    FROM nacini\_isporuke

        WHERE id = id\_p;

    IF brojac = 0 THEN

    SIGNAL SQLSTATE "45102" SET MESSAGE\_TEXT = "Greška u brisanju, način isporuke ne postoji!";

  ELSE

    DELETE

      FROM nacini\_isporuke

      WHERE id = id\_p;

  END IF;

END //

DELIMITER ;

Opis: Procedura koja služi za brisanje načina isporuke po unesenom ID-u. Prvo se provjerava da li uneseni ID načina isporuke postoji. Ako ne postoji, baca grešku.

Poziv procedure:

CALL obrisi\_nacin\_isporuke(7);

Procedura za dodavanje popusta

DELIMITER //

CREATE PROCEDURE dodaj\_popust(proizvod\_id\_p INTEGER, postotak\_popusta\_p DECIMAL(5,2), datum\_pocetka\_p DATE, datum\_zavrsetka\_p DATE)

BEGIN

    INSERT INTO popusti (proizvod\_id, postotak\_popusta, datum\_pocetka, datum\_zavrsetka) VALUES (proizvod\_id\_p, postotak\_popusta\_p, datum\_pocetka\_p, datum\_zavrsetka\_p);

END //

DELIMITER ;

Opis: Procedura za dodavanje popusta za određeni proizvod. Ovdje nismo koristili nikakve provjere kod unosa, zato što imamo okidač za to, koji provjerava jedinstvenost popusta za određeni proizvod.

Poziv procedure:

CALL dodaj\_popust(56, 30, '2025-01-01', '2025-01-10');

Procedura za brisanje popusta po ID-u

DELIMITER //

CREATE PROCEDURE obrisi\_popust(id\_p INTEGER)

BEGIN

  DECLARE brojac INTEGER DEFAULT 0;

  SELECT COUNT(\*) INTO brojac

    FROM popusti

        WHERE id = id\_p;

    IF brojac = 0 THEN

    SIGNAL SQLSTATE "45103" SET MESSAGE\_TEXT = "Greška u brisanju, popust ne postoji!";

  ELSE

    DELETE

      FROM popusti

      WHERE id = id\_p;

  END IF;

END //

DELIMITER ;

Opis: Procedura koja služi za brisanje popusta po unesenom ID-u. Prvo se provjerava da li uneseni ID popusta postoji. Ako ne postoji, baca grešku.

Poziv procedure:

CALL obrisi\_popust(41);

Procedura za ažuriranje popusta

DELIMITER //

CREATE PROCEDURE azuriraj\_popust(popust\_id INTEGER, novi\_postotak DECIMAL(5,2))

BEGIN

  DECLARE brojac INTEGER DEFAULT 0;

  SELECT COUNT(\*) INTO brojac

    FROM popusti

        WHERE id = popust\_id;

    IF brojac = 0 THEN

    SIGNAL SQLSTATE "45103" SET MESSAGE\_TEXT = "Greška u ažuriranju, popust ne postoji!";

  ELSE

    UPDATE popusti SET postotak\_popusta = novi\_postotak WHERE id = popust\_id;

  END IF;

END //

DELIMITER ;

Opis: Procedura koja služi za ažuriranje već postojećih popusta. Prvo se provjerava da li traženi popust postoji. Ako ne postoji, baca grešku.

Poziv funkcije:

CALL azuriraj\_popust(41, 10);

## Procedure Fran

A computer screen shot of a code

Description automatically generated

Ova procedura omogućuje upravljanje povratima proizvoda s vremenskim ograničenjem (15 dana od datuma narudžbe). Ako pokušate izvršiti povrat nakon tog roka, postupak sprječava unos i generira odgovarajuću grešku. Ovaj pristup osigurava da se povrati mogu obraditi samo unutar dopuštenog vremenskog okvira.

A computer code with text

Description automatically generated

DodajStanjeIsporuke je procedura za praćenje isporuke narudžbi. Ona prima informacije o ID-u narudžbe, trenutnom stanju narudžbe, datumu narudžbe i načinu isporuke. Na osnovu načina isporuke, iz tablice vrste\_dostave preuzima trajanje dostave u danima i koristi ga za izračun predviđenog datuma isporuke. Nakon toga, unosi stanje narudžbe, datum ažuriranja i izračunati će datum isporuke u tablicu stanje\_isporuke.

A computer code with text

Description automatically generated

DodajPovratProizvoda je procedura koja omogućava unos podataka o povratu proizvoda u tablicu povrati\_proizvoda. Procedura prima četiri parametra: ID narudžbe, datum povrata, razlog povrata i status povrata.

A screenshot of a computer program

Description automatically generated

Procedure za rad s korisnicima obuhvatća tri funkcionalnosti: dodavanje, ažuriranje i brisanje korisnika. Procedura DodajKorisnika omogućava unos novog korisnika u tablicu korisnici, pri čemu prima ime, prezime, email i telefon korisnika. AzurirajKorisnika ažurira podatke postojećeg korisnika na osnovu njegovog ID-a, pri čemu se u tablicu mjenjaju vrijednosti za ime, prezime, email i telefon. ObrisiKorisnika omogućava brisanje korisnika iz baze podataka na osnovu njegovog ID-a.

## Procedure Josip

A computer screen shot of text

Description automatically generated

* Ova procedura omogućuje dodavanje novog zapisa u tablicu **placanja**.
* **Ulazni parametri:**
* p\_narudzba\_id: ID narudžbe kojoj pripada plaćanje.
* p\_nacin\_placanja: Način plaćanja (npr. 'kartica' ili 'pouzeće').
* p\_iznos: Iznos plaćanja.
* p\_datum\_placanja: Datum kada je plaćanje izvršeno.
* **Funkcija:** Procedura unosi novi zapis u tablicu **placanja** s podacima koji su proslijeđeni kroz ulazne parametre.

A computer screen shot of a black screen

Description automatically generated

* Ova procedura omogućuje dodavanje novog kupona u tablicu **kuponi**.
* **Ulazni parametri:**
* p\_kod: Jedinstveni kod kupona.
* p\_postotak\_popusta: Postotak popusta koji kupon omogućuje.
* p\_datum\_pocetka: Datum od kada je kupon aktivan.
* p\_datum\_zavrsetka: Datum do kada je kupon aktivan.
* p\_max\_iskoristenja: Maksimalan broj iskorištenja kupona.
* **Funkcija:** Procedura unosi novi zapis u tablicu **kuponi** s proslijeđenim parametrima.

## Procedure Leo

-- Procedura: Dodavanje korisnika (Leo)

DELIMITER //

CREATE PROCEDURE dodaj\_korisnika(

IN p\_ime VARCHAR(255),

IN p\_prezime VARCHAR(255),

IN p\_email VARCHAR(255),

IN p\_lozinka VARCHAR(255),

IN p\_adresa TEXT,

IN p\_grad VARCHAR(255),

IN p\_telefon VARCHAR(20)

)

BEGIN

IF EXISTS (SELECT 1 FROM korisnici WHERE email = p\_email) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Korisnik sa ovim email-om već postoji!';

ELSE

INSERT INTO korisnici (ime, prezime, email, lozinka, adresa, grad, telefon, tip\_korisnika, datum\_registracije)

VALUES (p\_ime, p\_prezime, p\_email, p\_lozinka, p\_adresa, p\_grad, p\_telefon, 'kupac', CURDATE());

END IF;

END//

DELIMITER ;

Procedura dodaj\_korisnika omogućava dodavanje novog korisnika u bazu podataka. Prije unosa podataka, provjerava da li već postoji korisnik s istim mailom kako bi se osigurala jedinstvenost. Ako korisnik s unesenim mailom već postoji, procedura generira SQL grešku pomoću SIGNAL SQLSTATE. U slučaju da mail ne postoji, novi korisnik se unosi u tablicu korisnici zajedno sa svim podacima: ime, prezime, email, lozinka, adresa, grad, telefon, tip korisnika i datum registracije. Ova procedura osigurava referencijalni integritet i sprječava dupliciranje podataka u bazi.

CREATE PROCEDURE prikazi\_preporuke(korisnik\_id INT, page INT, per\_page INT)

BEGIN

DECLARE offset\_value INT;

-- Izračunavanje offset-a

SET offset\_value = (page - 1) \* per\_page;

SELECT

p.id AS proizvod\_id,

p.naziv,

p.opis,

p.cijena

FROM proizvodi p

WHERE p.kategorija\_id IN (

-- Preporučujemo proizvode iz kategorija koje je korisnik dodao u wishlist ili kupio

SELECT DISTINCT p2.kategorija\_id

FROM proizvodi p2

JOIN wishlist w ON p2.id = w.proizvod\_id

WHERE w.korisnik\_id = korisnik\_id

UNION

SELECT DISTINCT p3.kategorija\_id

FROM proizvodi p3

JOIN stavke\_narudzbe s ON p3.id = s.proizvod\_id

JOIN narudzbe n ON s.narudzba\_id = n.id

WHERE n.korisnik\_id = korisnik\_id

)

AND p.id NOT IN (

-- Iznimamo proizvode koje je korisnik već dodao u wishlist

SELECT proizvod\_id

FROM wishlist

WHERE korisnik\_id = korisnik\_id

)

ORDER BY p.cijena DESC

LIMIT per\_page OFFSET offset\_value; -- Koristimo izračunati offset

END//

DELIMITER ;

Procedura „prikazi\_preporuke“ koristi podatke o korisničkim aktivnostima, poput proizvoda koje je korisnik dodao u wishlistu ili kupio, kako bi generirala personalizirane preporuke proizvoda. Na temelju tih podataka, aplikacija preporučuje proizvode iz kategorija koje su korisniku već bile zanimljive, čime se povećava relevantnost preporuka. Također, izuzimaju se proizvodi koje je korisnik već dodao u wishlistu, čime se izbjegava ponavljanje. Ova procedura implementira i paginaciju, omogućujući korisnicima da pregledavaju proizvode stranica po stranica, čime se smanjuje opterećenje aplikacije i poboljšava korisničko iskustvo.

Kod je osmišljen kako bi omogućio brzo i učinkovito dohvaćanje podataka o proizvodima koji pripadaju kategorijama koje je korisnik već pratio, bez potrebe za ponavljanjem složenih SQL upita u aplikaciji. Upit koristi UNION za kombiniranje kategorija proizvoda iz wishlist-a i narudžbi, dok ORDER BY sortira proizvode po cijeni. Paginacija se postiže korištenjem LIMIT i OFFSET, što omogućava prikazivanje samo određenog broja proizvoda po stranici. Na taj način, aplikacija ostaje efikasna i optimizirana, dok korisnici uživaju u personaliziranim i lako dostupnim preporukama.

-- Procedura: Brisanje korisnika (Leo)

DELIMITER //

CREATE PROCEDURE obrisi\_korisnika(

IN p\_korisnik\_id INT

)

BEGIN

DECLARE korisnik\_ima\_aktivne\_narudzbe BOOLEAN;

-- Provjera da li korisnik ima aktivne narudžbe koje nisu 'dostavljeno'

SELECT EXISTS (SELECT 1 FROM narudzbe WHERE korisnik\_id = p\_korisnik\_id AND status\_narudzbe != 'dostavljeno')

INTO korisnik\_ima\_aktivne\_narudzbe;

IF korisnik\_ima\_aktivne\_narudzbe THEN

SIGNAL SQLSTATE '45001'

SET MESSAGE\_TEXT = 'Korisnik ima aktivne narudžbe i ne može se obrisati!';

ELSE

-- Brisanje povezanih podataka u preporuceni\_proizvodi

DELETE FROM preporuceni\_proizvodi WHERE korisnik\_id = p\_korisnik\_id;

-- Brisanje povezanih recenzija

DELETE FROM recenzije\_proizvoda WHERE korisnik\_id = p\_korisnik\_id;

-- Brisanje povezanih računa

DELETE FROM racuni WHERE korisnik\_id = p\_korisnik\_id;

-- Brisanje povezanih podataka

DELETE FROM placanja WHERE narudzba\_id IN (SELECT id FROM narudzbe WHERE korisnik\_id = p\_korisnik\_id);

DELETE FROM stavke\_narudzbe WHERE narudzba\_id IN (SELECT id FROM narudzbe WHERE korisnik\_id = p\_korisnik\_id);

DELETE FROM narudzbe WHERE korisnik\_id = p\_korisnik\_id;

DELETE FROM wishlist WHERE korisnik\_id = p\_korisnik\_id;

-- Na kraju, brisanje korisnika

DELETE FROM korisnici WHERE id = p\_korisnik\_id;

END IF;

END //

DELIMITER ;

Ova SQL procedura „obrisi\_korisnika“ omogućava brisanje korisnika iz sustava, uz provjeru ima li korisnik aktivne narudžbe koje nisu označene kao 'dostavljeno'. Procedura prima jedan parametar: p\_korisnik\_id, koji predstavlja identifikator korisnika koji treba biti obrisan. Prvo, procedura provodi provjeru ima li korisnik aktivne narudžbe (koje nisu dostavljene). Ako korisnik ima aktivne narudžbe, procedura generira grešku s porukom „Korisnik ima aktivne narudžbe i ne može se obrisati!“. Ako korisnik nema aktivne narudžbe, procedura nastavlja s brisanjem svih povezanih podataka: recenzija, stavki narudžbi, narudžbi, preporučenih proizvoda, računa i wishlist-a. Na kraju, ako su svi povezani podaci izbrisani, korisnik se briše iz tablice korisnici. Ova procedura osigurava da se korisnici s aktivnim narudžbama ne mogu slučajno obrisati, čime se sprječavaju potencijalni gubici podataka i nepravilnosti u poslovnom procesu.

-- Procedura: Ažuriranje korsinika(Leo)

DELIMITER //

CREATE PROCEDURE azuriraj\_korisnika(

IN p\_korisnik\_id INT,

IN p\_ime VARCHAR(255),

IN p\_prezime VARCHAR(255),

IN p\_email VARCHAR(255),

IN p\_adresa TEXT,

IN p\_grad VARCHAR(255),

IN p\_telefon VARCHAR(20)

)

BEGIN

UPDATE korisnici

SET

ime = p\_ime,

prezime = p\_prezime,

email = p\_email,

adresa = p\_adresa,

grad = p\_grad,

telefon = p\_telefon

WHERE id = p\_korisnik\_id;

END//

DELIMITER ;

Procedura „azuriraj\_korisnika“ omogućava ažuriranje podataka korisnika u tablici korisnici. Procedura prihvaća podatke kao što su ime, prezime, email, adresa, grad i telefon, zajedno s jedinstvenim ID-jem korisnika. Koristeći SQL naredbu UPDATE, ažurira sve navedene podatke u tablici korisnici na temelju unesenog ID-ja. Ova procedura omogućava jednostavno i sigurno ažuriranje podataka korisnika u bazi, uz očuvanje postojećih veza i integriteta.

-- Procedura: Ažuriraj tip korisnika (Leo)

DELIMITER //

CREATE PROCEDURE azuriraj\_tip\_korisnika(IN p\_korisnik\_id INT, IN p\_tip\_korisnika ENUM('kupac', 'admin'))

BEGIN

UPDATE korisnici

SET tip\_korisnika = p\_tip\_korisnika

WHERE id = p\_korisnik\_id;

END//

DELIMITER ;

Ova SQL procedura „azuriraj\_tip\_korisnika“ omogućava ažuriranje tipa korisnika u sustavu. Procedura prima dva parametra: p\_korisnik\_id, koji označava identifikator korisnika kojem treba biti promijenjen tip, i p\_tip\_korisnika, koji predstavlja novi tip korisnika koji može biti 'kupac' ili 'admin'. Procedura ažurira vrijednost u koloni tip\_korisnika u tablici korisnici za korisnika sa specificiranim ID-em (p\_korisnik\_id). Ova procedura omogućava administratorima da lako mijenjaju status korisnika, na primjer, da promotivni korisnici postanu administratori ili obrnuto, bez potrebe za ručnim ažuriranjem podataka u bazi.

-- Procedura: Dodaj u wishlist (Leo)

DELIMITER //

CREATE PROCEDURE dodaj\_u\_wishlist(korisnik\_id INT, proizvod\_id INT, grupa VARCHAR(255))

BEGIN

-- Dodavanje proizvoda u wishlist (pretpostavljamo da je provjera duplikata već obavljena kroz triger)

INSERT INTO wishlist (korisnik\_id, proizvod\_id, grupa)

VALUES (korisnik\_id, proizvod\_id, grupa);

END//

DELIMITER ;

Ova SQL procedura „dodaj\_u\_wishlist“ omogućava dodavanje proizvoda u wishlist korisnika. Procedura prima tri parametra: korisnik\_id (identifikator korisnika), proizvod\_id (identifikator proizvoda) i grupa (kategorija ili grupa u koju proizvod pripada). Unutar procedure izvršava se SQL upit za unos novog proizvoda u tablicu wishlist s navedenim podacima. Pretpostavlja se da je provjera duplikata već obavljena kroz okidač (trigger) kako bi se osiguralo da isti proizvod ne može biti dodan više puta.

-- Procedura: Ukloni u wishlist (Leo)

DELIMITER //

CREATE PROCEDURE ukloni\_proizvod\_iz\_wishliste(p\_korisnik\_id INT, p\_proizvod\_id INT)

BEGIN

DELETE FROM wishlist WHERE korisnik\_id = p\_korisnik\_id AND proizvod\_id = p\_proizvod\_id;

END//

DELIMITER ;

Ova SQL procedura „ukloni\_proizvod\_iz\_wishliste“ omogućava uklanjanje proizvoda iz wishlist-a korisnika. Procedura prima dva parametra: korisnik\_id (identifikator korisnika) i proizvod\_id (identifikator proizvoda). Unutar procedure izvršava se SQL upit za brisanje odgovarajućeg proizvoda iz tablice wishlist koji pripada korisniku s navedenim korisnik\_id i proizvod\_id. Ova funkcionalnost omogućava korisnicima da uklone proizvode koje su prethodno dodali u svoju wishlist.

# Okidači

## Okidači Loren

**Okidač za obavijest o niskoj zalihi**

**CREATE TEMPORARY TABLE** privremene\_obavijesti (

poruka **TEXT**,

vrijeme\_kreiranja **DATETIME**

);

DELIMITER //

**CREATE TRIGGER** au\_ObavijestNiskaZaliha

**AFTER UPDATE ON** proizvodi

**FOR EACH ROW**

**BEGIN**

**IF NEW**.kolicina\_na\_skladistu < 5 **THEN**

**INSERT INTO** privremene\_obavijesti (poruka, vrijeme\_kreiranja)

**VALUES** (

**CONCAT**('Upozorenje: Zaliha za proizvod "', **NEW**.naziv, '" je pala ispod 5.'),

**NOW**()

);

**END IF**;

**END** //

DELIMITER ;

Okidač „au\_ObavijestNiskaZaliha“ automatski generira obavijest kada zaliha proizvoda padne ispod 5. Obavijest se sprema u privremenu tablicu „privremene\_obavijesti“ koju kreiramo prije samog okidača. Okidač provjerava je li nova količina manja od 5. Ako je, u privremenu tablicu „privremene\_obavijesti“ unosi poruku „Zaliha za proizvod 'proizvod' je pala ispod 5.“ i trenutni datum i vrijeme.

**Okidač koji ne dopušta recenziju ako korisnik nije kupio proizvod ili ako je već napisao recenziju**

DELIMITER //

**CREATE TRIGGER** bi\_RestrikcijaRecenzije

**BEFORE INSERT ON** recenzije\_proizvoda

**FOR EACH ROW**

**BEGIN**

**IF NOT EXISTS** (

**SELECT** 1

**FROM** stavke\_narudzbe sn

**JOIN** narudzbe n **ON** sn.narudzba\_id = n.id

**WHERE** sn.proizvod\_id = **NEW**.proizvod\_id **AND** n.korisnik\_id = **NEW**.korisnik\_id

) **THEN**

**SIGNAL SQLSTATE '45505'**

**SET MESSAGE\_TEXT** = 'Korisnik nije kupio ovaj proizvod i ne može ostaviti recenziju.';

**END IF**;

**IF EXISTS** (

**SELECT** 1

**FROM** recenzije\_proizvoda

**WHERE** proizvod\_id = **NEW**.proizvod\_id **AND** korisnik\_id = **NEW**.korisnik\_id

) **THEN**

**SIGNAL SQLSTATE '45506'**

**SET MESSAGE\_TEXT** = 'Korisnik je već ostavio recenziju za ovaj proizvod.';

**END IF**;

**END** //

DELIMITER ;

Okidač „bi\_RestrikcijaRecenzije“ sprječava dodavanje recenzije za proizvod ako korisnik nije kupio taj proizvod. Prije dodavanja novog podatka u tablicu „recenzije\_proizvoda“, provjerava se postoji li odgovarajuća narudžba u tablici „stavke\_narudzbe“, gdje se podudaraju „proizvod\_id“ iz narudžbe i „korisnik\_id“ iz tablice „recenzije\_proizvoda“. Ako podudaranje ne postoji, generira se greška "Korisnik nije kupio ovaj proizvod i ne može ostaviti recenziju". Dodatno se provjerava da li već postoji recenzija za isti „proizvod\_id“ i „korisnik\_id“ u tablici „recenzije\_proizvoda“. Ako postoji, generira se greška "Korisnik je već ostavio recenziju za ovaj proizvod".

## Okidači Morena

**Okidač za praćenje status isporuke određene narudžbe**

DELIMITER //

CREATE TRIGGER dodaj\_pracenje\_nakon\_narudzbe

AFTER INSERT ON narudzbe

FOR EACH ROW

BEGIN

INSERT INTO pracenje\_isporuka (narudzba\_id, status\_isporuke, datum\_isporuke)

VALUES (NEW.id, 'u pripremi', NULL);

END//

DELIMITER ;

Ovaj nam se okidač aktivira nakon umetanja novog retka u tablicu narudzbe. On automatski dodaje zapis u tablicu pracenje\_isporuke i postavlja vrijednost statusa isporuke „na cekanju“. Također postavlja vrijednost u stupcu datuma isporuke na NULL jer datum još nije poznat.

**Okidač automatski ažurira status narudžbe u tablici narudzbe na 'dostavljeno' kada se status isporuke u tablici pracenje\_isporuka promijeni na 'dostavljeno'.**

DELIMITER //

CREATE TRIGGER azuriraj\_status\_narudzbe\_na\_dostavljeno

AFTER UPDATE ON pracenje\_isporuka

FOR EACH ROW

BEGIN

IF NEW.status\_isporuke = 'dostavljeno' THEN

-- Ažurira status narudžbe na 'dostavljeno'

UPDATE narudzbe

SET status\_narudzbe = 'dostavljeno'

WHERE id = NEW.narudzba\_id;

END IF;

END //

DELIMITER ;

Ovaj se okidač aktivira nakon što se ažurira redak u tablici „pracenje\_isporuke“ na „dostavljeno“. Tada on automatski ažurira i „status\_narudzbe“ u tablici narudzbe.

**Okidač koji automatski ažurira status narudžbe u tablici narudzbe na 'poslano' kada se status isporuke u tablici pracenje\_isporuka promijeni na 'poslano'.**

DELIMITER //

0

CREATE TRIGGER azuriraj\_status\_narudzbe\_na\_poslano

AFTER UPDATE ON pracenje\_isporuka

FOR EACH ROW

BEGIN

IF NEW.status\_isporuke = 'poslano' THEN

-- Ažurira status narudžbe na 'poslano'

UPDATE narudzbe

SET status\_narudzbe = 'poslano'

WHERE id = NEW.narudzba\_id;

END IF;

END //

DELIMITER ;

**Okidač automatski vraća količinu proizvoda na skladište kada se povrati proizvod (kad se doda zapis u tablicu povrati\_proizvoda).**

DELIMITER //

CREATE TRIGGER povrat\_proizvoda\_skladiste

AFTER INSERT ON povrati\_proizvoda

FOR EACH ROW

BEGIN

-- Povećava količinu proizvoda na skladištu kad je proizvod vraćen

UPDATE proizvodi

SET kolicina\_na\_skladistu = kolicina\_na\_skladistu + (SELECT kolicina FROM stavke\_narudzbe WHERE id = NEW.stavka\_id)

WHERE id = (SELECT proizvod\_id FROM stavke\_narudzbe WHERE id = NEW.stavka\_id);

END //

DELIMITER ;

## Okidači Bruno

Okidač koji osigurava da proizvod može imati samo jedan aktivni popust

DELIMITER //

CREATE TRIGGER bi\_popusti

BEFORE INSERT ON popusti

FOR EACH ROW

BEGIN

  DECLARE brojac INTEGER;

  SELECT COUNT(\*) INTO brojac

    FROM popusti

    WHERE NEW.proizvod\_id = proizvod\_id AND NEW.datum\_pocetka <= datum\_zavrsetka AND NEW.datum\_zavrsetka >= datum\_pocetka;

    IF brojac > 0 THEN

    SIGNAL SQLSTATE '45100'

    SET MESSAGE\_TEXT = 'Popust za ovaj proizvod već postoji u

    zadanom razdoblju.';

    END IF;

END; //

DELIMITER ;

Opis: Okidač koji se poziva prije unosa novog popusta u tablicu. Osigurava jedinstvenost popusta za određeni proizvod.

Okidač koji osigurava automatsko brisanje popusta na proizvode kada istekne datum završetka

DELIMITER //

CREATE TRIGGER bi\_1\_popusti

BEFORE INSERT ON popusti

FOR EACH ROW

BEGIN

  IF NEW.datum\_zavrsetka < CURDATE() THEN

        DELETE FROM popusti WHERE id = NEW.id;

    END IF;

END; //

DELIMITER ;

Opis: Okidač koji se poziva prije unosa popusta u tablicu i osigurava automatsko brisanje popusta kada istekne datum završetka.

Okidač koji sprječava unos negativne cijene isporuke

DELIMITER //

CREATE TRIGGER bi\_nacini\_isporuke

BEFORE INSERT ON nacini\_isporuke

FOR EACH ROW

BEGIN

  IF NEW.cijena < 0 THEN

    SIGNAL SQLSTATE "45101" SET MESSAGE\_TEXT= "Cijena isporuke ne može biti negativna";

  END IF;

END//

DELIMITER ;

Opis: Okidač koji se poziva prije unosa načina isporuke u tablicu i osigurava da cijena načina isporuke ne može biti negativna.

## Okidači Fran

A screen shot of a computer code

Description automatically generated

postavi\_status\_narudzbe se koristi za automatsko postavljanje početnog statusa narudžbe na 'u obradi' prilikom unosa nove narudžbe, ako korisnik nije specificirao status.

Detalji triggera:

* Djelovanje:
  + Trigger se aktivira prije unosa nove narudžbe u tablicu "narudzbe", zahvaljujući klauzuli BEFORE INSERT.
  + Provodi se provjera na polje status\_narudzbe koje označava status narudžbe.
* Logika triggera
  + Ako novi unos status\_narudzbe nije specificiran (tj. NULL), trigger automatski postavlja status\_narudzbe na vrijednost 'u obradi'.
  + NEW označava redak koji se trenutno unosi u tablicu, pa tako NEW.status\_narudzbe referencira vrijednost statusa koja će biti unesen.
* Svrha:
  + Osigurava da sve nove narudžbe uvijek imaju početni status 'u obradi', osim ako je korisnik izričito postavio drugačiji status. Ovo može biti korisno za održavanje dosljednosti i za automatizaciju početne obrade narudžbi.

## Okidači Josip

A screen shot of a computer

Description automatically generated

* Ovaj okidač sprječava unos narudžbi ako je kupon već iskorišten maksimalan broj puta.
* Aktivira se **prije umetanja (BEFORE INSERT)** u tablicu **narudzbe**.
* **Koraci:**
* Provjerava koliko puta je kupon već iskorišten pomoću **COUNT** funkcije.
* Ako broj iskorištenja kupona (iskorištenja) premašuje maksimalan broj dozvoljenih iskorištenja (max\_iskoristenja), okidač zaustavlja unos i generira poruku greške.

A computer screen with white and blue text

Description automatically generated

* Ovaj okidač automatski postavlja trenutni datum (CURDATE()) kao datum izdavanja računa, ako datum nije ručno unesen.
* Aktivira se **prije umetanja (BEFORE INSERT)** u tablicu **racuni**.
* **Koraci:**
* Provjerava je li polje **datum\_izdavanja** prazno (NULL).
* Ako je prazno, automatski postavlja trenutni datum kao datum izdavanja.

## Okidači Leo

-- Okidač: Automatsko postavljanje datuma registracije korisnika (Leo)

DELIMITER //

CREATE TRIGGER postavi\_datum\_registracije

BEFORE INSERT ON korisnici

FOR EACH ROW

BEGIN

IF NEW.datum\_registracije IS NULL THEN

SET NEW.datum\_registracije = CURDATE();

END IF;

END//

DELIMITER ;

Ovaj okidač izvršava se prije svakog unosa (BEFORE INSERT) u tablicu korisnici. Njegova svrha je automatski postaviti vrijednost atributa datum\_registracije na trenutni datum (CURDATE()), ukoliko nije eksplicitno definirano prilikom unosa. Ovim se osigurava da svaki korisnik koji se unosi u bazu ima ispravno postavljen datum registracije. Okidač doprinosi dosljednosti i točnosti podataka u tablici korisnici, eliminirajući mogućnost unosa zapisa s nedostajućim ili neispravnim datumom.

-- Okidač: spreči duplikate wishlist (Leo)

DELIMITER //

CREATE TRIGGER spreči\_duplikate\_wishlist

BEFORE INSERT ON wishlist

FOR EACH ROW

BEGIN

IF EXISTS (

SELECT 1

FROM wishlist

WHERE korisnik\_id = NEW.korisnik\_id AND proizvod\_id = NEW.proizvod\_id

) THEN

SIGNAL SQLSTATE '45002'

SET MESSAGE\_TEXT = 'Proizvod je već u wishlist-u ovog korisnika!';

END IF;

END//

DELIMITER ;

Ovaj okidač se izvršava prije svakog unosa (BEFORE INSERT) u tablicu wishlist. Njegova svrha je spriječiti dodavanje duplikata u tablicu wishlist, tj. isti proizvod ne može biti dodan u wishlist istog korisnika više puta. Provjerava postojanje odgovarajućeg zapisa koristeći EXISTS s odgovarajućim kriterijima (korisnik\_id i proizvod\_id). Ukoliko zapis već postoji, okidač generira SQL grešku s porukom „Proizvod je već u wishlistu ovog korisnika!“ koristeći SIGNAL SQLSTATE. Ovaj okidač je ključan za održavanje integriteta podataka i sprječavanje redundantnih zapisa u tablici wishlist.