

Sustav za rezervaciju čuvara ljubimaca

Dokumentacija projekta iz Baze Podataka I

Nositelj kolegija doc. dr. sc. Goran Oreški

Izvođač Romeo Šajina, mag. inf

Tim Tim 33

Autori

Luka Vilagoš 0303114421

Patricia Lazić 0303114869

Dinko Nađ 0303113989

Paula Vorih 0303114507

Alma Reka 0303116340

Sadržaj

- [Sustav za rezervaciju čuvara ljubimaca](#)
- [1. Uvod](#)
 - [1.1. Opis Odabranog Procesa](#)
 - [1.2. Minimalni Zahtjevi](#)
 - [1.3. Cilj](#)
- [2. Opis poslovnog procesa](#)
- [3. ER Diagram](#)
- [4. Shema relacijskog modela](#)
- [5. Logičke shema baze podataka](#)
- [6. Tablice](#)
 - [6.1. Dinko Nađ](#)
 - [6.1.1. Tablica usluge](#)
 - [6.1.2. Tablica cuvar](#)
 - [6.1.3. Tablica aktivnost](#)
 - [6.1.4. Tablica cuvar aktivnost](#)
 - [6.2. Patricia Lazić](#)
 - [6.2.1. Tablica vrsta](#)
 - [6.2.2. Tablica pasmina](#)
 - [6.2.3. Tablica ljubimac](#)
 - [6.2.4. Tablica rezervacija ljubimac](#)
 - [6.3. Luka Vilagoš](#)
 - [6.3.1. Tablica vlasnik](#)
 - [6.3.2. Tablica tip](#)
 - [6.3.3. Tablica znacajka](#)
 - [6.3.4. Tablica ustanova znacajka](#)
 - [6.4. Paula Vorih](#)
 - [6.4.1. Tablica ustanova](#)
 - [6.4.2. Tablica cuvar ustanova](#)
 - [6.4.3. Tablica mjesto interesa](#)
 - [6.4.4. Tablica regija](#)
 - [6.5. Alma Reka](#)
 - [6.5.1. Tablica korisnik](#)
 - [6.5.2. Tablica korisnik ljubimac](#)
 - [6.5.3. Tablica rezervacija](#)
 - [6.5.4. Tablica recenzija](#)
- [7. Upiti](#)
 - [7.1. Dinko Nađ](#)

- [7.1.1. Upit](#)
 - [7.1.2. Upit](#)
 - [7.1.3. Upit](#)
 - [7.1.4. Upit](#)
- [7.2. Patricia Lazić](#)
 - [7.2.1. Upit](#)
 - [7.2.2. Upit](#)
 - [7.2.3. Upit](#)
 - [7.2.4. Upit](#)
- [7.3. Luka Vilagoš](#)
 - [7.3.1. Upit](#)
 - [7.3.2. Upit](#)
 - [7.3.3. Upit](#)
 - [7.3.4. Upit](#)
- [7.4. Paula Vorih](#)
 - [7.4.1. Upit](#)
 - [7.4.2. Upit](#)
 - [7.4.3. Upit](#)
 - [7.4.4. Upit](#)
- [7.5. Alma Reka](#)
 - [7.5.1. Upit](#)
 - [7.5.2. Upit](#)
 - [7.5.3. Upit](#)
 - [7.5.4. Upit](#)
- [8. Pogledi](#)
 - [8.1. Dinko Nađ](#)
 - [8.1.1. Pogled](#)
 - [8.1.2. Pogled](#)
 - [8.1.3. Pogled](#)
 - [8.1.4. Pogled](#)
 - [8.2. Patricia Lazić](#)
 - [8.2.1. Pogled](#)
 - [8.2.2. Pogled](#)
 - [8.2.3. Pogled](#)
 - [8.2. 4. Pogled](#)
 - [8.3. Luka Vilagoš](#)
 - [8.3.1. Pogled](#)
 - [8.3.2. Pogled](#)
 - [8.3.3. Pogled](#)

- [8.3.4. Pogled](#)
- [8.4. Paula Vorih](#)
 - [8.4.1. Pogled](#)
 - [8.4.2. Pogled](#)
 - [8.4.3. Pogled](#)
 - [8.4.4. Pogled](#)
- [8.5. Alma Reka](#)
 - [8.5.1. Pogled](#)
 - [8.5.2. Pogled](#)
 - [8.5.3. Pogled](#)
 - [8.5.4. Pogled](#)
- [9. Okidači](#)
 - [9.1. Luka Vilagoš](#)
 - [9.1.1. Okidač](#)
 - [9.2.2. Okidač](#)
- [10. Generiranje podataka](#)
 - [10.1. LLM](#)
 - [10.2. Python](#)
- [11. Zaključak](#)

1. Uvod

1.1. Opis Odabranog Procesa

Projekt koji smo modelirali odnosi se na sustav za rezervaciju čuvara ljubimaca. Ovaj sustav omogućuje korisnicima da rezerviraju usluge čuvara za svoje ljubimce u različitim ustanovama koje nude specifične značajke poput velikih dvorišta ili igrališta. Proces uključuje upravljanje informacijama o ustanovama, njihovim vlasnicima, značajkama ustanova, tipovima ustanova, te čuvarima koji su povezani s tim ustanovama. Korisnici također mogu ocjenjivati ustanove i usluge koje koriste, što pomaže u održavanju kvalitete i zadovoljstva korisnika.

1.2. Minimalni Zahtjevi

Baza Podataka Struktura baze podataka koja podržava sve relevantne entitete kao što su ustanove, vlasnici, značajke, tipovi ustanova, čuvari, usluge, korisnici i rezervacije.

Veze među Entitetima Jasno definirane veze između entiteta, uključujući mnogostruke veze između čuvara i ustanova te validacija da je usluga povezana s čuvarom koji radi u odabranoj ustanovi.

Funkcionalnost Rezervacija Mogućnost korisnika da kreiraju rezervacije koje uključuju validaciju povezanosti usluge i ustanove putem čuvara.

Ocjenjivanje Mehanizam koji omogućuje korisnicima da ocjenjuju ustanove i automatsko ažuriranje prosječne ocjene ustanove.

1.3. Cilj

Cilj ovog sustava je omogućiti jednostavan i učinkovit način za korisnike da pronađu i rezerviraju usluge čuvara za svoje ljubimce u ustanovama koje zadovoljavaju njihove specifične potrebe. Sustav treba osigurati visoku razinu integriteta podataka kroz validaciju rezervacija i omogućiti transparentne ocjene ustanova kako bi korisnici mogli donositi informirane odluke. Također, sustav treba olakšati administraciju ustanova, upravljanje čuvarima, te praćenje i analizu korisničkih ocjena i rezervacija.

2. Opis poslovnog procesa

Poslovni proces koji smo ovdje modelirali poprilično je složen. Uveli smo pojednostavljenja u skoro sve, ako ne i sve dijelove njega, unatoč tome, vjerujemo da je proces dovoljno detaljan da se može vidjeti ideja i pokazati kompetentnost zamišljene tvrtke.

Kako se naš sustav temelji na rezervaciji čuvara ljubimaca, morali smo imati neku reprezentaciju ustanova, vlasnika i značajki u bazi podataka. To smo postigli s relacijama ustanova, vlasnik, tip i značajka, koje zajedno čine srž našeg modela.

Relacija ustanova čuva osnovne informacije o svakoj ustanovi kao što su naziv, adresa, broj telefona, email, radno vrijeme, opis, web stranica, kapacitet i ocjena. Svaka ustanova također ima svoj tip koji je povezan s relacijom tip. Ovdje smo definirali različite tipove ustanova, poput 'Pet Spa', 'Čuvanje pasa', itd., koji pružaju specifične usluge.

Relacija vlasnik sadrži osnovne podatke o vlasnicima ustanova, uključujući njihovo ime, prezime, broj telefona, email, datum rođenja, adresu, grad, državu i spol. Vlasnici su ključni za upravljanje ustanovama te stoga održavamo detaljne informacije o njima.

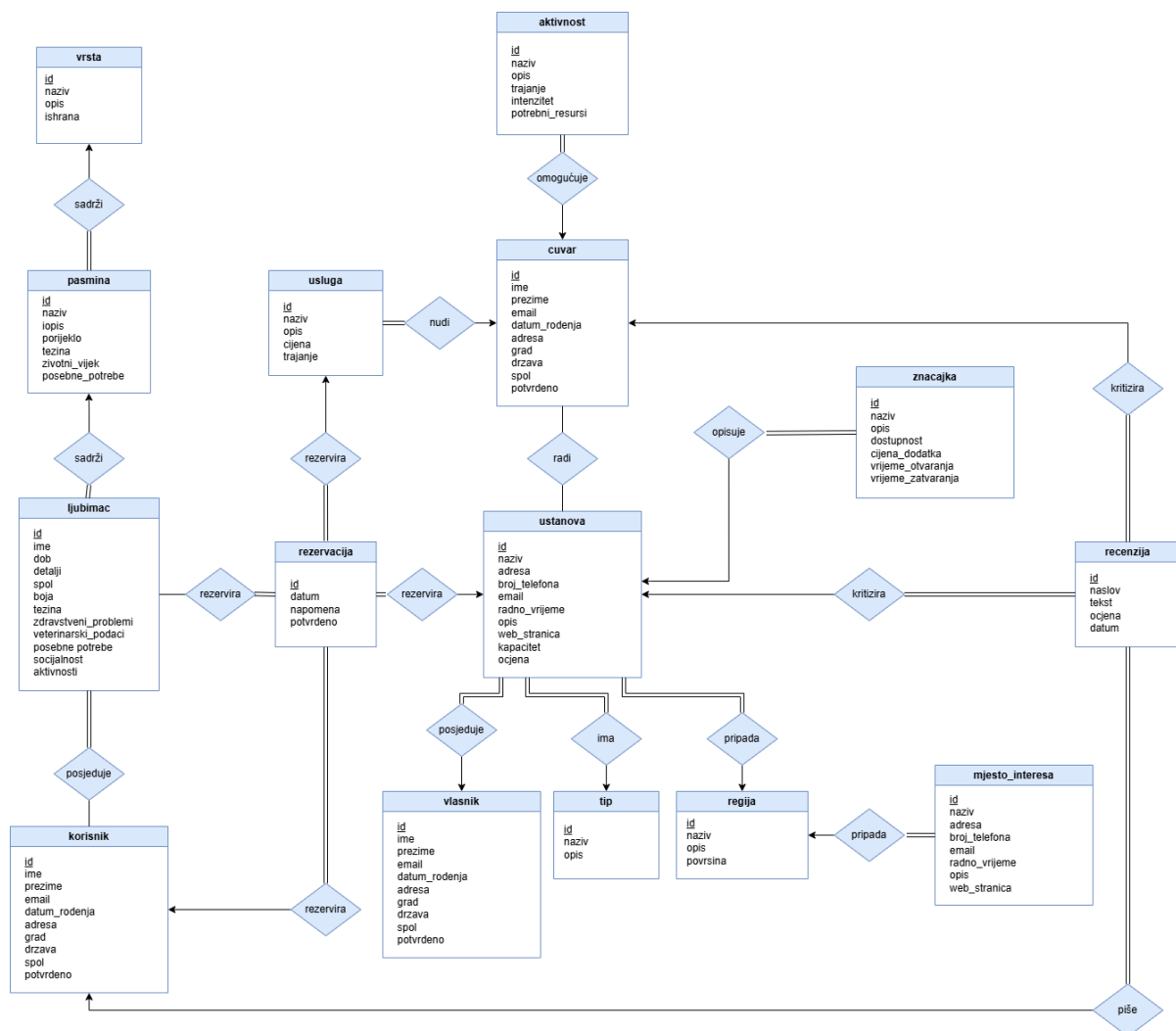
Relacija značajka opisuje različite značajke koje ustanova može imati, poput 'Veliko dvorište', 'Igralište', itd. Za svaku značajku bilježimo naziv, opis, vrijeme otvaranja i zatvaranja te dostupnost. Relacija ustanova_značajka povezuje ustanove s njihovim značajkama, omogućujući nam da identificiramo koje značajke su dostupne u kojoj ustanovi.

Pogledi koje smo kreirali pružaju korisne informacije i pojednostavljuju pristup podacima. Pogled `ustanova_tip_vlasnik_s_igralištem` prikazuje sve relevantne podatke o ustanovama s značajkom 'Igralište'. Pogled `broj_ustanova_tipa_pet_spa_odredenog_vlasnika` pokazuje sve vlasnike koji imaju ustanove tipa 'Pet Spa' i koliko takvih ustanova imaju. Pogled `ustanova_detalji` daje detaljan prikaz svih podataka o ustanovama, uključujući podatke o tipu, vlasniku i značajkama. Pogled `vlasnici_iz_zagreba_i_njihove_ustanove` prikazuje sve vlasnike iz Zagreba i njihove ustanove.

Sve ove relacije i pogledi omogućuju nam da efikasno pratimo i upravljamo ustanovama, njihovim vlasnicima i značajkama, čime osiguravamo visoku razinu usluge za korisnike našeg sustava. Naš model omogućava detaljnu analizu i pregled podataka, što je ključno za uspješno vođenje poslovanja u sektoru rezervacija čuvara ljubimaca.

3. ER Diagram

ER diagram izrađen u [draw.io \(drawio.com\)](https://draw.io) alatu. Detaljno prikazuje odnose između relacija standardnom notacijom.



4. Shema relacijskog modela

vrsta(id, naziv, opis, ishrana)

pasmina(id, naziv, vrsta_id, opis, porijeklo, tezina, zivotni_vijek, posebne_potrebe)

ljubimac(id, ime, pasmina_id, dob, detalji, spol, boja, tezina, zdravstveni_problemi, veterinarski_podaci, posebne_potrebe, socijalnost, aktivnosti)

korisnik(id, ime, prezime, broj_telefona, email, datum_rodenja, adresa, grad, drzava, spol, potvrdeno)

korisnik_ljubimac(id, korisnik_id, ljubimac_id)

znacajka(id, naziv, opis, dostupnost, cijena_dodatka, vrijeme_otvaranja, vrijeme_zatvaranja)

tip(id, naziv, opis)

vlasnik(id, ime, prezime, broj_telefona, email, datum_rodenja, adresa, grad, drzava, spol, potvrdeno)

regija(id, naziv, opis, površina)

mjesto_interesa(id, naziv, adresa, broj_telefona, email, radno_vrijeme, opis, web_stranica, regija_id)

ustanova(id, naziv, adresa, broj_telefona, email, radno_vrijeme, opis, web_stranica, kapacitet, ocjena, regija_id, tip_id, vlasnik_id)

ustanova_znacajka(id, ustanova_id, znacajka_id)

aktivnost(id, naziv, opis, trajanje, intenzitet, potrebni_resursi)

cuvar(id, ime, prezime, broj_telefona, email, datum_rodenja, adresa, grad, drzava, spol, potvrdeno)

cuvar_aktivnost(id, cuvar_id, aktivnost_id)

cuvar_ustanova(id, cuvar_id, ustanova_id)

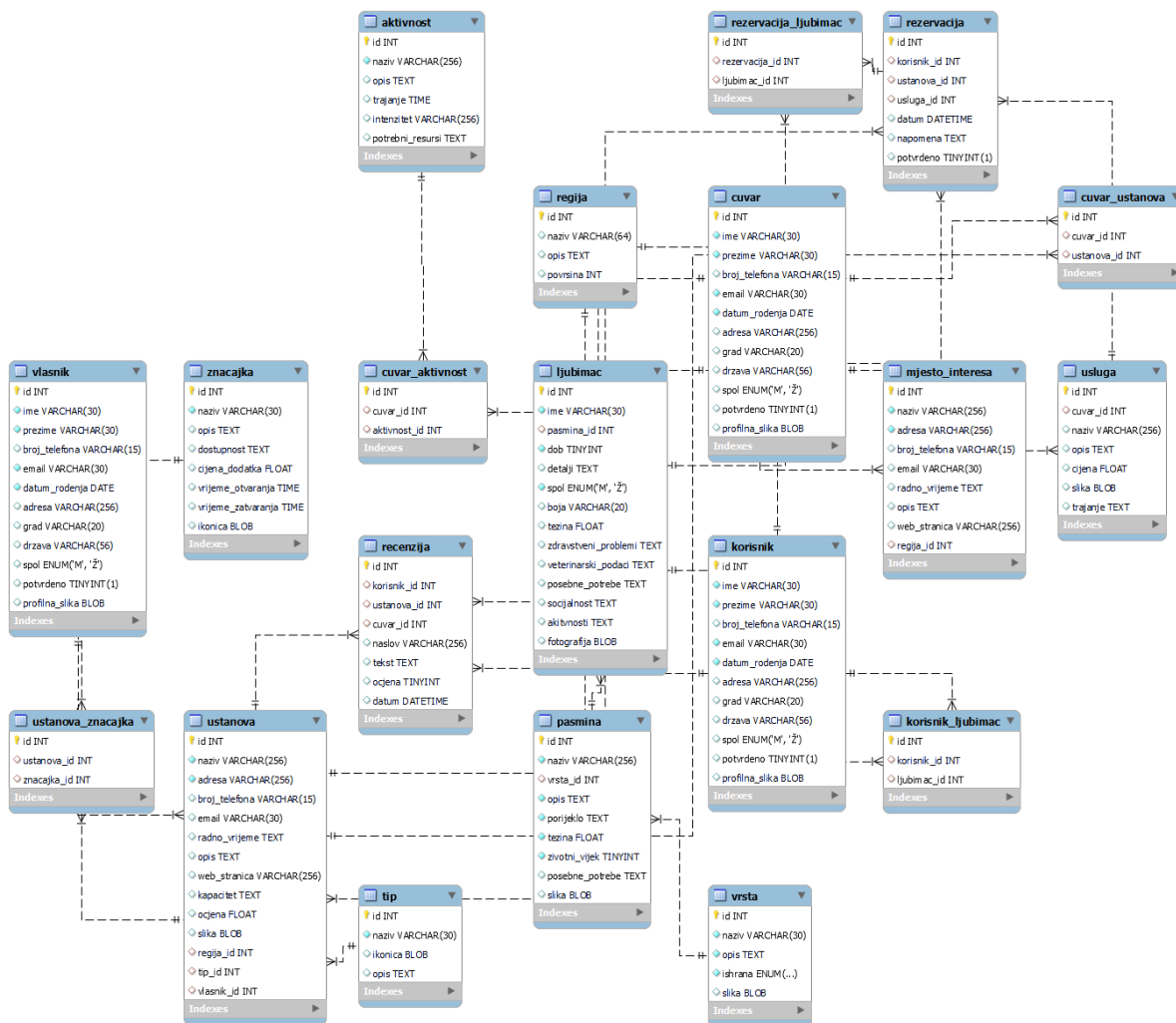
usluga(id, cuvar_id, naziv, opis, cijena, trajanje)

recenzija(id, korisnik_id, ustanova_id, cuvar_id, naslov, tekst, ocjena, datum)**rezervacija**(id, korisnik_id, ustanova_id, usluga_id, datum, napomena, potvrđeno)

```
rezervacija_ljubimac(id, rezervacija_id, ljubimac_id)
```

5. Logičke shema baze podataka

Logička shema je generirana u MySQL Workbenchu. Prikazuje detaljan prikaz baze podataka, svih relacija, veza te ograničenja.



6. Tablice

6.1. Dinko Nađ

6.1.1. Tablica usluge

Tablica "usluga" predstavlja pojedine usluge koje čuvari nude. Svaka usluga se sastoji od atributa: id,cuvar_id, naziv, opis, cijena i trajanje.

Atribut "id" je tipa INT s AUTO_INCREMENT svojstvom, što znači da se vrijednost ovoga atributa automatski povećava za svaku novu uslugu. Ovaj atribut je definiran kao PRIMARY KEY jer označava specifičnu brojčanu vrijednost svake usluge.

Atribut "cuvar_id" označava identifikator entiteta "cuvar" koji je povezan s uslugom te je označen sa INT. Ograničenje FOREIGN KEY osigurava da se svaki unos u

atributu "cuvar_id" referencira na postojeći unos u tablici "cuvar", pri čemu se na brisanje entiteta u tablici "cuvar" postavlja vrijednost "cuvar_id" na NULL (ON DELETE SET NULL).

Atribut "naziv" je tipa VARCHAR(256) koji ima maksimalnu duljinu od 256 znakova. Prednost VARCHARA je što on zauzima samo onoliko memorije koliko je potrebno za svaki unos. Ograničenje NOT NULL koji osigurava da svaka usluga ima svoj naziv.

Atribut "opis" je tipa TEXT te služi za detaljniji opis usluge koja se nudi. TEXT je tip podataka koji se koristi za pohranu veće količine tekstualnih podataka. TEXT nema unaprijed definiranu maksimalnu duljinu.

Atribut "cijena" predstavlja cijenu usluge. Postavljena je na FLOAT s 2 decimalna mjesta i ima ograničenje NOT NULL koja osigurava da svaka usluga ima svoju cijenu.

Atribut "Trajanje" je tipa TEXT koje opisuje koliko traje određena usluga.

```
CREATE TABLE usluga (  
    id INT AUTO_INCREMENT,  
    cuvar_id INT,  
    naziv VARCHAR (256),  
    opis TEXT,  
    cijena FLOAT(2),  
    trajanje TEXT,  
  
    CONSTRAINT PRIMARY KEY (id),  
    CONSTRAINT FOREIGN KEY (cuvar_id) REFERENCES cuvar (id) ON DELETE SET NULL  
);
```

6.1.2. Tablica cuvar

Tablica "cuvar" predstavlja osobe koje čuvaju životinje. Svaki čuvar se sastoji od atributa: id, ime, prezime, broj_telefona, email, datum_rođenja, adresu, grad, drzavu, spol i potvrđeno.

Atribut "id" je tipa INT s AUTO_INCREMENT svojstvom, što znači da se vrijednost ovoga atributa automatski povećava za svakog novog čuvara. Ovaj atribut je definiran kao PRIMARY KEY jer označava specifičnu brojčanu vrijednost čuvara.

Atributi "ime" i "prezime" su tipa VARCHAR(30) koji ima maksimalnu duljinu od 30 znakova, ali zauzima samo upisani dio u memoriju. Imaju ograničenje od NOT NULL koji omogućava da svaki čuvar mora imati svoje ime i prezime.

Atribut "broj_telefona" je tipa VARCHAR(15) te mu je maksimalna duljina unosa 15 znakova.

Atribut "email" je tipa VARCHAR(30) i koristi se za pohranu email adrese čuvara. Maksimalna duljina unosa je 30 znakova, što bi trebalo biti dovoljno za većinu email adresa. Ograničenje NOT NULL osigurava da svaki čuvar mora imati unesen email.

Atribut "datum_rođenja" je tipa DATE i koristi se za pohranu datuma čuvara. Ovaj tip podataka omogućuje lakši rad sa datumima prilikom pisanja upita. Najčešće pohranjuje datume u formatu godina-mjesec-dan. Ograničenje NOT NULL osigurava da svaki čuvar mora imati uneseni datum rođenja.

Atribut "adresa" je tipa VARCHAR(256) te nam omogućuje pohranu adrese čuvara. Maksimalna duljina je 256 znakova.

Atribut "grad" je tipa VARCHAR(20) koji nam omogućuje unos grada iz kojeg čuvar dolazi. Maksimalna duljina je 20 znakova.

Atribut "drzava" je tipa VARCHAR(56). Omogućava nam unos države iz koje dolazi čuvar. Maksimalna duljina je 56 znakova.

Atribut "spol" je tipa ENUM koja može uzeti samo jednu od ponuđenih vrijednost 'M' ili 'Ž', označavajući muški ili ženski spol čuvara.

Atribut "potvrđeno" je tipa BOOL koji označava jel čuvar potvrđen u sustavu ili nije. Sa 1 bit će označeno da je čuvar u sustavu, a sa 0 da nije.

```
CREATE TABLE cuvar (  
    id INT AUTO_INCREMENT,  
    ime varchar(30) NOT NULL,  
    prezime varchar(30) NOT NULL,  
    broj_telefona VARCHAR(15),  
    email VARCHAR(30) NOT NULL,  
    datum_rodenja DATE NOT NULL,  
    adresa VARCHAR(256),  
    grad VARCHAR(20),  
    drzava VARCHAR(56),  
    spol ENUM('M', 'Ž'),  
    potvrđeno BOOL,  
  
    CONSTRAINT PRIMARY KEY (id)  
);
```

6.1.3. Tablica aktivnost

Tablica "aktivnost" predstavlja aktivnosti koje nude čuvari životinja. Tablica se sastoji od atributa: id, naziv, opis, trajanje, intenzitet i potrebni_resursi.

Atribut "id" je tipa INT s AUTO_INCREMENT svojstvom, što znači da se vrijednost ovoga atributa automatski povećava za svaku novu aktivnost. Ovaj atribut je definiran kao PRIMARY KEY jer označava specifičnu brojčanu vrijednost aktivnosti.

Atribut "naziv" je tipa VARCHAR(256) u kojem se upisuje naziv aktivnosti koje nude čuvari. Maksimalna duljina je 256 znakova. Ograničenje NOT NULL osigurava da naziv aktivnosti bude popunjen.

Atribut "opis" je tipa TEXT te nam služi za malo detaljniji opis aktivnosti. U TEXT tip moguće je pohraniti velike količine podataka te ne moramo unaprijed definirati veličinu.

Atribut "trajanje" je tipa TIME i označava koliko vremena će trajati pojedina aktivnost. Sa ograničenjem NOT NULL osigurali smo da svaka aktivnost mora imati svoje vrijeme trajanja.

Atributom "intenzitet" odredili smo kojim se intenzitetom izvodi određena aktivnost. Tipa je VARCHAR(256) te ima maksimalnu duljinu od 256 znakova.

Atribut "potrebni_resursi" je tipa TEXT te služi da bi ljudi znali sta treba čuvaru da obavi određenu aktivnost.

```
CREATE TABLE aktivnost (
    id INT AUTO_INCREMENT,
    naziv varchar(256) NOT NULL,
    opis TEXT,
    trajanje TIME,
    intenzitet VARCHAR(256),
    potrebni_resursi TEXT,

    CONSTRAINT PRIMARY KEY (id)
);
```

6.1.4. Tablica cuvar_aktivnost

Ova tablica omogućuje vezu između čuvara i aktivnosti. Ova tablica sadrži attribute: id, cuvar_id, aktivnost_id.

Atribut "id" je tipa INT s AUTO_INCREMENT svojstvom, što znači da se vrijednost ovoga atributa automatski povećava za svaku novu vezu između čuvara i aktivnosti. Ovaj atribut je definiran kao PRIMARY KEY jer označava specifičnu brojčanu vrijednost.

Atribut "cuvar_id" je strani ključ koji smo postigli ograničenjem FOREIGN KEY koji referencira stupac "id" u tablici "cuvar". To znači da svaki unos u ovom stupcu mora biti valjani "id" čuvara iz tablice "cuvar". Ograničenje "ON DELETE SET NULL" znači da ako se obriše redak u tablici "cuvar", vrijednost ovog stupca bit će postavljena na "NULL" u svim povezanim redovima u tablici "cuvar_aktivnost".

Atribut "aktivnost_id" je strani ključ koji smo postigli ograničenjem FOREIGN KEY koji referencira stupac "id" u tablici "aktivnost". Svaki unos u ovom stupcu mora biti valjani "id" aktivnosti iz tablice "aktivnost". Ograničenje "ON DELETE SET NULL" znači da ako se obriše redak u tablici "aktivnost", vrijednost ovog stupca bit će postavljena na "NULL" u svim povezanim redovima u tablici "cuvar_aktivnost".

```
CREATE TABLE cuvar_aktivnost (
    id INT AUTO_INCREMENT,
    cuvar_id INT,
    aktivnost_id INT,

    CONSTRAINT PRIMARY KEY (id),
    CONSTRAINT FOREIGN KEY (cuvar_id) REFERENCES cuvar (id) ON DELETE SET NULL,
    CONSTRAINT FOREIGN KEY (aktivnost_id) REFERENCES aktivnost (id) ON DELETE SET NULL
);
```

6.2. Patricia Lazić

6.2.1. Tablica vrsta

Tablica „vrsta“ odnosi se na vrstu korisnikove životinje. Sadrži attribute: id, naziv, opis i ishrana.

Atribut „id INT AUTO_INCREMENT“ definiran je tipom INT jer pohranjuje brojčanu vrijednost koja je jedinstvena za svaku pojedinu vrstu. Atribut sadrži AUTO_INCREMENT svojstvo čime automatski generira jedinstvenu vrijednost za svaki redak umetnut u tablicu. Najčešće se koristi za primarne ključeve jer time omogućujemo da se sustav baze podataka automatski obrađuje bez potrebe za

ručnim unošenjem jedinstvenog cijelog broja za svaki novi red.

Atribut „naziv“ označava ime svake pojedine vrste. Za ovaj atribut odabran je tip podatka VARCHAR(30). Ova vrsta podataka omogućava dinamičku alokaciju memorije, što znači da će se koristiti samo onoliko memorije koliko je potrebno za svaki unos, umjesto da se rezervira fiksna količina memorije za svaki zapis. Broj u zagradama, (30), navodi maksimalnu duljinu niza koji se može pohraniti u ovom stupcu što je 30 znakova u suprotnom će izbaciti error. NOT NULL je ograničenje koje onemogućuje da stupac sadrži NULL vrijednost pri čemu svaki red u tablici mora imati vrijednost za ovaj stupac i ne može ostati prazan.

Atribut „opis“ je tipa TEXT koji služi za unos opisa vrste. TEXT je tip podataka koji se koristi za pohranu velikih količina tekstualnih podataka. Za razliku od VARCHAR-a, koji ima navedenu maksimalnu duljinu. TEXT je prikladan za pohranjivanje duljih nizova bez unaprijed definirane maksimalne duljine. Također sadrži NOT NULL ograničenje koje osigurava da stupac ne može imati praznu vrijednost, odnosno NULL.

Atribut „ishrana“ sadrži ENUM tip podatka koji se koristi za definiranje stupca koji može sadržavati samo fiksni skup vrijednosti. Svaka vrijednost u ENUM-u predstavlja jednu od dopuštenih opcija za taj stupac što u prijevodu znači da će prihvatiti samo jednu od unaprijed definiranih opcija ('mesožder', 'biljožder', 'svežder'). Sadrži NOT NULL pri čemu je obavezno definirati ishranu.

```
CREATE TABLE vrsta (  
  id INT AUTO_INCREMENT,  
  naziv VARCHAR(30) NOT NULL,  
  opis TEXT NOT NULL,  
  ishrana ENUM('mesožder', 'biljožder', 'svežder') NOT NULL,  
  CONSTRAINT PRIMARY KEY (id)  
);
```

6.2.2. Tablica pasmina

Tablica „pasmina“ sadržava podatke o skupini životinja unutar iste vrste koje dijele slične genetske karakteristike i fizičke osobine. Tablica sadržava attribute: id, naziv, vrsta_id, opis, porijeklo, težina, životinjski_vijek te posebne potrebe.

Atribut „id INT AUTO_INCREMENT“ definira prvi stupac pod nazivom „id“ s tipom podatka integer. Određen je kao primarni ključ te se automatski povećava za svaki novi red umetnut u tablicu. AUTO_INCREMENT je svojstvo koje automatski generira jedinstvenu vrijednost za svako novi red umetnut u tablicu. Kada je novi redak umetnut u tablicu, a vrijednost nije eksplicitno navedena za stupac „id“, MySQL automatski dodjeljuje sljedeću sekvencijalnu vrijednost cijelog broja stupcu.

Atribut „naziv“ se koristi za pohranjivanje informacija o nazivu entiteta to jest korisnikove pasmine. Sadržava tip podatka VARCHAR(256) koji označava varijabilni niz znakova. VARCHAR omogućava pohranu niza znakova promjenjive duljine s maksimalnom duljinom 256 znakova. Sadrži ograničenje NOT NULL pri čemu je potrebno ispuniti atribut „naziv“.

Atribut „vrsta_id INT“ definira treći stupac pod nazivom „vrsta_id“ s tipom podatka integer. Namijenjen je za držanje referenci na primarni ključ druge relacije putem ograničenja stranog ključa. Ograničenje CONSTRAINT FOREIGN KEY (vrsta_id) REFERENCES vrsta (id) ON DELETE SET NULL osigurava referentni integritet između dvije tablice što znači da svaka vrijednost u stupcu vrsta_id tablice pasmina mora postojati u stupcu id tablice vrsta. Također, ON DELETE SET NULL specificira ponašanje kada se referencirani redak u tablici vrsta izbriše. U tom slučaju, postavlja vrijednost vrsta_id na NULL u pasmina tablici, osiguravajući održavanje referentnog integriteta.

Ovaj odnos omogućava lakšu organizaciju podataka i dohvaćanje na temelju povezanih informacija.

Atribut „opis“ predstavlja opširan opis korisnikove pasmine. Koristi tip podatka TEXT koji se koristi za pohranu velikih količina tekstualnih podataka i ujedno osigurava da svaki zapis mora imati vrijednost koja nije NULL korištenjem NOT NULL ograničenja.

Atribut „porijeklo“ označava porijeklo korisnikove pasmine kojeg je tipa TEXT pri čemu se može opisati detaljno porijeklo pasmine. Također sadrži ograničenje NOT NULL zbog čega svaka pasmina mora imati jasno definirano svoje porijeklo.

Atribut „težina“ specificira težinu korisnikove pasmine. Atribut koristi decimalni tip podatka FLOAT te broj u zagradama, u ovom slučaju, (2), predstavlja širinu prikaza to jest pohranu brojeva sa pomičnim zarezom sa širinom prikaza 2. Koristi ograničenje NOT NULL pri čemu je nužno ispuniti redak u tablici sa vrijednošću tipa float.

Atribut „životni_vijek“ predstavlja životni vijek ili očekivani životni vijek korisnikove pasmine. Sadržava TINYINT tip podatka koji se najčešće koristi za pohranjivanje cijelih brojeva unutar malog raspona. Obično zahtijeva 1 bajt pohrane i može predstavljati vrijednosti od 0 do 255 sa UNSIGNED TINYINT. UNSIGNED je atribut u SQL-u koji se koristi za označavanje da stupac može pohraniti samo pozitivne vrijednosti cijelog broja koje predstavljaju životni vijek zbog čega vrijednost ne može biti negativna. Također osigurava da svaki zapis mora imati vrijednost koja nije NULL pri čemu svaka pasmina ima svoj životni vijek.

Atribut „posebne_potrebe“ sadržava posebne potrebe pasmine. Definiran je tipom TEXT pri čemu može pohraniti veliku količinu podataka, koji potencijalno predstavljaju posebne potrebe ili zahtjeve, i nema nikakva ograničenja što znači da dopušta prisutnost NULL vrijednosti. Ova fleksibilnost u dopuštanju NULL vrijednosti može biti korisna kada se radi s opcionalnim informacijama jer ne mora biti da svaka pasmina ima posebne potrebe i ograničenja.

```
CREATE TABLE pasmina (  
    id INT AUTO_INCREMENT,  
    naziv VARCHAR(256) NOT NULL,  
    vrsta_id INT,  
    opis TEXT NOT NULL,  
    porijeklo TEXT NOT NULL,  
    tezina FLOAT(2) NOT NULL,  
    zivotni_vijek TINYINT UNSIGNED NOT NULL,  
    posebne_potrebe TEXT,  
  
    CONSTRAINT PRIMARY KEY (id),  
    CONSTRAINT FOREIGN KEY (vrsta_id) REFERENCES vrsta (id) ON DELETE SET NULL  
);
```

6.2.3. Tablica ljubimac

Tablica „ljubimac“ sadržava podatke o korisnikovu ljubimcu. Uključuje attribute: id, ime, pasmina_id, dob, detalji, spol, boja, težina, zdravstveni problemi, veterinarski podaci, posebne potrebe, socijalnost i aktivnost.

Atribut „id“ je tipa INT te se koristi za jedinstvenu identifikaciju svakog ljubimca u bazi podataka. Ograničenje CONSTRAINT PRIMARY KEY (id) definira id stupac kao primarni ključ tablice čime pomoću AUTO INCREMENT svojstva automatski generira jedinstvene, sekvencijalne vrijednosti za svaki novi red umetnut u tablicu.

Atribut „ime“ se koristi za pohranu imena kućnih ljubimaca. Sadržava vrstu podataka VARCHAR, varijabilni znak, što znači da stupac može pohraniti nizove promjenjive duljine. Maksimalna duljina mu je 30 znakova te sadržava NOT NULL ograničenje što osigurava da vrijednost mora biti navedena za ovaj stupac u svakom redu.

Atribut „pasma_id“ je tipa podatka INT i koristi se za stvaranje odnosa između tablice ljubimac i tablice pasmina.. Stupac pasmina_id je strani ključ koji upućuje na stupac id (primarni ključ) u tablici pasmina i time se uspostavlja poveznica između dvije tablice, pokazujući da svaki ljubimac pripada određenoj pasmini. CONSTRAINT FOREIGN KEY (pasma_id) definira ograničenje stranog ključa na atributu pasmina_id. ON DELETE SET NULL određuje da ako se referencirani redak u tablici pasmina izbriše, pasmina_id u tablici ljubimac će biti postavljen na NULL.

Atribut „dob“ označava datum rođenja ili dob korisnikovog ljubimca. Sadržava tip podatka TINYINT koji je UNSIGNED što znači da dopušta samo ne-negativne vrijednosti, od 0 do 225 što je i više nego dovoljno godina za starost ljubimca. Sadržava NOT NULL ograničenje čime onemogućuje da vrijednost bude NULL.

Atribut „detalji“ označava neke specifične značajke ljubimaca. Tip podatka je TEXT čime je moguće pohraniti veliku količinu tekstualnih podataka. Ne sadrži nikakva ograničenja pri tome dopušta prisutnost NULL vrijednosti.

Atribut „spol“ je ENUM tip podatka koji je specifičan po tome što ograničava vrijednosti koje se mogu unijeti u stupac na unaprijed definirani skup opcija. U ovom slučaju su to vrijednosti 'M' i 'Ž' koje označavaju spol ljubimca. Sadrži NOT NULL ograničenje pri čemu je potrebno definirati spol.

Atribut „boja“ označava boju ljubimca. Za ovaj atribut odabran je tip podatka VARCHAR koji omogućuje maksimalnu duljinu od 20 znakova. Ne sadrži nikakva ograničenja stoga je moguća prisutnost NULL vrijednosti.

Atribut „težina“ specificira težinu korisnikovog ljubimca. Atribut koristi decimalni tip podatka FLOAT i širinu prikaza koji iznosi 2. Sadrži NOT NULL ograničenje te je nužno ispuniti redak u tablici.

Atribut „zdravstveni_problemi“ označava zdravstvene poteškoće i moguće smetnje u svakodnevnom životu. Koristi tip podatka TEXT koji omogućuje pohranu velike količine tekstualnih podataka bez unaprijed definirane maksimalne duljine. Ne sadrži nikakva ograničenja i time naznačuje da vrijednost može biti NULL to jest da ljubimci koji nemaju zdravstvene probleme mogu ostaviti redak praznim.

Atribut „veterinarski_podaci“ sadržava značajne podatke i napute od veterinara kojih se treba pridržavati. Definiran je tipom podatka TEXT te nema nikakva ograničenja.

Atribut „posebne_potrebe“ specificira dodatne zahtjeve ili uvjete koji se odnose na fizičke ili ponašajne potrebe ljubimca. Definiran je tipom podatka TEXT i nema nikakva ograničenja.

Atribut „socijalnost“ je tipa podatka TEXT te se odnosi na sposobnost i sklonost kućnih ljubimaca, za interakciju s ljudima i drugim životinjama. To uključuje podložnost afekciji, komunikaciji i prilagodljivosti. Nema nikakva ograničenja.

Atribut „aktivnosti“ sadržava podatke o različitim aktivnostima i stanjima ljubimaca. Na primjer, može sadržavati informacije poput šetnje, igre, odmora, spavanja itd. Definiran je tipom TEXT i time se svaka od ovih aktivnosti može biti zabilježena kao zasebna vrijednost. Nema nikakva ograničenja.

```
CREATE TABLE ljubimac (  
  id INT AUTO_INCREMENT,
```

```

ime VARCHAR(30) NOT NULL,
pasmina_id INT,
dob TINYINT UNSIGNED NOT NULL,
detalji TEXT,
spol ENUM('M', 'Ž') NOT NULL,
boja varchar(20),
tezina FLOAT(2),
zdravstveni_problemi TEXT,
veterinarski_podaci TEXT,
posebne_potrebe TEXT,
socijalnost TEXT,
akitivnosti TEXT,

CONSTRAINT PRIMARY KEY (id),
CONSTRAINT FOREIGN KEY (pasmina_id) REFERENCES pasmina (id) ON DELETE SET
NULL
);

```

6.2.4. Tablica rezervacija_ljubimac

Tablica "rezervacija_ljubimac" uspostavlja odnos više-na-više između dvije relacije, rezervacija i ljubimac i time su međusobno povezane. Tablica sadržava sljedeće attribute: id, rezervacija_id i ljubimac_id.

Atribut „id“ je tipa INT sa svojstvom AUTO_INCREMENT, što omogućuje automatsko povećanje njegove vrijednosti sa svakom novom vezom između rezervacije i kućnog ljubimca. Ovaj atribut definiran je kao PRIMARY KEY pomoću CONSTRAINT PRIMARY KEY svojstva čime osigurava svoju jedinstvenost.

Atribut „rezervacija_id“ pohranjuje id rezervacije povezane s određenim kućnim ljubimcem. Služi kao strani ključ koji referencira na „id“ stupac u tablici rezervacija. Ograničenje stranog ključa osigurava da vrijednosti u ovom stupcu odgovaraju važećim ID-ovima u rezervacijskoj tablici. Ograničenje ON DELETE SET NULL podrazumijeva da ako se redak u tablici rezervacija izbriše, vrijednost u ovom stupcu bit će postavljena na NULL..

Atribut ljubimac_id pohranjuje id kućnog ljubimca povezanog s određenom rezervacijom. Služi kao strani ključ koji referencira na „id“ stupac u tablici ljubimac osiguravajući da svaki unos u ovom stupcu mora biti važeći ID iz tablice ljubimac. Ograničenje ON DELETE SET NULL osigurava da ako se red u tablici ljubimac izbriše, vrijednost u ovom stupcu bude postavljena na NULL u svim povezanim redovima u tablici rezervacija_ljubimac.

```

CREATE TABLE rezervacija_ljubimac (
    id INT AUTO_INCREMENT,
    rezervacija_id INT,
    ljubimac_id INT,

    CONSTRAINT PRIMARY KEY (id),
    CONSTRAINT FOREIGN KEY (rezervacija_id) REFERENCES rezervacija (id) ON DELETE
SET NULL,
    CONSTRAINT FOREIGN KEY (ljubimac_id) REFERENCES ljubimac (id) ON DELETE SET
NULL
);

```

6.3. Luka Vilagoš

6.3.1. Tablica vlasnik

Tablica vlasnik je dizajnirana za pohranu podataka o vlasnicima ljubimaca. Sadržava attribute: id, ime, prezime, broj_telefona, email, datum_rođenja, adresa, grad, država, spol i potvrdu.

Atribut "id" postavljen je kao PRIMARY KEY tipa INT i automatski se povećava za svaki novi red umetnut u tablicu pomoću AUTO INCREMENT svojstva.

Atribut „ime“ koristi se za pohranu imena čuvara. Atribut je tipa VARCHAR te mu je maksimalna duljina unosa 30 znakova, što je dovoljno za unos imena. Sadržava NOT NULL ograničenje koje osigurava obavezan unos ovog podatka unutar baze podataka.

Atribut „prezime“ sadržava podatke o prezimenu čuvara. Sadržava VARCHAR tip podatka kojemu maksimalna duljina unosa iznosi 30 znakova. Sadržava NOT NULL ograničenje kojemu osigurava obavezan unos prezimena.

Atribut „broj_telefona“ je tipa VARCHAR te sadržava osobni broj mobitela ili telefona čuvara. Maksimalna duljina mu iznosi 15 znakova što je dovoljno za unos broja znamenki mobilnog uređaja. Sadržava NOT NULL ograničenje koje osigurava obavezan unos podatka.

Atribut „email“ sadržava email čuvara. Tipa je VARCHAR te mu je maksimalna duljina do 30 znakova. Sadržava NOT NULL ograničenje koje osigurava da je podatak obavezan unesen.

Atribut „datum_rođenja“ sadržava podatke o rođenju čuvara. Koristi tip podatka DATE koji je specifičan za pohranjivanje datumskih vrijednosti. Datumi su najčešće pohranjeni u formatu YYYY-MM-DD (godina, mjesec, dan). Koristan je za pohranu i upravljanje informacijama o datumu unutar baze podataka čime osigurava integritet podataka te lakše operacije povezane s datumima. Sadržava NOT NULL ograničenje koje osigurava obavezan unos datuma rođenja.

Atribut „adresa“ je tipa VARCHAR te sadržava podatke o adresi vlasnika. Maksimalna duljina unosa mu je 256 znakova te sadržava NOT NULL ograničenje.

Atribut „grad“ sadržava podatak o gradu u kojem vlasnik trenutno živi. Tipa je VARCHAR te mu maksimalna duljina iznosi 20 znakova. Sadržava NOT NULL ograničenje koje osigurava da je grad obavezno napisan.

Atribut „država“ je tipa VARCHAR te sadržava podatke o državi iz koje vlasnik dolazi. Maksimalna duljina je do 56 znakova te sadržava NOT NULL ograničenje koje onemogućuje prisutnost NULL vrijednosti.

Atribut „spol“ sadržava ENUM tip podatka koji se koristi za definiranje stupca koji može sadržavati samo ograničen skup unaprijed definiranih vrijednosti. ENUM jasno pokazuje koje su vrijednosti dopuštene, poboljšavajući jasnoću koda i lakoću održavanja. Svaka vrijednost predstavlja jednu od dozvoljenih opcija što su u ovom slučaju 'M' i 'Ž' i bilo što osim toga će izbaciti error. Sadržava NOT NULL ograničenje čime osigurava da je spol definiran.

Atribut „potvrđeno“ predstavlja je li je određeni aspekt vezan uz vlasnika ljubimaca potvrđen to jest legitiman. Sadržava BOOL tip podatka čime omogućuje samo dvije vrijednosti: True i False. Ne sadržava nikakvo ograničenje.

```
CREATE TABLE vlasnik (  
  id INT AUTO_INCREMENT,  
  ime varchar(30) NOT NULL,  
  prezime varchar(30) NOT NULL,
```

```

    broj_telefona VARCHAR(15),
    email VARCHAR(30) NOT NULL,
    datum_rodenja DATE NOT NULL,
    adresa VARCHAR(256),
    grad VARCHAR(20),
    drzava VARCHAR(56),
    spol ENUM('M', 'Ž'),
    potvrđeno BOOL,

    CONSTRAINT PRIMARY KEY (id)
);

```

6.3.2. Tablica tip

Tablica tip sadržava tipove ustanove i njegove karakteristike. Uključuje attribute: id, naziv i opis.

Atribut „id“ je tipa INT i sadržava AUTO_INCREMENT svojstvo čime se vrijednost povećava prilikom svakog novog upisa. Ograničenje CONSTRAINT PRIMARY KEY (id) definira id stupac kao primarni ključ relacije osiguravajući njegovu jedinstvenost.

Atribut „naziv“ je tipa VARCHAR (256) što omogućuje dinamičku alokaciju memorije, umjesto da se rezervira fiksna količina memorije za svaki novi zapis. Sadržava podatke o nazivu ustanove koje mogu varirati ovisno o njihovoj usluzi. Maksimalna duljina mu iznosi 256 znakova u suprotnome će izbaciti error. Sadržava NOT NULL ograničenje čime sprječava unos NULL vrijednosti to jest da polje naziv ostane prazno.

Atribut „opis“ je tipa TEXT i time se omogućuje pohrana velikih količina tekstualnih podataka. Namijenjen je za pohranu detaljnog opisa ili dodatnih informacija o retku. Ne sadrži nikakva ograničenja i time naznačuje da vrijednost može biti NULL to jest da opis tipa može ostati prazan.

```

CREATE TABLE tip (
    id INT AUTO_INCREMENT,
    naziv varchar(256) NOT NULL,
    opis TEXT,

    CONSTRAINT PRIMARY KEY (id)
);

```

6.3.3. Tablica značajka

Tablica značajka se odnosi na posebne značajke ustanove. Sadržava attribute: id, naziv, opis, dostupnost, cijena_dodatka, vrijeme_otvaranja i vrijeme_zatvaranja.

Atribut „id“ je tipa INT i definiran je kao primarni ključ pomoću ograničenja CONSTRAINT PRIMARY KEY. AUTO INCREMENT svojstvo omogućuje automatsku generaciju vrijednosti za svaki novi red i time osigurava jedinstvenost.

Atribut „naziv“ je tipa VARCHAR i u njemu se pohranjuju podaci vezani za naziv raznih značajki kao što su igrališta ili pet spa. Maksimalna duljina mu iznosi do 256 znakova te sadržava NOT NULL ograničenje što naznačuje da je obavezno upisati naziv posebnih značajki.

Atribut „opis“ je tipa TEXT i omogućuje pohranu velike količine tekstualnih podataka. Atribut služi za opis svake značajke i njihove karakteristike. Ne sadržava nikakva ograničenja što omogućuje prisutnost NULL vrijednosti.

Atribut „dostupnost“ definiran je tipom TEXT i služi za pohranu dostupnosti pojedine značajke. Ne sadržava NOT NULL ograničenje što omogućuje da polje ostane prazno.

Atribut „cijena_dodatka“ specificira cijenu dodatnih značajki po dogovoru. Ovaj atribut koristi vrstu podataka FLOAT, omogućujući pohranjivanje brojeva s pomičnim zarezom s preciznošću prikaza od 2 decimalna mjesta. Atribut ne sadrži nikakva ograničenja i time je dozvoljeno ostaviti polje praznim.

Atribut „vrijeme_otvaranja“ pohranjuje vrijeme otvaranja značajke ili usluge. Koristi podatkovni tip TIME koji je značajan za pohranjivanje vremenskih vrijednosti u formatu HH:MM:SS. TIME omogućuje da se točno odredi vrijeme kada su značajke dostupne i vraća 00:00:00 ako izraz nije datum/vrijeme i ako je izraz NULL. Ne sadržava nikakva ograničenja čime omogućuje prisutnost NULL vrijednosti.

Atribut „vrijeme_zatvaranja“ pohranjuje vrijeme zatvaranja značajke ili usluge. Definiran je tipom podatka TIME koji omogućuje pohranu vremenskih vrijednosti u formatu HH:MM:SS. Ne sadržava nikakva ograničenja što omogućuje prisutnost NULL vrijednosti.

```
CREATE TABLE znacajka (  
  id INT AUTO_INCREMENT,  
  naziv VARCHAR(256) NOT NULL,  
  opis TEXT,  
  dostupnost TEXT,  
  cijena_dodatka FLOAT(2),  
  vrijeme_otvaranja TIME,  
  vrijeme_zatvaranja TIME,  
  
  CONSTRAINT PRIMARY KEY (id)  
);
```

6.3.4. Tablica ustanova_znacajka

Tablica „ustanova_znacajka“ definirana je odnosom više na više između 2 entiteta, ustanova i značajka i time omogućuje njihovu povezanost. Svaki zapis u ovoj tablici povezuje jednu ustanovu s jednom značajkom. Sadržava atribute: id, ustanova_id i znacajka_id.

Atribut „id“ je tipa INT s AUTO INCREMENT svojstvom što omogućava da se vrijednost atributa automatski poveća za svaku novu vezu između ustanove i značajke. Atribut je definiran kao PRIMARY KEY jer označava specifičnu numeričku vrijednost.

Atribut „ustanova_id“ je strani ključ uspostavljen ograničenjem FOREIGN KEY, koji upućuje na stupac „id“ u tablici „ustanova“. To znači da svaki unos u ovom stupcu mora biti važeći „id“ iz tablice „ustanova“. Ograničenje ON DELETE SET NULL podrazumijeva da ako se redak u tablici „ustanova“ izbriše, vrijednost u ovom stupcu će biti postavljena na NULL u svim pridruženim redcima u tablici „ustanova_znacajka“.

Atribut „znacajka_id“ je strani ključ koji referencira „id“ iz tablice „znacajka“ što osigurava da svaki novi unos u ovom stupcu mora biti važeći „id“ iz tablice „znacajka“. Sadržava ograničenje ON DELETE SET NULL koje osigurava da ako se redak u tablici „znacajka“ izbriše, vrijednost u ovom stupcu će biti postavljena na NULL.

```
CREATE TABLE ustanova_znacajka (  
    id INT AUTO_INCREMENT,  
    ustanova_id INT,  
    znacajka_id INT,  
  
    CONSTRAINT PRIMARY KEY (id),  
    CONSTRAINT FOREIGN KEY (ustanova_id) REFERENCES ustanova (id) ON DELETE SET  
NULL,  
    CONSTRAINT FOREIGN KEY (znacajka_id) REFERENCES znacajka (id) ON DELETE SET  
NULL  
);
```

6.4. Paula Vorih

6.4.1. Tablica ustanova

Tablica “ustanova” sadržava podatke o ustanovama gdje klijenti mogu ostaviti svoje ljubimce na čuvanju. Ona sadrži attribute: id, naziv, adresa, broj_telefona, email, radno_vrijeme, opis, web_stranica, kapacitet, ocjena, regija_id, tip_id i vlasnik_id.

Atribut “id” je tipa INT koji se svojstvom AUTO_INCREMENT povećava prilikom svakog novog umetanja id vrijednosti. “id” označava svaki jedinstveni skup vrijednosti koji se nalazi u ovoj tablici. Ovaj atribut je također primarni ključ ove tablice.

Atribut “naziv” tipa VARCHAR(256) označava naziv ustanove u kojima borave ljubimci. NOT NULL ograničenje sprečava unos NULL vrijednosti, odnosno sprečava da ime ustanove bude prazno polje.

Atribut “adresa” tipa VARCHAR(256) označava adresu ustanove u obliku broja i naziva ulice. NOT NULL ograničenje sprečava unos NULL vrijednosti, odnosno sprečava da adresa ustanove bude prazno polje.

Atribut “broj_telefona” je tipa VARCHAR(15) kako bi se mogao pohraniti broj telefona ustanove čija prva znamenka može biti 0 i koji između troznamenkastih vrijednosti ima simbol “-”.

Atribut “email” je tipa VARCHAR(30) gdje se pohranjuje adresa e-pošte svake ustanove.

Atribut “radno_vrijeme” je tipa TEXT gdje se pohranjuje radno vrijeme svake ustanove. Vrijednosti su navedene u 24-satnom formatu oblika HH:MM-HH:MM.

Atribut “opis” je tipa TEXT. Sadrži kratak opis pojedine ustanove.

Atribut “web_stranica” je tipa VARCHAR(256). U njemu se nalaze web stranice od pojedine ustanove.

Atribut “kapacitet” je tipa TEXT. U njemu se nalaze kapaciteti pojedine ustanove u obliku “[broj] kućnih ljubimaca”.

Atribut “ocjena” je tipa FLOAT(1) koji sadrži prosječnu ocjenu svake ustanove zaokruženu na jednu decimalu.

Atribut “regija_id” je tipa INT i on sadrži id-eve regija u kojoj se nalaze pojedine ustanove. Ovaj atribut je definiran kao strani ključ koji je vezan sa id-em, to jest, primarnim ključem tablice “regija”. Za ovaj atribut je također definirano svojstvo ON DELETE SET NULL koji će osigurati da se u slučaju brisanja ove vrijednosti iz roditeljske tablice ona postavi na NULL u tablici dijete.

Atribut “tip_id” je tipa INT i on sadrži id-eve svih tipova aktivnosti kojima je pojedina ustanova namijenjena. Ovaj atribut je definiran kao strani ključ koji je vezan sa id-em, to jest, primarnim ključem tablice “tip”. Ima definirano svojstvo ON DELETE SET NULL koji će osigurati da se u slučaju brisanja ove vrijednosti iz roditeljske tablice ona postavi na NULL u tablici dijete.

Atribut “vlasnik_id” je tipa INT i on sadrži sve id-eve vlasnika pojedine ustanove. Postoje vlasnici koji su vlasnici više od jedne ustanove. Ovaj atribut je definiran kao strani ključ koji je vezan sa id-em, to jest, primarnim ključem tablice “vlasnik”. Ima definirano svojstvo ON DELETE SET NULL koji će osigurati da se u slučaju brisanja ove vrijednosti iz roditeljske tablice ona postavi na NULL u tablici dijete.

```
CREATE TABLE ustanova (  
    id INT AUTO_INCREMENT,  
    naziv VARCHAR(256) NOT NULL,  
    adresa VARCHAR(256) NOT NULL,  
    broj_telefona VARCHAR(15),  
    email VARCHAR(30),  
    radno_vrijeme TEXT,  
    opis TEXT,  
    web_stranica VARCHAR(256),  
    kapacitet TEXT,  
    ocjena FLOAT(1),  
    regija_id INT,  
    tip_id INT,  
    vlasnik_id INT,  
  
    CONSTRAINT PRIMARY KEY (id),  
    CONSTRAINT FOREIGN KEY (regija_id) REFERENCES regija (id) ON DELETE SET NULL,  
    CONSTRAINT FOREIGN KEY (tip_id) REFERENCES tip (id) ON DELETE SET NULL,  
    CONSTRAINT FOREIGN KEY (vlasnik_id) REFERENCES vlasnik (id) ON DELETE SET  
NULL  
);
```

6.4.2. Tablica cuvar_ustanova

Tablica “cuvar_ustanova” prikazuje u kojoj ustanovi će vrijeme provoditi pojedini čuvar. Ona sadrži attribute: id, cuvar_id i ustanova_id.

Atribut “id” je tipa INT koji se svojstvom AUTO_INCREMENT povećava prilikom svakog novog umetanja id vrijednosti. “id” označava svaki jedinstveni skup vrijednosti koji se nalazi u ovoj tablici. Ovaj atribut je također primarni ključ ove tablice.

Atribut “cuvar_id” je tipa INT i on sadrži sve id-eve čuvara koji čuvaju u pojedinoj ustanovi. Ovaj atribut je definiran kao strani ključ koji je vezan sa id-em, to jest, primarnim ključem tablice “cuvar”. Ima definirano svojstvo ON DELETE SET NULL koji će osigurati da se u slučaju brisanja ove vrijednosti iz roditeljske tablice ona postavi na NULL u tablici dijete.

Atribut “ustanova_id” je tipa INT i on sadrži id-eve svih ustanova gdje se nalazi pojedini čuvar. Ovaj atribut je definiran kao strani ključ koji je vezan sa id-em, to jest, primarnim ključem tablice “ustanova”. Ima definirano svojstvo ON DELETE SET NULL koji će osigurati da se u slučaju brisanja ove vrijednosti iz roditeljske tablice ona postavi na NULL u tablici dijete.

```
CREATE TABLE cuvar_ustanova (
    id INT AUTO_INCREMENT,
    cuvar_id INT,
    ustanova_id INT,

    CONSTRAINT PRIMARY KEY (id),
    CONSTRAINT FOREIGN KEY (cuvar_id) REFERENCES cuvar (id) ON DELETE SET NULL,
    CONSTRAINT FOREIGN KEY (ustanova_id) REFERENCES ustanova (id) ON DELETE SET NULL
);
```

6.4.3. Tablica mjesto_interesa

Tablica "mjesto_interesa" označava različite atrakcije na temu kućnih ljubimaca i njihovih vlasnika za svaku regiju. Te atrakcije mogu biti unutarnje, poput dućana, ili vanjske, poput memorijalnih parkova. Sadrži atribute: id, naziv, adresa, broj_telefona, email, radno_vrijeme, opis, web_stranica i regija_id.

Atribut "id" je tipa INT koji se svojstvom AUTO_INCREMENT povećava prilikom svakog novog umetanja id vrijednosti. "id" označava svaki jedinstveni skup vrijednosti koji se nalazi u ovoj tablici. Ovaj atribut je također primarni ključ ove tablice.

Atribut "naziv" tipa VARCHAR(256) označava naziv pojedine atrakcije. NOT NULL ograničenje sprečava unos NULL vrijednosti, odnosno sprečava da ime atrakcije bude prazno polje.

Atribut "adresa" tipa VARCHAR(256) označava adresu atrakcije u obliku broja i naziva ulice. NOT NULL ograničenje sprečava unos NULL vrijednosti, odnosno sprečava da adresa atrakcije bude prazno polje.

Atribut "broj_telefona" je tipa VARCHAR(15) kako bi se mogao pohraniti broj telefona atrakcije čija prva znamenka može biti 0 i koji između troznamenkastih vrijednosti ima simbol "-". Neke atrakcije nemaju broj telefona.

Atribut "email" je tipa VARCHAR(30) gdje se pohranjuje adresa e-pošte za pojedinu atrakciju. Neke atrakcije nemaju adresu e-pošte.

Atribut "radno_vrijeme" je tipa TEXT gdje se pohranjuje radno vrijeme svake atrakcije. Vrijednosti su navedene u 24-satnom formatu oblika HH:MM-HH:MM. Neke atrakcije su uvijek otvorene pa njihovo radno vrijeme je opisano kao "Always Open" ili "Uvijek otvoreno".

Atribut "opis" je tipa TEXT. Sadrži kratak opis pojedine atrakcije.

Atribut "web_stranica" je tipa VARCHAR(256). U njemu se nalaze web stranice od pojedine atrakcije.

Atribut "regija_id" je tipa INT i on sadrži id-eve regija u kojoj se nalaze pojedine atrakcije. Ovaj atribut je definiran kao strani ključ koji je vezan sa id-em, to jest, primarnim ključem tablice "regija". Za ovaj atribut je također definirano svojstvo ON DELETE SET NULL koji će osigurati da se u slučaju brisanja ove vrijednosti iz roditeljske tablice ona postavi na NULL u tablici dijete.

```
CREATE TABLE mjesto_interesa (
    id INT AUTO_INCREMENT,
    naziv VARCHAR(256) NOT NULL,
    adresa VARCHAR(256) NOT NULL,
    broj_telefona VARCHAR(15),
```

```

email VARCHAR(30),
radno_vrijeme TEXT,
opis TEXT,
web_stranica VARCHAR(256),
regija_id INT,

CONSTRAINT PRIMARY KEY (id),
CONSTRAINT FOREIGN KEY (regija_id) REFERENCES regija (id) ON DELETE SET NULL
);

```

6.4.4. Tablica regija

Tablica "regija" navodi sve županije na području Republike Hrvatske uz kratak opis svake od njih. Sadrži attribute: id, naziv, opis i površina.

Atribut "id" je tipa INT koji se svojstvom AUTO_INCREMENT povećava prilikom svakog novog umetanja id vrijednosti. "id" označava svaki jedinstveni skup vrijednosti koji se nalazi u ovoj tablici. Ovaj atribut je također primarni ključ ove tablice.

Atribut "naziv" je tipa VARCHAR(64). U njemu se nalaze imena pojedinih županija u Republici Hrvatskoj, uključujući Grad Zagreb.

Atribut "opis" je tipa TEXT. Sadrži kratak opis pojedine županije.

Atribut "povrsina" je tipa INT i on navodi površinu pojedine županije izražene kao broj četvornih kilometara.

```

CREATE TABLE regija (
    id INT AUTO_INCREMENT,
    naziv varchar(64),
    opis TEXT,
    površina INT,

    CONSTRAINT PRIMARY KEY (id)
);

```

6.5. Alma Reka

6.5.1. Tablica korisnik

Tablica korisnik pohranjuje informacije o korisnicima sustava. Svaki korisnik predstavlja osobu koja koristi sustav za upravljanje rezervacijama i recenzijama.

Atributi talice korisnik su:

id (INT AUTO_INCREMENT) ovaj atribut je primarni ključ tablice. Jedinstveno indentificira svakog korisnika.

ime (VARCHAR(30)) ovaj atribut predstavlja ime korisnika.

prezime (VARCHAR(30)) ovaj atribut predstavlja prezime korisnika.

broj_telefona(VARCHAR(15)) ovaj atribut predstavlja broj telefona korisnika. Može biti NULL ili prazan ako korisnik nije dao broj telefona.

email (VARCHAR(30)) ovaj atribut predstavlja email adresu korisnika koja se koristi za komunikaciju i identifikaciju.

datum_rodenja (DATE) ovaj atribut predstavlja datum rođenja korisnika. Pomaže u određivanju starosti korisnika.

adresa (VARCHAR(256)) ovaj atribut predstavlja adresu korisnika. Može sadržavati naziv ulice, broj i druge detalje adrese.

grad (VARCHAR(20)) ovaj atribut predstavlja grad u kojem korisnik živi.

drzava (VARCHAR(56)) ovaj atribut predstavlja državu u kojoj korisnik prebiva.

spol (ENUM('M', 'Ž')) ovaj atribut predstavlja spol korisnika. M označava muško, a Ž žensko.

potvrđeno (BOOL) ovaj atribut predstavlja logičku vrijednost koja označava je li korisnički račun potvrđen ili verificiran.

```
CREATE TABLE korisnik (  
    id INT AUTO_INCREMENT,  
    ime varchar(30) NOT NULL,  
    prezime varchar(30) NOT NULL,  
    broj_telefona VARCHAR(15),  
    email VARCHAR(30) NOT NULL,  
    datum_rodenja DATE NOT NULL,  
    adresa VARCHAR(256),  
    grad VARCHAR(20),  
    drzava VARCHAR(56),  
    spol ENUM('M', 'Ž'),  
    potvrđeno BOOL,  
  
    CONSTRAINT PRIMARY KEY (id)  
);
```

6.5.2. Tablica korisnik_ljubimac

Tablica korisnik_ljubimac stvara odnos više-prema-više između tablica korisnik i ljubimac. Prati koji ljubimci pripadaju kojim korisnicima.

Atributi tablice korisnik_ljubimac su:

id (INT AUTO_INCREMENT) ovaj atribut predstavlja primarni ključ tablice. Jedinstveno identifikira svaki zapis.

korisnik_id (INT) ovaj atribut predstavlja strani ključ koji referencira id iz tablice korisnik. Označava korisnika koji posjeduje ljubimca.

ljubimac_id (INT) ovaj atribut predstavlja strani ključ koji referencira id iz tablice ljubimac. Označava ljubimca kojeg posjeduje korisnik.

```
CREATE TABLE korisnik_ljubimac (
    id INT AUTO_INCREMENT,
    korisnik_id INT,
    ljubimac_id INT,

    CONSTRAINT PRIMARY KEY (id),
    CONSTRAINT FOREIGN KEY (korisnik_id) REFERENCES korisnik (id) ON DELETE SET
NULL,
    CONSTRAINT FOREIGN KEY (ljubimac_id) REFERENCES ljubimac (id) ON DELETE SET
NULL
);
```

6.5.3. Tablica rezervacija

Tablica rezervacija prehranjuje informacije o rezervacijama koje korisnici obavljaju za usluge ili boravke u raznim ustanovama.

Atributi tablice rezervacija:

id (INT AUTO_INCREMENT) ovaj atribut predstavlja primarni ključ tablice. Jedinstveno identificira svaku rezervaciju.

korisnik_id (INT) ovaj atribut predstavlja strani ključ koji referencira id iz tablice korisnik. Označava korisnika koji je obavio rezervaciju.

ustanova_id (INT) ovaj atribut predstavlja strani ključ koji referencira id iz tablice ustanova. Označava ustanovu u kojoj je obavljena rezervacija.

usluga_id (INT) ovaj atribut predstavlja strani ključ koji referencira id iz tablice usluga. Označava uslugu koja je rezervirana.

datum (DATETIME) ovaj atribut predstavlja datum i vrijeme kada je rezervacija obavljena ili zakazana.

napomena (TEXT) ovaj atribut predstavlja dodatne napomene ili komentare o rezervaciji. Može uključivati posebne zahtjeve ili upute.

potvrđeno (BOOL) ovaj atribut predstavlja logičku vrijednost koja označava je li rezervacija potvrđena.

```
CREATE TABLE rezervacija (
    id INT AUTO_INCREMENT,
    korisnik_id INT,
    ustanova_id INT,
    usluga_id INT,
    datum DATETIME,
    napomena TEXT,
    potvrđeno BOOL,

    CONSTRAINT PRIMARY KEY (id),
    CONSTRAINT FOREIGN KEY (korisnik_id) REFERENCES korisnik (id) ON DELETE SET
NULL,
    CONSTRAINT FOREIGN KEY (ustanova_id) REFERENCES ustanova (id) ON DELETE SET
NULL,
    CONSTRAINT FOREIGN KEY (usluga_id) REFERENCES usluga (id) ON DELETE SET NULL
);
```

6.5.4. Tablica recenzija

Tablica recenzija pohranjuje recenzije koje korisnici pišu o ustanovama ili čuvarima.

Atributi tablice recenzija:

id(INT AUTO_INCREMENT) ovaj atribut predstavlja primarni ključ tablice. Jedinstveno identificira svaku recenziju.

korisnik_id(INT) ovaj atribut predstavlja strani ključ koji referencira id iz tablice korisnik. Označava korisnika koji je napisao recenziju.

ustanova_id(INT) ovaj atribut predstavlja strani ključ koji referencira id iz tablice ustanova. Označava ustanovu koja se recenzira.

cuvar_id(INT) ovaj atribut predstavlja strani ključ koji referencira id iz tablice cuvar. Označava čuvara koji se recenzira.

naslov(VARCHAR(256)) ovaj atribut predstavlja naslov recenzije. Pruža kratak sažetak sadržaja recenzije.

tekst(TEXT) ovaj atribut predstavlja detaljniji tekst recenzije. Uključuje korisnikove misli i mišljenja o ustanovi ili čuvaru.

ocjena(TINYINT UNSIGNED) ovaj atribut predstavlja ocjenu koju je dao korisnik, ocjena koja je obično na skali od 1 do 5.

datum(DATETIME) ovaj atribut predstavlja datum i vrijeme kada je recenzija napisana ili predana.

```
CREATE TABLE recenzija (  
    id INT AUTO_INCREMENT,  
    korisnik_id INT,  
    ustanova_id INT,  
    cuvar_id INT,  
    naslov VARCHAR(256),  
    tekst TEXT,  
    ocjena TINYINT UNSIGNED,  
    datum DATETIME,  
  
    CONSTRAINT PRIMARY KEY (id),  
    CONSTRAINT FOREIGN KEY (korisnik_id) REFERENCES korisnik (id) ON DELETE SET  
NULL,  
    CONSTRAINT FOREIGN KEY (ustanova_id) REFERENCES ustanova (id) ON DELETE SET  
NULL,  
    CONSTRAINT FOREIGN KEY (cuvar_id) REFERENCES cuvar (id) ON DELETE SET NULL  
);
```


7. Upiti

7.1. Dinko Nađ

7.1.1. Upit

Ovaj SQL upit izračunava koliko je tko usluga napravio od čuvara, od svakog čuvara izračunava ukupno zaradu te koliko prosječno uzima po usluzi.

```
SELECT cuvar.id , cuvar.ime, cuvar.prezime,
       COUNT(usluga.id) AS broj_usluga,
       SUM(usluga.cijena) AS ukupna_zarada,
       AVG(usluga.cijena) AS prosjecna_cijena
FROM cuvar
INNER JOIN usluga ON cuvar.id = usluga.cuvar_id
GROUP BY cuvar.grad, cuvar.id, cuvar.ime, cuvar.prezime
HAVING COUNT(usluga.id) > 0
ORDER BY prosjecna_cijena DESC;
```

U FROM dijelu odabire se dvije tablice koje su se trebale spojiti. U ovo slučaju, to su tablice "cuvar" i "usluga". Zatim se koristi INNER JOIN kako bi se spojili podatci iz tablica na temelju identifikatora čuvara ("cuvar.id") i identifikatora čuvara u uslugama ("usluga.cuvar_id").

Nakon što imamo spojene podatke, koristimo GROUP BY kako bismo grupirali te retke prema određenim stupcima. U ovom slučaju, grupiramo po identifikatoru čuvara, imenu, i prezimenu čuvara.

Nakon grupiranja, koristimo HAVING da bi primijenili filter na grupirane podatke. U ovom slučaju, koristimo "HAVING COUNT(usluga.id)>0" kako bi osigurao da se prikazuju samo čuvari koji imaju barem jednu uslugu.

Kada su podatci filtrirani i grupirani, koristim naredbu SELECT kako bi odabrali konkretne stupce koje želimo prikazati u rezultatima. U ovom dijelu koristimo 3 funkcije agregacije. Prva od njih je ("COUNT(usluga.id) AS broj_usluga") kako bi prebrojao koliko svaki čuvar ima usluga i spremio kao ("broj_usluga"). Pomoću funkcije ("SUM(usluga.cijena) AS ukupna_zarada") dobijemo koliko je ukupno svaki čuvar zaradio. I na kraju sa funkcijom ("AVG(usluga.cijena) AS prosjecna_cijena") dobijemo prosječnu cijenu usluga za svakog čuvara. U SELECT naredbi uz već navedene 3 funkcije agregacije ostavljeno je i identifikacija, ime i prezime čuvara.

I na samom kraju, izvodi se ORDER BY kako bi se sortirali svi rezultati prema ("prosjecna_cijena") u silaznom redoslijedu, od najveće do najmanje.

7.1.2. Upit

Ovaj SQL upit izračunava koliko je svaki čuvar mora raditi(u minutama).

```
SELECT cuvar.id, cuvar.ime, cuvar.prezime,
       ROUND(SUM(TIME_TO_SEC(aktivnost.trajanje) / 60), 2) AS ukupno_trajanje_min
FROM cuvar
INNER JOIN cuvar_aktivnost ON cuvar.id = cuvar_aktivnost.cuvar_id
INNER JOIN aktivnost ON cuvar_aktivnost.aktivnost_id = aktivnost.id
GROUP BY cuvar.id, cuvar.ime, cuvar.prezime
ORDER BY ukupno_trajanje_min DESC;
```

U FROM dijelu navode se tablice "cuvar", "cuvar_aktivnost", "aktivnost" iz kojih ćemo dohvatiti podatke. Nakon dohvaćanja podataka izvršava se INNER JOIN kako bi se spojili podaci iz tablica "cuvar" i "cuvar_aktivnost" na temelju identifikatora čuvara ("cuvar.id") i identifikatora čuvara u aktivnostima ("cuvar_aktivnost.cuvar_id"). Ova operacija osigura da dobijemo samo one retke iz tablica koje se podudaraju prema identifikatoru čuvara.

Nakon toga, ponovno se izvršava INNER JOIN kako bismo spojili podatke iz prethodno spojenih tablica s tablicom "aktivnost" na temelju identifikatora aktivnosti ("cuvar_aktivnost.aktivnost_id") i identifikatora aktivnosti ("aktivnost.id"). Ova operacija osigurava da dobijemo samo one retke koji se podudaraju prema identifikatoru aktivnosti.

Nakon što su podaci spojeni, koristimo GROUP BY kako bismo grupirali rezultate prema identifikatoru čuvara, imenu i prezimenu čuvara. Ova operacija omogućava da se podaci agregiraju po svakom čuvaru.

Zatim se koristi funkcija SUM zajedno s funkcijom ROUND kako bi se izračunalo ukupno trajanje svih aktivnosti za svakog čuvara u minutama s dvije decimale. Trajanje aktivnosti se prvo pretvara u sekunde, dijeli se s 60 da bi se dobilo u minutama, zatim se izračunava ukupna vrijednost, a na kraju se rezultat zaokružuje na dvije decimale.

Na kraju, rezultati se sortiraju prema ukupnom trajanju aktivnosti u silaznom redoslijedu, od najduže do najkraće.

7.1.3. Upit

Ovaj SQL upit izračunava prosjek cijena svih usluga i vraća usluge koje su iznad tog prosjeka u gradu Zagrebu.

```
SELECT DISTINCT usluga.naziv, usluga.opis
FROM usluga
INNER JOIN cuvar ON usluga.cuvar_id = cuvar.id
INNER JOIN cuvar_aktivnost ON cuvar.id = cuvar_aktivnost.cuvar_id
INNER JOIN aktivnost ON cuvar_aktivnost.aktivnost_id = aktivnost.id
WHERE cuvar.grad = 'Zagreb'
AND usluga.cijena > (
    SELECT AVG(usluga.cijena)
    FROM usluga
    INNER JOIN cuvar ON usluga.cuvar_id = cuvar.id
    INNER JOIN cuvar_aktivnost ON cuvar.id = cuvar_aktivnost.cuvar_id
    INNER JOIN aktivnost ON cuvar_aktivnost.aktivnost_id = aktivnost.id
    WHERE cuvar.grad = 'Zagreb'
);
```

U FROM dijelu odabiremo tablice "usluga", "cuvar", "cuvar_aktivnosti" i "aktivnost". Zatim koristimo INNER JOIN kako bismo spojili podatke iz tih tablica na temelju njihovih međusobnih veza. Spajali smo ih prema identifikatoru čuvara "usluga.cuvar_id" i "cuvar.id", druga veza je između "cuvar.id" i "cuvar_aktivnost.cuvar_id" te u trećem "cuvar_aktivnosti.aktivnosti_id" i "aktivnosti_id".

U WHERE uvjetu navodimo da želimo podatke samo za čuvare koji su iz Zagreba. Ovo osigurava da se u rezultatima prikažu samo usluge čuvara iz Zagreba.

Koristimo podupit unutar WHERE uvjeta kako bismo izračunali prosječnu cijenu usluga. INNER JOIN-om smo istim principom kao i gore spojili sve tablice te izračunali prosječnu vrijednost usluga. U glavnom WHERE dijelu zatim smo usporedili sve vrijednosti i prosječnu vrijednost. Samo usluge čija je cijena veća od prosjeka bit će uključena u rezultate.

Na kraju svega, izvodi se SELECT DISTINCT kako bismo ostavili samo stupce "usluga.naziv" i "usluga.opis". Naredba DISTINCT nam omogućuje da se uklone duplikati iz rezultata.

7.1.4. Upit

Ovaj SQL upit izračunava najveće cijene usluga prema gradovima u kojim čuvari žive.

```
SELECT cuvar.grad, cuvar.ime, cuvar.prezime, usluga.naziv, usluga.opis,
usluga.cijena
FROM usluga
JOIN cuvar ON usluga.cuvar_id = cuvar.id
WHERE usluga.cijena = (
    SELECT MAX(usluga_u2.cijena)
    FROM usluga AS usluga_u2
    JOIN cuvar AS cuvar_c2 ON usluga_u2.cuvar_id = cuvar_c2.id
    WHERE cuvar_c2.grad = cuvar.grad
)
ORDER BY cuvar.grad;
```

U FROM dijelu odabiremo tablice "usluga" i "cuvar". Koristimo INNER JOIN kako bismo spojili podatke iz tih tablica na temelju njihovih međusobnih veza. Spajamo ih prema identifikatoru čuvara "usluga.cuvar_id" i "cuvar.id".

U WHERE dijelu navodimo da želimo podatke samo za čuvare koji su obavljali najskuplje usluge u svakom gradu.

Koristimo podupit unutar WHERE uvjeta kako bismo izračunali maksimalnu cijenu usluge za svaki grad. Podupit traži maksimalnu cijenu usluge u istom gradu kao što je grad trenutnog čuvara "cuvar.grad".

U podupitu se koristi MAX funkcija za pronalaženje maksimalne cijene usluge, a zatim se uspoređuje s cijenom usluge u vanjskom upitu.

Pomoću naredbe SELECT odabiremo stupce koje želimo prikazati u rezultatima: "cuvar.grad", "cuvar.ime", "cuvar.prezime", "usluga.naziv", "usluga.opis" te "usluga.cijena".

Rezultati se sortiraju sa naredbom ORDER BY abecednim redom.

7.2. Patricia Lazić

7.2.1. Upit

Prikaži najmlađeg ljubimca za svaku pasminu

```
SELECT
    p.naziv AS pasmina_naziv,
    ljubimac.ime AS ljubimac_ime,
    ljubimac.dob AS ljubimac_dob,
    ljubimac.boja AS ljubimac_boja,
    ljubimac.tezina AS ljubimac_tezina
FROM pasmina p
JOIN ljubimac ON p.id = ljubimac.pasmina_id
WHERE ljubimac.dob = (
    SELECT MIN(lj.dob)
    FROM ljubimac lj
    WHERE lj.pasmina_id = p.id
)
```

```
WHERE lj.pasmina_id = p.id
)
ORDER BY p.naziv;
```

U ovom upitu, odabiremo podatke iz tablica pasmina i ljubimac koristeći INNER JOIN kako bismo ih spojili na temelju njihovih međusobnih veza. U WHERE dijelu koristimo podupit kako bismo pronašli najmlađeg ljubimca za svaku pasminu. Podupit koristi MIN funkciju za pronalaženje minimalne dobi ljubimca za određenu pasminu. Odabrane stupce sortiramo abecednim redom prema nazivu pasmine.

7.2.2. Upit

Prikaži ljubimca sa više od jedne rezervacije

```
SELECT
  l.id AS ljubimac_id,
  l.ime AS ljubimac_ime,
  p.naziv AS pasmina_naziv,
  v.naziv AS vrsta_naziv,
  COUNT(r1.rezervacija_id) AS broj_rezervacija
FROM ljubimac l
JOIN pasmina p ON l.pasmina_id = p.id
JOIN vrsta v ON p.vrsta_id = v.id
JOIN rezervacija_ljubimac r1 ON l.id = r1.ljubimac_id
GROUP BY l.id, p.naziv, v.naziv
HAVING COUNT(r1.rezervacija_id) > 1
ORDER BY broj_rezervacija DESC, l.ime ASC;
```

Ovaj upit koristi INNER JOIN za spajanje tablica ljubimac, pasmina, vrsta i rezervacija_ljubimac kako bi dobio podatke o ljubimcima s više od jedne rezervacije. Koristimo GROUP BY za grupiranje ljubimaca prema njihovom ID-u, nazivu pasmine i vrsti, te HAVING za filtriranje ljubimaca s više od jedne rezervacije. Rezultati se sortiraju prema broju rezervacija i imenu ljubimca.

7.2.3. Upit

Prikaži najčešću boju ljubimca po vrsti

```
SELECT vrsta.naziv AS vrsta_naziv, ljubimac.boja, COUNT(*) AS broj_ljubimaca
FROM vrsta
JOIN pasmina ON vrsta.id = pasmina.vrsta_id
JOIN ljubimac ON pasmina.id = ljubimac.pasmina_id
GROUP BY vrsta.id, ljubimac.boja
HAVING COUNT(*) = (
  SELECT MAX(boja_count)
  FROM (
    SELECT COUNT(*) AS boja_count
    FROM pasmina AS p2
    JOIN ljubimac AS l2 ON p2.id = l2.pasmina_id
    WHERE p2.vrsta_id = vrsta.id
    GROUP BY l2.boja
  ) AS broj_boja
)
ORDER BY vrsta.naziv;
```

Ovaj upit koristi INNER JOIN za spajanje tablica vrsta, pasmina i ljubimac kako bi dobio podatke o najčešćoj boji ljubimca po vrsti. Koristimo GROUP BY za grupiranje po vrsti i boji, te HAVING za filtriranje boje s najvećim brojem ljubimaca. Podupit koristi MAX funkciju za pronalaženje najčešće boje ljubimca za određenu vrstu. Rezultati se sortiraju abecednim redom prema nazivu vrste.

7.2.4. Upit

Prikaži najčešću pasminu ljubimca

```
SELECT p.id AS id_pasmina, p.naziv AS pasmina_naziv, COUNT(*) AS broj_ljubimaca
FROM ljubimac l
JOIN pasmina p ON l.pasmina_id = p.id
GROUP BY p.id, p.naziv
ORDER BY broj_ljubimaca DESC;
```

Ovaj upit koristi INNER JOIN za spajanje tablica ljubimac i pasmina kako bi dobio podatke o najčešćoj pasmini ljubimca. Koristimo GROUP BY za grupiranje pasmina prema njihovom ID-u i nazivu, te ORDER BY za sortiranje rezultata prema broju ljubimaca u silaznom redoslijedu.

7.3. Luka Vilagoš

7.3.1. Upit

Prikaži sve značajke određene ustanove

```
SELECT z.naziv, z.opis
FROM ustanova_znacajka uz
JOIN znacajka z ON uz.znacajka_id = z.id
WHERE uz.ustanova_id = 1;
```

Ovaj upit odabire naziv i opis značajki određene ustanove (s ID-om 1) iz tablica ustanova_znacajka i znacajka koristeći INNER JOIN. Na temelju ustanova_id iz tablice ustanova_znacajka filtriramo značajke za specifičnu ustanovu.

7.3.2. Upit

Prikaži sve ustanove i njihovog vlasnika

```
SELECT u.naziv AS ustanova, v.ime AS vlasnik_ime, v.prezime AS vlasnik_prezime
FROM ustanova u
JOIN vlasnik v ON u.vlasnik_id = v.id;
```

Ovaj upit odabire naziv ustanova i ime te prezime vlasnika koristeći INNER JOIN na tablice ustanova i vlasnik na temelju vlasnik_id. Prikazuje sve ustanove zajedno s njihovim vlasnicima.

7.3.3. Upit

Pronađi tipove ustanova sa određenom značajkom

```
SELECT u.naziv AS ustanova, t.naziv AS tip
FROM ustanova u
JOIN ustanova_znacajka uz ON u.id = uz.ustanova_id
JOIN znacajka z ON uz.znacajka_id = z.id
JOIN tip t ON u.tip_id = t.id
WHERE z.naziv = 'veliko dvorište';
```

Ovaj upit odabire naziv ustanove i tip ustanove koristeći INNER JOIN na tablice ustanova, ustanova_znacajka, znacajka i tip kako bi pronašao ustanove koje imaju značajku 'Veliko dvorište'. Filtrira rezultate prema nazivu značajke.

7.3.4. Upit

Pronađi sve ustanove koje imaju značajku koja je dostupna između 12:00:00 i 19:00:00

```
SELECT u.naziv AS ustanova, z.naziv AS znacajka
FROM ustanova u
JOIN ustanova_znacajka uz ON u.id = uz.ustanova_id
JOIN znacajka z ON uz.znacajka_id = z.id
WHERE z.id IN (
    SELECT z1.id
    FROM znacajka z1
    WHERE z1.vrijeme_otvaranja <= '12:00:00' AND z1.vrijeme_zatvaranja >=
'19:00:00'
);
```

Ovaj upit odabire naziv ustanove i značajke za ustanove koje imaju značajku dostupnu između 12:00:00 i 19:00:00. Koristi podupit za pronalaženje ID-ova značajki koje su dostupne u tom vremenskom razdoblju.

7.4. Paula Vorih

7.4.1. Upit

Ispiši sve ustanove u karlovačkoj regiji uz detalje čuvara u tim ustanovama.

```
SELECT u.naziv AS naziv_ustanove, u.adresa AS adresa_ustanove, r.naziv AS
naziv_regije, c.ime AS ime_cuvara, c.prezime AS prezime_cuvara, c.email AS
email_cuvara
FROM ustanova u
JOIN cuvar_ustanova cu ON u.id = cu.ustanova_id
JOIN cuvar c ON cu.cuvar_id = c.id
JOIN regija r ON u.regija_id = r.id
WHERE r.naziv = 'karlovačka';
```

Sa "FROM ustanova u" odabiremo tablicu ustanova i dajemo joj alias u. Potom koristimo JOIN operaciju nad "cuvar_ustanova" sa aliasem "cu" sa ON uvjetom da pridružimo "ustanova.id" sa jednakom vrijednosti "cuvar_ustanova.ustanova_id" kako bi dobili odnose između ustanova i čuvara. Sljedeće JOIN pridruživanje je "cuvar" sa aliasem "c" sa ON uvjetom pridruživanja "cuvar_ustanova.cuvar_id" sa jednakom vrijednosti "cuvar.id" kako bi dobili podatke o pojedinim čuvarima. Zatim radimo JOIN nad tablicom regija s aliasom "r" gdje sa ON uvjetom pridružujemo "ustanova.regija" sa jednakom vrijednosti iz "regija.id". Pomoću "WHERE r.naziv = 'Karlovačka'"

dolazimo do željene regije za koju želimo vidjeti podatke, u ovom slučaju "Karlovačka". U SELECT djelu odabiremo stupce koje želimo da se prikažu, koji su u ovom slučaju "u.naziv" sa aliasem "naziv_ustanove", "u.adresa" sa aliasem "adresa_ustanove", "r.naziv" sa aliasem "naziv_regije", "c.ime" sa aliasem "ime_cuvara", "c.prezime" sa aliasem "prezime_cuvara" te "c.email" sa aliasem "email_cuvara".

7.4.2. Upit

Ispiši sve ustanove koje imaju veću ocjenu od prosječne ocjene unutar svojih regija.

```
SELECT u.naziv AS naziv_ustanove, u.adresa AS adresa_ustanove, u.ocjena AS
ocjena_ustanove, r.naziv AS naziv_regije, r.opis AS opis_regije
FROM ustanova u
JOIN regija r ON u.regija_id = r.id
WHERE u.ocjena > (
    SELECT AVG(u2.ocjena)
    FROM ustanova u2
    WHERE u2.regija_id = u.regija_id
);
```

Sa "FROM ustanova u" odabiremo tablicu "ustanova" i dajemo joj alias "u". Nakon toga radimo JOIN tablice "ustanova" sa tablicom "regija" s aliasom "r" gdje sa ON uvjetom pridružujemo "ustanova.regija_id" sa jednakom vrijednosti iz "regija.id". U WHERE izrazu radimo podupit gdje želimo da "ustanova.ocjena" bude veća nego prosječna ocjena ustanovi u pojedinoj regiji. U podupitu tablici "ustanova" dajemo alias "u2" kako bi ga ralikovali sa aliasom "u" iz glavnog upita. U podupitu koristimo "WHERE u2.regija_id = u.regija_id" kako bi osigurali da se koriste srednje vrijednosti istih regija koje smo dobili iz glavnog upita. U SELECT dijelu glavnog upita odabiremo stupce koje želimo da se prikažu, koji su u ovom slučaju "u.naziv" s aliasom "naziv_ustanove", "u.adresa" s aliasom "adresa_ustanove", "u.ocjena" s aliasom "ocjena_ustanove", "r.naziv" s aliasom "naziv_regije" te "r.opis" sa aliasom "opis_regije".

7.4.3. Upit

Ispiši sve regije u kojima je prosječna ocjena svih ustanova veća od 4.4. Ocjene moraju biti poredane od najveće prema najmanjoj.

```
SELECT r.naziv AS naziv_regije, AVG(u.ocjena) AS prosjecna_ocjena
FROM ustanova u
JOIN regija r ON u.regija_id = r.id
GROUP BY r.naziv
HAVING AVG(u.ocjena) > 4.4
ORDER BY prosjecna_ocjena DESC;
```

Sa "FROM ustanova u" odabiremo tablicu "ustanova" i dajemo joj alias "u". Zatim radimo JOIN tablice "ustanova" sa tablicom "regija" s aliasom "r" gdje sa ON uvjetom pridružujemo "ustanova.regija_id" sa jednakom vrijednosti iz "regija.id". Potom grupiramo po nazivu regije kako bismo dobili prosječnu vrijednost sa funkcijom "AVG(u.ocjena)". Sa HAVING uvjetom zadajemo da prosječna ocjena ustanove u nekoj regiji mora biti veća od 4.4. Koristimo "ORDER BY prosjecna_ocjena DESC" kako bismo dobili poredak prosječnih ocjena u silaznom redoslijedu. U SELECT dijelu odabiremo stupce "r.naziv" s aliasom "naziv_regije" i funkcijom agregacije "AVG(u.ocjena)" dobivamo prosječnu ocjenu ustanova i dajemo joj alias "prosjecna_ocjena".

7.4.4. Upit

Ispiši informacije o čuvarima i ustanovama u kojima borave u regijama koje počinju sa slovom P.

```
SELECT c.ime AS ime_cuvara, c.prezime AS prezime_cuvara, c.email AS email_cuvara,  
       u.naziv AS naziv_ustanove, u.ocjena AS ocjena_ustanove, r.naziv AS  
naziv_regije  
FROM cuvar c  
JOIN cuvar_ustanova cu ON c.id = cu.cuvar_id  
JOIN ustanova u ON cu.ustanova_id = u.id  
JOIN regija r ON u.regija_id = r.id  
WHERE r.naziv LIKE 'P%';
```

Sa "FROM cuvar c" odabiremo tablicu "cuvar" i dajemo joj alias "c". Zatim koristimo JOIN operaciju nad "cuvar_ustanova" sa aliasom "cu" sa ON uvjetom da pridružimo "cuvar.id" sa jednakom vrijednosti "cuvar_ustanova.cuvar_id" kako bi dobili odnose između čuvara ustanova i čuvara. To isto radimo za tablice "ustanova" i "regija" da ih spojimo sa prikladnim id vrijednostima. "WHERE r.naziv LIKE 'P%'" se koristi kako bi odabrali sve nazive čije ime počinje sa slovom P. Na kraju SELECT naredbom odabiremo stupce "c.ime" s aliasom "ime_cuvara", "c.prezime" s aliasom "prezime_cuvara", "c.email" s aliasom "email_cuvara", "u.naziv" s aliasom "naziv_ustanove", "u.ocjena" s aliasom "ocjena_ustanove" te "r.naziv" s aliasom "naziv_regije".

7.5. Alma Reka

7.5.1. Upit

Ovaj upit će pronaći sve korisnike koji su stariji od 18 godina.

```
SELECT *  
FROM korisnik  
WHERE DATEDIFF(CURDATE(), datum_rođenja) > (18 * 365);
```

Odabrat će sve stupce iz tablice 'korisnik' za korisnike koji su stariji od 18 godina. Funkcija 'DATEDIFF' izračunava razliku u danima između trenutnog datuma ('CURDATE()') i 'datum_rođenja' (datum rođenja) za svakog korisnika. Ako je razlika veća od 18 puta 365 dana (otprilike 18 godina), korisnik se smatra starijim od 18 godina i uključuje se u rezultat.

7.5.2. Upit

Ovaj upit će pronaći sve rezervacije koje su zakazane unutar sljedeća dva tjedna.

```
SELECT *  
FROM rezervacija  
WHERE datum BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 2 WEEK);
```

Odabrat će sve stupce iz tablice 'rezervacija' za rezervacije koje se događaju unutar sljedeća dva tjedna. Stupac 'datum' (datum rezervacije) mora biti između trenutnog datuma ('CURDATE()') i datuma dva tjedna od trenutnog datuma ('DATE_ADD(CURDATE() INTERVAL 2 WEEK)'). Operator 'BETWEEN' se koristi za filtriranje rezultata u ovom rasponu datuma.

7.5.3. Upit

Ovaj upit će dohvatiti sve recenzije s ocjenom 3 ili niže

```
SELECT *  
FROM recenzija  
WHERE ocjena <= 3;
```

Odabrat će sve stupce iz tablice 'recenzija' gdje je 'ocjena' manja ili jednaka 3. Ovo učinkovito dohvaća sve recenzije koje imaju nisku ocjenu, što se može koristiti za identificiranje problema ili područja za poboljšanje.

7.5.4. Upit

Računa prosječnu ocjenu recenzija

```
SELECT SUM(ocjena)/count(ocjena) AS prosjecna_ocjena  
FROM recenzija;
```

Izračunat će prosječnu ocjenu svih recenzija u tablici 'recenzija'. Zbraja sve ocjene ('SUM(ocjena)') i dijeli taj zbroj s ukupnim brojem ocjena ('count(ocjena)'). Rezultirajuća prosječna ocjena naziva se 'prosjecna_ocjena'. Ovaj upit omogućuje analizu ukupnog zadovoljstva korisnika ili kvalitete proizvoda ili usluge.

8. Pogledi

8.1. Dinko Nađ

8.1.1. Pogled

Koliko rezervacija ima svaka aktivnost.

```
CREATE VIEW broj_rezervacija AS  
SELECT aktivnost.naziv AS vrsta_aktivnosti, COUNT(*) AS broj_rezervacija  
FROM cuvar  
JOIN cuvar_aktivnost ON cuvar.id = cuvar_aktivnost.cuvar_id  
JOIN aktivnost ON cuvar_aktivnost.aktivnost_id = aktivnost.id  
GROUP BY aktivnost.naziv  
ORDER BY broj_rezervacija DESC;
```

Kreira se pogled sa nazivom „broj_rezervacija“ koji sadrži informacije o broju rezervacije za svaku vrstu aktivnosti.

U FROM dijelu odabiremo tablice „cuvar“, „aktivnost“ i „cuvar_aktivnost“.

Koristimo JOIN ključne riječi kako bismo spojili podatke iz različitih tablica. Prvo, spajamo tablice „cuvar“ i „cuvar_aktivnosti“ prema identifikatoru čuvara. Zatim spajamo tablicu „cuvar_aktivnost“ s tablicom „aktivnost“ prema identifikatoru aktivnosti. Ovo omogućuje da se podatci povežu na temelju veza između čuvara i njihovih aktivnosti.

U GROUP BY dijelu grupiramo rezultate prema nazivu aktivnosti. Ovo osigurava da će rezultati agregirati po svakoj jedinstvenoj vrsti aktivnosti.

U SELECT dijelu odabiremo stupce koje želimo prikazati u rezultatima. U ovom slučaju, odabiremo naziv aktivnosti koji smo preimenovali u „vrsta_aktivnosti“ i koristimo funkciju COUNT(*) kako bismo izbrojali koliko puta se svaka aktivnost pojavljuje.

Naredbom ORDER BY sortiramo rezultate prema broju rezervacija po aktivnosti silazno. To znači da će aktivnosti s najvećim rezervacijama biti prikazane prve u rezultatima.

Nakon što je pogled kreiran, testiramo ga izvršavanjem upita „SELECT * FROM broj_rezervacija“ što prikazuje sve podatke koji su pohranjeni u tom pogledu.

8.1.2. Pogled

Koliko ima muških i koliko ima ženskih čuvara

```
CREATE VIEW spolovi AS
SELECT
    cuvar.spol,
    COUNT(DISTINCT cuvar.id) AS broj_cuvara,
    ROUND(AVG(DATEDIFF(CURRENT_DATE, cuvar.datum_rodenja) / 365), 1) AS
    prosjecna_dob_godina,
    SUM(usluga.cijena) AS suma_cijena
FROM cuvar
JOIN usluga ON cuvar.id = usluga.cuvar_id
GROUP BY cuvar.spol;
```

Ovaj kod kreira pogled nazvan „spolovi“ koji zadrži informacije o čuvarima grupiranim po spolu.

U FROM dijelu navodimo koje dvije tablice, u ovom slučaju to su „cuvar“ i „usluga“. Zatim spajamo podatke iz obje tablice koristeći JOIN naredbu. To znači da će svaki red iz tablice „cuvar“ pronaći odgovarajuće retke u tablici „usluga“ prema uvjetu spajanje (cuvar.id = usluga.cuvar_id).

Nakon što su podatci spojeni, pomoću naredbe GROUP BY grupiramo rezultate prema spolu čuvara, kako je definirano u (GROUP BY cuvar.spol). Program će ih grupirati tako da se svi muški i ženski stajati zasebno.

Zatim se izvodi SELECT naredba u kojoj odabiremo koje podatke želimo prikazati. U ovom slučaju, to uključuje spol čuvara, broj jedinstvenih čuvara, prosječnu dob čuvara i ukupnu sumu cijena usluga.

Funkcija COUNT broji jedinstvene identifikatore čuvara u svakoj grupi prema spolu, što omogućava da se odredi ukupan broj različitih čuvara unutar svake spolne kategorije. Naredba DISTINCT koristimo da bi eliminirali duplikati. Broj čuvara smo spremili kao „broj_cuvara“

Pomoću funkcije DATEDIFF odredili smo koliko je vremena prošlo od datuma rođenja do sada. Podijeli smo taj broj sa 365 da bi dobili godine. Funkcijom AVG dobili smo prosjeke svih godina. I na kraju smo preko ROUND funkcije zaokružili taj broj na 1 decimalu. Tu vrijednost smo spremili pod nazivom „prosjecna_dob_godina“.

Funkcija SUM zbraja sve cijene usluga koje su pružili čuvari unutar svake grupe prema spolu, dajući ukupnu zaradu od usluga za svaki spol.

Nakon što je pogled kreiran, testiramo ga izvršavanjem „SELECT * FROM spolovi“, što prikazuje sve podatke koji su pohranjeni u tom pogledu.

8.1.3. Pogled

Ovaj kod broji aktivnosti s različitim intenzitetima za svaki grad.

```
CREATE VIEW intenzitet_po_gradu AS
SELECT
    cuvar.grad AS puni_naziv_grada,
    SUM(aktivnost.intenzitet = 'srednje') AS broj_srednjeg,
    SUM(aktivnost.intenzitet = 'nisko') AS broj_niskog,
    SUM(aktivnost.intenzitet = 'visoko') AS broj_visokog
FROM cuvar
JOIN cuvar_aktivnost ON cuvar.id = cuvar_aktivnost.cuvar_id
JOIN aktivnost ON cuvar_aktivnost.aktivnost_id = aktivnost.id
GROUP BY cuvar.grad
ORDER BY cuvar.grad;
```

U ovom kodu stvaramo pogled „intezitet_po_gradu“ koji daje detaljan pregled broja aktivnosti različitih intenziteta za svaki grad.

U FROM dijelu odabiremo tablice „cuvar“, „aktivnost“ i „cuvar_aktivnost“.

Koristimo JOIN ključne riječi kako bismo spojili podatke iz različitih tablica. Prvo, spajamo tablice „cuvar“ i „cuvar_aktivnosti“ prema identifikatoru čuvara. Zatim spajamo tablicu „aktivnost“ s tablicom „cuvar_aktivnost“ prema identifikatoru aktivnosti. Ovo omogućuje da se podatci povežu na temelju veza između čuvara i njihovih aktivnosti.

Koristimo funkciju SUM zajedno sa logičkim izrazima kako bismo izračunali broj aktivnosti različitih intenziteta za svaki grad. Za svaki grad, logički izraz provjerava svaku aktivnost i dodaje 1 ako je njezin intenzitet „srednji“, inače dodaje 0. Isti postupak se radi i za niske i visoke intenzitete.

Kada se svi podatci povežu i izračunaju, koristimo naredbu GROUP BY izraz kako bismo grupirali podatke po punim nazivima grada. Ovo osigurava da dobijemo rezultate za svaki grad pojedinačno.

Na kraju, koristimo ORDER BY izraz kako bismo sortirali rezultate po punim nazivima grada. Ovo nam omogućava da rezultat pregledamo u uređenom redoslijedu.

Nakon što je pogled kreiran, izvršava se upit koji prikazuje sve podatke iz toga pogleda.

8.1.4. Pogled

Prikaži sve aktivnosti koje traju duže od 1 sata i imaju više od 5 rezervacija.

```
CREATE VIEW dugacke_popularne_rezervacije AS
SELECT * FROM aktivnost
WHERE trajanje > '01:00:00'
AND id IN (
    SELECT aktivnost_id
    FROM cuvar_aktivnost
    GROUP BY aktivnost_id
    HAVING COUNT(*) > 5);
```

U ovom kodu stvaramo pogled „dugacke_popularne_rezervacije“ koji daje sve aktivnosti koje traju duže od 1 sat i imaju više od 5 rezervacija.

Unutar FROM dijela se odabire se tablica aktivnosti iz koje se u WHERE dijelu selektira sve retke gdje je trajanje aktivnosti veće od jednog sata.

U podupitu se selektiraju „aktivnost_id“ iz tablice „cuvar_aktivnosti“, nakon toga se grupira prema „aktivnost_id“ i prebrojava one koje se pojavljuju više od 5 puta za svaki ID aktivnost.

Na kraju, glavni upit provjerava jesu li ID-jevi aktivnosti iz rezultata podupita prisutni u glavnoj tablici „aktivnost“.

Na samom kraju poziva se pogled tako da se selektira sve podatke iz tog pogleda.

8.2. Patricia Lazić

8.2.1. Pogled

Detaljan prikaz ljubimca

```
CREATE VIEW detaljan_prikaz_ljubimac AS
SELECT
  l.id AS ljubimac_id,
  l.ime AS ljubimac_ime,
  l.dob AS ljubimac_dob,
  l spol AS ljubimac_spol,
  l.boja AS ljubimac_boja,
  l.tezina AS ljubimac_tezina,
  l.zdravstveni_problemi AS ljubimac_zdravstveni_problemi,
  l.veterinarski_podaci AS ljubimac_veterinarski_podaci,
  l.posebne_potrebe AS ljubimac_posebne_potrebe,
  l.socijalnost AS ljubimac_socijalnost,
  l.aktivnosti AS ljubimac_aktivnosti,
  p.naziv AS pasmina_naziv,
  v.naziv AS vrsta_naziv,
  (SELECT COUNT(r1.rezervacija_id) FROM rezervacija_ljubimac r1 WHERE
r1.ljubimac_id = l.id) AS broj_rezervacija,
  (SELECT GROUP_CONCAT(r1.rezervacija_id ORDER BY r1.rezervacija_id ASC SEPARATOR
', ') FROM rezervacija_ljubimac r1 WHERE r1.ljubimac_id = l.id) AS id_rezervacija
FROM ljubimac l
JOIN pasmina p ON l.pasmina_id = p.id
JOIN vrsta v ON p.vrsta_id = v.id;
```

Ovaj pogled prikazuje detaljne informacije o svakom ljubimcu, uključujući podatke o pasmini, vrsti, te broju i popisu rezervacija. Koristi se INNER JOIN za spajanje tablica ljubimac, pasmina i vrsta. Podupiti se koriste za izračun broja rezervacija i popisa ID-ova rezervacija za svakog ljubimca.

8.2.2. Pogled

Statistika za tablicu vrsta. Prikazuje prosječne, maksimalne i minimalne vrijednosti određenih podataka za ljubimce, grupirano po vrsti.

```
CREATE VIEW vrsta_statistika AS
SELECT
  v.id AS id,
  v.naziv AS naziv,
  AVG(l.dob) AS prosjecna_dob,
```

```

MIN(l.dob) AS minimalna_dob,
MAX(l.dob) AS maksimalna_dob,
STDDEV(l.dob) AS standardna_devijacija_dob,
SUM(l.dob) AS ukupna_dob,
AVG(l.tezina) AS prosjecna_tezina,
MIN(l.tezina) AS najmanja_tezina,
MAX(l.tezina) AS najveca_tezina,
STDDEV(l.tezina) AS standardna_devijacija_tezina,
SUM(l.tezina) AS ukupna_tezina,
COUNT(l.id) AS broj_ljubimaca
FROM vrsta v
JOIN pasmina p ON v.id = p.vrsta_id
JOIN ljubimac l ON p.id = l.pasmina_id
GROUP BY v.id;

```

Ovaj pogled prikazuje statističke podatke za svaku vrstu, uključujući prosječnu, minimalnu, maksimalnu i standardnu devijaciju za dob i težinu ljubimaca, kao i ukupne vrijednosti. Koristi se INNER JOIN za spajanje tablica vrsta, pasmina i ljubimac, te GROUP BY za grupiranje po ID-u vrste.

8.2.3. Pogled

Prikaz svih ljubimaca i njihove rezervacije, uključujući i datum rezervacije.

```

CREATE VIEW ljubimci_s_rezervacijama AS
SELECT
    l.id AS ljubimac_id,
    l.ime AS ljubimac_ime,
    p.naziv AS pasmina_naziv,
    v.naziv AS vrsta_naziv,
    rl.rezervacija_id AS rezervacija_id,
    r.datum AS rezervacija_datum
FROM
    ljubimac l
JOIN pasmina p ON l.pasmina_id = p.id
JOIN vrsta v ON p.vrsta_id = v.id
LEFT JOIN rezervacija_ljubimac rl ON l.id = rl.ljubimac_id
LEFT JOIN rezervacija r ON rl.rezervacija_id = r.id;

```

Ovaj pogled prikazuje osnovne podatke o ljubimcima i njihovim rezervacijama, uključujući identifikator i ime ljubimca, naziv pasmine, naziv vrste, identifikator rezervacije i datum rezervacije. Pogled koristi JOIN operacije za spajanje tablica ljubimac, pasmina, vrsta, rezervacija_ljubimac i rezervacija.

8.2.4. Pogled

Prikazuje statistiku za ljubimce grupirano po pasmini.

```

CREATE VIEW pasmina_statistika AS
SELECT
    p.id AS pasmina_id,
    p.naziv AS pasmina_naziv,
    p.opis AS pasmina_opis,
    p.porijeklo AS pasmina_porijeklo,
    p.tezina AS pasmina_tezina,

```

```

p.zivotni_vijek AS pasmina_zivotni_vijek,
v.naziv AS vrsta_naziv,
COUNT(l.id) AS broj_ljubimaca,
AVG(l.dob) AS prosjecna_dob_ljubimaca,
MIN(l.dob) AS najmanja_dob_ljubimaca,
MAX(l.dob) AS najveca_dob_ljubimaca,
AVG(l.tezina) AS prosjecna_tezina_ljubimaca,
MIN(l.tezina) AS najmanja_tezina_ljubimaca,
MAX(l.tezina) AS najveca_tezina_ljubimaca,
(SELECT AVG(rez_count)
FROM (
    SELECT COUNT(r1.rezervacija_id) AS rez_count
    FROM ljubimac l2
    LEFT JOIN rezervacija_ljubimac r1 ON l2.id = r1.ljubimac_id
    WHERE l2.pasmina_id = p.id
    GROUP BY l2.id
) AS broj_rez) AS prosjecan_broj_rezervacija_po_ljubimcu
FROM pasmina p
LEFT JOIN vrsta v ON p.vrsta_id = v.id
LEFT JOIN ljubimac l ON p.id = l.pasmina_id
GROUP BY p.id, v.naziv;

```

Ovaj pogled prikazuje statističke podatke za svaku pasminu, uključujući broj ljubimaca, prosječnu, minimalnu, maksimalnu dob i težinu ljubimaca, te prosječan broj rezervacija po ljubimcu. Koristi se LEFT JOIN za spajanje tablica pasmina, vrsta i ljubimac, te GROUP BY za grupiranje po ID-u pasmine i nazivu vrste. Podupit se koristi za izračun prosječnog broja rezervacija po ljubimcu za svaku pasminu.

8.3. Luka Vilagoš

8.3.1. Pogled

Pogled prikazuje sve relevantne podatke o ustanovi s značajkom 'Igralište'

```

CREATE VIEW ustanova_tip_vlasnik_s_igralištem AS
SELECT u.naziv AS ustanova_naziv, t.naziv AS tip_naziv, v.ime AS vlasnik_ime,
v.prezime AS vlasnik_prezime, z.naziv AS znacajka_naziv
FROM ustanova u
JOIN tip t ON u.tip_id = t.id
JOIN vlasnik v ON u.vlasnik_id = v.id
JOIN ustanova_znacajka uz ON u.id = uz.ustanova_id
JOIN znacajka z ON uz.znacajka_id = z.id
WHERE z.naziv = 'Igralište'
GROUP BY u.id, u.naziv, t.naziv, v.ime, v.prezime, z.naziv;

```

Ovaj pogled prikazuje sve relevantne podatke o ustanovama koje imaju značajku 'Igralište', uključujući naziv ustanove, tip, ime i prezime vlasnika, te naziv značajke. Koristi INNER JOIN za spajanje tablica i GROUP BY za grupiranje podataka po ID-u ustanove, nazivu ustanove, tipu, imenu i prezimenu vlasnika te nazivu značajke.

8.3.2. Pogled

Pogled prikazuje sve vlasnike koji imaju ustanove tipa 'Pet Spa' i koliko takvih ustanova imaju

```
CREATE VIEW broj_ustanova_tipa_pet_spa_odredenog_vlasnika AS
SELECT v.ime, v.prezime, (
    SELECT COUNT(*)
    FROM ustanova u
    WHERE u.vlasnik_id = v.id AND u.tip_id IN (
        SELECT t.id
        FROM tip t
        WHERE t.naziv = 'Pet Spa'
    )
) AS broj_ustanova
FROM vlasnik v
WHERE v.id IN (
    SELECT u.vlasnik_id
    FROM ustanova u
    JOIN tip t ON u.tip_id = t.id
    WHERE t.naziv = 'Pet Spa'
);
```

Ovaj pogled prikazuje sve vlasnike koji imaju ustanove tipa 'Pet Spa' i broj takvih ustanova. Koristi podupite za brojanje ustanova tipa 'Pet Spa' za svakog vlasnika i filtrira vlasnike koji posjeduju takve ustanove.

8.3.3. Pogled

Pogled prikazuje detaljan prikaz svih podataka o ustanovama

```
CREATE VIEW ustanova_detalji AS
SELECT
    u.id AS ustanova_id,
    u.naziv AS ustanova_naziv,
    u.adresa AS ustanova_adresa,
    u.broj_telefona AS ustanova_broj_telefona,
    u.email AS ustanova_email,
    u.radno_vrijeme AS ustanova_radno_vrijeme,
    u.opis AS ustanova_opis,
    u.web_stranica AS ustanova_web_stranica,
    u.kapacitet AS ustanova_kapacitet,
    u.ocjena AS ustanova_ocjena,
    t.id AS tip_id,
    t.naziv AS tip_naziv,
    t.opis AS tip_opis,
    v.ime AS vlasnik_ime,
    v.prezime AS vlasnik_prezime,
    v.broj_telefona AS vlasnik_broj_telefona,
    v.email AS vlasnik_email,
    v.datum_rodenja AS vlasnik_datum_rodenja,
    v.adresa AS vlasnik_adresa,
    v.grad AS vlasnik_grad,
    v.drzava AS vlasnik_drzava,
    v.spol AS vlasnik_spol,
```

```

GROUP_CONCAT(z.naziv ORDER BY z.naziv ASC SEPARATOR ', ') AS znacajke,
(SELECT COUNT(*) FROM ustanova_znacajka uz WHERE uz.ustanova_id = u.id) AS
broj_znacajki,
(SELECT GROUP_CONCAT(CONCAT(z1.naziv, ' (', z1.dostupnost, ')') ORDER BY
z1.naziv ASC SEPARATOR ', ')
FROM znacajka z1
JOIN ustanova_znacajka uz1 ON z1.id = uz1.znacajka_id
WHERE uz1.ustanova_id = u.id) AS znacajke_detalj
FROM ustanova u
JOIN tip t ON u.tip_id = t.id
JOIN vlasnik v ON u.vlasnik_id = v.id
JOIN ustanova_znacajka uz ON u.id = uz.ustanova_id
JOIN znacajka z ON uz.znacajka_id = z.id
GROUP BY u.id, t.id, v.id;

```

Ovaj pogled prikazuje detaljan prikaz svih podataka o ustanovama, uključujući podatke o ustanovi, tipu, vlasniku i značajkama. Koristi INNER JOIN za spajanje tablica i GROUP BY za grupiranje podataka po ID-u ustanove, tipu i vlasniku.

8.3.4. Pogled

Pogled prikazuje sve vlasnike koji žive u Zagrebu i njihove ustanove

```

CREATE VIEW vlasnici_iz_zagreba_i_njihove_ustanove AS
SELECT v.ime, v.prezime, v.grad, GROUP_CONCAT(u.naziv ORDER BY u.naziv ASC
SEPARATOR ', ') AS ustanove
FROM ustanova u
JOIN vlasnik v ON u.vlasnik_id = v.id
WHERE v.grad = 'Zagreb'
GROUP BY v.id
ORDER BY v.prezime, v.ime;

```

Ovaj pogled prikazuje sve vlasnike koji žive u Zagrebu i njihove ustanove. Koristi INNER JOIN za spajanje tablica ustanova i vlasnik, filtrira rezultate prema gradu 'Zagreb' i koristi GROUP BY za grupiranje po ID-u vlasnika. Rezultati se sortiraju prema prezimenu i imenu vlasnika.

8.4. Paula Vorih

8.4.1. Pogled

Pogled prikazuje broj ustanova u svakoj regiji u silaznom redoslijedu.

```

CREATE VIEW broj_ustanova_po_regiji AS
SELECT r.naziv AS naziv_regije,
COUNT(u.id) AS broj_ustanova
FROM ustanova u
JOIN regija r ON u.regija_id = r.id
GROUP BY r.naziv
ORDER BY broj_ustanova DESC;

```


Kreiramo pogled koji se zove "broj_ustanova_po_regiji" sa "CREATE VIEW broj_ustanova_po_regiji AS" naredbom. "FROM ustanova u" predstavlja glavnu tablicu iz koje izvlačimo podatke, a to je "ustanova", i dajemo joj alias "u". Sa operacijom JOIN pridružujemo tablicu "regija" s aliasom "r" tako da se "ustanova.regija_id" i "regija.id" podudaraju. Grupiramo sve po nazivu regije kako bismo mogli prebrojati sve ustanove u nekoj regiji. Nakon toga silazno sortiramo broj ustanova za svaku regiju. Sa "SELECT r.naziv AS naziv_regije" odabiremo stupac "naziv" u tablici "regija" i dajemo mu alias "naziv_regije". Unutar SELECT dijela sadržana je i funkcija agregacije koja broji ustanove.

8.4.2. Pogled

Pogled prikazuje mjesta interesa u svakoj regiji poredano po regijama s najvećom do najmanje površine.

```
CREATE VIEW mjesto_interesa_u_regiji_i_info AS
SELECT m.id AS mjesto_interesa_id,
       m.naziv AS mjesto_interesa_naziv,
       m.adresa AS mjesto_interesa_adresa,
       r.naziv AS regija_naziv,
       r.opis AS regija_opis,
       r.povrsina AS regija_povrsina
FROM mjesto_interesa m
JOIN regija r ON m.regija_id = r.id
ORDER BY regija_povrsina DESC;
```

Kreiramo pogled koji se zove "mjesto_interesa_u_regiji_i_info" sa "CREATE VIEW mjesto_interesa_u_regiji_i_info AS" naredbom. "FROM mjesto_interesa m" predstavlja glavnu tablicu iz koje izvlačimo podatke, a to je "mjesto_interesa", i dajemo joj alias "m". Sa operacijom JOIN pridružujemo tablicu "regija" s aliasom "r" tako da se "mjesto_interesa.regija_id" i "regija.id" podudaraju. Potom silazno sortiramo regije po njihovim površinama. Na kraju sa SELECT naredbom odabiremo sve potrebne stupce i dajemo im prikladne aliase.

8.4.3. Pogled

Pogled prikazuje broj čuvara u svakoj regiji.

```
CREATE VIEW broj_cuvara_po_regiji AS
SELECT r.naziv AS regija_naziv,
       COUNT(c.id) AS broj_cuvara
FROM cuvar c
JOIN cuvar_ustanova cu ON c.id = cu.cuvar_id
JOIN ustanova u ON cu.ustanova_id = u.id
JOIN regija r ON u.regija_id = r.id
GROUP BY r.naziv;
```

Kreiramo pogled koji se zove "broj_cuvara_po_regiji" sa "CREATE VIEW broj_cuvara_po_regiji AS" naredbom. "FROM cuvar c" predstavlja glavnu tablicu iz koje izvlačimo podatke, a to je "cuvar", i dajemo joj alias "c". Koristimo join operacije nad tablicama "cuvar_ustanova", "ustanova" i "regija" sa ON uvjetom gdje spajamo id vrijednosti iz relevantnih tablica. Izvodimo grupiranje po nazivu regije, zatim brojimo čuvare sa funkcijom "COUNT(c.id) AS broj_cuvara" koja je uz "r.naziv AS regija_naziv" sadržana u SELECT dijelu.

8.4.4. Pogled

Pogled prikazuje prosječnu ocjenu ustanova u svakoj regiji te najbolju ocjenu ustanove u toj regiji.

```
CREATE VIEW prosjecna_ocjena_i_top_ustanova_po_regiji AS
WITH top_ustanova AS (
    SELECT u.regija_id,
           u.naziv AS naziv_top_ustanove,
           u.ocjena AS ocjena_top_ustanove
    FROM ustanova u
    JOIN (
        SELECT regija_id,
               MAX(ocjena) AS max_ocjena
        FROM ustanova
        GROUP BY regija_id
      ) max_u ON u.regija_id = max_u.regija_id AND u.ocjena = max_u.max_ocjena
)
SELECT r.naziv AS naziv_regije,
       AVG(u.ocjena) AS prosjecna_ocjena,
       tu.naziv_top_ustanove,
       tu.ocjena_top_ustanove
FROM ustanova u
JOIN regija r ON u.regija_id = r.id
JOIN top_ustanova tu ON r.id = tu.regija_id
GROUP BY r.naziv, tu.naziv_top_ustanove, tu.ocjena_top_ustanove;
```

Kreiramo pogled koji se zove "prosjecna_ocjena_i_top_ustanova_po_regiji" sa "CREATE VIEW prosjecna_ocjena_i_top_ustanova_po_regiji AS" naredbom. Pomoću WITH klauzule kreiramo pomoćnu tablicu top_ustanova kojoj dajemo alias "tu" u "JOIN top_ustanova tu ON r.id = tu.regija_id" operaciji. Unutar WITH klauzule imamo glavnu tablicu "ustanova" s aliasom "u" koju spajamo sa sadržajem podupita gdje računamo maksimalnu ocjenu pojedine regije. U ON uvjetu nakon podupita dobivamo najveću ocjenu za svaku regiju. Ispod WITH dijela također imamo glavnu tablicu "ustanova" s aliasom "u" gdje pridružujemo vrijednosti iz "regija" s aliasom "r" da se vrijednosti "ustanova.regija_id" i "regija_id" podudaraju te pridružujemo "top_ustanova" s aliasom "tu" što smo dobili iz gornjeg dijela s uvjetom da se podudaraju "regija.id" i "top_ustanova.regija_id". Kreiramo grupu stupaca "r.naziv", "tu.naziv_top_ustanove" i "tu.ocjena_top_ustanove" i prilikom odabira stupaca u SELECT dijelu računamo prosječnu ocjenu kako bi na kraju dobili prosječne ocjene i ocjene ustanove sa najvišom ocjenom u toj regiji.

8.5. Alma Reka

8.5.1. Pogled

Ovaj upit će dohvatiti sve potvrđene rezervacije

```
CREATE VIEW confirmed_reservations AS
SELECT *
FROM rezervacija
WHERE potvrđeno = TRUE;
```

Prikaz 'confirmed_reservation' odabire sve stupce iz tablice 'rezervacija' za rezervacije koje su potvrđene ('potvrđeno' = TRUE'). Ovaj prikaz pojednostavljuje dohvaćanje potvrđenih rezervacija, olakšavajući upravljanje i analizu potvrđenih rezervacija.

8.5.2. Pogled

Dohvati mi korisnike s više od jednog ljubimca

```
CREATE VIEW korisnici_sa_vise_ljubimaca AS
SELECT korisnik.id, korisnik.ime, korisnik.prezime,
COUNT(korisnik_ljubimac.ljubimac_id) AS pet_count
FROM korisnik
JOIN korisnik_ljubimac ON korisnik.id = korisnik_ljubimac.korisnik_id
GROUP BY korisnik.id
HAVING pet_count > 1;
```

Prikaz 'korisnici_sa_vise_ljubimaca' odabire ID, ime i prezime korisnika, zajedno s brojem njihovih kućnih ljubimca ('pet_count' > 1). Ovaj prikaz je koristan za identificiranje i upravljanje korisnicima s više kućnih ljubimaca.

8.5.3. Pogled

Dohvati mi sve korisnike koji su iz Hrvatske i grada Zagreba

```
CREATE VIEW korisnici_u_zagrebu AS
SELECT *
FROM korisnik
WHERE drzava = 'Hrvatska'
AND grad = 'Zagreb';
```

Prikaz 'korisnici_u_zagrebu' odabire sve stupce iz tablice 'korisnik' za korisnike koji žive u Hrvatskoj ('drzava' = Hrvatska) i čiji je grad prebivališta Zagreb ('grad' = zagreb). Ovaj prikaz pomaže u fokusiranju na korisnike koji se nalaze u Zagrebu za ciljane usluge ili komunikaciju.

8.5.4. Pogled

Dohvati mi sve korisnike koji imaju barem jednu recenziju

```
CREATE VIEW korisnici_s_recenzijama AS
SELECT korisnik.id, korisnik.ime, korisnik.prezime, COUNT(recenzija.id) AS
broj_recenzija
FROM korisnik
JOIN recenzija ON korisnik.id = recenzija.korisnik_id
GROUP BY korisnik.id, korisnik.ime, korisnik.prezime
HAVING review_count > 0;
```

Prikaz 'korisnici_s_recenzijama' odabire ID, ime i prezime korisnika, zajedno s brojem recenzija koje su napisali iz tablica 'korisnik' i 'recenzija'. Uključuje samo korisnike koji su napisali barem jednu recenziju ('broj_recenzija' > 0). Ovaj prikaz je koristan za identificiranje aktivnih korisnika koji daju povratne informacije, što može biti vrijedno za poboljšanje usluga i angažman korisnika.

9. Okidači

9.1. Luka Vilagoš

9.1.1. Okidač

Ažurira ocjenu ustanove nakon nove recenzije

```
DELIMITER //
```

```
CREATE TRIGGER azuriraj_ocjenu_nakon_inserta
AFTER INSERT ON recenzija
FOR EACH ROW
BEGIN
    DECLARE new_avg_ocjena DECIMAL(3, 2);
    SELECT AVG(ocjena) INTO new_avg_ocjena
    FROM recenzija
    WHERE ustanova_id = NEW.ustanova_id;
    UPDATE ustanova
    SET ocjena = new_avg_ocjena
    WHERE id = NEW.ustanova_id;
END;
//
DELIMITER ;
```

Ovaj okidač ažurira prosječnu ocjenu ustanove svaki put kada se ubaci nova recenzija. Nakon umetanja nove recenzije u tablicu recenzija, trigger izračunava novu prosječnu ocjenu za odgovarajuću ustanovu koristeći AVG funkciju. Zatim ažurira tablicu ustanova s novom prosječnom ocjenom.

9.1.2. Okidač

Osigurava da odabrana ustanova i usluga su povezane čuvarom

```
DELIMITER //
```

```
CREATE TRIGGER validiraj_usluga_cuvar_prije_inserta
BEFORE INSERT ON rezervacija
FOR EACH ROW
BEGIN
    DECLARE v_count INT;
    SELECT COUNT(*)
    INTO v_count
    FROM usluga u
    JOIN cuvar c ON u.cuvar_id = c.id
    JOIN cuvar_ustanova cu ON c.id = cu.cuvar_id
    WHERE u.id = NEW.usluga_id
    AND cu.ustanova_id = NEW.ustanova_id;
    IF v_count = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Odabrana usluga_id nije asocirana sa danom ustanova_id
kroz cuvara';
    END IF;
END;
//
DELIMITER ;
```

```
//  
DELIMITER ;
```

Ovaj trigger osigurava da se nova rezervacija može umetnuti samo ako je odabrana usluga povezana s danom ustanovom preko čuvara. Prije umetanja nove rezervacije u tablicu rezervacija, trigger provjerava postoji li odgovarajuća veza između usluge, čuvara i ustanove. Ako veza ne postoji, trigger podiže grešku i sprječava umetanje.

10. Generiranje podataka

10.1. LLM

Za generiranje podataka koristili smo određene Large Language modele poput GPT 3.5, GPT 4 GPT 4o. Ovu metodu smo koristili za generiranje svih podataka osim za M:M tablice. Započeli smo generaciju sa promptom poput slejdećeg:

This is my table:

```
CREATE TABLE znacajka (  
  id INT AUTO_INCREMENT,  
  naziv VARCHAR(256) NOT NULL,  
  opis TEXT,  
  dostupnost TEXT,  
  cijena_dodatka FLOAT(2),  
  vrijeme_otvaranja TIME,  
  vrijeme_zatvaranja TIME,  
  CONSTRAINT PRIMARY KEY (id)  
  
);
```

I need you to generate as much data as you can. The context for the data is a pet hotel type service where pet owners can take their pets to certain facilities for play or to take care of them while they are away. This particular table contains certain aspects of facilities that might benefit the pet owner. For example a large yard, playground, pet spa etc. The pets range from dogs and cats to birds and snakes so keep in mind the more exotic animals, ignore the id field.

Upit se sastoji od sljedeće strukture. Prvo smo dali SQL kod za tablicu za koju želimo da generira podatke. Nakon toga smo dali kontekst što se nalazi u tablici, primjer podataka koji bi se mogli nalaziti u tablici. Ako tablica uključuje ikakav strani ključ naveli smo u kojem okviru brojeva se nalazi pošto sve tablice koriste AUTO INCREMENT kao ID. Nakon što nam da odgovor u obliku:

Sure! Based on the provided table structure and the context of a pet hotel service, I'll generate sample data in CSV format. This data will include various facilities and aspects beneficial to pet owners.

Here is the CSV format for your table znacajka:

```
naziv,opis,dostupnost,cijena_dodatka,vrijeme_otvaranja,vrijeme_zatvaranja  
"Large Yard","A large yard where pets can run freely and play with other  
pets.", "Available",10.00,"08:00:00","20:00:00"  
"Playground","A dedicated playground with toys and agility equipment for  
dogs.", "Available",15.00,"08:00:00","18:00:00"
```

```
"Pet Spa","A spa for pets offering grooming, massages, and
relaxation.", "Available",25.00,"09:00:00","17:00:00"
"Bird Aviary","A spacious aviary where birds can fly and
socialize.", "Available",12.50,"08:00:00","16:00:00"
"Reptile House","A controlled environment for reptiles to stay
comfortable.", "Limited Availability",20.00,"10:00:00","18:00:00"
"Cat Lounge","A quiet and cozy lounge specifically designed for
cats.", "Available",8.00,"09:00:00","19:00:00"
"Small Animal Playpen","An enclosed play area for small animals like rabbits and
guinea pigs.", "Available",5.00,"07:00:00","19:00:00"
"Fish Pond","A tranquil pond for fish with daily feeding and
maintenance.", "Available",18.00,"09:00:00","17:00:00"
...
```

Nakon toga bi uzeli dobivene csv podatke, proveli ih kroz google translate kako bi bili na hrvatskom i spremili u .csv file na računalu. U slučaju grešaka u podacima dali smo daljnje upite poput sljedećeg:

```
Remove the quotation marks but be careful not to accidentally create more data
columns with wild commas in the data. If there is, just remove it.
```

Nakon toga bi dobili odgovor sa ispravljenim podacima te bi ispravili našu csv datoteku.

10.2. Python

Za generiranje many to many relacijskih tablica koristili smo sljedeću python skriptu:

```
import csv
import random

file_path = "example.csv"

with open(file_path, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["example_id", "example_id"])
    counter = 1
    for i in range(1, 60):
        writer.writerow([i, counter])
        if counter == 39:
            counter = 0
        counter += 1

    writer.writerow([30, 1])

print("Data has been written to", file_path)
```

Skripta prvo specifikira ime datoteke u koju želimo spremiti podatke. Nadalje, koristeći metodu `open()` otvorimo novu vezu do datoteke u koju ćemo pisati. Koristeći metodu `writer` iz klase `csv` koju smo importali na vrhu datoteke stvorimo objekt pisača koji će pisati u našu otvorenu csv datoteku. Nakon toga iskoristimo to što su svi id atributi tipa `AUTO INCREMENT` te jednostvanom `for-in` petljom i `counter` varijablom generiramo potrebne podatke. Parametre `for` petlje i logiku `countera` smo mijenjali po zahtjevima tablica. Nakon toga se veza sa datotekom zatvara i na konzolu se ispisuje poruka.

11. Zaključak

U ovom projektu smo uspješno kreirali cjelokupnu bazu podataka za sustav za rezervaciju čuvara ljubimaca. Naša baza podataka sadrži 20 tablica koje obuhvaćaju sve relevantne entitete kao što su ustanove, vlasnici, značajke, tipovi ustanova, čuvari, usluge, korisnici i rezervacije. Svaki član tima pridonio je pisanjem četiri upita i četiri pogleda, čime smo osigurali sveobuhvatan pregled i analizu podataka unutar sustava.

Osim dizajniranja baze podataka, generirali smo podatke za sve tablice kako bismo simulirali stvarno okruženje i omogućili testiranje funkcionalnosti sustava. Kreirali smo i nekoliko okidača (triggers) kako bismo osigurali integritet podataka i omogućili automatsko ažuriranje ključnih informacija, poput prosječnih ocjena ustanova nakon svake nove recenzije.

Tijekom razvoja, poseban naglasak stavili smo na validaciju i osiguranje povezanosti usluga i čuvara unutar ustanova, čime smo osigurali točnost i dosljednost rezervacija. Naš sustav omogućava korisnicima jednostavnu i sigurnu rezervaciju usluga čuvara za ljubimce, dok administracija ustanova može efikasno upravljati resursima i analizirati korisničke ocjene i povratne informacije.

Ovaj projekt demonstrira našu sposobnost modeliranja složenih poslovnih procesa, dizajniranja i implementacije baze podataka te osiguravanja funkcionalnosti putem upita, pogleda i okidača. Naša baza podataka je robustan i pouzdan alat koji može podržati operativne potrebe sustava za rezervaciju čuvara ljubimaca, te pruža čvrstu osnovu za budući razvoj i proširenje sustava.