Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

DOKUMENTACIJA UZ PROJEKTNI ZADATAK

SUSTAV ZA UPRAVLJANJE RESTORANOM

TIM 8

Dino Červar
Lenny Dagostini
Mihael Domjan
Sven Gerenčir
Josip Tomo Licardo

Studijski smjer: Informatika
Kolegij: Baze podataka 2
Mentor: doc. dr. sc. Goran Oreški

SADRŽAJ

UVOD

Kao što smo naveli u prvom sastanku tima, za temu projekta odlučili smo jednoglasno “Sustav za upravljanje restoranom”. Voditelj tima i osoba za dokumentaciju pratili su i zapisivali rad programera koji su implementirali bazu podataka. Kroz period našeg rada održali smo nekoliko sastanaka, ali naša komunikacija je bila gotovo svakodnevna.

Relacije koje smo zadali u našem projektu su : rezervacija, dostava_stavka, stol, otpis, otpis_stavka, dostava, adresa, meni, stavka_meni, racun, dobavljac, rezije, namirnica, nacin_placanja, kategorija_rezije, osoba, djelatnik, stavka_racun, kategorija_namirnica, nabava_stavka, catering_narucitelj, zanimanje, catering_zahtjev, djelatnik_smjena, catering_stavka, sadrzi_alergen, nabava, alergen, catering, smjena i djelaticni_catering.
Svaka relacija sadrži svoj zasebni ID (primarni ključ) zbog jedinstvenosti, a isto tako i neke strane ključeve.

Svrha i cilj našega projekta bila je prikazati sustav upravljanja restoranom što više realnijim.

PODJELA I SAŽETAK ULOGA

Dino Červar

- Programer
- Popunjava tablice sa zapisima (dodaje insertove)
- Izrada tablica
- Izrada sheme baze podataka (relacijski model)

Lenny Dagostini

- Programer
- Popunjava tablice sa zapisima (dodaje insertove)
- Izrada upita, pogleda, funkcija, procedura, okidača

Mihael Domjan

- Osoba za komunikaciju
- Izrada dokumentacije
- Izrada ER – dijagrama

Sven Gerenčir

- Programer
- Izrada web sučelja
- Izrada sheme baze podataka (relacijski model)

Josip Tomo Licardo

- Voditelj
- Programer
- Popunjava tablice sa zapisima (dodaje insertove)
- Izrada upita, pogleda, funkcija, procedura, okidača
- Izrada sheme baze podataka (relacijski model)

VODITELJ:

Dosta veliku ulogu u našem timu imao je voditelj. Njegova uloga je bila da nadgleda proces izrade projekta. Davao nam je motivaciju za rad. Upozoravao je na greške i davao neke nove ideje. Isto tako vodio je brigu o rokovima te da se sve preda na vrijeme bez kašnjenja. Kao složan tim oslobodili smo ga od dodjela uloga, već smo to odradili međusobnim dogovorom.

OSOBA ZA KOMUNIKACIJU i OSOBA ZA IZRADU DOKUMENTACIJE:

Njegov zadatak je da uz voditelja organizira sastanke kada je za to bilo potrebe i naravno sve zapisivati. Svaki sastanak bio je organiziran tako da se prilagodi svakom članu tima. Pomoću zapisnika osoba za komunikaciju svakom je članu dala na uvid nove ideje projekta i stvari koje su možda promakle. Isto tako njegova obveza bila je izrada “Izvješća s prvog sastanka”, “Sažetka projekta”, izrada dokumentacije.

PROGRAMERI:

Njihov zadatak bio je izrada baze podataka i web sučelja. Sučelje je napravljeno pomoću PHP-a. Ostali alati koje su koristili su HTML, CSS, Javascript. S obzirom da su programeri našeg tima već bili upoznati sa tim programima odlučili su se za njih radi lakšeg i bržeg rada.

ER DIJAGRAM

Adresa <-> dobavljač

Veza „ima“ povezuje tablice adresa i dobavljač.

Dobavljač <-> nabava

Veza „ima“ povezuje tablice dobavljač i nabava

Nabava <-> nabava_stavka

Veza „sadrži“ povezuje tablice nabava i nabava_stavka

Nabava_stavka <-> namirnica

Veza „sadrži“ povezuje tablice nabava_stavka i namirnica.

Otpis <-> otpis_stavka

Veza „sadrži“ povezuje tablice otpis i otpis_stavka

Otpis_stavka <-> namirnica

Veza „sadrži“ povezuje tablice otpis_stavka i namirnica.

Namirnica <-> kategorija_namirnica

Veza „ima“ povezuje tablice namirnica i kategorija_namirnica.

Namirnica <-> stavka_meni

Veza „ima“ povezuje tablice namirnica i stavka_meni.

Stavka_meni <-> meni

Veza „ima“ povezuje tablice stavka_meni i meni.

Meni <-> sadrži_alergen

Veza „ima“ povezuje tablice meni i sadrži_alergen.

Alergen <-> sadrži_alergen

Veza „ima“ povezuje tablice alergen i sadrži_alergen.

Meni <-> stavka_račun

Veza „sadrži“ povezuje tablice meni i stavka_račun.

Meni <-> dostava_stavka

Veza „sadrži“ povezuje tablice meni i dostava_stavka.

Meni <-> catering_stavka

Veza „sadrži“ povezuje tablice meni i catering_stavka.

Račun <-> način_plačanja

Veza „sadrži“ povezuje tablice račun i način_plačanja.

Račun <-> stavka_račun

Veza „ima“ povezuje tablice račun i stavka_račun.

Adresa <-> dostava

Veza „ima“ povezuje tablice adresa i dostava.

Račun <-> stol

Veza „sadrži“ povezuje tablice račun i stol.

Račun <-> djelatnik

Veza „sadrži“ povezuje tablice račun i djelatnik.

Stol <-> rezervacija

Veza „sadrži“ povezuje tablice stol i rezervacija.

Rezervacija <-> osoba

Veza „rezervira“ povezuje tablice rezervacija i osoba.

Osoba <-> djelatnik

Veza „je“ povezuje tablice osoba i djelatnik

Osoba <-> catering_naručitelj

Veza „je“ povezuje tablice osoba i catering_naručitelj.

Djelatnik <-> zanimanje

Veza „ima“ povezuje tablice djelatnik i zanimanje.

Djelatnik <-> djelatnici_catering

Veza „ima“ povezuje tablice djelatnik i djelatnici_catering.

Djelatnik <-> djelatnik_smjena

Veza „ima“ povezuje tablice djelatnik i djelatnik_smjena.

Djelatnik_smjena <-> smjena

Veza „sadrži“ povezuje tablice djelatnik_smjena i smjena.

Catering_naručitelj <-> catering_zahtjev

Veza „pravi“ povezuje tablice catering_naručitelj i catering_zahtjev.

Catering_zahtjev <-> catering

Veza „sadrži“ povezuje tablice catering_stavka i catering.

Catering <-> djelatnici_catering

Veza „ima“ povezuje tablice catering i djelatnici_catering.

Catering <-> catering_stavka

Veza „sadrži“ povezuje tablice catering i catering_stavka.

Osoba <-> dostava_stavka

Veza „sadrži“ povezuje tablice osoba i dostava_stavka.

Režije <-> kategorija_režije

Veza „ima“ povezuje tablice režije i kategorija_režije.

TABLICE

TABLICA OSOBA

Sastoji se od atributa id, ime, prezime, broj_mob i email. Atribut id je primarni ključ tipa integer. Atributi ime, prezime, broj_mob i email su tipa varchar sa ograničenjem znakova (50, 10 i 30)

```
CREATE TABLE osoba (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    ime VARCHAR(50) NOT NULL,
    prezime VARCHAR(50) NOT NULL,
    broj_mob VARCHAR(10),
    email VARCHAR(30)
);
```

TABLICA ZANIMANJE

Sastoji se od atributa id, naziv i placa_hrk. Atribut id je primarni ključ tipa integer. Atribut naziv je tipa varchar sa ograničenjem znakova 50. Dok je atribut placa_hrk tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Check postavlja zahtjev unosa.

```
CREATE TABLE zanimanje (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    naziv VARCHAR(50) NOT NULL,
    placa_hrk DECIMAL(10, 2) DEFAULT 3750.00,
    CHECK (placa_hrk >= 3750.00)
);
```

TABLICA DJELATNIK

Sastoji se od atributa id, id_osoba, oib, datum_rođenja, datum_zaposlenja, id_zanimanje i zaposlen. Primarni ključ tipa integer je atribut id. Id_osoba je strani ključ tipa integer. Atribut oib je tipa char koji se sastoji od 11 znakova. Atribut datum_rođenja je tipa date. Atribut datum_zaposlenja je tipa datetime jer trebami zapisati datum i vrijeme. Id_zanimanje je strani ključ tipa integer. Atribut zaposlen je tipa char sa ograničenjem znakova 1. Check postavlja zahtjev unosa.

```
CREATE TABLE djelatnik (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   id_osoba INTEGER NOT NULL,
   oib CHAR(11) NOT NULL,
   datum_rodenja DATE NOT NULL,
   datum_zaposlenja DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
   id_zanimanje INTEGER NOT NULL,
   zaposlen CHAR(1) NOT NULL DEFAULT "D",
   CHECK (zaposlen IN("D", "N")),
   FOREIGN KEY (id_osoba) REFERENCES osoba (id),
   FOREIGN KEY (id_zanimanje) REFERENCES zanimanje (id),
   CONSTRAINT id_osoba_unique UNIQUE (id_osoba),
   CONSTRAINT oib_unique UNIQUE (oib)
);
```

TABLICA ADRESA

Sastoji se od atributa id, drzava, grad, ulica i post_broj. Atribut id je primarni ključ tipa integer. Ostali atributi (drzava, grad, ulica, post_broj) su tipa varchar sa ograničenjem znakova (50 i 10).

```
CREATE TABLE adresa (
        id INTEGER PRIMARY KEY AUTO_INCREMENT,
        drzava VARCHAR (50) NOT NULL,
        grad VARCHAR (50) NOT NULL,
        ulica VARCHAR (50) NOT NULL,
        post_broj VARCHAR(10) NOT NULL
);
```

TABLICA DOBAVLJAČ

Sastoji se od atributa id, naziv, id_adresa, oib, broj_mob i vrsta_usluge. Atribut id je primarni ključ tipa integer. Atribut naziv je tipa varchar sa ograničenjem znakova 50. Id_adresa je strani ključ tipa integer. Atribut oib je tipa char koji se sastoji od 11 znakova. Atribut broj_mob je tipa varchar sa ograničenjem znakova 50 i atribut vrsta_usluge je tipa varchar sa ograničenjem znakova 50.

```
CREATE TABLE dobavljac (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   naziv VARCHAR(50) NOT NULL,
   id_adresa INTEGER NOT NULL,
   oib CHAR(11) NOT NULL UNIQUE,
   broj_mob VARCHAR(10) NOT NULL UNIQUE,
   vrsta_usluge VARCHAR(50) NOT NULL,
   FOREIGN KEY (id_adresa) REFERENCES adresa (id)
);
```

TABLICA STOL

Sastoji se od atributa id, broj_stola, rajon_stola i broj_gostiju_kapacitet. Atribut id je primarni ključ tipa integer, dok su atributi broj_stola i rajon_stola tipa varchar sa ograničenjem znakova 4. Atribut broj_gostiju_kapacitet je tipa integer.

```
CREATE TABLE stol (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   broj_stola VARCHAR(4) NOT NULL,
   rajon_stola VARCHAR(4) NOT NULL,
   broj_gostiju_kapacitet INTEGER
);
```

TABLICA NAČIN PLAĆANJA

Sastoji se od atributa id i naziv. Atribut id je primarni ključ tipa integer. Atribut naziv je tipa varchar sa ograničenjem znakova 10.

```
CREATE TABLE nacini_placanja (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   naziv VARCHAR(10) NOT NULL
);
```

TABLICA RAČUN

Sastoji se od atributa id, sifra, id_nacin_placanja, id_stol, id_djelatnik, vrijeme_izdavanja i iznos_hrk. Atribut id je primarni ključ tipa integer. Atribut sifra je tipa varchar sa ograničenjem znakova 10. Atributi id_nacin_placanja, id_djelatnik i id_stol su strani ključevi tipa integer. Atribut vrijeme_izdavanja je tipa datetime jer moramo zapisati datum i vrijeme. Atribut iznos_hrk je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane).

```
CREATE TABLE racun (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   sifra VARCHAR(10) NOT NULL,
   id_nacin_placanja INTEGER NOT NULL,
   id_stol INTEGER NOT NULL,
   id_djelatnik INTEGER NOT NULL,
   vrijeme_izdavanja DATETIME NOT NULL DEFAULT NOW(),
   iznos_hrk DECIMAL(10, 2) DEFAULT 0.00,
   FOREIGN KEY (id_nacin_placanja) REFERENCES nacini_placanja (id),
   FOREIGN KEY (id_stol) REFERENCES stol (id),
   FOREIGN KEY (id_djelatnik) REFERENCES djelatnik (id)
);
```

TABLICA ALERGEN

Sastoji se od atributa id i naziv. Atribut id je primarni ključ tipa integer. Atribut naziv je tipa varchar sa ograničenjem znakova 50.

```
CREATE TABLE alergen (
        id INTEGER PRIMARY KEY AUTO_INCREMENT,
        naziv VARCHAR(50) NOT NULL UNIQUE
);
```

TABLICA MENI

Sastoji se od atributa id, naziv_stavke, cijena_hrk i aktivno. Atribut id je primarni ključ tipa integer. Atribut naziv_stavke je tipa varchar sa ograničenjem znakova 70. Atribut cijena_hrk je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Atribut aktivno je tipa char što znači da se sastoji od 1 znaka. Check postavlja zahtjev unosa.

```
CREATE TABLE meni (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    naziv_stavke VARCHAR(70) NOT NULL,
    cijena_hrk DECIMAL(10, 2) NOT NULL,
    aktivno CHAR(1) DEFAULT "D",
    CHECK (aktivno IN ("D", "N"))
);
```

TABLICA SADRŽI ALERGEN

Sastoji se od atributa id, id_meni, id_alergen. Atribut id je primarni ključ tipa integer. Atributi id_meni i id_alergen su strani ključevi tipa integer.

```
CREATE TABLE sadrzi_alergen (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_meni INTEGER NOT NULL,
    id_alergen INTEGER NOT NULL,
    FOREIGN KEY (id_meni) REFERENCES meni (id),
    FOREIGN KEY (id_alergen) REFERENCES alergen (id)
);
```

TABLICA KATEGORIJA NAMIRNICA

Sastoji se od atributa id i naziv. Atribut je primarni ključ tipa integer. Atribut naziv je tipa varchar sa ograničenjem znakova 50.

```
CREATE TABLE kategorija_namirnica (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    naziv VARCHAR(50) NOT NULL
);
```

TABLICA NAMIRNICA

Sastoji se od atributa id, naziv, id_kategorija, kolicina_na_zalihi i mjerna_jedinica. Atribut id je primarni ključ tipa integer. Atribut naziv je tipa varchar sa ograničenjem znakova 50. Atribut id_količina je strani ključ tipa integer. Atribut kolicina_na_zalihi je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Atribut mjerna_jedinica je tipa varchar sa ograničenjem znakova 20. Check postavlja zahtjev unosa.

```
CREATE TABLE namirnica (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    naziv VARCHAR(50) NOT NULL UNIQUE,
    id_kategorija INTEGER NOT NULL,
    kolicina_na_zalihi DECIMAL (10, 2) NOT NULL,
    mjerna_jedinica VARCHAR(20) NOT NULL,
    FOREIGN KEY (id_kategorija) REFERENCES kategorija_namirnica (id),
    CHECK (kolicina_na_zalihi >= 0)
);
```

TABLICA STAVKA MENI

Sastoji se od atributa id, id_namirnica, kolicina, id_meni. Atribut id je primarni ključ tipa integer. Atributi id_namirnica i id_meni su strani ključevi tipa integer. Atribut kolicina je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Check postavlja zahtjev unosa. Unique zahtjeva da se unosu u atributu ne ponavljaju.

```
CREATE TABLE stavka_meni (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_namirnica INTEGER NOT NULL,
    kolicina DECIMAL (10, 2),
    id_meni INTEGER NOT NULL,
    FOREIGN KEY (id_namirnica) REFERENCES namirnica (id),
    FOREIGN KEY (id_meni) REFERENCES meni (id),
    UNIQUE (id_namirnica, id_meni),
    CHECK (kolicina > 0)
);
```

TABLICA STAVKA RAČUN

Sastoji se od atributa id, id_racun, id_meni, kolicina, cijena_hrk. Atribut id je primarni ključ tipa integer. Atributi id_racun i id_meni su strani ključevi tipa integer. Atribut kolicina je tipa integer. Atribut cijena_hrk je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane).

```
CREATE TABLE stavka_racun (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_racun INTEGER NOT NULL,
    id_meni INTEGER NOT NULL,
    kolicina INTEGER NOT NULL,
    cijena_hrk DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (id_racun) REFERENCES racun (id),
    FOREIGN KEY (id_meni) REFERENCES meni (id),
    UNIQUE (id_racun, id_meni)
);
```

TABLICA REZERVACIJA

Sastoji se od atributa id, id_stol, id_osoba, zeljeni_datum, vrijeme_od, vrijeme_do, broj_gostiju. Atribut id je primarni ključ tipa integer. Atributi id_stol i id_osoba su strani ključevi tipa integer. Atribut zeljeni_datum je tipa date. Atributi vrijeme_od i vrijeme_do su tipa time. Atribut broj_gostiju je tipa integer.

```
CREATE TABLE rezervacija (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_stol INTEGER NOT NULL,
    id_osoba INTEGER NOT NULL,
    zeljeni_datum DATE NOT NULL,
    vrijeme_od TIME NOT NULL,
    vrijeme_do TIME NOT NULL,
    broj_gostiju INTEGER NOT NULL,
    FOREIGN KEY (id_stol) REFERENCES stol (id),
    FOREIGN KEY (id_osoba) REFERENCES osoba (id)
);
```

TABLICA CATERING NARUČITELJ

Sastoji se od atributa id, id_osoba i oib. Atribut id je primarni ključ tipa integer. Atribut id_osoba je strani ključ tipa integer. Atribut oib je tipa char što znači da se sastoji od 11 znakova.

```
CREATE TABLE catering_narucitelj (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_osoba INTEGER NOT NULL,
    oib CHAR(11) NOT NULL UNIQUE,
    FOREIGN KEY (id_osoba) REFERENCES osoba (id)
);
```

TABLICA CATERING ZAHTJEV

Sastoji se od atributa id, id_narucitelj, id_adresa, zeljeni_datum i datum_zahtjeva. Atribut id je primarni ključ tipa integer. Atributi id_narucitelj i id_adresa su strani ključevi tipa integer. Atribut zeljeni datum je tipa date.

```
CREATE TABLE catering_zahtjev (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_narucitelj INTEGER NOT NULL,
    id_adresa INTEGER NOT NULL,
    opis TEXT,
    zeljeni_datum DATE NOT NULL,
    datum_zahtjeva TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_narucitelj) REFERENCES catering_narucitelj (id),
    FOREIGN KEY (id_adresa) REFERENCES adresa (id)
);
```

TABLICA CATERING

Sastoji se od atributa id, id_zahtjev, cijena_hrk, datum_izvrsenja i uplaceno. Atribut id je primarni ključ tipa integer. Atribut id_zahtjev je strani ključ tipa integer. Atribut cijena_hrk je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Atribut datum_izvrsenja je tipa timestamp (sadrži datum i vrijeme). Atribut uplaceno je tipa char koja se sastoji od 1 znaka. Check postavlja zahtjev unosa.

```
CREATE TABLE catering (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_zahtjev INTEGER NOT NULL,
    cijena_hrk DECIMAL(10, 2) DEFAULT 0.00,
    datum_izvrsenja TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    uplaceno CHAR(1) NOT NULL,
    FOREIGN KEY (id_zahtjev) REFERENCES catering_zahtjev (id),
    CHECK (uplaceno IN ("D", "N"))
);
```

TABLICA CATERING STAVKA

Sastoji se od atributa id, id_catering, id_meni, kolicina i cijena_hrk. Atribut id je primarni ključ tipa integer. Atributi id_catering i id_meni su strani ključevi tipa integer. Atribut kolicina je tipa integer. Atribut cijena_hrk je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Check postavlja zahtjev unosa.

```
CREATE TABLE catering_stavka (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_catering INTEGER NOT NULL,
    id_meni INTEGER NOT NULL,
    kolicina INTEGER NOT NULL,
    cijena_hrk DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (id_catering) REFERENCES catering (id),
    FOREIGN KEY (id_meni) REFERENCES meni (id),
    CHECK (kolicina > 0)
);
```

TABLICA DJELATNICI CATERING

Sastoji se od atributa id, id_catering i id_djelatnik. Atribut id je primarni ključ tipa integer. Atributi id_catering i id_djelatnik su strani ključevi tipa integer. Unique zahtjeva da se unosi u atributu ne ponavljaju.

```
CREATE TABLE djelatnici_catering (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_catering INTEGER NOT NULL,
    id_djelatnik INTEGER NOT NULL,
    UNIQUE(id_catering, id_djelatnik),
    FOREIGN KEY (id_catering) REFERENCES catering (id),
    FOREIGN KEY (id_djelatnik) REFERENCES djelatnik (id)
);
```

TABLICA NABAVA

Sastoji se od atributa id, id_dobavljac, iznos_hrk, podmireno i datum. Atribut id je primarni ključ tipa integer. Atribut id_dobavljac je strani ključ tipa integer. Atribut iznos_hrk je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Atribut podmireno je tipa char što znači da se sastoji od 1 znamenke. Atribut datum je tipa date. Check postavlja zahtjev kod unosa.

```
CREATE TABLE nabava (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   id_dobavljac INTEGER NOT NULL,
   opis TEXT,
   iznos_hrk DECIMAL(10, 2) DEFAULT 0.00,
   podmireno CHAR(1) NOT NULL,
   datum DATE NOT NULL,
   FOREIGN KEY (id_dobavljac) REFERENCES dobavljac (id),
   CHECK (podmireno IN ("D", "N"))
);
```

TABLICA NABAVA STAVKA

Sastoji se od atributa id, id_nabava, id_namirnica, kolicina i cijena_hrk. Atribut id je primarni ključ tipa integer. Atributi id_nabava i id_namirnica su vanjski ključevi tipa integer. Atribut kolicina je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Atribut cijena_hrk je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Check postavlja zahtjev kod unosa.

```
CREATE TABLE nabava_stavka (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   id_nabava INTEGER NOT NULL,
   id_namirnica INTEGER NOT NULL,
   kolicina DECIMAL(10, 2) NOT NULL,
   cijena_hrk DECIMAL(10, 2) NOT NULL,
   FOREIGN KEY (id_nabava) REFERENCES nabava (id),
   FOREIGN KEY (id_namirnica) REFERENCES namirnica (id),
   CHECK (kolicina > 0)
);
```

TABLICA OTPIS

Sastoji se od atributa id i datum. Atribut id je primarni ključ tipa integer. Atribut datum je tipa timestamp (sadrži datum i vrijeme).

```
CREATE TABLE otpis (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   datum TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   opis TEXT
);
```

TABLICA OTPIS STAVKA

Sastoji se od atributa id, id_otpis, id_namirnica i kolicina. Atribut id je primarni ključ tipa integer. Atributi id_otpis i id_namirnica su vanjski ključevi tipa integer. Atribut kolicina je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Check postavlja zahtjev kod unosa.

```
CREATE TABLE otpis_stavka (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   id_otpis INTEGER NOT NULL,
   id_namirnica INTEGER NOT NULL,
   kolicina DECIMAL(10, 2) NOT NULL,
   FOREIGN KEY (id_otpis) REFERENCES otpis (id),
   FOREIGN KEY (id_namirnica) REFERENCES namirnica (id),
   CHECK (kolicina > 0)
);
```

TABLICA KATEGORIJA REŽIJE

Sastoji se od atributa id i naziv. Atribut id je primarni ključ tipa integer. Atribut naziv je tipa varchar sa ograničenjem znakova 50.

```
CREATE TABLE kategorija_rezije (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   naziv VARCHAR(50) NOT NULL UNIQUE
);
```

TABLICA REŽIJE

Sastoji se od atributa id, iznos_hrk, datum, id_kategorija i placeno. Atribut id je primarni ključ tipa integer. Atribut iznos_hrk je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Atribut datum je tipa date. Atribut id_kategorija je strani ključ tipa integer. Atribut placeno je tipa char koji se sastoji od 1 znaka. Check postavlja zahtjev kod unosa.

```
CREATE TABLE rezije (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   iznos_hrk DECIMAL(10, 2) NOT NULL,
   datum DATE NOT NULL,
   id_kategorija INTEGER NOT NULL,
   placeno CHAR(1) NOT NULL,
   FOREIGN KEY (id_kategorija) REFERENCES kategorija_rezije (id),
   CHECK (placeno IN ("D", "N"))
);
```

TABLICA SMJENA

Sastoji se od atributa id, naziv, pocetak_radnog_vremena i kraj_radnog_vremena. Atribut id je primarni ključ tipa integer. Atribut naziv je tipa varchar sa ograničenjem unakova 50. Atributi pocetak_radnog_vremena i kraj_radnog vremena su tipa time.

```
CREATE TABLE smjena (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   naziv VARCHAR (50) NOT NULL,
   pocetak_radnog_vremena TIME NOT NULL,
   kraj_radnog_vremena TIME NOT NULL
);
```

TABLICA DJELATNIK SMJENA

Sastoji se od atributa id, id_djelatnik, id_smjena i datum. Atribut id je primarni ključ tipa integer. Atributi id_djelatnik i id_smjena su strani ključevi tipa integer. Atribut datum je tipa date.

```
CREATE TABLE djelatnik_smjena (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   id_djelatnik INTEGER NOT NULL,
   id_smjena INTEGER NOT NULL,
   datum DATE NOT NULL,
   FOREIGN KEY (id_djelatnik) REFERENCES djelatnik (id),
   FOREIGN KEY (id_smjena) REFERENCES smjena (id)
);
```

TABLICA DOSTAVA

Sastoji se od atributa id, id_osoba, id_adresa, datum, cijena_hrk i izvrsena. Atribut id je primarni ključ tipa integer. Atributi id_osoba i id_adresa su strani ključevi tipa integer. Atribut datum je tipa date. Atribut cijena_hrk je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Atribut izvrsena je tipa char što znači da se sastoji od 1 znamenke. Check postavlja zahtjev kod unosa.

```
CREATE TABLE dostava (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   id_osoba INTEGER NOT NULL,
   id_adresa INTEGER NOT NULL,
   datum DATE NOT NULL,
   cijena_hrk DECIMAL(10, 2) DEFAULT 0.00,
   izvrsena CHAR(1) NOT NULL,
   FOREIGN KEY (id_osoba) REFERENCES osoba (id),
   FOREIGN KEY (id_adresa) REFERENCES adresa (id),
   CHECK (izvrsena IN ("D", "N"))
);
```

TABLICA DOSTAVA STAVKA

Sastoji se od atributa id, id_dostava, id_meni, kolicina i cijena_hrk. Atribut id je primarni ključ tipa integer. Atributi id_dostava i id_meni su strani ključevi tipa integer. Atribut kolicina je tipa integer. Atribut cijena_hrk je tipa decimal 10,2 što znači da je ograničen sa 10 znakova (8 sa lijeve i 2 sa desne strane). Check postavlja zahtjev kod unosa.

```
CREATE TABLE dostava_stavka (
   id INTEGER PRIMARY KEY AUTO_INCREMENT,
   id_dostava INTEGER NOT NULL,
   id_meni INTEGER NOT NULL,
   kolicina INTEGER NOT NULL,
   cijena_hrk DECIMAL(10, 2) NOT NULL,
   FOREIGN KEY (id_dostava) REFERENCES dostava (id),
   FOREIGN KEY (id_meni) REFERENCES meni (id),
   CHECK (kolicina > 0)
);
```

TRIGGERI

1.TRIGGER
Ovaj trigger provjerava da li je količina namirnica na zalihi dostatna, izračunava iznos stavke računa i ukupni iznos računa te smanjuje količinu namirnica na zalihi.

```
DELIMITER //
CREATE TRIGGER bi_stavka_racun
   BEFORE INSERT ON stavka_racun
   FOR EACH ROW
BEGIN
   DECLARE l_cijena_stavke DECIMAL (10, 2);

   CALL provjeri_kolicinu_na_zalihi(new.id_meni, new.kolicina);

   SELECT cijena_hrk INTO l_cijena_stavke
                FROM meni
      WHERE meni.id = new.id_meni;

         SET new.cijena_hrk = l_cijena_stavke * new.kolicina;

   UPDATE racun
                SET iznos_hrk = iznos_hrk + new.cijena_hrk
                WHERE id = new.id_racun;

         CALL smanji_kolicinu_na_zalihi(new.id_meni, new.kolicina);
END//
DELIMITER ;
```

2.TRIGGER
Sljedeći trigger ptovjerava da li je količina namirnica na zalihi dostatna, izračunava iznos stavke catering i ukupni iznos cateringa te smanjuje količinu namirnica na zalihi.

```
DELIMITER //
CREATE TRIGGER bi_catering_stavka
   BEFORE INSERT ON catering_stavka
   FOR EACH ROW
BEGIN
         DECLARE l_cijena_stavke DECIMAL (10, 2);

   CALL provjeri_kolicinu_na_zalihi(new.id_meni, new.kolicina);

   SELECT cijena_hrk INTO l_cijena_stavke
                FROM meni
      WHERE meni.id = new.id_meni;

         SET new.cijena_hrk = l_cijena_stavke * new.kolicina;
```

```
   UPDATE catering
                SET cijena_hrk = cijena_hrk + new.cijena_hrk
                WHERE id = new.id_catering;

   CALL smanji_kolicinu_na_zalihi(new.id_meni, new.kolicina);
END//
DELIMITER ;
```

3.TRIGGER
Ovaj trigger izračunava ukupni iznos nabave i povećava količinu namirnica na zalihi.

```
DELIMITER //
CREATE TRIGGER bi_nabava_stavka
   BEFORE INSERT ON nabava_stavka
   FOR EACH ROW
BEGIN
   UPDATE nabava
                SET iznos_hrk = iznos_hrk + new.cijena_hrk
                WHERE id = new.id_nabava;

   UPDATE namirnica
                SET kolicina_na_zalihi = kolicina_na_zalihi + new.kolicina
      WHERE namirnica.id = new.id_namirnica;
END//
DELIMITER ;
```

4.TRIGGER
Provjerava da li je količina namirnica na zalihi dostatna, izračunava iznos stavke dostave i ukupni iznos dostave te smanjuje količinu namirnica na zalihi.

```
DELIMITER //
CREATE TRIGGER bi_dostava_stavka
   BEFORE INSERT ON dostava_stavka
   FOR EACH ROW
BEGIN
        DECLARE l_cijena_stavke DECIMAL (10, 2);

   CALL provjeri_kolicinu_na_zalihi(new.id_meni, new.kolicina);

   SELECT cijena_hrk INTO l_cijena_stavke
                FROM meni
      WHERE meni.id = new.id_meni;
```

```
        SET new.cijena_hrk = l_cijena_stavke * new.kolicina;

    UPDATE dostava
                SET cijena_hrk = cijena_hrk + new.cijena_hrk
                WHERE id = new.id_dostava;

        CALL smanji_kolicinu_na_zalihi(new.id_meni, new.kolicina);
END//
DELIMITER ;
```

5.TRIGGER
Provjerava da li je količina namirnica na zalihi dostatna i smanjuje količinu na zalihi.

```
DELIMITER //
CREATE TRIGGER bi_otpis_stavka
        BEFORE INSERT ON otpis_stavka
    FOR EACH ROW
BEGIN

DECLARE l_kolicina_na_zalihi DECIMAL(10, 2);

SELECT kolicina_na_zalihi INTO l_kolicina_na_zalihi
        FROM namirnica
    WHERE id = new.id_namirnica;

        IF l_kolicina_na_zalihi - new.kolicina < 0 THEN
                SIGNAL SQLSTATE '45000'
       SET MESSAGE_TEXT = "Količina na zalihi preniska!";
        ELSE
                UPDATE namirnica
                SET kolicina_na_zalihi = kolicina_na_zalihi - new.kolicina
                        WHERE new.id_namirnica = namirnica.id;
    END IF;

END//
DELIMITER ;
```

FUNKCIJE

1. Sljedeća funkcija kreira šifru računa. Ova funkcija prima parametre id_racuna tipa integer i na temelju toga vraća CHAR(6) tip podataka o tome koji je id predan.

```
DELIMITER //
CREATE FUNCTION kreiraj_sifru_racuna (id_racuna INTEGER) RETURNS CHAR(6)
DETERMINISTIC
BEGIN

        IF (id_racuna BETWEEN 1 AND 9) THEN
                RETURN CONCAT("00000", id_racuna);
        ELSEIF (id_racuna BETWEEN 10 AND 99) THEN
                RETURN CONCAT("0000", id_racuna);
        ELSEIF (id_racuna BETWEEN 100 AND 999) THEN
                RETURN CONCAT("000", id_racuna);
        ELSEIF (id_racuna BETWEEN 1000 AND 9999) THEN
                RETURN CONCAT("00", id_racuna);
        ELSEIF (id_racuna BETWEEN 10000 AND 99999) THEN
                RETURN CONCAT("0", id_racuna);
        ELSEIF (id_racuna BETWEEN 100000 AND 999999) THEN
                RETURN CONVERT(id_racuna, CHAR);
   END IF;

END //
DELIMITER ;
```

2. Funkcija koja kreira šifru računa u slučaju da koristimo autoincrement (ne prima id kao parametar). Vraća CHAR(6) tip podatka.

```
DELIMITER //
CREATE FUNCTION kreiraj_sifru_racuna_autoincrement () RETURNS CHAR(6)
DETERMINISTIC
BEGIN

        DECLARE novi_id INTEGER DEFAULT NULL;

   SELECT (id + 1) INTO novi_id
                FROM racun
                ORDER BY id DESC
                LIMIT 1;

        IF novi_id IS NULL THEN
                SET novi_id = 1;
        END IF;
```

```
        RETURN kreiraj_sifru_racuna(novi_id);

END //
DELIMITER ;
```

3. Funkcija koja za uneseni datum i vrijeme vraća podatak o tome da li je određeni stol slobodan (nema rezervacije). Prima parametre p_id_stol tipa integer, p_datum tipa date, p_vrijeme_od i p_vrijeme_do tipa time. Vraća CHAR(2) tip podataka.

```
DELIMITER //
CREATE FUNCTION stol_dostupan (p_id_stol INTEGER, p_datum DATE, p_vrijeme_od
TIME, p_vrijeme_do TIME) RETURNS CHAR(2)
DETERMINISTIC
BEGIN

        IF (SELECT COUNT(*)
                        FROM rezervacija
                        WHERE zeljeni_datum = p_datum
                                AND (p_vrijeme_od BETWEEN vrijeme_od AND vrijeme_do
                                OR p_vrijeme_do BETWEEN vrijeme_od AND vrijeme_do)
                                AND id_stol = p_id_stol) > 0
        THEN
                RETURN "NE";
        ELSE
                RETURN "DA";
        END IF;

END //
DELIMITER ;
```

4. Funkcija koja vraća da li je kapacitet nekog stola dovoljan za određen broj osoba. Prima parametre p_id_stol i p_broj_gostiju tipa integer. Vraća CHAR(2) tip podataka.

```
DELIMITER //
CREATE FUNCTION kapacitet_stola_dovoljan (p_id_stol INTEGER, p_broj_gostiju
INTEGER) RETURNS CHAR(2)
DETERMINISTIC
BEGIN

IF (SELECT COUNT(*)
                FROM stol
                WHERE p_broj_gostiju <= broj_gostiju_kapacitet
                AND id = p_id_stol) = 1
        THEN
                RETURN "DA";
```

```
        ELSE
                RETURN "NE";
        END IF;

END //
DELIMITER ;
```

UPITI

1.UPIT
Upit koji prikazuje ukupnu zaradu po mjesecima (racun + dostava + catering).

```
SELECT mjesec, SUM(ukupno) AS ukupna_zarada
        FROM (
                (SELECT CONCAT(MONTH(vrijeme_izdavanja), "/",
YEAR(vrijeme_izdavanja)) AS mjesec, SUM(iznos_hrk) AS ukupno
                        FROM racun
                        GROUP BY mjesec)
                UNION ALL
                (SELECT CONCAT(MONTH(datum), "/", YEAR(datum)) AS mjesec,
SUM(cijena_hrk) AS ukupno
                        FROM dostava
                        GROUP BY mjesec)
                UNION ALL
                (SELECT CONCAT(MONTH(datum_izvrsenja), "/", YEAR(datum_izvrsenja))
AS mjesec, SUM(cijena_hrk) AS ukupno
                        FROM catering
                        GROUP BY mjesec)
        ) AS zarade
   GROUP BY mjesec
   ORDER BY STR_TO_DATE((CONCAT("01/", mjesec)),'%d/%m/%Y') DESC;
```

2.UPIT
Upit koji prikazuje koje namirnice se najviše koriste u jelima na trenutno aktivnom meniju (uzimajući u obzir količinu u kojoj se namirnice koriste).

```
SELECT namirnica.*
        FROM stavka_meni
   INNER JOIN namirnica
   ON namirnica.id = stavka_meni.id_namirnica
   INNER JOIN meni
   ON meni.id = stavka_meni.id_meni
   WHERE meni.aktivno = "D"
        GROUP BY id_namirnica
   ORDER BY SUM(kolicina) DESC
   LIMIT 3;
```

3.UPIT
Upit koji prikazuje sva jela i alergene koje sadrže (uz pomoć funkcije).

```
DELIMITER //
CREATE FUNCTION sadrzi_alergene (p_id_meni INTEGER) RETURNS VARCHAR(100)
DETERMINISTIC
BEGIN
	RETURN (SELECT GROUP_CONCAT(" ", a.naziv)
				FROM meni m
				LEFT JOIN sadrzi_alergen sa
				ON m.id = sa.id_meni
				LEFT JOIN alergen a
				ON a.id = sa.id_alergen
				WHERE m.id = p_id_meni);
END //
DELIMITER ;

SELECT naziv_stavke AS jelo, sadrzi_alergene(id) AS alergeni_u_jelu
	FROM meni;
```

4.UPIT
Upit koji prikazuje sve stavke u meniju izdane na računima u ukupnoj količini većoj od 30.

```
DELIMITER //
CREATE FUNCTION br_prodanih_veci_od (p_id_meni INTEGER, p_kolicina INTEGER)
RETURNS CHAR(2)
DETERMINISTIC
BEGIN
	IF p_kolicina < (SELECT SUM(kolicina)
					FROM stavka_racun
					WHERE id_meni = p_id_meni)
	THEN
		RETURN "DA";
	ELSE
		RETURN "NE";
	END IF;
END //
DELIMITER ;

SELECT *
	FROM meni
  HAVING br_prodanih_veci_od(id, 30) = "DA";
```

5.UPIT
Upit (funkcija) koji prikazuje broj izdanih računa između dva datuma.

```
DELIMITER //
CREATE FUNCTION br_racuna_izmedu (p_datum_od DATETIME, p_datum_do
DATETIME) RETURNS INTEGER
DETERMINISTIC
BEGIN
        RETURN (SELECT COUNT(*)
                                FROM racun
                                WHERE vrijeme_izdavanja
                                BETWEEN p_datum_od AND p_datum_do);
END //
DELIMITER ;

SELECT br_racuna_izmedu(STR_TO_DATE('01.01.2021.', '%d.%m.%Y.'),
STR_TO_DATE('01.05.2021.', '%d.%m.%Y.'));
```

6.UPIT
Upit koji prikazuje ime i prezime svakog zaposlenika, njihov email, posao koji obavljaju u restoranu i broj odrađenih sati u prosincu 2021. godine.

```
SELECT d.id, ime, prezime, email, z.naziv,
COALESCE(SUM(HOUR(SUBTIME(kraj_radnog_vremena, pocetak_radnog_vremena))), 0)
AS broj_odradenih_sati
        FROM djelatnik d
   LEFT JOIN osoba o
   ON d.id_osoba = o.id
   LEFT JOIN zanimanje z
   ON d.id_zanimanje = z.id
   LEFT JOIN djelatnik_smjena ds
   ON ds.id_djelatnik = d.id
   LEFT JOIN smjena s
   ON ds.id_smjena = s.id
   WHERE (YEAR(ds.datum) = 2021 AND MONTH(ds.datum) = 12) OR (ds.datum) IS NULL
   GROUP BY (d.id);
```

7.UPIT
Upit koji prikazuje iznos režija po kvartalu tijekom 2021. godine.

```
SELECT CONCAT(kvartali.kvartal, ". kvartal") AS kvartal, COALESCE(tmp.ukupno, 0) AS
ukupan_iznos
        FROM (SELECT 1 AS kvartal UNION SELECT 2 UNION SELECT 3 UNION
SELECT 4) AS kvartali
                LEFT JOIN (SELECT QUARTER(datum) AS kvartal, SUM(iznos_hrk)
AS ukupno
                                FROM rezije
                                WHERE YEAR(datum) = 2021
                                GROUP BY QUARTER(datum)) AS tmp
                ON kvartali.kvartal = tmp.kvartal;
```

8.UPIT
Upit koji prikazuje sve namirnice sa dodatnim stupcem u kojem je navedeno da li je ta namirnica barem jednom otpisana, te ako je, u kolikoj ukupnoj količini.

```
SELECT namirnica.naziv,
        (CASE WHEN otpis_stavka.id IS NOT NULL
                THEN "Postoji otpis za namirnicu"
                ELSE "Nema otpisa" END) AS otpis,
        COALESCE(SUM(otpis_stavka.kolicina), 0) AS otpisana_kolicina
    FROM namirnica
  LEFT JOIN otpis_stavka
  ON namirnica.id = otpis_stavka.id_namirnica
  GROUP BY namirnica.id
  ORDER BY otpis DESC;
```

9.UPIT
Upit koji prikazuje sve djelatnike koji su barem jednom bili gosti restorana (tj. ako su barem jednom napravili rezervaciju, naručili catering ili dostavu).

```
SELECT osoba.*
        FROM djelatnik
  INNER JOIN osoba
  ON djelatnik.id_osoba = osoba.id
  WHERE osoba.id IN (SELECT DISTINCT id_osoba
                                FROM rezervacija
                        UNION
                        SELECT DISTINCT id_osoba
                                FROM catering_narucitelj
                        UNION
                        SELECT DISTINCT id_osoba
                                FROM dostava);
```

10.UPIT
Upit koji prikazuje zaradu po stolu u 2021. godine, te broj računa i broj srednja zarada po kapacitetu i računu (ako je taj broj nizak, to bi mogla biti indikacija da pozicija zbog nekog razloga privlači manje grupe).

```
SELECT id_stol, zarada_stola, iznos_hrk,
ROUND(zarada_stola/broj_racuna/broj_gostiju_kapacitet,2) as
zarada_po_kapacitetu_i_računu
        FROM (SELECT id_stol, iznos_hrk, broj_gostiju_kapacitet, SUM(iznos_hrk) as
zarada_stola, count(iznos_hrk) as broj_racuna
                FROM racun
                LEFT JOIN stol on racun.id_stol=stol.id
                WHERE (YEAR(vrijeme_izdavanja) = 2021)
                GROUP BY (id_stol)) as tmp
                ORDER BY (zarada_stola) DESC;
```

11.UPIT
Upit koji prikazuje datum kada je zadnje prodana svaka stavka menija.

```
SELECT naziv_stavke, MAX(temp.vrijeme_izdavanja) as zadnji_datum_prodaje
        FROM(SELECT vrijeme_izdavanja, naziv_stavke
                        FROM racun
                        LEFT JOIN stavka_racun
        ON racun.id= stavka_racun.id_racun
                        LEFT JOIN meni
        ON stavka_racun.id_meni=meni.id) as temp
        group by temp.naziv_stavke
        order by zadnji_datum_prodaje DESC;
```

12.UPIT
Upit koji prikazuje prosječnu nabavnu cijenu za namirnice koje su se barem jednom nabavile.

```
SELECT namirnica.naziv,
                CONCAT(ROUND((SUM(cijena_hrk) / SUM(kolicina)), 2), " kn / ",
mjerna_jedinica) AS prosjecna_nabavna_cijena
        FROM nabava_stavka
    INNER JOIN namirnica
    ON namirnica.id = nabava_stavka.id_namirnica
    GROUP BY namirnica.id;
```

13.UPIT
Upit koji prikazuje kategorija namirnica s najvećom potrošnjom po kvartalu.

```
SELECT naziv as kategorija_namirnica, MAX(suma) as ukupna_potrosnja,
CONCAT(
(CASE
 WHEN MONTH(vrijeme_izdavanja) IN (1,2,3)
 THEN "Kvartal 1. "
 WHEN MONTH(vrijeme_izdavanja) IN (4,5,6)
 THEN "Kvartal 2. "
  WHEN MONTH(vrijeme_izdavanja) IN (7,8,9)
 THEN "Kvartal 3. "
  WHEN MONTH(vrijeme_izdavanja) IN (10,11,12)
 THEN "Kvartal 4. "
 END),
 YEAR(vrijeme_izdavanja)) AS Kvartal
FROM
(SELECT naziv, vrijeme_izdavanja, TOT, SUM(TOT) as suma
FROM
(SELECT *
FROM (SELECT stavka_meni.kolicina*stavka_racun.kolicina AS TOT, mjerna_jedinica,
kategorija_namirnica.naziv, vrijeme_izdavanja
        FROM stavka_meni
                JOIN stavka_racun
      ON stavka_meni.id_meni=stavka_racun.id_meni
                JOIN racun
      ON stavka_racun.id_racun=racun.id
                JOIN namirnica
      ON namirnica.id= stavka_meni.id_namirnica
                JOIN kategorija_namirnica
      ON kategorija_namirnica.id= namirnica.id_kategorija) as temp
      WHERE mjerna_jedinica NOT IN ("komad")
UNION
SELECT *
FROM (SELECT stavka_meni.kolicina*stavka_racun.kolicina*.5 AS TOT, mjerna_jedinica,
kategorija_namirnica.naziv, vrijeme_izdavanja
        FROM stavka_meni
                JOIN stavka_racun
      ON stavka_meni.id_meni=stavka_racun.id_meni
                JOIN racun
      ON stavka_racun.id_racun=racun.id
                JOIN namirnica
      ON namirnica.id= stavka_meni.id_namirnica
```

```
                JOIN kategorija_namirnica
        ON kategorija_namirnica.id= namirnica.id_kategorija) as temp
        WHERE mjerna_jedinica IN ("komad")) as temp
GROUP BY QUARTER (vrijeme_izdavanja), naziv) as temp
GROUP BY QUARTER (vrijeme_izdavanja);
```

14.UPIT

Upit koji prikazuje cijenu sastojka svakog jela da su svi nabavljeni na najgoru cijenu dosad, te računa marginu za taj najgori slučaj. Sastojci za koju nemamo upisanu nabavu su izabačeni, a jelo ako nemamo upisanu nabavu za nijedan sastojak.

```
SELECT naziv_stavke, cijena_hrk, SUM(najveca_cijena*kolicina) AS
najveca_cijena_sastojka, (cijena_hrk - najveca_cijena*kolicina) AS najmanja_margina,
(CASE WHEN (cijena_hrk - najveca_cijena*kolicina) >= 0
    THEN " "
    ELSE "Mogući gubitak!" END) AS napomena
FROM meni
JOIN stavka_meni
ON stavka_meni.id_meni=meni.id
LEFT JOIN
(SELECT id_namirnica,
                MAX(cijena_hrk/kolicina) AS najveca_cijena
        FROM nabava_stavka
    INNER JOIN namirnica
    ON namirnica.id = nabava_stavka.id_namirnica
    GROUP BY namirnica.id) as temp
ON temp.id_namirnica=stavka_meni.id_namirnica
WHERE najveca_cijena IS NOT NULL
GROUP BY naziv_stavke
ORDER BY najmanja_margina
;
```

15.UPIT
Upit koji prikazuje ukupne rashode po mjesecu (plaće+rezije+nabava).

```
SELECT mjesec, ukup+SUM(placa_hrk) as Ukupni_rashodi
FROM
(SELECT mjesec, SUM(ukupno) as ukup
        FROM (
                (SELECT CONCAT(MONTH(datum), "/", YEAR(datum)) AS mjesec,
SUM(iznos_hrk) AS ukupno
                        FROM rezije
                        GROUP BY mjesec)
                UNION ALL
                (SELECT CONCAT(MONTH(datum), "/", YEAR(datum)) AS mjesec,
SUM(iznos_hrk) AS ukupno
                        FROM nabava
                        GROUP BY mjesec)
        ) AS zarade
   GROUP BY mjesec
   ORDER BY STR_TO_DATE((CONCAT("01/", mjesec)),'%d/%m/%Y') DESC) as temp
   JOIN djelatnik
   JOIN zanimanje
   ON zanimanje.id=djelatnik.id_zanimanje
   WHERE zaposlen="D"
   GROUP BY mjesec;
   ;
```

16.UPIT
Upit koji računa srednju zaradu preko računa po danu podijeljena po satima.

```
SELECT ROUND(SUM(tot)/(SELECT COUNT(DISTINCT DATE(vrijeme_izdavanja)) FROM
racun), 2), sat
FROM
(SELECT SUM(iznos_hrk) as tot, HOUR(vrijeme_izdavanja) AS sat,
COUNT(DATE(vrijeme_izdavanja)) AS datum
FROM racun
GROUP BY sat, vrijeme_izdavanja) AS temp
GROUP BY sat
ORDER BY sat;
```

POGLEDI

1.POGLED
Pogled pod nazivom aktivni_ meni koji prikazuje trenutni meni tako da filtrira po atributu “aktivno”.

```
CREATE VIEW aktivni_meni AS
SELECT *
        FROM meni
   WHERE aktivno = "D";
```

2.POGLED
Pogled nadolazeci_caterinzi koji prikazuje sve cateringe u budućnosti (te za koje zahtjeve su vezani) zajedno sa brojem zaposlenika koji su zaduženi za taj catering.

```
CREATE VIEW nadolazeci_caterinzi AS
SELECT c.id catering_id,
                c.datum_izvrsenja,
      c.uplaceno,
      cz.id zahtjev_id,
      cz.zeljeni_datum,
      cz.datum_zahtjeva,
      COUNT(dc.id) AS broj_zaposlenika
         FROM catering c
         INNER JOIN djelatnici_catering dc
   ON c.id = dc.id_catering
   INNER JOIN catering_zahtjev cz
   ON c.id_zahtjev = cz.id
   WHERE datum_izvrsenja IS NULL
   GROUP BY c.id;
```

3.POGLED
Pogled trenutni_djelatnici koji prikazuje sve trenutne djelatnike po atributu “zaposlen”.

```
CREATE VIEW trenutni_djelatnici AS
        SELECT *
                FROM djelatnik
                WHERE zaposlen = "D";
```

4.POGLED
Pogled najveci_br_rezervacija koji prikazuje goste koji su napravili najveći broj rezervacija, te prikazuje da li je ta osoba ujedno i zaposlenik.

```
CREATE VIEW najveci_br_rezervacija AS
SELECT osoba.*, COUNT(*) AS broj_rezervacija,
                (CASE WHEN osoba.id IN (SELECT id_osoba FROM djelatnik)
                        THEN "Da"
        ELSE "Ne" END) AS osoba_je_djelatnik
        FROM rezervacija
   INNER JOIN osoba
   ON osoba.id = rezervacija.id_osoba
   GROUP BY osoba.id
   ORDER BY broj_rezervacija DESC
   LIMIT 5;
```

5.POGLED
Pogled najcesce_adrese_dostave koji prikazuje na koje adrese se najčešće vrši dostava.

```
CREATE VIEW najcesce_adrese_dostave AS
SELECT adresa.*
        FROM dostava
   INNER JOIN adresa ON adresa.id = dostava.id_adresa
   GROUP BY adresa.id
   ORDER BY COUNT(*) DESC
   LIMIT 3;
```

6.POGLED
Pogled prosjecan_iznos_racuna koji prikazuje prosječan iznos računa po mjesecu tijekom prošle godine.

```
CREATE VIEW prosjecan_iznos_racuna AS
SELECT CONCAT(MONTH(vrijeme_izdavanja), "/", YEAR(vrijeme_izdavanja)) AS mjesec,
                ROUND(AVG(iznos_hrk), 2) AS prosjecan_iznos
        FROM racun
   WHERE YEAR(vrijeme_izdavanja) = YEAR(CURRENT_TIMESTAMP) - 1
   GROUP BY MONTH(vrijeme_izdavanja);
```

7.POGLED
Pogled namirnice_za_nabavu koji prikazuje namirnice čija je količina na zalihi niska u odnosu na količinu u kojoj se koriste u jelima (tj. prikazuje namirnice koje bi trebalo nabaviti) - uzimaju se podaci jela izdanih na računima u periodu od zadnjih godinu dana

```
CREATE VIEW namirnice_za_nabavu AS
SELECT namirnica.naziv,
                namirnica.kolicina_na_zalihi,
      (kolicina * SUM(br_narucenih_jela)) AS utroseno_zadnjih_god_dana,
      (namirnica.kolicina_na_zalihi / (kolicina * SUM(br_narucenih_jela))) AS omjer
        FROM stavka_meni
   INNER JOIN (SELECT id_meni, SUM(kolicina) AS br_narucenih_jela
                                        FROM stavka_racun
                                        INNER JOIN racun ON racun.id =
stavka_racun.id_racun
                                        WHERE vrijeme_izdavanja >= DATE_SUB(NOW(),
INTERVAL 1 YEAR)
                                        GROUP BY id_meni) AS br_narucenih_jela
        ON stavka_meni.id_meni = br_narucenih_jela.id_meni
   INNER JOIN namirnica ON namirnica.id = id_namirnica
   GROUP BY id_namirnica
   ORDER BY omjer ASC
   LIMIT 10;
```

PROCEDURE

1.PROCEDURA
Procedura koja za određeno jelo prikazuje koje namirnice se koriste i u kolikoj količini za to jelo.

```
DELIMITER //
CREATE PROCEDURE sastojci (IN p_id_meni INTEGER, OUT status_jela VARCHAR(100))
BEGIN
        DROP TABLE IF EXISTS sastojci_jela;
   CREATE TEMPORARY TABLE sastojci_jela (
                naziv_jela VARCHAR(70),
      naziv_namirnice VARCHAR(50),
      kolicina DECIMAL (10, 2),
      mjerna_jedinica VARCHAR(20),
      kolicina_na_zalihi DECIMAL (10, 2),
      status_jela VARCHAR(100)
   );

   SET status_jela = "Jelo se nalazi na trenutnom meniju.";

   IF (SELECT COUNT(*)
                        FROM meni
                        WHERE id = p_id_meni AND aktivno = "N") > 0
        THEN
                SET status_jela = "Jelo se ne nalazi u trenutnom meniju!";
        END IF;

 INSERT INTO sastojci_jela
   SELECT meni.naziv_stavke, namirnica.naziv, stavka_meni.kolicina, mjerna_jedinica,
kolicina_na_zalihi, status_jela
                FROM meni
                INNER JOIN stavka_meni
                ON meni.id = stavka_meni.id_meni
                INNER JOIN namirnica
                ON stavka_meni.id_namirnica = namirnica.id
                WHERE meni.id = p_id_meni;
END //
DELIMITER ;
```

2.PROCEDURA
Procedura koja smanjuje količinu namirnica na zalihi za određeni meni

```
DELIMITER //
CREATE PROCEDURE smanji_kolicinu_na_zalihi (IN p_id_meni INTEGER, IN
p_kolicina_jela INTEGER)
BEGIN

DECLARE l_id_namirnica INTEGER;
DECLARE l_kolicina DECIMAL (10, 2);

DECLARE cur CURSOR FOR
        SELECT id_namirnica, kolicina
                FROM stavka_meni
                WHERE id_meni = p_id_meni;

DECLARE EXIT HANDLER FOR NOT FOUND BEGIN END;

OPEN cur;

smanji_kolicinu: LOOP
        FETCH cur INTO l_id_namirnica, l_kolicina;
   UPDATE namirnica
                SET kolicina_na_zalihi = kolicina_na_zalihi - l_kolicina * p_kolicina_jela
      WHERE namirnica.id = l_id_namirnica;
        END LOOP smanji_kolicinu;

CLOSE cur;

END //
DELIMITER ;
```

3.PROCEDURA
Procedura koja provjerava da li za određeno jelo postoji dovoljna količina namirnica na zalihi.

```
DELIMITER //
CREATE PROCEDURE provjeri_kolicinu_na_zalihi (IN p_id_meni INTEGER, IN
p_kolicina_jela INTEGER)
BEGIN

DECLARE l_kolicina_na_zalihi DECIMAL (10, 2);
DECLARE l_id_namirnica INTEGER;
DECLARE l_naziv_namirnice VARCHAR (50);
DECLARE l_kolicina DECIMAL (10, 2);
DECLARE error_kolicina VARCHAR(100) DEFAULT "Količina na zalihi preniska za
namirnicu: ";

DECLARE cur CURSOR FOR
        SELECT id_namirnica, kolicina
                FROM stavka_meni
                WHERE id_meni = p_id_meni;

DECLARE EXIT HANDLER FOR NOT FOUND BEGIN END;

OPEN cur;

provjeri_kolicinu: LOOP
        FETCH cur INTO l_id_namirnica, l_kolicina;
   SELECT kolicina_na_zalihi INTO l_kolicina_na_zalihi
                FROM namirnica
      WHERE id = l_id_namirnica;
   IF (l_kolicina_na_zalihi - l_kolicina * p_kolicina_jela) < 0 THEN
                SELECT naziv INTO l_naziv_namirnice
                        FROM namirnica
                        WHERE id = l_id_namirnica;
      SET error_kolicina = CONCAT(error_kolicina, l_naziv_namirnice);
      SIGNAL SQLSTATE "45000"
      SET MESSAGE_TEXT = error_kolicina;
   END IF;
END LOOP provjeri_kolicinu;

CLOSE cur;

END //
DELIMITER ;
```

4.PROCEDURA
Procedura koja "briše" jelo s menija -> postavlja atribut 'aktivno' na "N".

```
DELIMITER //
CREATE PROCEDURE obrisi_jelo (p_id_jela INTEGER)
BEGIN
        IF (SELECT COUNT(*)
                        FROM meni
        WHERE id = p_id_jela) = 0
        THEN
                SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Jelo sa tim id-em ne postoji u tablici meni!';
        ELSEIF (SELECT COUNT(*)
                        FROM meni
        WHERE id = p_id_jela
                                AND aktivno = "N") = 1
        THEN
                SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Jelo je već neaktivno!';
        ELSE
                UPDATE meni
                SET aktivno = "N"
      WHERE id = p_id_jela;
   END IF;
END //
DELIMITER ;
```

5.PROCEDURA
Procedura koja dodaje jelo na meni.

```
DELIMITER //
CREATE PROCEDURE dodaj_jelo (p_naziv_stavke VARCHAR(70), p_cijena_hrk
DECIMAL(10, 2))
BEGIN
        -- aktivno jelo s tim nazivom već postoji -> javlja grešku:
        IF (SELECT COUNT(*)
                        FROM meni
                        WHERE naziv_stavke = p_naziv_stavke
        AND aktivno = "D") > 0
        THEN
                SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Jelo već postoji!';
        -- neaktivno jelo s tim nazivom već postoji -> aktivira ga:
        ELSEIF (SELECT COUNT(*)
                        FROM meni
                        WHERE naziv_stavke = p_naziv_stavke
                                AND aktivno = "N") > 0
        THEN
                UPDATE meni
                        SET aktivno = "D"
        WHERE naziv_stavke = p_naziv_stavke;
        -- inače, dodaje jelo:
   ELSE
                INSERT INTO meni (naziv_stavke, cijena_hrk) VALUES (p_naziv_stavke,
p_cijena_hrk);
   END IF;
END //
DELIMITER ;
```

6.PROCEDURA
Procedura koja modificira tablicu stavka_meni za određeno jelo (uređuje popis namirnica za to jelo). Ako je p_kolicina NULL, briše stavku.

```
DELIMITER //
CREATE PROCEDURE uredi_stavka_meni
(p_naziv_jela VARCHAR(70),
p_naziv_namirnice VARCHAR(50),
p_kolicina DECIMAL(10, 2),
OUT status_procedure VARCHAR(100))
BEGIN

DECLARE l_id_meni INTEGER DEFAULT NULL;
DECLARE l_id_namirnica INTEGER DEFAULT NULL;

SELECT id INTO l_id_meni
        FROM meni
        WHERE naziv_stavke = p_naziv_jela;

SELECT id INTO l_id_namirnica
        FROM namirnica
        WHERE naziv = p_naziv_namirnice;

IF l_id_meni IS NULL THEN
        SET status_procedure = "Jelo tog naziva ne postoji!";
ELSEIF l_id_namirnica IS NULL THEN
        SET status_procedure = "Namirnica tog naziva ne postoji!";
ELSEIF p_kolicina IS NULL THEN
        DELETE FROM stavka_meni
                WHERE id_namirnica = l_id_namirnica AND id_meni = l_id_meni;
        SET status_procedure = "Stavka obrisana!";
ELSEIF p_kolicina <= 0 THEN
        SET status_procedure = "Količina mora biti pozitivan broj!";
ELSEIF (SELECT COUNT(*)
                FROM stavka_meni
                WHERE id_namirnica = l_id_namirnica
                        AND id_meni = l_id_meni) > 0
THEN
        SET status_procedure = "Već postoji unos za to jelo i namirnicu!";
ELSE
        INSERT INTO stavka_meni (id_namirnica, kolicina, id_meni) VALUES
                (l_id_namirnica, p_kolicina, l_id_meni);
        SET status_procedure = "Stavka dodana!";
END IF;
```

```
END //
DELIMITER ;
```

7.PROCEDURA
Procedura koja stvara novi zahtjev za catering

```
DELIMITER //
CREATE PROCEDURE stvori_catering_zahtjev
(IN p_id_narucitelj INTEGER,
IN p_id_adresa INTEGER,
IN p_opis TEXT,
IN p_zeljeni_datum DATE,
OUT status_zahtjeva VARCHAR(100))
BEGIN

	DECLARE l_id_narucitelj INTEGER DEFAULT NULL;
	DECLARE l_id_adresa INTEGER DEFAULT NULL;

	SELECT id INTO l_id_narucitelj
		FROM catering_narucitelj
		WHERE id = p_id_narucitelj;

	SELECT id INTO l_id_adresa
		FROM adresa
		WHERE id = p_id_adresa;

	IF (p_zeljeni_datum) < CURRENT_TIMESTAMP THEN
		SET status_zahtjeva = "Zahtjev odbijen; željeni datum ne može biti u
prošlosti!";
	ELSEIF l_id_narucitelj IS NULL THEN
		SET status_zahtjeva = "Zahtjev odbijen; naručitelj ne postoji u evidenciji!";
	ELSEIF l_id_adresa IS NULL THEN
		SET status_zahtjeva = "Zahtjev odbijen; adresa ne postoji u evidenciji!";
	ELSE
		INSERT INTO catering_zahtjev (id_narucitelj, id_adresa, opis, zeljeni_datum)
VALUES
			(l_id_narucitelj, l_id_adresa, p_opis, p_zeljeni_datum);
		SET status_zahtjeva = "Catering zahtjev stvoren.";
	END IF;

END //
DELIMITER ;
```

8. I 9. PROCEDURA
Procedure koja stvaraju novi otpis.

```
DELIMITER //
CREATE PROCEDURE dodaj_stavku_za_otpis (p_naziv_namirnice VARCHAR(50),
p_kolicina DECIMAL(10, 2))
BEGIN

        DECLARE l_id_namirnica INTEGER DEFAULT NULL;

        CREATE TEMPORARY TABLE IF NOT EXISTS tmp_otpis_stavka (
                id_namirnica INTEGER NOT NULL,
                kolicina DECIMAL(10, 2)
        );

        SELECT id INTO l_id_namirnica
                FROM namirnica
                WHERE naziv = p_naziv_namirnice;

        IF l_id_namirnica IS NULL THEN
                SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Namirnica sa tog naziva ne postoji!';
        ELSEIF p_kolicina <= 0 THEN
                SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Količina mora biti pozitivan broj!';
        ELSE
                INSERT INTO tmp_otpis_stavka VALUES (l_id_namirnica, p_kolicina);
        END IF;

END //
DELIMITER ;


DELIMITER //
CREATE PROCEDURE stvori_otpis ()
BEGIN

        DECLARE l_id_otpis INTEGER;
   DECLARE l_id_namirnica INTEGER;
   DECLARE l_kolicina DECIMAL(10, 2);

        DECLARE cur CURSOR FOR
                SELECT id_namirnica, kolicina
                        FROM tmp_otpis_stavka;

        DECLARE EXIT HANDLER FOR NOT FOUND
```

```
        BEGIN
                DELETE FROM tmp_otpis_stavka;
        END;

        -- stvara novi otpis
        INSERT INTO otpis VALUES ();

    -- dohvaća id zadnje unesenog otpisa
    SELECT id INTO l_id_otpis
                FROM otpis
                ORDER BY id DESC
                LIMIT 1;

    OPEN cur;

    unesi_stavke: LOOP
                FETCH cur INTO l_id_namirnica, l_kolicina;
                INSERT INTO otpis_stavka (id_otpis, id_namirnica, kolicina)
                        VALUES (l_id_otpis, l_id_namirnica, l_kolicina);
        END LOOP unesi_stavke;

        CLOSE cur;

END //
DELIMITER ;
```

10.PROCEDURA

Procedura koja za upisani vrijeme i datum prikazuje djelatnike koji su bili na poslu.

```
DELIMITER //
CREATE PROCEDURE prisustvo_radnika(IN p_datum DATE, p_sat TIME)
BEGIN
        DROP TABLE IF EXISTS prisutni_radnici;
    CREATE TEMPORARY TABLE prisutni_radnici(id_radnika INTEGER, ime VARCHAR(20),
prezime VARCHAR(20), zanimanje VARCHAR(30));
        INSERT INTO prisutni_radnici
        SELECT djelatnik.id, osoba.ime, osoba.prezime, zanimanje.naziv
                FROM djelatnik
                JOIN osoba
```

```
            ON djelatnik.id_osoba=osoba.id
            JOIN zanimanje
            ON djelatnik.id_zanimanje=zanimanje.id
            JOIN djelatnik_smjena
            ON djelatnik_smjena.id_djelatnik = djelatnik.id_osoba
    JOIN smjena
    ON djelatnik_smjena.id_smjena = smjena.id
    WHERE p_sat >= smjena.pocetak_radnog_vremena AND
    p_sat <= smjena.kraj_radnog_vremena AND
    djelatnik_smjena.datum = p_datum;
END //
DELIMITER ;
```

11.PROCEDURA

Procedura koja stornira zadani račun tako da stvori račun sa jednakim i negativnim stavkama.

```
DELIMITER //
CREATE PROCEDURE storno_racuna(IN p_id INTEGER, p_djelatnik INTEGER)
BEGIN
DECLARE l_id INTEGER;
DECLARE l_datum DATETIME;
DECLARE l_stol INTEGER;
DECLARE l_plac INTEGER;
DECLARE l_id2 INTEGER;
DECLARE size INTEGER;
DECLARE begginning INTEGER;
DECLARE meni INTEGER;
DECLARE l_kolicina INTEGER;



SET l_id= (SELECT MAX(id)+1 FROM racun);
SET l_datum = (SELECT vrijeme_izdavanja FROM racun WHERE id=p_id);
SET l_stol = (SELECT id_stol FROM racun WHERE id=p_id);
SET l_plac = (SELECT id_nacin_placanja FROM racun WHERE id=p_id);
SET l_id2= (SELECT MAX(id)+1 FROM stavka_racun);


INSERT INTO racun VALUES(l_id, kreiraj_sifru_racuna_autoincrement(), l_plac, l_stol,
p_djelatnik, l_datum, 0);
SELECT COUNT(id_meni) FROM stavka_racun WHERE id_racun=p_id INTO size;
```

```
SELECT MIN(id) FROM stavka_racun WHERE id_racun=p_id INTO begginning;

REPEAT
SELECT id_meni FROM (SELECT MIN(id), id_meni FROM stavka_racun WHERE
id_racun=p_id)AS temp INTO meni;
SELECT kolicina FROM stavka_racun WHERE begginning=id INTO l_kolicina;
INSERT INTO stavka_racun VALUES(l_id2,l_id, meni, -l_kolicina, 0); -- INSERT INTO
stavka_racun VALUES(l_id2,l_id, meni, kolicina, 0);
SET l_id2=l_id2+1;
SET begginning=begginning+1;
SET size=size-1;
UNTIL size<=0
END REPEAT;


END //
DELIMITER ;
```

12.PROCEDURA

Procedura koja za određeni catering u privremenu tablicu sprema sve zaposlenike koji su zaduženi za taj catering.

```
DELIMITER //
CREATE PROCEDURE prikazi_djelatnike_catering (p_id_catering INTEGER)
BEGIN

DROP TABLE IF EXISTS tmp_djelatnici_catering;
CREATE TEMPORARY TABLE tmp_djelatnici_catering (
        ime VARCHAR(50),
   prezime VARCHAR(50),
   broj_mob VARCHAR(10),
   email VARCHAR(30),
   oib CHAR(11),
   datum_zaposlenja DATE
);

INSERT INTO tmp_djelatnici_catering
        SELECT osoba.ime, osoba.prezime, osoba.broj_mob, osoba.email, djelatnik.oib,
djelatnik.datum_zaposlenja
                FROM djelatnici_catering
                INNER JOIN djelatnik ON djelatnik.id = id_djelatnik
                INNER JOIN osoba ON osoba.id = id_osoba
                WHERE id_catering = p_id_catering;

END //
```

```
DELIMITER ;
```

13.PROCEDURA
Procedura koja postavlja datum_izvrsenja za određeni catering na trenutni datum (ako kao parametar p_datum_izvrsenja primi NULL), a inače postavlja datum_izvrsenja na vrijednost tog parametra.

```
DELIMITER //
CREATE PROCEDURE postavi_datum_izvrsenja_catering (p_id_catering INTEGER,
p_datum_izvrsenja DATE, OUT p_status VARCHAR(100))
BEGIN

        DECLARE l_id_catering INTEGER DEFAULT NULL;
   DECLARE l_datum_izvrsenja DATE DEFAULT NULL;

        SELECT id, datum_izvrsenja INTO l_id_catering, l_datum_izvrsenja
                FROM catering
      WHERE id = p_id_catering;

        IF l_id_catering IS NULL THEN
                SET p_status = "Catering sa tim id-em ne postoji!";
        ELSEIF l_datum_izvrsenja IS NOT NULL THEN
                SET p_status = "Catering već ima datum izvršenja!";
        ELSEIF p_datum_izvrsenja IS NULL THEN
                UPDATE catering
                        SET datum_izvrsenja = CURRENT_TIMESTAMP
        WHERE id = l_id_catering;
                SET p_status = CONCAT("Postavljen datum izvršenja na današnji datum za
catering s id-em: ", l_id_catering);
   ELSEIF p_datum_izvrsenja > CURRENT_TIMESTAMP THEN
                SET p_status = "Datum izvršenja ne može biti u budućnosti!";
        ELSE
                UPDATE catering
                        SET datum_izvrsenja = p_datum_izvrsenja
        WHERE id = l_id_catering;
                SET p_status = CONCAT("Postavljen datum izvršenja na ",
p_datum_izvrsenja,  " za catering s id-em: ", l_id_catering);
   END IF;

END //
DELIMITER ;
```

14.PROCEDURA
Procedura za kreiranje rezervacije

```
DELIMITER //
CREATE PROCEDURE kreiraj_rezervaciju
(p_id_stol INTEGER,
p_id_osoba INTEGER,
p_zeljeni_datum DATE,
p_vrijeme_od TIME,
p_vrijeme_do TIME,
p_broj_gostiju INTEGER,
OUT status_rezervacije VARCHAR(100))
BEGIN

DECLARE l_id_stol INTEGER DEFAULT NULL;
DECLARE l_id_osoba INTEGER DEFAULT NULL;

SELECT id INTO l_id_osoba
        FROM osoba
   WHERE id = p_id_osoba;

SELECT id INTO l_id_stol
        FROM stol
   WHERE id = p_id_stol;

IF l_id_osoba IS NULL THEN
        SET status_rezervacije = "Osoba sa navedenim id-em ne postoji!";
ELSEIF l_id_stol IS NULL THEN
        SET status_rezervacije = "Stol sa navedenim id-em ne postoji!";
ELSEIF p_broj_gostiju <= 0 THEN
        SET status_rezervacije = "Broj gostiju mora biti pozitivan broj!";
ELSEIF p_zeljeni_datum < CURRENT_TIMESTAMP THEN
        SET status_rezervacije = "Željeni datum mora biti u budućnosti!";
ELSEIF p_vrijeme_od < "10:00" OR p_vrijeme_do > "23:00" THEN
        SET status_rezervacije = "Željeno vrijeme nije unutar radnog vremena restorana!";
ELSEIF kapacitet_stola_dovoljan(p_id_stol, p_broj_gostiju) = "NE" THEN
        SET status_rezervacije = "Broj gostiju prevelik za odabrani stol!";
ELSEIF (stol_dostupan (p_id_stol, p_zeljeni_datum, p_vrijeme_od, p_vrijeme_do)) = "NE"
THEN
        SET status_rezervacije = "Odabrani stol je zauzet u željenom vremenu!";
ELSE
        INSERT INTO rezervacija (id_stol, id_osoba, zeljeni_datum, vrijeme_od, vrijeme_do,
broj_gostiju)
                VALUES (l_id_stol, l_id_osoba, p_zeljeni_datum, p_vrijeme_od,
p_vrijeme_do, p_broj_gostiju);
```

```
        SET status_rezervacije = "Rezervacija kreirana!";
END IF;

END //
DELIMITER ;
```

15.PROCEDURA
Procedura za dodavanje novog djelatnika

```
DELIMITER //
CREATE PROCEDURE dodaj_djelatnika
(p_ime VARCHAR(50),
p_prezime VARCHAR(50),
p_broj_mob VARCHAR(10),
p_email VARCHAR (30),
p_oib CHAR(11),
p_datum_rodenja DATE,
p_id_zanimanje INTEGER,
OUT status_transakcije VARCHAR(100))
BEGIN

DECLARE l_id_osoba INTEGER;

DECLARE unique_ogranicenje_prekrseno CONDITION FOR 1062;
DECLARE zanimanje_ne_postoji CONDITION FOR 1452;

DECLARE EXIT HANDLER FOR unique_ogranicenje_prekrseno
        BEGIN
                ROLLBACK;
                        SET status_transakcije = CONCAT("Već postoji djelatnik sa oib-om: ",
p_oib);
   END;

DECLARE EXIT HANDLER FOR zanimanje_ne_postoji
        BEGIN
                ROLLBACK;
                SET status_transakcije = "Zanimanje sa tim id-em ne postoji!";
        END;

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;

INSERT INTO osoba (ime, prezime, broj_mob, email) VALUES
        (p_ime, p_prezime, p_broj_mob, p_email);
```

```
SELECT MAX(id) INTO l_id_osoba
        FROM osoba;

INSERT INTO djelatnik (id_osoba, oib, datum_rodenja, id_zanimanje) VALUES
        (l_id_osoba, p_oib, p_datum_rodenja, p_id_zanimanje);

SET status_transakcije = "Djelatnik dodan!";

COMMIT;

END //
DELIMITER ;
```

16. i 17. PROCEDURA
Procedure koje stvaraju stavku za nabavu

```
DELIMITER //
CREATE PROCEDURE dodaj_stavku_za_nabavu (p_naziv_namirnice VARCHAR(50),
p_kolicina DECIMAL(10, 2), p_cijena_hrk NUMERIC(10,2))
BEGIN

        DECLARE l_id_namirnica INTEGER DEFAULT NULL;


   CREATE TEMPORARY TABLE IF NOT EXISTS tmp_nabava_stavka (
                id_namirnica INTEGER NOT NULL,
                kolicina DECIMAL(10, 2),
      cijena_hrk NUMERIC(10,2)
        );

        SELECT id INTO l_id_namirnica
                FROM namirnica
                WHERE naziv = p_naziv_namirnice;

        IF l_id_namirnica IS NULL THEN
                SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Namirnica sa tog naziva ne postoji!';
        ELSEIF p_kolicina <= 0 THEN
                SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Količina mora biti pozitivan broj!';
        ELSEIF p_cijena_hrk = 0 THEN
                SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Cijena ne može biti negativna!';
        ELSE
                INSERT INTO tmp_nabava_stavka VALUES (l_id_namirnica, p_kolicina,
p_cijena_hrk);
```

```
        END IF;

END //
DELIMITER ;



DELIMITER //
CREATE PROCEDURE stvori_nabavu (p_dobavljac INTEGER, p_opis VARCHAR(300),
p_podmireno CHAR(1), p_datum DATE)
BEGIN

        DECLARE l_id_nabava INTEGER;
    DECLARE l_id_namirnica INTEGER;
    DECLARE l_kolicina DECIMAL(10, 2);
    DECLARE l_cijena NUMERIC(10, 2);

        DECLARE cur CURSOR FOR
                SELECT id_namirnica, kolicina, cijena_hrk
                        FROM tmp_nabava_stavka;

        DECLARE EXIT HANDLER FOR NOT FOUND
                BEGIN
                        DELETE FROM tmp_nabava_stavka;
                END;
        IF p_dobavljac IS NULL THEN
                SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Obavezan je upis dobavljaca!';
        ELSEIF p_podmireno NOT IN('D', 'N') THEN
                SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'status podmireno more biti "D" ili "N"!';
        ELSEIF p_datum IS NULL THEN
                SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Obavezan je upis datuma!';
        ELSE


                INSERT  INTO  nabava (id_dobavljac, opis, podmireno, datum)
                VALUES (p_dobavljac, p_opis, p_podmireno, p_datum);

                -- dohvaća id zadnje unesene nabave
                SELECT id INTO l_id_nabava
                        FROM nabava
                        ORDER BY id DESC
                        LIMIT 1;

                OPEN cur;
```

```
            unesi_stavke: LOOP
                FETCH cur INTO l_id_namirnica, l_kolicina, l_cijena;
                INSERT INTO nabava_stavka (id_nabava, id_namirnica, kolicina,
cijena_hrk)
                    VALUES (l_id_nabava, l_id_namirnica, l_kolicina, l_cijena);
            END LOOP unesi_stavke;

            CLOSE cur;
        END IF;

END //
DELIMITER ;
```

18.PROCEDURA
Procedura koji stavlja zadanu nabavu u status podmireno

```
DELIMITER //
CREATE PROCEDURE podmiri_nabavu (p_id INTEGER)
BEGIN

DECLARE l_podmireno CHAR(1);

SELECT podmireno INTO l_podmireno
FROM nabava
WHERE p_id=id;
IF l_podmireno='D'THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'nabava je već u statusu podmireno!';
ELSEIF l_podmireno IS NULL THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'ne postoji nabava sa zadani ID-om!';
ELSE
        UPDATE nabava
   SET podmireno='D'
   WHERE id=p_id;
END IF;
END //
DELIMITER ;
```

19.PROCEDURA
Procedura koji stavlja zadanu reziju u status podmireno

```
DELIMITER //
CREATE PROCEDURE podmiri_reziju (p_id INTEGER)
BEGIN

DECLARE l_placeno CHAR(1);

SELECT placeno INTO l_placeno
FROM rezije
WHERE p_id=id;
IF l_placeno='D'THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'režija je već u statusu podmireno!';
ELSEIF l_placeno IS NULL THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'ne postoji režija sa zadani ID-om!';
ELSE
        UPDATE rezije
   SET placeno='D'
   WHERE id=p_id;
END IF;
END //
DELIMITER ;
```

OVLASTI

Dolje su navedene ovlasti za gosta. Dodavane su redoslijedom kojim su bile potrebne za funkcioniranje na web sučelju.

```
DROP USER IF EXISTS gost;
CREATE USER gost IDENTIFIED BY 'password';
GRANT SELECT, INSERT ON restoran.osoba TO gost;
GRANT SELECT, INSERT ON restoran.adresa TO gost;
GRANT SELECT, INSERT ON restoran.rezervacija TO gost;
GRANT SELECT, INSERT ON restoran.catering_zahtjev TO gost;
GRANT SELECT ON restoran.aktivni_meni TO gost;
GRANT SELECT ON restoran.meni TO gost;
GRANT SELECT ON restoran.alergen TO gost;
GRANT SELECT ON restoran.kategorija_namirnica TO gost;
GRANT SELECT ON restoran.stavka_meni TO gost;
```

GRANT REFERENCES ON restoran. TO gost;*
GRANT EXECUTE ON PROCEDURE restoran.kreiraj_rezervaciju TO gost;
GRANT ALL PRIVILEGES ON restoran.gost_dostava TO gost;
GRANT SELECT ON restoran.sadrzi_alergen TO gost;
GRANT SELECT, INSERT ON restoran.dostava TO gost;
GRANT SELECT, INSERT ON restoran.dostava_stavka TO gost;
GRANT SELECT, INSERT ON restoran.catering_narucitelj TO gost;

Dolje su navedene ovlasti za konobara. Dodavane su redoslijedom kojim su bile potrebne za funkcioniranje na web sučelju.

DROP USER IF EXISTS konobar;
CREATE USER konobar IDENTIFIED BY 'password';
GRANT SELECT ON restoran.meni TO konobar;
GRANT SELECT ON restoran.aktivni_meni TO konobar;
GRANT ALL PRIVILEGES ON restoran.konobar_racun TO konobar;
GRANT SELECT ON restoran.sadrzi_alergen TO konobar;
GRANT SELECT ON restoran.alergen TO konobar;
GRANT SELECT, INSERT ON restoran.osoba TO konobar;
GRANT SELECT, INSERT ON restoran.djelatnik TO konobar;
GRANT SELECT ON restoran.stol TO konobar;
GRANT REFERENCES ON restoran. TO konobar;*
GRANT SELECT, INSERT ON restoran.racun TO konobar;
GRANT SELECT, INSERT ON restoran.stavka_racun TO konobar;
GRANT EXECUTE ON FUNCTION restoran.kreiraj_sifru_racuna_autoincrement TO konobar;

Dolje su navedene ovlasti za poslovodju. Dodavane su redoslijedom kojim su bile potrebne za funkcioniranje na web sučelju.

DROP USER IF EXISTS poslovodja;
CREATE USER poslovodja IDENTIFIED BY 'password';
GRANT SELECT, INSERT, UPDATE ON restoran.djelatnik TO poslovodja;
GRANT SELECT, INSERT ON restoran.osoba TO poslovodja;
GRANT REFERENCES ON restoran. TO poslovodja;*
GRANT ALL PRIVILEGES ON restoran.tablicajutro TO poslovodja;
GRANT ALL PRIVILEGES ON restoran.tablicavecer TO poslovodja;
GRANT SELECT ON restoran.zanimanje TO poslovodja;
GRANT SELECT, INSERT, UPDATE ON restoran.djelatnik_smjena TO poslovodja;
GRANT SELECT ON restoran.smjena TO poslovodja;
GRANT SELECT, INSERT, UPDATE ON restoran.namirnica TO poslovodja;
GRANT SELECT, INSERT, UPDATE ON restoran.meni TO poslovodja;
GRANT SELECT ON namirnice_za_nabavu TO poslovodja;
GRANT EXECUTE ON PROCEDURE restoran.dodaj_stavku_za_otpis TO poslovodja;
GRANT EXECUTE ON PROCEDURE restoran.stvori_otpis TO poslovodja;
GRANT ALL PRIVILEGES ON restoran.tmp_nabava_stavka TO poslovodja;
GRANT EXECUTE ON PROCEDURE restoran.stvori_nabavu TO poslovodja;
GRANT SELECT ON restoran.catering_zahtjev TO poslovodja;

GRANT SELECT, INSERT, UPDATE ON restoran.catering TO poslovodja;
GRANT SELECT ON restoran.adresa TO poslovodja;
GRANT SELECT ON restoran.catering_narucitelj TO poslovodja;
GRANT SELECT ON restoran.nadolazeci_caterinzi TO poslovodja;
GRANT SELECT ON restoran.aktivni_meni TO poslovodja;
GRANT ALL PRIVILEGES ON restoran.trenutni_catering TO poslovodja;
GRANT ALL PRIVILEGES ON restoran.tablicadjelatnika TO poslovodja;
GRANT SELECT, INSERT ON restoran.sadrzi_alergen TO poslovodja;
GRANT SELECT, INSERT ON restoran.alergen TO poslovodja;
GRANT SELECT, INSERT ON restoran.catering_stavka TO poslovodja;
GRANT SELECT, INSERT, UPDATE ON restoran.catering TO poslovodja;
GRANT SELECT, INSERT ON djelatnici_catering TO poslovodja;
GRANT ALL PRIVILEGES ON restoran.namirnicenoves TO poslovodja;
GRANT ALL PRIVILEGES ON restoran.tablicaalergeni TO poslovodja;
GRANT SELECT,INSERT ON restoran.stavka_meni TO poslovodja;
GRANT SELECT ON restoran.racun TO poslovodja;
GRANT SELECT ON restoran.dostava TO poslovodja;
GRANT SELECT ON restoran.rezije TO poslovodja;
GRANT SELECT ON restoran.nabava TO poslovodja;
GRANT SELECT ON restoran.prosjecan_iznos_racuna TO poslovodja;
GRANT SELECT ON restoran.najveci_br_rezervacija TO poslovodja;
GRANT SELECT ON restoran.najcesce_adrese_dostave TO poslovodja;
GRANT EXECUTE ON PROCEDURE restoran.dodaj_djelatnika TO poslovodja;

WEB SUČELJE

Web sučelje je podijeljeno na tri dijela. Prvotni index.php služi samo kako bi se moglo lako navigirati između dijelova koji pripadaju ovlaštenim osobama(gost, konobar, poslovodja). Dok su datoteke od gosta i konobara vezane svaka za jedan file, nije bilo potrebno stavljati dodatne mape.Kod poslovodje postoji šest dodatnih mapa kako bi bilo lakše navigirati se njima.

PODJELA I NAMJENA POJEDINIH PHP DATOTEKA

1.U rootu se nalaze mape *gost, poslovodja, konobar*, te *index.php* koja služi za navigaciju i restart sesije.

1.1.U gost mapi se nalaze:

gost_konekcija.php za uspostavljanje veze s ovlastima gosta
gn_kolicina.php za operacije nad stavkama za dostavu gostu
gost_adresa.php za postavljanje podataka o adresi
gost_info.php za postavljanje podataka o gostu
gost_cateringzahtjev.php za ispunjavanja zahtjev
gost_rezervacijastola.php za izvršavanje rezervacije
izvrsi_dostavu.php izvršava dostavu
gost_sucelje.php povezuje sve zajedno

1.2.U konobar mapi se nalaze:

konobar_konekcija.php za uspostavljanje veze s ovlastima konobara
kn_kolicina.php za operacije nad stavkama za račun
konobar_info.php za odabiranje šifre konobara
konobar_odaberistol.php za odabir stola
izvrsi_racun.php unosi trenutni račun
konobar_sucelje.php povezuje sve zajedno

1.3.U poslovodja mapi se nalaze:

poslovodja_konekcija.php za uspostavljanje veze s ovlastima poslovođe
poslovodja_info.php za odabiranje šifre poslovođe
poslovodja_sucelje.php povezuje dvije datoteke iznad i ostale mape mape *psmjene,djelatnik,pnabava,pcatering,pmeni,panaliza*

1.3.1. psmjene:

p_smjenaoduzmi.php oduzima osobe iz smjene
p_smjenadodaj.php dodaje osobe u smjene
odabirsmjene.php sadrži u sebi sve što je potrebno za unos u smjenu uključujući pozive datotekama iznad
smjenaotprije.php sadrži rad ljudi u određenoj smjeni prije
danotprije.php postavlja smjeneotprije na jutarnju i večernju
p_dodajdatum.php odabire datum koji ćemo mijenjati
p_izvrsiupissmjene.php izvršava upis smjene s parametrima iz datoteke odabirsmjene.php
poslovodja_smjene.php povezuje sve u sebi od gore, vrši petlju nad danima i smjenama otprije

1.3.2. djelatnik:

p_izmjenadjel.php mijenja status zaposlen nezaposlen kod djelatnika
p_djelatnikad.php postavlja tablicu za djelatnike
p_novidjelatnik.php dodaje novog djelatnika
poslovodja_djelatnici.php veže navedeno i ima form koji poziva datoteku p_novidjelatnik.php

1.3.3. pnabava

stanje_namirnica.php prikazuje trenutno stanje svih namirnica
namirnicenabava.php prikazuje namirnice koje bi trebali naručiti obzirom na prošlu godinu
dodajnaotpis.php vrši otpis jedne stavke
nabavastavkaform.php ima form za dodavanje stavki, poziva datoteku nabavastavka.php
novanabava.php upisuje u bazu podatke o nabavi
nabavaform.php ima form za unošenje podataka samoj nabavi za ispravno izvršenu nabavu i poziva novanabava.php
poslovodja_skladiste.php veže navedeno i prikazuje form za otpis, te tablicu za stanje trenutne nabave

1.3.4. pcatering

p_ispunicatering.php dodaje sve u bazu
p_dodajc.php dodaje i oduzima djelatnika na trenutnom cateringu
pn_kolicina dodaje ili smanjuje ovisno o operaciji stavke iz menija

p_zahtjevcatering.php veže navedeno, služi kao sučelje za odgovaranje na zahtjeve i stvaranje cateringa
neispunjeni_caterinzi.php prikazuje zahtjeve za catering koji nisu ispunjeni i prosljeđuje id za zahtjev koji se treba obraditi na datoteku p_zahtjevcatering.php
p_uplacencat.php mijenja stanje cateringa na uplaćeno
p_cateringad.php prikazuje cateringe koji nisu uplaćeni
poslovodja_catering.php veže navedeno

1.3.5. pmeni

izvrsiunosmeni.php unosi u bazu novu stavku s namirnicama i alergenima
dodajalergen.php dodaje ili oduzima alergen s nove stavke menija
postavikolicinu.php postavlja količinu namirnice za koju je pozvana
dodajnamirnicu.php dodaje ili oduzima namirnicu s nove stavke menija
poslovodja_dodajmeni2.php pozvana u poslovodja_dodajmeni.php postavlja naziv nove stavke i njenu cijenu
poslovodja_dodajmeni.php funkcionira kao sučelje za dodavanje nove stavke u meni, veže navedeno
poslovodja_stavkameni.php aktivira deaktivira stavku menija na bazi
poslovodja_meniad.php postavlja tablicu za aktiviranje i deaktiviranje stavki s menija, te pruža navigaciju za poslovodja_dodajmeni.php

1.3.6. panaliza

p_prometmjeseci.php prikazuje prihod po mjesecima
p_rashodmjeseci.php prikazuje rashod po mjesecima
p_racunprosjek.php prikazuje prosjek računa po mjesecima
p_radnici.php prikazuje sate koje su radnici odradili u prošlom(12.mjesecu)
p_rezervacije.php prikazuje osobe s najviše rezervacija
p_najcescedostave.php prikazuje adrese s najviše dostava
poslovodja_analiza.php sadrži sve navedeno

ZAKLJUČAK

Cilj nam je bio što jasnije opisati sustav u našem slučaju za upravljanje restoranom. Svaki član tima, od voditelja do programera i osobe za komunikaciju odradio je maksimalno svoj zadatak. Ovaj projekt uveliko nam je proširio znanje i otvorio neke nove vidike u svijetu baza podataka.