

**SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN**

Petar Matišić

RENT-A-CAR

SEMINAR

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Petar Matišić

Matični broj: 0016145882

Studij: Informacijsko i programsko inženjerstvo

RENT-A-CAR

SEMINAR

Mentor:

dr. sc. Bogdan Okreša Đurić

Varaždin, siječanj 2023.

Izjava o izvornosti

Izjavljujem da je ovaj seminar izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvatanjem odredbi u sustavu FOI Radovi

Sažetak

Aplikacija za *rent-a-car* je softverski alat koji se instalira na računalo i omogućuje vlasnicima *rent-a-car* tvrtki da upravljaju svojim poslovanjem. Takve aplikacije obično uključuju funkcionalnosti poput evidencije najma vozila, računovodstva, vođenja evidencije o klijentima i vozilima, te generiranja izvješća. One također mogu omogućiti automatizirano obavješćavanje klijenata o promjenama u dostupnosti vozila, te olakšati rezervacije putem integracije s web stranicom ili drugim kanalima prodaje. *Desktop* aplikacije za *rent-a-car* pomažu tvrtkama da poboljšaju efikasnost i smanje troškove povezane s upravljanjem najmom vozila.

Ključne riječi: najam, vozilo, baza podataka, PostgreSQL, C#

Sadržaj

1.	Uvod	1
2.	Opis aplikacijske domene	2
3.	Teorijski uvod	3
3.1.	Relacijske baze podataka	3
3.1.1.	Aktivne baze podataka	4
3.1.2.	Temporalne baze podataka	5
3.1.3.	Opis tehnologija korištenih kod izrade projekta	6
4.	Model baze podataka	7
4.1.	Prikaz i opis ERA modela	7
4.1.1.	Okidači	9
4.1.2.	Upiti	17
5.	Implementacija i primjeri korištenja	21
5.0.1.	Kategorija	23
5.0.2.	Klijent	23
5.0.3.	Lokacija	24
5.0.4.	Model	24
5.0.5.	Način plaćanja	25
5.0.6.	Najam	25
5.0.7.	Osiguranje	26
5.0.8.	Osiguranje najma	26
5.0.9.	Početna stranica	27
5.0.10.	Proizvođač	27
5.0.11.	Račun	28
5.0.12.	Vozilo	28
6.	Zaključak	29
	Popis literature	30
	Popis slika	32
1.	GitHub	34

1. Uvod

Ovaj projektni rad bavit će se modeliranjem baze podataka za *rent-a-car*. *Rent-a-car* je općenito iznajmljivanje vozila. U samome procesu iznajmljivanja vozila javljaju se razni entiteti, među kojima postoje neki odnosi, a o kojima će se nešto više reći u ostalim poglavljima.

Kroz ovaj projektni rad bit će opisana teorijska podloga korištenih tehnologija, aplikacijska domena, postupak izrade ERA modela (logički i relacijski model baze podataka) te opis alata za izradu. Zatim će se objasniti korišteni okidači i primjeri upita. Nadalje će biti opisana tehnologija za izradu aplikacije, te prikaz same funkcionalnosti razvijene aplikacije.

Pomoć pri izradi upita korištena je knjiga [1]. Kod kreiranja okidača korištena je knjiga [1].

Za izradu baze podataka i razvoj aplikacije korišteni su sljedeći alati: PostgreSQL [2], C# [3], Npgsql [4] te DataGrip 2022.2.5 [5] i Visual Studio 2022 [3]. PostgreSQL ima lokalni poslužitelj koji se pokreće preko DataGrip-a. DataGrip je alat koji služi za modeliranje baze podataka, kreiranje tablica, dodavanja veza među tablicama, kreiranja okidača, izvršavanja upita i dr. Visual Studio 2022 je razvojno okruženje u kojemu je napravljena aplikacija u obliku *Windows* formi.

2. Opis aplikacijske domene

Domena aplikacije je *rent-a-car*, odnosno iznajmljivanje vozila. *Rent-a-car* je danas sve popularniji, gdje agencije iznajmljuju vozila drugim osobama. Središte aplikacijske domene je najam, na kojeg se vežu neki drugi entiteti poput klijenta, vozila, osiguranja, lokacije, računa.

Popis entiteta:

- Najam,
- Klijent,
- Lokacija,
- Račun,
- Način plaćanja,
- Vozilo,
- Kategorija,
- Model,
- Proizvođač,
- Osiguranje.

Uz navedene entitete, javlja se još jedan entitet koji je po prirodi veze više na više ($M : N$), a to je entitet Osiguranje najma. Glavni cilj aplikacije je omogućiti izvođenje operacija unosa, čitanja, brisanja i ažuriranja nad bazom podataka, te izvršavanje nekoliko upita. Bitno je napomenuti kako je poslovna logika same aplikacije napravljena na razini baze podataka koristeći aktivne i temporalne vrste baza podataka kroz funkcije i okidače. Sama aplikacija je izrađena u programskom jeziku C# u razvojnom okruženju Visual Studio 2022 kao *Windows Forms* aplikacija.

Središte aplikacije čini entitet **najam**, kojeg određuju datum početka i završetka najma, lokacija gdje će klijent preuzeti vozilo te gdje će ga vratiti iznajmljivaču. Osim toga na entitet najam vezemo i podatke o vozilu, klijentu preko vanjskog ključa na te entitete.

Klijenta čini osoba kojoj se iznajmljuje vozilo, a atributi koji ju određuju su njegova šifra, ime, prezime, datum rođenja, mjesto stanovanja, kontakt broj te broj vozačke dozvole. Uz klijenta drugi važan entitet je i **vozilo**, a atributi koji ga određuju su registracijska oznaka, godina proizvodnje, kilometraža, boja, datum tehničkog pregleda te vanjski ključevi na model, kategoriju te lokaciju. Nakon završetka najma izdaje se **račun** korisniku, kojeg određuju atributi ID računa, datum izdavanja, iznos, atribut "iskorišteno" koji označava popust na najam auta na neke periode te vanjski ključ na način plaćanja, koji daje sve potrebne podatke klijentu o plaćanju. Uz sve to postoji i entitet osiguranje, kojeg određuju atributi naziv, opis te polica osiguranja.

3. Teorijski uvod

3.1. Relacijske baze podataka

Relacijske baze podataka (engl. *Relational database management systems, RDBMS*) [1] su softverski sustavi koji se koriste za upravljanje relacijskim bazama podataka. Relacijske baze podataka su strukturirane na način da se podaci pohranjuju u tablicama koje su povezane međusobno vezama.

Svaka tablica sadrži niz redaka i stupaca, a svaki redak predstavlja jedan zapis, a svaki stupac jednu određenu vrstu podatka. Na primjer, u tablici "osobe" može se pohraniti ime, prezime i godine svake osobe. Veze između tablica omogućuju povezivanje podataka iz više tablica te se na taj način mogu dobiti kompleksniji podaci. Dakle, relacije između tablica su uspostavljene pomoću ključeva. Postoje dvije vrste ključeva: primarni ključ i vanjski ključ [1]. Primarni ključ je jedinstven i ne smije se ponavljati unutar jedne tablice, dok vanjski ključ povezuje red u jednoj tablici s primarnim ključem u drugoj tablici.

Jedna od glavnih prednosti relacijskih baza podataka je njihova strukturiranost i normaliziranost podataka [1]. To znači da su podaci pohranjeni na optimalan način te da se u tablicama ne ponavljaju podaci. To omogućuje lakše ažuriranje i održavanje podataka te se sprječava pojava grešaka. Relacijske baze podataka također omogućuju brzo izvršavanje različitih upita i izvještaja. Jedna od još nekoliko prednosti relacijskih baza podataka je njihov integritet podataka [1]. Integritet podataka je mjera točnosti i cjelovitosti podataka u bazi podataka. Relacijske baze podataka omogućuju korištenje integritetnih ograničenja kako bi se osiguralo da se podaci unose ispravno i da se ne gube pri ažuriranjima.

Međutim, relacijske baze podataka imaju i neke nedostatke. Jedan od najčešćih je ograničenost u obradi velikih količina podataka, osobito podataka koji se ne lako mogu strukturirati u tablicama [1]. Ako se struktura podataka mijenja, to može zahtijevati promjene u cijeloj bazi podataka, što može biti vremenski zahtjevno i skupo. Također, relacijske baze podataka mogu biti skupe u smislu hardverskih i softverskih zahtjeva te zahtijevaju stručno održavanje i administraciju kako bi se osigurala njihova optimalna radna sposobnost i sigurnost [1]. Iako se često koriste u velikim korporativnim okruženjima, postoje i drugi tipovi baza podataka koji mogu biti prikladniji za manja poduzeća ili individualne korisnike.

3.1.1. Aktivne baze podataka

Aktivne baze podataka (engl. *Active Databases*) [1] su specifičan tip baza podataka koje koriste aktivne pravila za upravljanje podacima i njihovim manipulacijama. Ove baze podataka se razlikuju od tradicionalnih relacijskih baza podataka time što imaju integriranu mehanizam za upravljanje pravilima i događajima, što ih čini pogodnim za rješavanje složenih poslovnih zadataka.

Takve vrste baze podataka su programirane da reagiraju na specifične događaje koji se događaju unutar same baze podataka ili u okruženju u kojem se baza nalazi. Ovaj pristup bazama podataka omogućuje im da proaktivno djeluju na temelju događaja i podataka koji su im dostupni, bez potrebe da ih prvo pozove program koji koristi tu bazu podataka [1].

Primjeri događaja koji mogu pokrenuti aktivnost u aktivnoj bazi podataka uključuju promjenu podataka u bazi, dostupnost određenih podataka, vrijeme ili specifičnu akciju korisnika. Na primjer, aktivna baza podataka može imati pravilo da automatski pošalje poruku e-pošte svima koji su zainteresirani za promjenu cijene određenog proizvoda u bazi podataka.

Aktivne baze podataka imaju nekoliko prednosti u odnosu na tradicionalne relacijske baze podataka. Oni omogućuju bolju automatizaciju poslova i smanjuju potrebu za ručnim intervencijama, što može dovesti do ušteda vremena i smanjenja pogrešaka. Također, aktivne baze podataka mogu biti bolje prilagođene specifičnim potrebama korisnika jer mogu reagirati na specifične događaje na način koji je programiran od strane korisnika [1].

Međutim, aktivne baze podataka također imaju neke nedostatke. Jedan od najvećih je složenost njihovog razvoja i održavanja, što može biti skupo i zahtijevati visoko stručne radnike. Također, aktivne baze podataka ne podržavaju sve standarde SQL-a, što može otežati integraciju s drugim sustavima [1]. Pored toga, aktivne baze podataka često imaju ograničenja u pogledu veličine i performansi, što ih čini manje pogodnima za velike aplikacije ili one koje zahtijevaju visoku razinu performansi. Iako imaju neke nedostatke, aktivne baze podataka i dalje imaju svoje mjesto u razvoju aplikacija i mogu biti korisne u određenim okolnostima.

3.1.2. Temporalne baze podataka

Temporalne baze podataka [1] su vrsta relacijskih baza podataka koja ima dodatni kapacitet za spremanje i praćenje vremenskih informacija. Ove baze podataka omogućuju korisnicima da spremaju i prate podatke u sklopu vremenskog konteksta, što je korisno u mnogim aplikacijama gdje je važno pratiti kako se podaci mijenjaju kroz vrijeme. Primjeri takvih aplikacija uključuju financijske sustave, zdravstveni sustavi i sustave za upravljanje projektima.

Jedna od glavnih prednosti temporalnih baza podataka je njihova sposobnost da efikasno prate i spremaju podatke o promjenama u vremenu. Omogućuju korisnicima da lako dohvate podatke u određenom vremenskom rasponu ili da prate promjene u podacima tijekom vremena. Također, temporalne baze podataka često imaju bolje mehanizme za povezivanje podataka u vremenu, što olakšava rad s podacima koji su povezani s vremenom.

Međutim, temporalne baze podataka također imaju neke nedostatke. Jedan od najvećih je složenost njihovog razvoja i održavanja, što može biti skupo i zahtjevno. Osim toga, temporalne baze podataka mogu biti otežane u upravljanju, što može dovesti do poteškoća u pristupu i obradi podataka. Također, korištenje temporalnih baza podataka može dovesti do povećanja veličine baze podataka, što može uzrokovati probleme s performansama [1]. Sve u svemu, temporalne baze podataka su korisne u specifičnim situacijama, ali se treba pažljivo razmisliti o njihovoj uporabi u širem kontekstu.

3.1.3. Opis tehnologija korištenih kod izrade projekta

PostgreSQL [2] je jedna od najpopularnijih i najstarijih otvorenih relacijskih baza podataka. Napravljen je 1990. godine i od tada se stalno razvija. Podržava većinu standarda SQL-a i ima mnoge napredne značajke poput podrške za transakcije, indekse, trigere, procedurne jezike, geospacialne podatke i mnogo toga drugog. PostgreSQL se često koristi u komercijalnim aplikacijama, ali također se može besplatno koristiti u svrhu razvoja i testiranja. Ima veliku zajednicu korisnika i razvojnu ekipu koja se trudi održati visok kvalitet i sigurnost baze podataka. Ovaj softverski alat je moguće preuzeti na poveznici:

<https://www.postgresql.org/download/>

C# je programski jezik koji je razvijen od strane Microsofta za upotrebu u razvoju aplikacija za Microsoftove platforme, kao što su Windows i .NET [3]. To je jezik visokog nivoa koji se bazira na C i C++, a nudi mnoge modernije karakteristike kao što su tipiziranje varijabli unaprijed i automatsko upravljanje memorijom. C# se često koristi u razvoju web aplikacija, igara i mobilnih aplikacija te se smatra jednim od najpopularnijih programskih jezika danas.

Integrirano razvojno okruženje **Microsoft Visual Studio** [3] je alat za razvoj softvera koji omogućuje programerima da razvijaju aplikacije za različite platforme, uključujući Windows, Mac, Android i iOS. Ono pruža sve potrebne alate za pisanje, testiranje i debugiranje koda, te integriranu podršku za različite programske jezike, uključujući C#, C++, Python i mnoge druge. Visual Studio također uključuje integraciju s različitim verzionim kontrolama, što olakšava rad u timu na projektima i omogućava lakše održavanje koda. Ovaj softverski alat je moguće preuzeti na poveznici:

<https://visualstudio.microsoft.com/downloads/>

Npgsql [4] je ADO.NET *provider* za PostgreSQL baze podataka. Omogućuje .NET aplikacijama da se spajaju na i rade s PostgreSQL bazama podataka. Ovaj softverski alat je moguće preuzeti na poveznici:

<https://www.npgsql.org/doc/installation.html>

JetBrains DataGrip [5] je integrirano razvojno okruženje (IDE) za upravljanje različitim vrstama relacijskih baza podataka. Podržava različite vrste baza podataka, uključujući MySQL, PostgreSQL, Oracle i SQL Server. DataGrip sadrži alate za rad s SQL-om, kao što su uređivač upita, debugger i integracija s repozitorijima kontrola izvornog koda. Također ima mogućnost rada s različitim bazama podataka u isto vrijeme, što ga čini pogodnim za razvoj aplikacija koja se temelji na više baza podataka. DataGrip je dostupan za Windows, MacOS i Linux operativne sustave. Ovaj softverski alat je moguće preuzeti na poveznici:

<https://www.jetbrains.com/datagrip/download/>

4. Model baze podataka

4.1. Prikaz i opis ERA modela

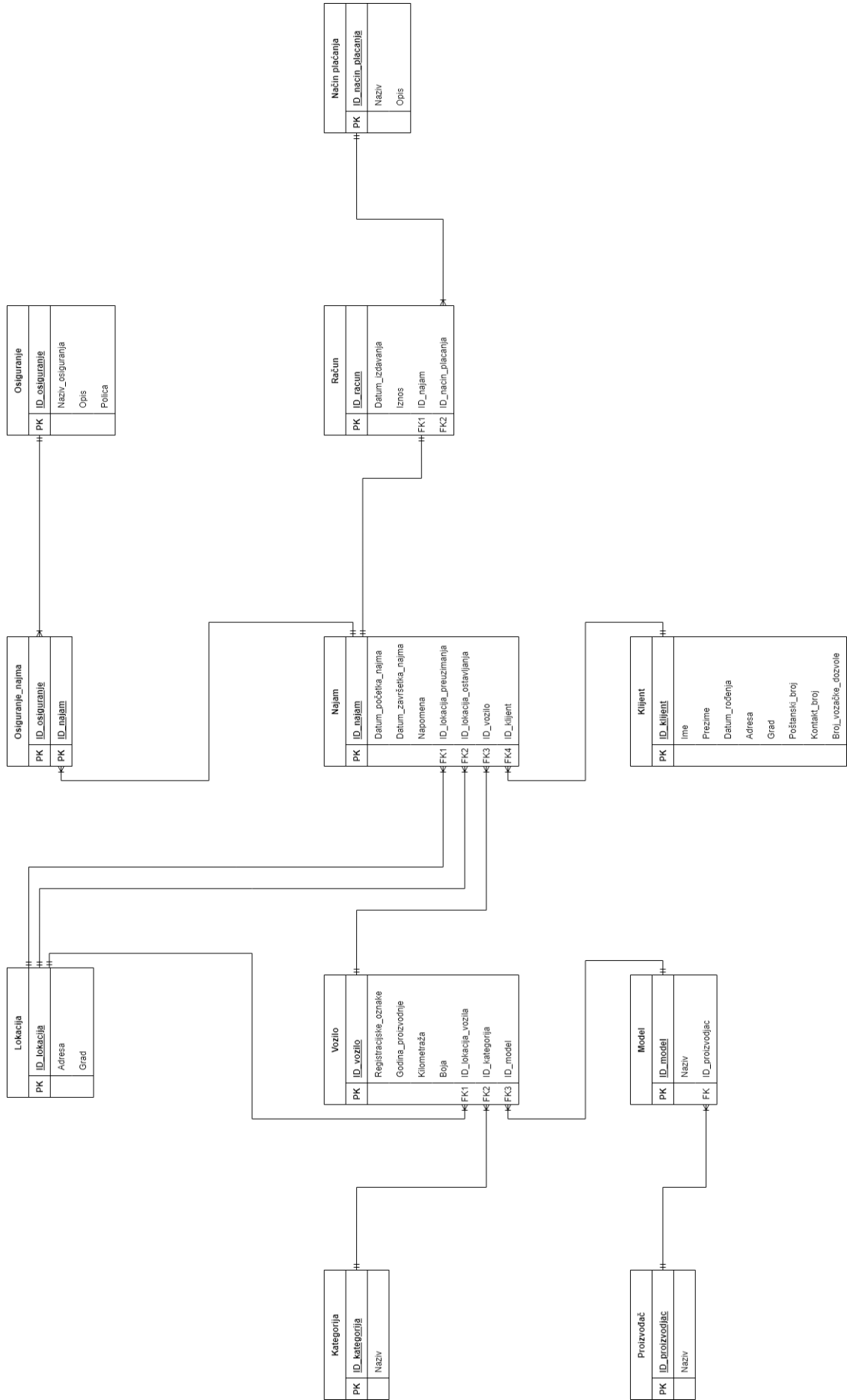
Na ERA modelu se prikazuje podatkovna struktura objekata i njihovi odnosi, odnosno kardinalnosti. Kardinalnost se postiže različitim vrstama veza između jedne, dvije ili više tablica. Na ERA modelu za Rent-a-car javljaju se binarne veze jedan naprema više, te jedna veza više naprema više.

Nakon što je izrađen ERA model on se pretvara u relacijski model. On je zasnovan na teoriji skupova [1]. Relacijska shema sastoji se od naziva i skupa atributa. Prema tome, relacijski model baze podataka za Rent-a-car je sljedeći:

- **najam** (IDnajam, datumpocetkanajma, datumzavrsetkanajma, napomena, *idlokacija*preuzimanja, *idlokacija*ostavljanja, *idvozilo*, *idklijent*, vrijediod, vrijedido),
- **klijent** (IDklijent, ime, prezime, datumrođenja, adresa, grad, postanski broj, kontakt broj, brojvozackedozvole),
- **vozilo** (IDvozilo, registracijskeoznake, godinaproizvodnje, kilometraza, boja, *idlokacijavozila*, *idkategorija*, *idmodel*, pregled),
- **racun** (IDracun, datumizdavanja, iznos, *idnajam*, *idnacinplacanja*, iskoristeno),
- **lokacija** (IDlokacija, adresa, grad),
- **model** (IDmodel, naziv, *idproizvodac*),
- **proizvodac** (IDproizvodac, naziv),
- **kategorija** (IDkategorija, naziv),
- **nacinplacanja** (IDnacinplacanja, naziv, opis),
- **osiguranje** (IDosiguranje, nazivosiguranja, opis, polica),
- **osiguranjenajma** (Idosiguranje, Idnajam).

Primarni ključevi su podcrtani, a vanjski ključevi u kurzivu.

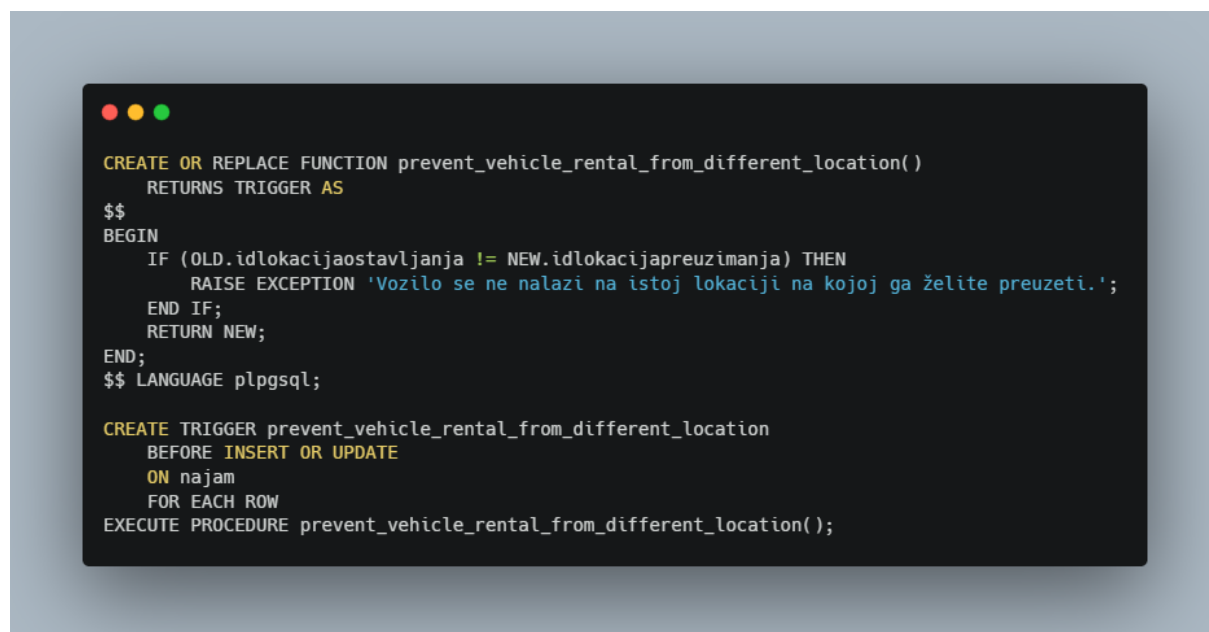
U nastavku na slici 1. je prikazan ERA model za aplikaciju za Rent-a-car. Model sadrži jedanaest entiteta, koji su međusobno povezani. Veze u modelu su $(1 : M)$ te $(N : M)$, ternarnih veza nema.



Slika 1: ERA model

4.1.1. Okidači

Kod razvoja aplikacije kreirano je osam okidača. Okidači se mogu kreirati u DataGrip-u, a mogu i izravno implementirati u C# kod. Kod izrade ovoga projekta okidači su kreirani u DataGrip-u na način da se odabere "New Query" te se napiše SQL kod okidača. Nakon što je kreiran, odabere se opcija "Run" te se okidač automatski spremi u mapi "routines" za tablicu za koju je kreiran.



Slika 2: Okidač 1

Prvi okidač koji je kreiran sprječava da se unajmi vozilo koje se ne nalazi na istoj lokaciji na kojoj ga klijent želi preuzeti. Ukoliko se lokacija preuzimanja vozila (NEW.idlokacijapreuzimanja) razlikuje od trenutne lokacije vozila (OLD.idlokacijaostavljanja), okidač će baciti iznimku s porukom "Vozilo se ne nalazi na istoj lokaciji na kojoj ga želite preuzeti.". Inače, okidač vraća nove vrijednosti za "najam". Okidač se izvršava prije INSERT ili UPDATE naredbe na tablici "najam" za svaki redak u tablici.

```

CREATE OR REPLACE FUNCTION update_vehicle_location_on_return()
    RETURNS TRIGGER AS
$$
BEGIN
    IF (NEW.idlokacijaostavljanja != OLD.idlokacijapreuzimanja) THEN
        UPDATE vozilo
        SET idlokacijavozila = NEW.idlokacijaostavljanja
        WHERE vozilo."IDvozilo" = NEW."idvozilo";
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_vehicle_location_on_return
    AFTER INSERT OR UPDATE
    ON najam
    FOR EACH ROW
    EXECUTE PROCEDURE update_vehicle_location_on_return();

```

Slika 3: Okidač 2

Drugi okidač u PostgreSQL-u koji služi ažuriranju lokacije vozila ukoliko klijent vrati vozilo na neku drugu lokaciju. Ovaj okidač se izvršava nakon unosa ili ažuriranja podataka u tablicu "najam". Ukoliko se lokacija povrata vozila razlikuje od lokacije preuzimanja vozila u tablici, okidač će ažurirati trenutnu lokaciju vozila u tablici "vozilo" tako da se postavi na lokaciju na kojoj je vozilo vraćeno. Okidač vraća nove podatke u tablici "najam".

```

CREATE OR REPLACE FUNCTION check_rental_validity()
    RETURNS TRIGGER AS
$$
BEGIN
    IF (NEW.vrijediod > NEW.vrijedido) THEN
        RAISE EXCEPTION 'Početak vremenskog raspona validnosti najma ne može biti poslije kraja
raspona.';
    ELSIF (NEW.datumpocetkanajma < NEW.vrijediod OR NEW.datumpocetkanajma > NEW.vrijedido) THEN
        RAISE EXCEPTION 'Datum početka najma nije unutar vremenskog raspona validnosti najma.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_rental_validity
    BEFORE INSERT OR UPDATE
    ON najam
    FOR EACH ROW
EXECUTE PROCEDURE check_rental_validity();

```

Slika 4: Okidač 3

Treći okidač `check_rental_validity` provjerava je li vrijeme početka najma unutar vremenskog raspona trajanja najma koji je naveden u tablici "najam". Ako nije, baca iznimku. Ako je početak vremenskog raspona validnosti najma poslije kraja raspona, također baca iznimku. Okidač se izvršava prije svakog INSERT-a ili UPDATE-a u tablici "najam".


```

CREATE OR REPLACE FUNCTION prevent_double_rentals()
    RETURNS TRIGGER AS
$$
BEGIN
    -- Provjera je li najam nekog klijenta aktivan
    IF (NEW.datumzavrsetkanajma IS NULL AND
        (SELECT COUNT(*) FROM najam WHERE idklijent = NEW.idklijent AND najam.datumzavrsetkanajma IS
        NULL) > 0) THEN
        RAISE EXCEPTION 'Ovaj klijent već ima aktivni najam vozila.';
    -- Provjera je li klijent već iznajmio neko vozilo u rasponu od datuma početka najma do datuma
    završetka najma
    ELSIF (NEW.datumzavrsetkanajma IS NULL AND
        (SELECT COUNT(*)
        FROM najam
        WHERE idklijent = NEW.idklijent
        AND datumpocetkanajma <= NEW.datumpocetkanajma
        AND datumzavrsetkanajma >= NEW.datumpocetkanajma) > 0) THEN
        RAISE EXCEPTION 'Ovaj klijent već ima iznajmljeno vozilo u odabranom vremenskom rasponu.';
    -- Provjera je li najam nekog vozila aktivan
    ELSIF (NEW.datumzavrsetkanajma IS NULL AND
        (SELECT COUNT(*) FROM najam WHERE idvozilo = NEW.idvozilo AND najam.datumzavrsetkanajma IS
        NULL) > 0) THEN
        RAISE EXCEPTION 'Ovo vozilo je već iznajmljeno.';
    -- Provjera je li vozilo već iznajmljeno u rasponu od datuma početka najma do datuma završetka
    najma
    ELSIF (NEW.datumzavrsetkanajma IS NULL AND
        (SELECT COUNT(*)
        FROM najam
        WHERE idvozilo = NEW.idvozilo
        AND datumpocetkanajma <= NEW.datumpocetkanajma
        AND datumzavrsetkanajma >= NEW.datumpocetkanajma) > 0) THEN
        RAISE EXCEPTION 'Ovo vozilo je već iznajmljeno u odabranom vremenskom rasponu.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER prevent_double_rentals
    BEFORE INSERT OR UPDATE
    ON najam
    FOR EACH ROW
    EXECUTE PROCEDURE prevent_double_rentals();

```

Slika 5: Okidač 4

Četvrti okidač je namijenjen za provjeru je li se neko vozilo ili neki klijent već nalazi u nekom aktivnom najmu ili da li je vozilo ili klijent već iznajmio neko vozilo, odnosno da se vozilo koristi u nekom zadanom vremenskom rasponu. Ako se bilo koja od tih situacija dogodila, okidač će baciti iznimku s odgovarajućom porukom o grešci. Okidač se izvršava prije unosa ili ažuriranja podatka u tablicu "najam".

```

CREATE OR REPLACE FUNCTION prevent_frequent_rentals()
  RETURNS TRIGGER AS
$$
BEGIN
  IF (NEW.datumzavrsetkanajma IS NULL AND (SELECT COUNT(*)
                                           FROM najam
                                           WHERE idklijent = NEW.idklijent
                                           AND najam.datumzavrsetkanajma IS NOT NULL
                                           AND datumzavrsetkanajma >= NEW.datumpocetkanajma -
                                           INTERVAL '7 days') >
      0) THEN
    RAISE EXCEPTION 'Ovaj klijent je unajmio vozilo u zadnjih 7 dana.';
  ELSIF (NEW.datumzavrsetkanajma IS NULL AND (SELECT COUNT(*)
                                           FROM najam
                                           WHERE idvozilo = NEW.idvozilo
                                           AND najam.datumzavrsetkanajma IS NOT NULL
                                           AND datumzavrsetkanajma >= NEW.datumpocetkanajma -
                                           INTERVAL '7 days') >
      0) THEN
    RAISE EXCEPTION 'Ovo vozilo je unajmljeno u zadnjih 7 dana.';
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER prevent_frequent_rentals
  BEFORE INSERT OR UPDATE
  ON najam
  FOR EACH ROW
  EXECUTE PROCEDURE prevent_frequent_rentals();

```

Slika 6: Okidač 5

Peti okidač se koristi za sprječavanje najma vozila od strane klijenta koji je unajmio vozilo u zadnjih 7 dana ili za sprječavanje najma vozila koje je unajmljeno u posljednjih 7 dana. Okidač se pokreće prije INSERT ili UPDATE naredbe na tabeli "najam" i provjerava je li datum završetka najma prazan. Ako je prazan, okidač provjerava je li postoji zapis u tabeli "najam" s istim ID-om klijenta ili vozila čiji je datum završetka najma nedavno. Ako postoji takav zapis, okidač izbacuje iznimku. Inače, okidač vraća novi zapis.

```

CREATE OR REPLACE FUNCTION prevent_unavailable_rentals()
    RETURNS TRIGGER AS
$$
BEGIN
    IF (NEW.datumzavrsetkanajma IS NULL AND (SELECT COUNT(*)
                                                FROM najam
                                                WHERE idvozilo = NEW.idvozilo
                                                AND najam.datumzavrsetkanajma IS NULL
                                                AND najam.vrijedido >= NEW.datumpocetkanajma) > 0) THEN
        RAISE EXCEPTION 'Ovo vozilo je trenutno u uporabi ili je unajmljeno u budućnosti.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER prevent_unavailable_rentals
    BEFORE INSERT OR UPDATE
    ON najam
    FOR EACH ROW
    EXECUTE PROCEDURE prevent_unavailable_rentals();

```

Slika 7: Okidač 6

Šesti okidač se poziva prije INSERT ili UPDATE naredbi na tablici "najam" i primjenjuje se na svaki red u tablici. Funkcija `prevent_unavailable_rentals()` provjerava je li vozilo trenutno u uporabi ili je unajmljeno u budućnosti. Ako je vozilo u uporabi ili je unajmljeno u budućnosti, funkcija baca iznimku "Ovo vozilo je trenutno u uporabi ili je unajmljeno u budućnosti.". Inače, vraća novi red u tablici "najam".

```

CREATE OR REPLACE FUNCTION check_vehicle_inspection_status()
    RETURNS TRIGGER AS
$$
BEGIN
    IF (NEW.pregled < NOW() - INTERVAL '12 months') THEN
        RAISE EXCEPTION 'Ovo vozilo nije bilo na pregledu u zadnjih 12 mjeseci.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_vehicle_inspection_status
    BEFORE INSERT OR UPDATE
    ON vozilo
    FOR EACH ROW
    EXECUTE PROCEDURE check_vehicle_inspection_status();

```

Slika 8: Okidač 7

Sedmi okidač provjerava je li se datum zadnjeg tehničkog pregleda vozila nalazi u posljednjih 12 mjeseci odnosno godini dana. Ako nije, izvršava se naredba `RAISE EXCEPTION` koja će ispisati poruku o grešci i prekinuti izvršavanje okidača. Inače, okidač vraća nove podatke u tablicu. Okidač se izvršava prije svakog `INSERT`-a ili `UPDATE`-a u tablici "vozilo" i to za svaki red u tablici pojedinačno.

```

CREATE OR REPLACE FUNCTION apply_rental_discount()
    RETURNS TRIGGER AS
$$
DECLARE
    idracun integer;
    iskor boolean;
BEGIN
    -- Dohvaćanje ID-a računa za određeni najam
    SELECT "IDracun" INTO idracun FROM racun WHERE racun.idnajam = NEW."IDnajam";
    -- Provjera da li je okidač već izvršen
    SELECT iskoristeno into iskor FROM racun WHERE "IDracun" = idracun;
    IF (iskor=true) THEN
        RETURN NEW;
    END IF;
    -- Provjera da li je razdoblje najma veće od 14 dana
    IF (NEW.datumzavrsetkanajma - NEW.datumpocetkanajma > 14) THEN
        -- Dohvaćanje ID-a računa za određeni najam
        SELECT "IDracun" INTO idracun FROM racun WHERE idnajam = NEW."IDnajam";
        -- Ažuriranje računa
        UPDATE racun
        SET iznos = iznos * 0.9,
            iskoristeno = true
        WHERE racun."IDracun" = idracun;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER apply_rental_discount
    AFTER INSERT OR UPDATE
    ON najam
    FOR EACH ROW
    EXECUTE PROCEDURE apply_rental_discount();

```

Slika 9: Okidač 8

Osmi i zadnji okidač se koristi za primjenu popusta na račun za najam vozila ukoliko je razdoblje najma duže od 14 dana. Okidač prvo dohvaća ID računa za najam koji se promijenio/unio u tablicu "najam" i provjerava je li već primijenjen popust na taj račun. Ako nije, provjerava se da li je razdoblje najma veće od 14 dana. Ako jest, iznos na računu se množi s 0.9 (što predstavlja popust od 10%) i označava se da je popust primijenjen. Okidač nakon toga vraća nove vrijednosti u tablicu "najam".

4.1.2. Upiti

Kod izrade ovoga projekta kreirani su sljedeći upiti:

- Upit koji prikazuje datume početka i završetka najma za određeni model vozila,
- upit koji prikazuje sva unajmljena vozila, te datume početka i završetka najma za određenog klijenta,
- upit koji prikazuje sva vozila koja se trenutno koriste,
- upit koji prikazuje sva vozila koja se ne koriste ali za određenu lokaciju.

The screenshot shows a Windows application window titled "Upiti". It contains four sections, each with a search bar and a "Prikaži" button, followed by a table of results.

Model: Octavia **Prikaži**

Naziv modela vozila i datumi najma:

	Naziv modela	Početak najma	Završetak najma
▶	Octavia	1.9.2022.	
	Octavia	1.2.2022.	30.4.2022.
	Octavia	1.2.2022.	30.4.2022.
	Octavia	1.2.2022.	30.4.2022.

Klijent: Petar **Prikaži**

Sva unajmljena vozila određenog klijenta:

	Ime klijenta	Prezime klijenta	Proizvođač auta	Naziv modela auta	Početak najma	Završetak najma
▶	Petar	Matišić	Mazda	CX-3	1.1.2022.	30.1.2022.
*	Petar	Matišić	Mazda	CX-3	10.1.2023.	20.1.2023.

Vozila koja se trenutno koriste: **Prikaži**

	Proizvođač auta	Model auta	Ime klijenta	Prezime klijenta	Početak najma
▶	Skoda	Octavia	Ivo	Ivić	1.9.2022.
*					

Lokacija: Zagreb **Prikaži**

Pregled vozila po lokacijama koji se ne koriste:

	Grad	Registracijska oznaka	Proizvođač auta	Model auta
▶	Zagreb	12345	Mazda	CX-3
*				

Slika 10: Windows forma upita

```

select model.naziv          as "Naziv modela",
       najam.datumpocetkanajma as "Početak najma",
       najam.datumzavrsetkanajma as "Završetak najma"
from model,
     vozilo,
     najam
where vozilo.idmodel = model."IDmodel"
     and najam.idvozilo = vozilo."IDvozilo"
     and model."naziv" = 'Octavia';

```

Slika 11: Upit 1

Kod prvog upita u zaglavlju rezultata prikazuje se naziv modela vozila te datum početka i završetka najma. Potrebno je spojiti tablice model, vozilo i najam, izjednačiti vanjske i primarne ključeve, te na kraju primarni ključ tablice model mora biti jednak vrijednosti koju korisnik unese u *textbox* u aplikaciji.

```

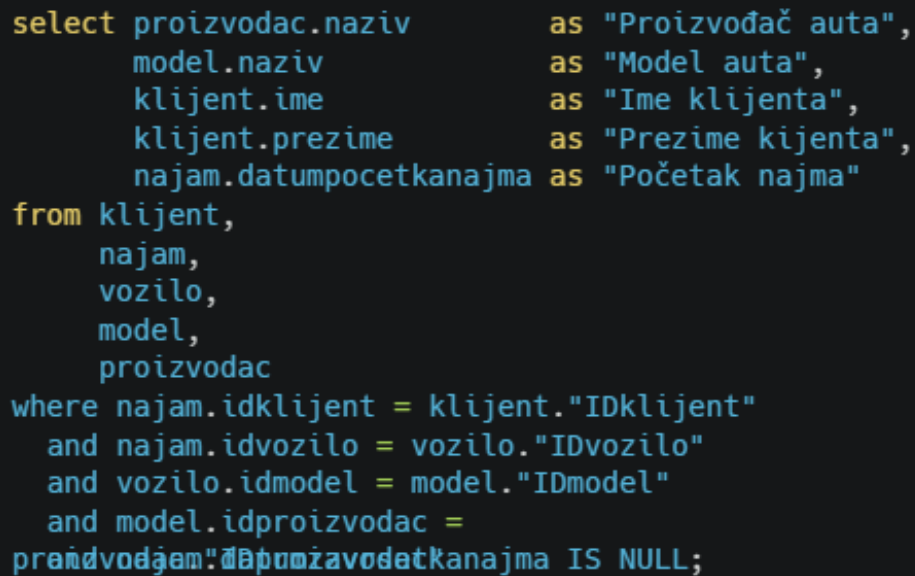
select klijent.ime          as "Ime klijenta",
       klijent.prezime      as "Prezime klijenta",
       proizvodac.naziv     as "Proizvođač auta",
       model.naziv          as "Naziv modela auta",
       najam.datumpocetkanajma as "Početak najma",
       najam.datumzavrsetkanajma as "Završetak najma"
from klijent
     join najam on (klijent."IDklijent" = najam.idklijent)
     join vozilo on (vozilo."IDvozilo" = najam.idvozilo)
     join model on (model."IDmodel" = vozilo.idmodel)
     join proizvodac on (proizvodac."IDproizvodac" =
       model.idproizvodac 'Petar');

```

Slika 12: Upit 2

Drugi upit prikazuje sva vozila koje je određeni klijent unajmio. U zaglavlju rezultata upita prikazuju se naziv proizvođača i modela, te datum početka i završetka najma. Spaja se ukupno 5 tablica: klijent, najam, vozilo, model, proizvođač. Izjednačavaju se primarni i vanjski

ključevi navedenih tablica te na kraju primarni ključ tablice *klijent* mora biti jednak vrijednosti upisanoj od strane korisnika u *textbox* u aplikaciji.

A screenshot of a terminal window with a dark background and light-colored text. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The text is a SQL query with syntax highlighting: 'select' is yellow, table names and column names are light blue, and string literals are white. The query joins five tables: klijent, najam, vozilo, model, and proizvodac. It selects specific columns and aliases for each, and includes a condition to check for NULL values in the 'datumzavršetkanajma' column.

```
select proizvodac.naziv      as "Proizvođač auta",
       model.naziv          as "Model auta",
       klijent.ime           as "Ime klijenta",
       klijent.prezime       as "Prezime klijenta",
       najam.datumpocetkanajma as "Početak najma"
from klijent,
     najam,
     vozilo,
     model,
     proizvodac
where najam.idklijent = klijent."IDklijent"
     and najam.idvozilo = vozilo."IDvozilo"
     and vozilo.idmodel = model."IDmodel"
     and model.idproizvodac =
     proizvodjac."IDproizvodac"
     and najam.datumzavršetkanajma IS NULL;
```

Slika 13: Upit 3

Treći upit koji je kreiran prikazuje nazive modela i proizvođača onih vozila koji se trenutno koriste. U zaglavlju rezultata upita se nalaze nazivi proizvođača i modela, te datum početka najma. Spaja se pet tablica, izjednačavaju se vanjski i primarni ključevi, te na kraju identifikator koristi li se neko vozilo ili ne je datum završetka najma. Ako je polje prazno to znači da se vozilo trenutno koristi.


```

select (select Grad
        from Lokacija
        where Lokacija."IDlokacija" = Vozilo.idlokacijavozila)          as "Grad",
        Vozilo.registracijskeoznake                                     as "Registracijska
oznaka",
        proizvodac.Naziv                                             as "Proizvodjac",
        (select Model.Naziv from Model where Model."IDmodel" = Vozilo.idmodel) as "Model"
from Vozilo
        join Najam on (Najam.idvozilo = Vozilo."IDvozilo")
        join Model on (Model."IDmodel" = Vozilo.idmodel)
        join proizvodac on (proizvodac."IDproizvodac" = Model.idproizvodac)
where Najam.datumzavrsetkanajma is not null
    and (select Lokacija.grad from Lokacija where Lokacija."IDlokacija" = Vozilo.idlokacijavozila) =
    'Zagreb'
except
select (select Grad
        from Lokacija
        where Lokacija."IDlokacija" = Vozilo.idlokacijavozila)          as "Grad",
        Vozilo.registracijskeoznake                                     as "Registracijska
oznaka",
        proizvodac.Naziv                                             as "Proizvodjac",
        (select Model.Naziv from Model where Model."IDmodel" = Vozilo.idmodel) as "Model"
from Vozilo
        join Najam on (Najam.idvozilo = Vozilo."IDvozilo")
        join Model on (Model."IDmodel" = Vozilo.idmodel)
        join proizvodac on (proizvodac."IDproizvodac" = Model.idproizvodac)
where Najam.datumzavrsetkanajma is null
    and (select Lokacija.grad from Lokacija where Lokacija."IDlokacija" = Vozilo.idlokacijavozila) =
    'Zagreb';

```

Slika 14: Upit 4

Četvrti kreirani upit vraća sva vozila koja se trenutno ne koriste za određenu lokaciju. U zaglavlju rezultata upita se nalaze naziv grada, registracijska oznaka vozila, kao i naziv proizvođača i modela vozila. Upit sadrži nekoliko podupita: dohvaćanje naziva grada iz tablice Lokacija, dohvaćanje naziva modela iz tablice Model. S obzirom da se iz tablice najam određuje koristi li se neko vozilo ili ne, potrebno je napraviti razliku s obzirom na atribut datum završetka najma.

5. Implementacija i primjeri korištenja

U ovom su poglavlju prikazane korištene forme za izradu aplikacije. Nadalje, prikazana je jedna implementacija kompletne forme (u ovom slučaju za formu "Najam" jer je ona najkompleksnija), a ostale forme su izrađene na sličan način kao forma "Najam" pa stoga nije potrebno sve ostale opisivati. Forma "Najam" sadrži gumbiće (Kreiraj, Obriši, Ažuriraj, Prikaži) zatim `textbox`-ove za upis podataka te `DataGridView` za prikaz podataka. U nastavku je prikazana klasa `Najam.cs`:

Prije svega, u linijama 1-7, učitavaju se potrebne biblioteke za rad sa PostgreSQL bazom podataka i grafičkim korisničkim sučeljem.

Nakon toga, u funkciji `frmNajam_Load` se otvara konekcija sa bazom podataka i izvršava se SQL naredba za dohvaćanje svih podataka iz tabele "najam". Rezultati se prikazuju u `DataGridView` kontroli `dgvNajmovi`.

U *event handler*-u `btnKreiraj_Click` se otvara konekcija sa bazom podataka i izvršava se SQL naredba za unošenje novog reda u tabelu "najam". Vrijednosti za svako polje se preuzimaju iz odgovarajućih tekstualnih polja. Ako naredba uspije, zatvara se konekcija i poziva se funkcija za prikaz podataka iz tabele najam u `DataGridView` kontroli. U slučaju bilo kakve greške, ista se prikazuje u `MessageBox`-u. Ako se konekcija uspješno otvori, obavezno se zatvara na kraju programskog bloka, a ako je došlo do ikakvih grešaka, tada u `finally` bloku.

U *event handler*-u `btnObriši_Click` se otvara konekcija sa bazom podataka i izvršava se SQL naredba za brisanje reda iz tabele "najam". Vrijednost za primarni ključ se preuzima iz tekstualnog polja `tbIDnajma`. Ako naredba uspije, zatvara se konekcija i poziva se funkcija za prikaz podataka iz tabele "najam" u `DataGridView` kontroli. U slučaju bilo kakve greške, ista se prikazuje u `MessageBox`-u. Ako se konekcija uspješno otvori, obavezno se zatvara na kraju programskog bloka, a ako je došlo do ikakvih grešaka, tada u `finally` bloku.

U *event handler*-u `btnAzuriraj_Click` se otvara konekcija sa bazom podataka i izvršava se SQL naredba za ažuriranje postojećeg reda u tabeli najam. Vrijednosti za svako polje se preuzimaju iz odgovarajućih tekstualnih polja. Ako naredba uspije, zatvara se konekcija i poziva se funkcija za prikaz podataka.

Funkcijom `prikazi_podatke` dohvaćaju se podatci iz baze `SELECT` upitom te ih se prikazuje u `DataGridView`-u. Ovu funkciju se poziva u *event handler*-ima `frmNajam_Load` i `btnPrikazi_Click`.

Forma "Najam" i njeni elementi se mogu pogledati u sekciji 5.0.6.

```

using Npgsql;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Common;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace RentACar
{
    public partial class frmNajam : Form
    {
        NpgsqlConnection con = new NpgsqlConnection("Server=localhost;Port=5432;Database=postgres;Username=postgres;Password=1234;");

        public frmNajam()
        {
            InitializeComponent();
        }

        public void prikazi_podatke()
        {
            con.Open();
            NpgsqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "select \IDnajam\ as \Šifra najma\", \IDkljent\ as \Šifra klijenta\", \IDvozilo\ as \Šifra vozila\", datum pocetkanajma as \Datum početka najma\", datumzavrsetkanajma as \Datum završetka najma\", idlokacijapreuzimanja as \Šifra lokacije preuzimanja\", idlokacijaostavljanja as \Šifra lokacije ostavljanja\", napomena as \Napomena\", vrijediod as \Vrijedi od\", vrijedido as \Vrijedi do\" from najam;";
            cmd.ExecuteNonQuery();
            DataTable dt = new DataTable();
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(cmd);
            da.Fill(dt);
            dgvNajmovi.DataSource = dt;
            con.Close();
        }

        private void frmNajam_Load(object sender, EventArgs e)
        {
            prikazi_podatke();
        }

        private void btnKreiraj_Click(object sender, EventArgs e)
        {
            try
            {
                con.Open();
                NpgsqlCommand cmd = con.CreateCommand();
                cmd.CommandType = CommandType.Text;
                cmd.CommandText = "insert into najam values('' + tbIDnajma.Text + '', '' + tbPocetak.Text + '', NULL, '' + tbNapomena.Text + '', '' + tbLokacijaPreuzimanja.Text + '', '' + tbLokacijaOstavljanja.Text + '', '' + tbIDvozila.Text + '', '' + tbIDklijenta.Text + '', NULL, NULL);";
                cmd.ExecuteNonQuery();
                con.Close();
                prikazi_podatke();
                MessageBox.Show("Zapis je unesen u bazu!");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                con.Close();
            }
        }

        private void btnObrisi_Click(object sender, EventArgs e)
        {
            try
            {
                con.Open();
                NpgsqlCommand cmd = con.CreateCommand();
                cmd.CommandType = CommandType.Text;
                cmd.CommandText = "delete from najam where \IDnajam\='' + tbIDnajma.Text + ''";
                cmd.ExecuteNonQuery();
                con.Close();
                prikazi_podatke();
                MessageBox.Show("Zapis je obrisao iz baze!");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                con.Close();
            }
        }

        private void btnAzuriraj_Click(object sender, EventArgs e)
        {
            try
            {
                con.Open();
                NpgsqlCommand cmd = con.CreateCommand();
                cmd.CommandType = CommandType.Text;
                cmd.CommandText = "update najam set datum pocetkanajma='' + tbPocetak.Text + '', datumzavrsetkanajma='' + tbZavrsetak.Text + '', idlokacijapreuzimanja='' + tbLokacijaPreuzimanja.Text + '', idlokacijaostavljanja='' + tbLokacijaOstavljanja.Text + '', \IDkljent\='' + tbIDklijenta.Text + '', \IDvozilo\='' + tbIDvozila.Text + '', napomena='' + tbNapomena.Text + '', vrijediod='' + tbVrijediOd.Text + '', vrijedido='' + tbVrijediDo.Text + '' where \IDnajam\='' + tbIDnajma.Text + ''";
                cmd.ExecuteNonQuery();
                con.Close();
                prikazi_podatke();
                MessageBox.Show("Zapis je ažuriran u bazi!");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                con.Close();
            }
        }

        private void btnPrikazi_Click(object sender, EventArgs e)
        {
            prikazi_podatke();
        }
    }
}

```

Slika 15: Klasa Najam.cs forme Najam

5.0.1. Kategorija

The 'Kategorije' form displays a table with two columns: 'ID kategorije' and 'Naziv kategorije'. The first row is selected, showing ID 1 and 'Kategorija 1'. Below the table are input fields for 'Šifra kategorije' and 'Naziv kategorije', and buttons for 'Kreiraj', 'Ažuriraj', 'Obriši', and 'Prikaži'.

ID kategorije	Naziv kategorije
1	Kategorija 1
2	Kategorija 2
3	Kategorija 3
4	Kategorija 4
5	Kategorija 5
6	Kategorija 6
*	

Šifra kategorije:

Naziv kategorije:

Kreiraj Ažuriraj Obriši Prikaži

Slika 16: Forma Kategorija

5.0.2. Klijent

The 'Klijenti' form displays a table with columns: 'Šifra klijenta', 'Ime', 'Prezime', 'Datum rođenja', 'Adresa', 'Grad', 'Poštanski broj', 'Kontakt broj', and 'Broj vozačke dozvole'. The first row is selected, showing ID 1, 'Petar', 'Matic', '24.4.2001.', 'Medimurska 18', 'Varazdin', '42000', '111', and '123456789'. To the left of the table are input fields for 'Šifra klijenta', 'Ime klijenta', 'Prezime klijenta', 'Datum rođenja', 'Adresa', 'Grad', 'Poštanski broj', 'Kontakt broj', and 'Broj vozačke dozvole'. Below these are buttons for 'Kreiraj', 'Ažuriraj', 'Obriši', and 'Prikaži'.

Šifra klijenta	Ime	Prezime	Datum rođenja	Adresa	Grad	Poštanski broj	Kontakt broj	Broj vozačke dozvole
1	Petar	Matic	24.4.2001.	Medimurska 18	Varazdin	42000	111	123456789
2	Ante	Antic	24.4.2002.	Medimurska 17	Zagreb		111222	987654321
3	Ivo	Ivic	24.4.1995.	Medimurska 19	Osijek	12000	222333	12345678
4	Ana	Anic	24.4.1989.	Medimurska 11	Rijeka	23000	333444	876543210
5	Marko	Markovic	24.4.1999.	Medimurska 12	Split		444555	123487654
6	Josip	Josipovic	24.4.2000.	Medimurska 14	Pula	43000	555666	876541234
7	Anto	Antovic	24.4.1997.	Vibaska 21	Dubrovnik	10000	999888	19283281
*								

Šifra klijenta:

Ime klijenta:

Prezime klijenta:

Datum rođenja:

Adresa:

Grad:

Poštanski broj:

Kontakt broj:

Broj vozačke dozvole:

Kreiraj Ažuriraj Obriši Prikaži

Slika 17: Forma Klijent

5.0.3. Lokacija

Šifra lokacije:

Adresa:

Grad:

	Šifra lokacije	Adresa	Grad
▶	1	Vukovarska 12	Zagreb
	2	Avenija Dubrovnik 15	Zagreb
	3	Tresnjevka 16	Zagreb
	4	Ducanska 12	Varazdin
	5	Riva 13	Split
	6	Čakovečka 19	Čakovec
	7	Osječka 17	Osijek
	8	Pulska 20	Pula
*			

Slika 18: Forma Lokacija

5.0.4. Model

Šifra modela:

Naziv modela:

Šifra proizvođača:

	Šifra modela	Naziv modela	Šifra proizvođača	Naziv proizvođača
▶	1	CX-3	1	Mazda
	2	Picasso	3	Citroen
	3	Octavia	2	Skoda
	4	Golf 5	5	VW
	5	RS6	6	Audi
	6	X5	7	BMW
	7	McLaren	8	Mercedes
	8	Cayman GT4	9	Porsche
	9	Phantom	10	Rolls Royce

Slika 19: Forma Model

5.0.5. Način plaćanja

Šifra načina:

Naziv plaćanja:

Opis:

Kreiraj Obrši

Ažuriraj Prikaži

	Šifra načina plaćanja	Naziv plaćanja	Opis plaćanja
▶	1	Gotovina	Na lokaciji.
	2	Gotovina	Postom.
	3	Mobitel	Na lokaciji.
	4	Mobitel	Online.
	5	Kartica	Na lokaciji.
	6	Kartica	Online.
*			

Slika 20: Forma Način plaćanja

5.0.6. Najam

Šifra najma:

Šifra klijenta:

Šifra vozila:

Početak najma:

Završetak najma:

Lokacija preuzimanja:

Lokacija ostavljanja:

Napomena:

Najam vrijedi od:

Najam vrijedi do:

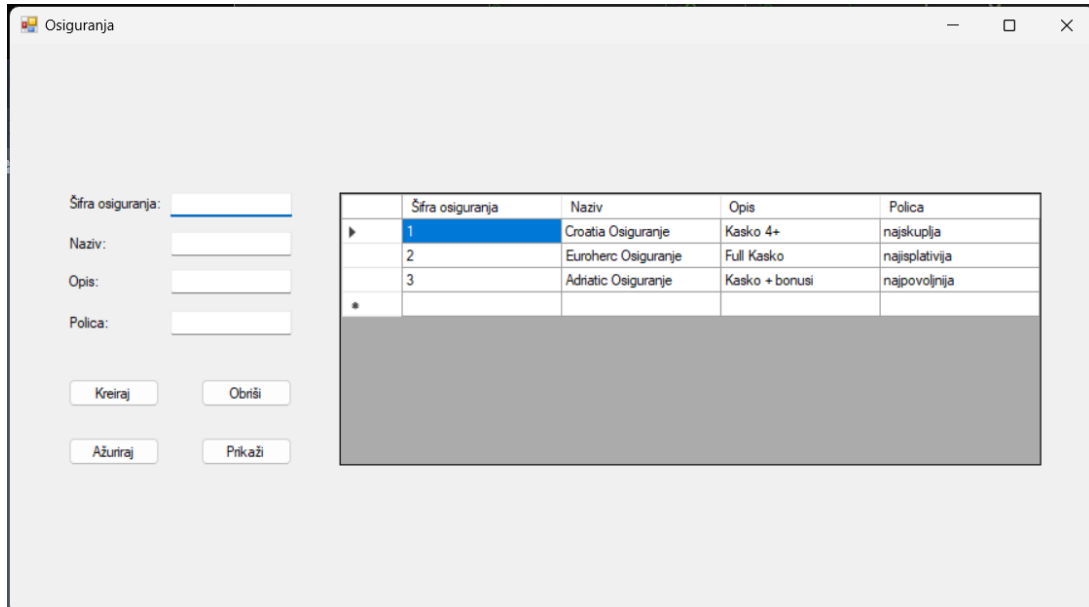
Kreiraj Obrši

Ažuriraj Prikaži

	Šifra najma	Šifra klijenta	Šifra vozila	Datum početka najma	Datum završetka najma	Šifra lokacije preuzimanja	Šifra lokacije ostavljanja	Napomena	Vrijedi od	Vrijedi do
▶	1	1	1	1.1.2022.	30.1.2022.	1	2	Najam auta n...		
	2	5	3	1.2.2022.	30.4.2022.	2	2	Najam auta n...		
	3	2	6	15.6.2022.	15.8.2022.	4	1	Najam auta n...		
	6	5	3	1.2.2022.	30.4.2022.	2	2	Najam auta n...		
	7	2	6	15.6.2022.	15.8.2022.	4	1	Najam auta n...		
	8	6	7	1.11.2022.	24.12.2022.	6	3	Najam auta d...		
	10	5	3	1.2.2022.	30.4.2022.	2	2	Najam auta n...		
	11	2	6	15.6.2022.	15.8.2022.	4	1	Najam auta n...		
	12	6	7	1.11.2022.	24.12.2022.	6	3	Najam auta d...		
	13	4	8	20.9.2022.	30.9.2022.	3	5	Najam auta n...		
	14	1	1	10.1.2023.	20.1.2023.	2	2	bez	1.1.2023.	30.1.2023.
	15	2	1	28.1.2023.	30.1.2023.	2	2	bez	1.1.2023.	30.1.2023.
	16	3	3	1.9.2022.		2	2	bez		
*										

Slika 21: Forma Najam

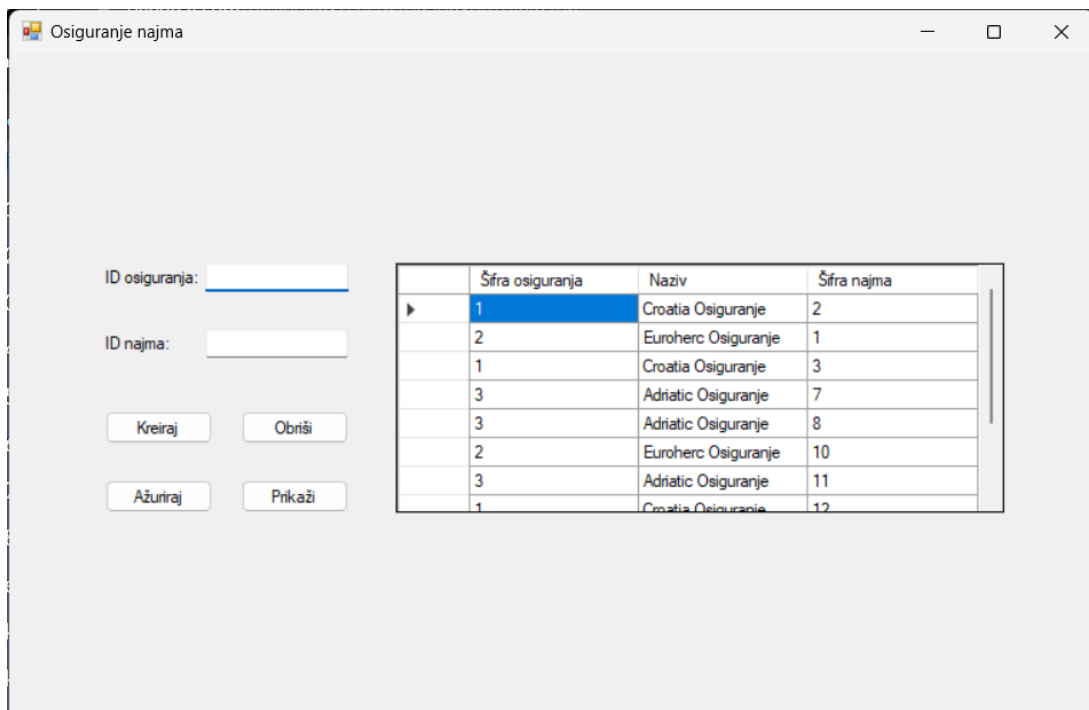
5.0.7. Osiguranje



	Šifra osiguranja	Naziv	Opis	Polica
▶	1	Croatia Osiguranje	Kasko 4+	najskuplja
	2	Euroherc Osiguranje	Full Kasko	najisplativija
	3	Adriatic Osiguranje	Kasko + bonusi	najpovoljnija
*				

Slika 22: Forma Osiguranje

5.0.8. Osiguranje najma



	Šifra osiguranja	Naziv	Šifra najma
▶	1	Croatia Osiguranje	2
	2	Euroherc Osiguranje	1
	1	Croatia Osiguranje	3
	3	Adriatic Osiguranje	7
	3	Adriatic Osiguranje	8
	2	Euroherc Osiguranje	10
	3	Adriatic Osiguranje	11
	1	Croatia Osiguranje	12

Slika 23: Forma Osiguranje najma

5.0.9. Početna stranica

Početna stranica

Rent-a-car App

Lokacija	Kategorija	Proizvođač
Model	Vozilo	Način plaćanja
Račun	Najam	Klijent
Osiguranje	Osiguranje najma	Upiti

Slika 24: Forma Početna stranica

5.0.10. Proizvođač

Proizvođači

Šifra proizvođača:

Naziv:

Kreiraj Obriši

Ažuriraj Prikaži

Šifra proizvođača	Naziv proizvođača
1	Mazda
2	Skoda
3	Citroen
4	Opel
5	VW
6	Audi
7	BMW
8	Mercedes

Slika 25: Forma Proizvođač

5.0.11. Račun

Šifra računa:

Šifra najma:

Datum izdavanja:

Način plaćanja:

Iznos:

Iskorišten kupon:

	Šifra računa	Šifra najma	Šifra načina plaćanja	Datum izdavanja	Iznos računa	Iskorišten kupon
▶	6	7	2	30.9.2022.	30	<input type="checkbox"/>
	3	3	5	15.8.2022.	200	<input type="checkbox"/>
	2	2	4	30.4.2022.	290	<input type="checkbox"/>
	5	6	6	23.12.2022.	170	<input type="checkbox"/>
	1	1	1	30.1.2022.	95	<input type="checkbox"/>
*						<input type="checkbox"/>

Slika 26: Forma Račun

5.0.12. Vozilo

Šifra vozila:

Registracijske oznake:

Šifra modela vozila:

Šifra kategorije:

Šifra lokacije vozila:

Datum proizvodnje:

Kilometraža:

Boja:

Datum teh. pregleda:

	IDvozilo	Registracijski oznake	Datum proizvodnje auta	Kilometraža	Boja	Šifra lokacije vozila	Šifra kategorije	Šifra modela vozila	Datum tehničkog pregleda
▶	1	12345	4.3.2011.	340000	crvena	1	3	1	
	3	98765	30.5.2020.	140000	roza	3	5	3	
	4	56789	28.7.2021.	90000	zelena	4	6	4	
	5	1234	15.1.2016.	230000	zuta	5	4	5	
	7	87654	24.2.2012.	280000	smeđa	7	2	7	
	8	45678	14.9.2014.	200000	narancasta	8	4	8	
	2	54321	26.2.2019.	180000	plava	7	5	2	
	6	43210	13.8.2018.	190000	siva	7	5	6	
*									

Slika 27: Forma Vozilo

6. Zaključak

Korišteni alati u ovom projektu su poprilično dobri za razvoj jednog ovakvog sustava. Prije svega, DataGrip je odličan alat koji omogućuje "profinjeno" kreiranje tablica, okidača, *view*-ova. Izrada samog modela preko dijagrama je nešto teža, prije svega spajanje tablica vezama te kardinalnosti, ali sve u svemu, kada se smisli dobar konceptualni model sustava, tada izrada samog modela postaje linearan problem, točnije stvari se rješavaju jedna za drugom (slijedno).

Izrada same aplikacije u Visual Studiu također nije preteška. Sve *Windows* forme se manje-više "crtaju", gumbići, *label*-i, *textbox*-ovi se *drag and drop*-aju, zatim *DataGridView*-ovi, koji su jako korisni, olakšavaju posao za tablični prikaz podataka, dok spajanje na bazu preko Npgsql providera također nije bio težak izazov.

Sama poslovna logika koja se temelji na razini baze preko aktivnih i temporalnih vrsta podataka, izrađena je preko funkcija i okidača. Izrada takvih vrsta baza podataka je bila nešto složenija i kompleksnija i tražila je poprilično jedan veći vremenski period u odnosu na prethodne "tehničke" izazove.

Navedene tehnologije za rad na ovakvom projektu su izabrane iz razloga zato što su zanimljive samom autoru ovog projekta, a i samim tim popularne u svijetu *IT*-a te *development*-a, točnije, naširoko su korištene, primjerice C# koji je univerzalan i za sami *frontend* i *backend*, dok je PostgreSQL baza podataka korištena u *large-scale* sustavima i tvrtkama.

Popis literature

- [1] M. Maleković i M. Schatten, „Teorija i primjena baza podataka,” *Fakultet organizacije i informatike, Varaždin*, 2017.
- [2] PostgreSQL, „PostgreSQL,” *Web resource: <http://www.PostgreSQL.org/about>*, 1996.
- [3] J. Templeman i D. Vitter, *Visual Studio: The .NET Framework Black Book*. Coriolis, 2002.
- [4] Npgsql. „Npgsql, dotnet access to postgresql,” Npgsql. (2022.), adresa: <https://www.npgsql.org/%5C#about> (pogledano 7. 1. 2023.).
- [5] DataGrip. „Datagrip,” DataGrip. (2022.), adresa: <https://www.jetbrains.com/help/datagrip/meet-the-product.html> (pogledano 8. 1. 2023.).

Popis slika

1.	ERA model	8
2.	Okidač 1	9
3.	Okidač 2	10
4.	Okidač 3	11
5.	Okidač 4	12
6.	Okidač 5	13
7.	Okidač 6	14
8.	Okidač 7	15
9.	Okidač 8	16
10.	Windows forma upita	17
11.	Upit 1	18
12.	Upit 2	18
13.	Upit 3	19
14.	Upit 4	20
15.	Klasa Najam.cs forme Najam	22
16.	Forma Kategorija	23
17.	Forma Klijent	23
18.	Forma Lokacija	24
19.	Forma Model	24
20.	Forma Način plaćanja	25
21.	Forma Najam	25
22.	Forma Osiguranje	26
23.	Forma Osiguranje najma	26
24.	Forma Početna stranica	27

25.	Forma Proizvođač	27
26.	Forma Račun	28
27.	Forma Vozilo	28

Prilozi

1. GitHub

Poveznica na repozitorij sustava za verzioniranje. (Potrebno je kliknuti na ovaj tekst.)