



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Ивана Маркановић, ЕЗ 46/2014

PROJEKTOVANJE BAZE PODATAKA AUTOPREVOZNIKA

ПРОЈЕКАТ

- Примењено софтверско инжењерство (ОАС) -

Нови Сад, 25.05.2021.

SADRŽAJ

OPIS REŠAVANOG PROBLEMA	3
OPIS KORIŠĆENIH TEHNOLOGIJA I ALATA	4
OPIS REŠENJA PROBLEMA	5
KORISNIČKI INTERFEJS	5
BAZA PODATAKA	7
PREDLOZI ZA DALJA USAVRŠAVANJA	12
LITERATURA	13

OPIS REŠAVANOG PROBLEMA

Projekat predstavlja primer projektovanja baze podataka za kompaniju autoprevoznika gradskog prevoza. Realizovan projekat predstavlja desktopnu aplikaciju gde pomoću korisničkog interfejsa korisnik manipuliše projektovanom bazom podataka.

Autoprevoznik predstavlja preduzeće čija je pretežna delatnost gradski i prigradski kopneni prevoz putnika. On vrši aktivnosti prevoza putnika autobusima, već određenim linijama, sa ukrcavanjem i iskrcavanjem putnika na određenim već utvrđenim autobuskim stajalištima.

Kao svaka kompanija, Autoprevoznik ima radnike. Razlikujemo sledeće:

- Kondukter, radnik od koga putnik kupuje kartu
- Kontroler, radnik koji proverava kupljenu kartu
- Vozač, radnik čija je obaveza da vozi autobuse sa putnicima po definisanim linijama.

Putnik kupuje kartu u autobusu, kao što smo već rekli, od konduktera za jednu vožnju, za datu liniju. Kupljena karta sadrži sva odgovarajuća svojstva, potrebna radi njene identifikacije i određivanja njene pripadnosti odgovarajućem autobusu.

Linija predstavlja putanju kojom se autobus kreće i prolazi kroz određena naselja. U naseljima se mogu nalaziti garaže radi čuvanja autobusa koji se trenutno nisu u upotrebi. Svaka linija ima jasno određene stanice na kojima putnici mogu ući ili sići sa autobusa.

Svaki od entiteta u bazi podataka Autoprevoznik sadrži odgovarajuća polja radi pružanja jedinstvene identifikacije i opširnijih informacija o samom entitetu.

OPIS KORIŠĆENIH TEHNOLOGIJA I ALATA

Razvoj baze podataka počinje sa Oracle SQL Developer Data Modeler koji predstavlja besplatni grafički alat pomoću kojeg je korisnik u mogućnosti da napravi, pregleda i izmeni logičke, relacione, fizičke, multi-dimenzionalne modele podataka. Pomoću njega na grafički i lagan način možemo projektovati ER model podataka i uz pomoć ugrađenih funkcija pretvoriti je u Relacioni model. [1]

SQL Server Management Studio (SSMS) predstavlja integrisano okruženje čiji je primarni zadatak pristup, konfiguracija, upravljanje i razvoj svih komponenti SQL Servera. SSMS pruža ujedinjeni program koji kombinuje grafičke alate sa većim brojem script editora koji pružaju pristup SQL serveru developerima i administratorima bazama podataka. Upotrebljava se za izdavanje upita, projektovanje i rukovođenje bazama podataka. Pomoću njega smo u mogućnosti da pišemo SQL komande i testiramo ih bez potrebe pokretanja aplikacije. [2]

Za bazu podataka, koristili smo ER model podataka i Relacioni model podataka koji smo napravili pomoću Oracle SQL Developer Data Modeler. Model podataka je skup međusobno povezanih podataka. ER model podataka nastoji da grafičkim putem prikaže model realnog sveta na koji se odnosi baza podataka. [3] Relacioni model podataka se bazira na predikatskom računu i teoriji skupova. Osnovna ideja je korišćenje relacija, koja predstavlja skupove podataka u logičkom obliku. Svaka relacija ima oblik tabele. [4]

Samo projektovanje baze podataka je moguće uraditi na više načina. Mogući pristupi projektovanja su: Database first, Code first i Model first. Database first pristup ili pristup projektovanja baze podataka prvo, kao što i naziv kaže, pruža mogućnost projektovanja baze podataka prvo i njego interesiranje u program ili mogućnost integrisanja već postojeće baze podataka u program. Code first je pristup pisanja klasa podataka u kodu aplikacije prvo i kasnije generisanje baze podataka na osnovu njih. Poslednji pristup je Model first koji podrazumeva kreiranje modela podataka pomoću dizajnera i generisanje klase podataka na osnovu samog modela. U projektu je korišćen Database first pristup. [5]

Upotreba SQL baze podataka podrazumeva i upotrebu SQL komandi (tj. trigera), uskladištenih procedura, indeksa i funkcija. Uskladištene procedure predstavljaju pripremljen SQL kod koji možemo sačuvati i upotrebiti ga iznova po potrebi. Trigeri su specijalni tip procedure koji se automatski izvršava u slučaju određenog događaja u bazi podataka, tj. u slučaju izmene podataka u bazi podataka. Indeksi su elementi SQL baze koji omogućavaju brže prolazanje i pristup podacima, kao i reorganizaciju podataka. Funkcije, slično uskladištenim procedurama, predstavljaju niz uređenih SQL komandi.

Aplikacija je razvijena kao Windows Presentation Format (WPF) aplikacija. WPF je UI framework koji kreira desktop korisničke aplikacije i deo je .NET frameworka. [6] .NET je besplatna, kros-platforma i open source platforma za razvoj aplikacija. [7]

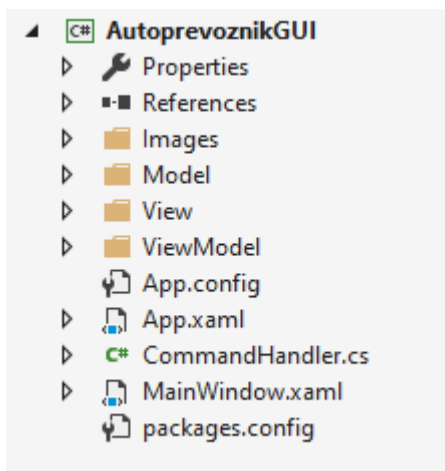
Obrazac korišćen tokom razvoja aplikacije je MVVM, odnosno Model-View-ViewModel.

- Model jednostavno drži podatke u sistemu i ne sadrži logiku rukovanja sa njima. U našem slučaju to nam je baza podataka.
- ViewModel služi kao konekcija između Modela i View. Njegova primarna uloga je snabdnevanje View podacima koji trebaju da se prikažu i pružanje interakcije sa podacima.
- View ima ulogu prikazivanja formatiranih podataka i delegiranja podataka dalje ka modelu. [8]

OPIS REŠENJA PROBLEMA

KORISNIČKI INTERFEJS

Aplikacija je implementirana kao WPF aplikacija i koristi MVVM obrazac. (Slika 1)

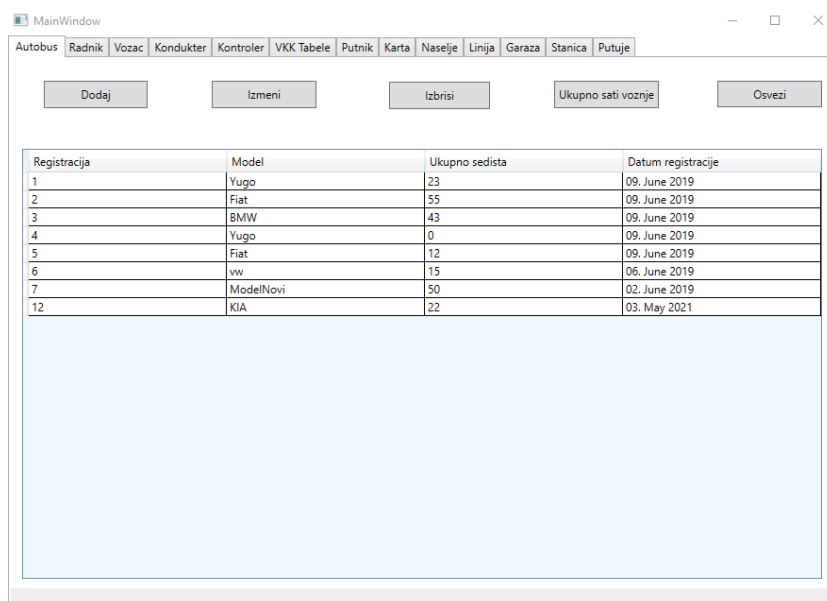


Slika 1. Struktura aplikacije

Pokretanjem aplikacije prikaže nam se početni ekran aplikacije (Slika 2) koji se sastoji iz tabova za biranje prikaza odgovarajućih tabela baze podataka. Svaki od tabova pokazuje tabelu sa vrednostima odgovarajuće tabele iz baze podataka, zajedno sa dugmadima za izvršenje mogućih funkcija nad tom tabelom i status barom na dnu ekrana. Standardne funkcije nad tabelama su:

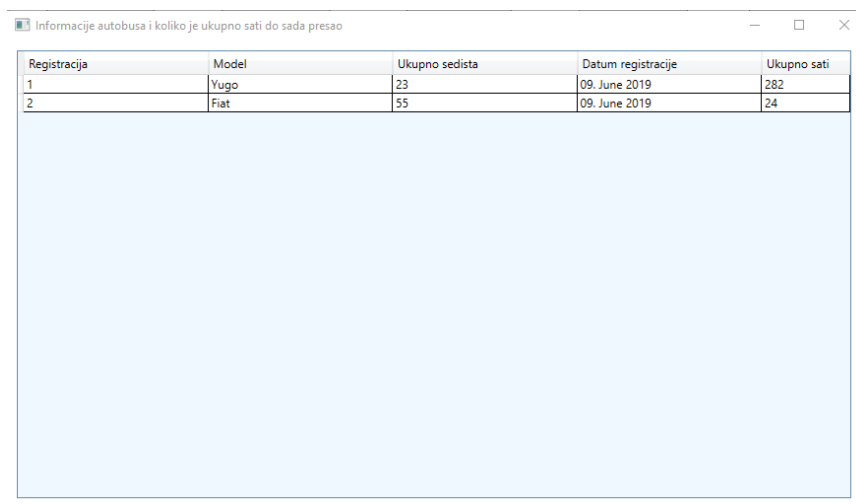
- Dodavanje novog entiteta u tabelu
- Izmena dodatog entiteta
- Brisanje entiteta iz tabele
- Osvežavanje tabele u slučaju promena u bazi podataka.
- Dodatne funkcije koje su karakteristične za pojedine tabele.

Npr. kod aplikacije prikazanoj na slici (Slika 2) vidimo ekran koji odgovara tabeli za Autobus i njene moguće funkcije. Dodatna funkcionalnost jeste prikazivanje ukupnog broja sati vožnje koji su provozali pojedini autobusi, koja u ovom slučaju predstavlja proceduru P_Autobus_sati (Slika 8)



Slika 2. Početni ekran aplikacije

Rezultati dodatnih funkcija će biti prikazane ili u otvorenom novom prozoru (Slika 3) ili u status baru na dnu ekrana. (Slika 4) Dat je primer rezultata procedure P_Naselje, koje je prikazano u status baru (Slika 4)



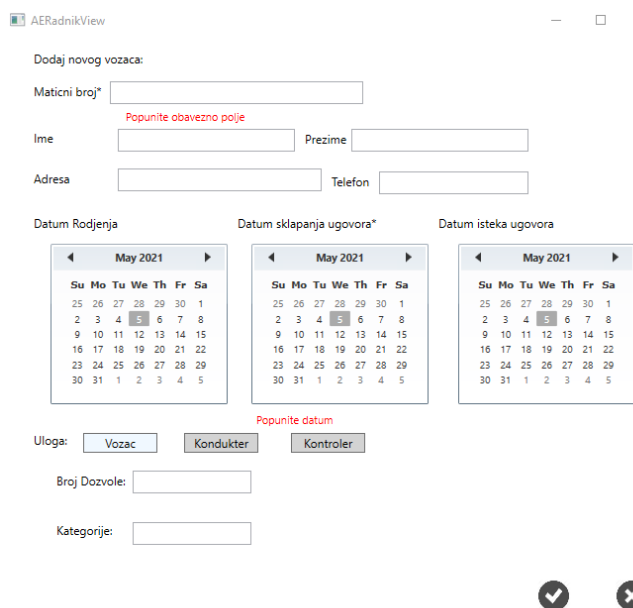
Registracija	Model	Ukupno sedista	Datum registracije	Ukupno sati
1	Yugo	23	09. June 2019	282
2	Fiat	55	09. June 2019	24

Slika 3. Prikaz nakon pritiska na Ukupan broj sati dugme i rezultat

INFO: Najposećenije mesto je Aviv, posećeno 3 puta.

Slika 4. Prikaz rezultata funkcije Max broj prolaska u status baru.

Pritiskom na dugme Dodaj, otvara se novi ekran koji prikuplja potrebne informacije o dodavanju određenog entiteta i prikazuje odgovarajuće poruke u slučaju greške (Slika 5)



Dodaj novog vozača:

Maticni broj*

Popunite obavezno polje

Ime Prezime

Adresa Telefon

Datum Rodjenja Datum sklapanja ugovora* Datum isteka ugovora

Popunite datum

Uloga: ☐ Vozač ☐ Kondukter ☐ Kontroler

Broj Dozvole:

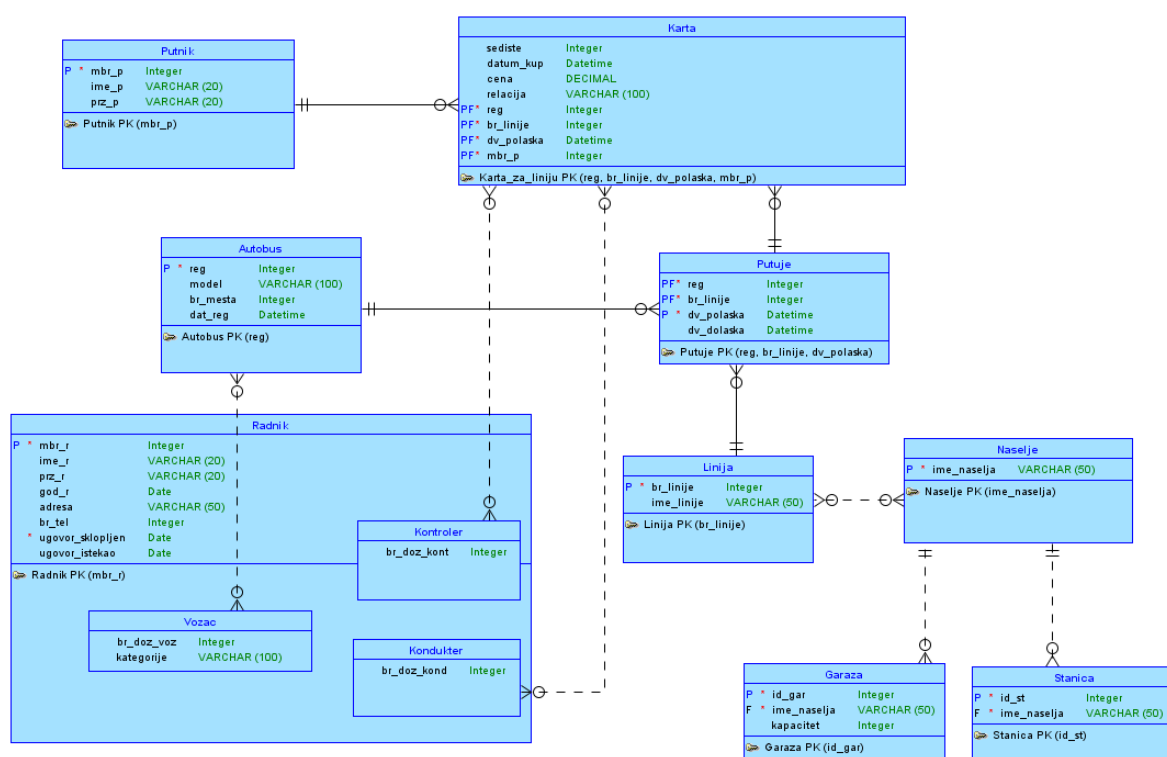
Kategorije:

Slika 5. Dodavanje novog Radnika

Funkcija izmene entiteta sličan je funkciji za dodavanje novog entiteta. Otvara se isti ekran, jedina razlika je da su sva polja već popunjena sa odgovarajućim vrednostima entiteta koji želimo da promenimo a prethodno smo ga selektovali iz tabele. Akcija brisanja entiteta vrši se selektovanjem odgovarajućeg entiteta iz tabele i pritiskom na dugme Izbrisi.

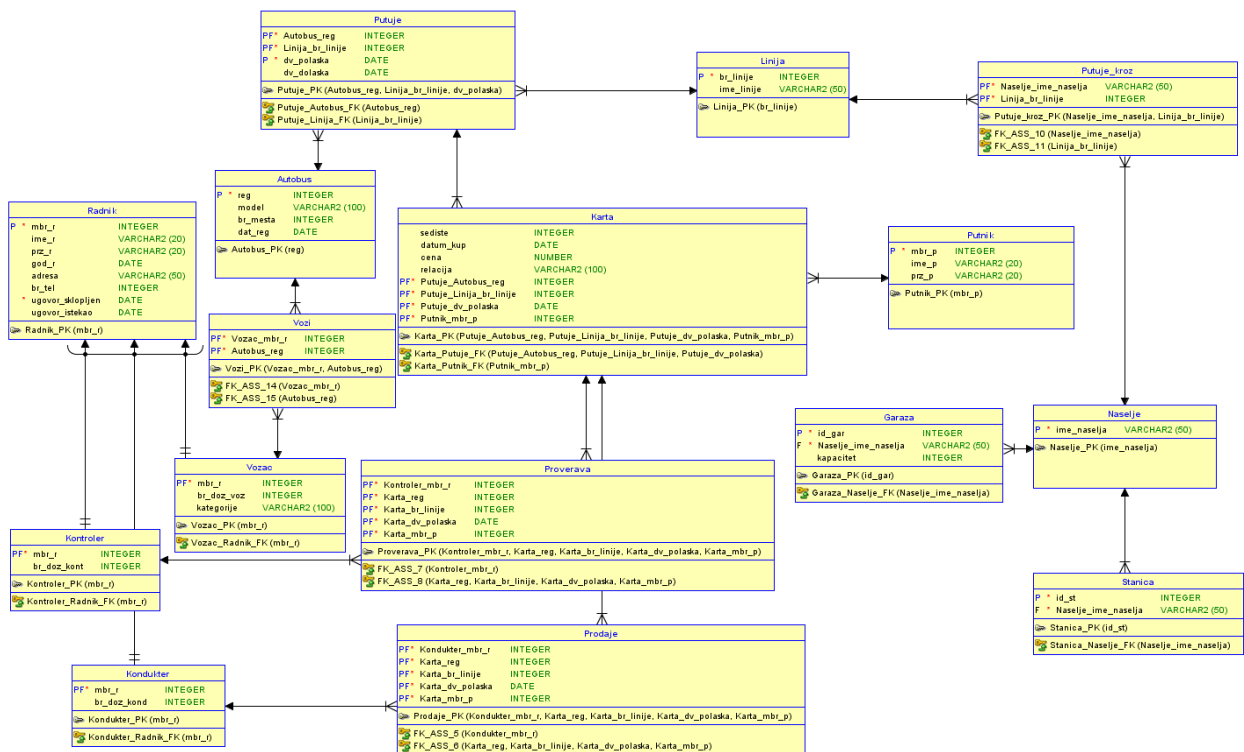
BAZA PODATAKA

Projektovanje baze podataka započinje se sa pisanjem specifikacije sistema. Specifikacija sistema je dokument tekstualnog oblika, koja iskazuje sve potrebne informacije o samom projektovanom sistemu kao i njegove funkcionalnosti. Pomoću nje, pravi se ER model podataka. (Slika 6)



Slika 6. ER model podataka

Prikazani ER model podataka smo napravili u Oracle SQL Developer Data Modeler i uz pomoć koga smo korišćenjem funkcije generisanja Relacionog modela podataka (Slika 7) dobili relacioni model sa slike.



Slika 7. Relacioni model podataka

Pisanje SQL procedura i funkcija je moguće u okviru samog Visual Studija ili u okviru Microsoft SQL Server Management Studiju (SSMS). SSMS olakšava pisanje i testiranja SQL komandi, bez potrebe pokretanja aplikacije radi testiranja samih komandi.

U okviru projekta imamo SQL procedure, funkcije, indeks i trigere. SQL procedure koje su nam od interesa su:

- P_Autobus_Sati – vraća informaciju o autobusu i koliko sati je on proveo ukupno u vožnji. Procedura se poziva pritiskom dugmeta Ukupno sati vožnje kod taba za Autobus. U nastavku imamo priloženu sliku njegovog koda (Slika 8) i njegovo pozivanje iz aplikacije (Slika 9).
- P_Naselje – prolazi kroz tabelu Naselje, računa za svako naselje koliko je puta putnik sa kartom prosao kroz njega i računa maksimalnu vrednost. Procedura, sa kodom (Slika 10), se poziva dugmetom iz taba Naselje i rezultat se prikazuje u status baru na dnu ekrana. (Slika 11)

```

CREATE PROCEDURE [dbo].[P_Autobus_sati]
AS
begin
    --Vraca informacije o autobusu i koliko sati je on ukupno proveo u voznji
    select * from(select p.Autobus_reg, hours = Sum(case when p.dv_polaska < CURRENT_TIMESTAMP and p.dv_dolaska < CURRENT_TIMESTAMP then DATEDIFF(HOUR, dv_polaska, dv_dolaska)
    when p.dv_polaska < CURRENT_TIMESTAMP and p.dv_dolaska > CURRENT_TIMESTAMP then DATEDIFF(hour, dv_polaska, CURRENT_TIMESTAMP) end) from Putuje p
    where dv_dolaska is not null group by Autobus_reg)p join Autobus a
    on p.Autobus_reg = a.reg
end
  
```

Slika 8. Prikaz koda procedure P_Autobus_Sati

Pozivanje procedure se obavlja iz koda, iz odgovarajućeg ViewModela, sa funkcijom prikazanom donjom slikom. (Slika 9)

```
var data = db.Database.SqlQuery<ResultAutobus>(@"exec dbo.P_Autobus_sati");
```

Slika 9. Pozivanje procedure P_Autobus_sati iz aplikacije


```

CREATE PROCEDURE [dbo].[P_Naselje]
(
    @returnValue varchar(100) output
)
AS
--Prolazi kroz tabelu Naselje, racuna za svaki koliko je puta je putnik sa kartom prosao kroz njega, i racuna maximum
declare @id_naselja varchar(50);
DECLARE @passes INT = 0;
DECLARE @NaseljeMax VARCHAR(50);
DECLARE @PassesMax INT = 0;
declare @lastNaselje varchar(50);

declare Naselje_cursor cursor
for
select ime_naselja from Naselje

open Naselje_cursor

while @@FETCH_STATUS = 0
BEGIN
    FETCH NEXT FROM Naselje_cursor INTO @id_naselja;

    IF(@passes > @PassesMax)
    BEGIN
        SET @PassesMax = @passes;
        set @NaseljeMax = @lastNaselje;
    end

    SET @passes = dbo.GetPasses(@id_naselja);
    set @lastNaselje = @id_naselja;

end

close Naselje_cursor;
deallocate Naselje_cursor;

--SELECT p.Naselje_ime_naselja, p.passed FROM(SELECT TOP 1 Naselje_ime_naselja,passed = COUNT(Naselje_ime_naselja) FROM Putuje_kroz ok
--right join Karta k on ok.Linija_br_linije = k.Putuje_Linija_br_linije group by Naselje_ime_naselja order by passed desc) p
set @returnValue = @NaseljeMax + ',' + cast(@PassesMax as varchar(10));

RETURN 0;

```

Slika 10. Prikaz koda procedure P_Naselje

```

var data = db.Database.SqlQuery<string>(@"declare @ReturnValue varchar(100);
exec dbo.P_Naselje @ReturnValue output
select @ReturnValue");

```

Slika 11. Pozivanje procedure P_Naselje iz aplikacije

Korisnički definisane SQL funkcije u projektu su nam potrebne radi dobijanja određenih vrednosti iz baze podataka. Koristimo radi izračunavanja određenih informacija iz sistema. Korišćene funkcije su nam

- GetPasses – izračunava koliko je putnika sa kupljenom kartom proslo kroz naselje. Funkciju koristimo u okviru procedure P_Naselje (Slika 10). Kod nam je priložen u nastavku (Slika 12).
- GetSeat – vraća sledeće sedište na redu za kupovinu u datom autobusu i pokreće se pri dodavanju (kupovini) nove karte. (Slika 13). Data funkcija se poziva iz aplikacije na sledeći način prikazan slikom (Slika 14).

```

CREATE FUNCTION [dbo].[GetPasses]
(
    @ime VARCHAR(50)
)
RETURNS INT
AS
BEGIN
    --Izracunava koliko putnika sa kupljenom kartom proslo kroz prosledjeno naselje
    DECLARE @numPassed INT = 0;
    SET @numPassed = (select COUNT(*) FROM Putuje_kroz pk RIGHT JOIN Karta k ON pk.Linija_br_linije = k.Putuje_Linija_br_linije WHERE pk.Naselje_ime_naselja = @ime);

    RETURN @numPassed
END

```

Slika 12. Prikaz koda funkcije GetPasses

```

CREATE FUNCTION [dbo].[GetSeat]
(
    @reg int,
    @br_linije int,
    @dvPolaska datetime
)
RETURNS INT
AS
BEGIN
    --Izracunava sledece sediste na redu za kupovinu u Autobusu
    declare @nextFreeSeat int = 0;
    declare @br_mesta int;
    declare @seatsBought int;

    set @br_mesta = (select br_mesta from Autobus where reg = @reg);

    if(@br_mesta is not null)
    begin
        set @seatsBought = (select bought = count(*) from Karta where Putuje_Autobus_reg = @reg and Putuje_Linija_br_linije = @br_linije and Putuje_dv_polaska = @dvPolaska);
        if(@seatsBought is not null)
        begin
            if(@seatsBought < @br_mesta)
            set @nextFreeSeat = @seatsBought + 1;
            else
            set @nextFreeSeat = 0;
        end
    end
    else
    set @nextFreeSeat = 0;

    RETURN @nextFreeSeat
END

```

Slika 13. Prikaz koda funkcije GetSeat

```

var data = db.Database.SqlQuery<int>(@"select dbo.GetSeat(@reg,@br_linije,@dvPolaska)", paramReg, paramLinija, paramDVPolaska);

```

Slika 14. Pozivanje funkcije GetSeat iz aplikacije

Indeksi su elementi SQL baze koji omogućavaju brže prolazanje i pristup podacima, kao i reorganizaciju podataka. Ne koristimo ih kod tabela koji imaju česte update ili insert operacije, kao ni kod onih tabela koji sadrže veliki broj NULL vrednosti. U okviru projekta, imamo jedan indeks, povezan za tabelu Putuje_Kroz, za kolonu Linija_br_linije (Slika 15). Tabela Putuje_Kroz povezuje tabele Linija i Naselje. Ona nam daje informacije koja linija prolazi kroz koja naselja.

```

CREATE INDEX [LinijaIndex]
ON [dbo].Putuje_Kroz
(Linija_br_linije)

```

Slika 15. Prikaz koda indeksa LinijaIndex

Trigeri kao što smo rekli su specijalni tip uskladištenih procedura koje se automatski izvršavaju u slučaju određenog događaja u bazi podataka. Događaji koji utiču da okidanje trigeru u projektu su događaji kreiranja novog entiteta. U projektu imamo sledeće trigere

- LinijaTrigger – pokreće se nakon kreiranja novog entiteta u tabeli Linija i vrši dodeljivanje datoj liniji onaj autobus koji je imao najmanje sati vožnje (Slika 16)
- VozacTrigger – izvršava se nad tabelom Vozac i pokreće se nakon kreiranja novog Radnika tipa Vozac. Njegova svrha je dodeljivanje vozaču autobus koji nema već dodeljenog vozača. U slučaju nedostatka slobodnih autobusa ispisuje se odgovarajuća poruka (Slika 17)

```

(CREATE TRIGGER [LinijaTrigger]
ON [dbo].[Linija]
AFTER INSERT
AS
    DECLARE @freeAutobus INT;
    DECLARE @br_linije INT;
    DECLARE @minAutobus INT;

    SELECT @br_linije = insertedLinija.br_linije from inserted insertedLinija;

    SELECT TOP 1 @freeAutobus=[reg] FROM Autobus a1
    LEFT JOIN Putuje p1 ON a1.reg = p1.Autobus_reg
    WHERE p1.Autobus_reg IS NULL

    IF(@freeAutobus IS NULL)
    BEGIN
        PRINT('izracunaj koji ima najmanje');

        set @minAutobus = (select p.Autobus_reg from (select top 1 Autobus_reg, hours=Sum(case when p.dv_polaska < CURRENT_TIMESTAMP and p.dv_dolaska < CURRENT_TIMESTAMP then DATEDIFF(HOUR, dv_polaska, dv_dolaska)
        when p.dv_polaska < CURRENT_TIMESTAMP and p.dv_dolaska > CURRENT_TIMESTAMP then DATEDIFF(HOUR, dv_polaska, CURRENT_TIMESTAMP) end) from Putuje p
        where dv_dolaska is not null group by Autobus_reg order by hours asc) p)

        IF(@minAutobus IS NULL)
        PRINT('nema autobusa');
        else
        INSERT INTO putuje values(@minAutobus,@br_linije,CURRENT_TIMESTAMP,NULL);
    END
    else
    INSERT INTO Putuje VALUES (@freeAutobus, @br_linije, CURRENT_TIMESTAMP,NULL);

```

Slika 16. Prikaz koda trigeru LinijaTrigger

```

CREATE TRIGGER [VozacTrigger]
ON [dbo].[Vozac]
AFTER INSERT
AS
    DECLARE @mbr_r int;
    DECLARE @reg INT;

    SELECT @mbr_r = insertedVozac.mbr_r FROM inserted insertedVozac;

    SELECT TOP 1 @reg=[reg] FROM Autobus a1
    LEFT JOIN Vozi v1 ON a1.reg = v1.Autobus_reg
    WHERE v1.Autobus_reg IS NULL

    IF(@reg is not NULL)
        INSERT INTO Vozi(Vozac_mbr_r,Autobus_reg) VALUES(@mbr_r,@reg)
    ELSE
        PRINT ('Nema slobodnih autobusa');

```

Slika 17. Prikaz koda trigera VozacTrigger

Primenu trigera VozacTrigger možemo videti na sledećem primeru. Radi primera, dodajemo novi Autobus i novog radnika tipa Vozac (Tabela 1). Nakon dodavanja novog Autobusa (Tabela 2) i uskoro posle njega i novog vozača, u našem slučaju poslednje vrednosti u tabelama, triger nad tabelom Vozac (Tabela 3) se okida i dodaje u tabelu Vozi (Tabela 4) novi entitet. Tabela Vozi predstavlja vezu između entiteta Autobus i Vozac i prikazuje koji vozači trenutno voze koje autobuse, kao što vidimo u tabeli. Novi entitet koji je dodan u tabelu Vozi predstavlja rezultat trigera VozacTrigger.

mbr_r	ime_r	prz_r	god_r	adresa	br_tel	ugovor_sklopljen	ugovor_istekao
1	Ime	Imenovic	2019-06-12	Adresa 2	1234232	2019-06-09	2019-06-22
2	Danilo	Danilovic	2019-06-09	Adresa 4	523452345	2019-06-09	2019-06-27
3	Marko	Markovic	2019-06-09	Adresa 221	234234	2019-06-09	NULL
4	Daca	Imenovic	2019-06-09	Adresa 666	64567	2019-06-09	NULL
5	Ime	Dacovic	2019-06-09	Adresa 112	1123123	2019-06-09	NULL
6	Andjela	Andjelovic	2019-06-09	Adresa 112	1123123	2019-06-09	NULL
7	Nikola	Nikolic	2019-06-09	Adresa 112	1123123	2019-06-09	NULL
8	Momir	Momirovic	2019-06-09	Adresa 112	1123123	2019-06-09	NULL
9	Nebojsa	Nebojsic	2019-06-09	Adresa 112	1123123	2019-06-09	NULL
10	Spasoje	Spasojevic	2019-06-09	Adresa 112	1123123	2019-06-09	NULL
11	Jovana	Jovanovic	2019-06-09	Adresa 112	1123123	2019-06-09	NULL
12	Milica	Milicic	2019-06-09	Adresa 112	1123123	2019-06-09	NULL
13	Sava	Savic	2019-06-09	Adresa 112	1123123	2019-06-09	NULL
14	Vozac	Novvi	2019-06-02	Adresa 2	5555	2019-06-11	2019-06-20
2213	Radnik	Radnikovic	2021-04-04	asdf	123123	2021-05-05	2021-05-06
33465	Marko	Maric	1969-07-09	Liman 1	123456	2019-02-07	2021-06-18

Tabela 1. Radnik

reg	model	br_mesta	dat_reg
1	Yugo	23	2019-06-09
2	Fiat	55	2019-06-09
3	BMW	43	2019-06-09
4	Yugo	0	2019-06-09
5	Fiat	12	2019-06-09
6	vw	15	2019-06-06
7	ModelNovi	50	2019-06-02
12	KIA	22	2021-05-03
87	Mercedes	20	2021-03-07

Tabela 2. Autobus

mbr_r	br_doiz_voz	kategorije
1	1	ABC
2	2	A1 A2
3	3	DE
4	4	B2 B3
5	5	F
6	5	F
7	5	F
14	5556	A1B1
2213	1234	A
33465	1234567	A3

Tabela 3. Vozac

Vozac_mbr_r	Autobus_reg
1	1
2	2
3	3
4	4
5	5
7	12
14	6
2213	7
33465	87

Tabela 4. Vozi

PREDLOZI ZA DALJA USAVRŠAVANJA

Prva pozitivna i samim tim i negativna strana projekta jeste sam način projektovanja i implementacije baze podataka Autoprevoznik. Projekat je urađen na jednostavan način, jer mu je primarna funkcija bila u edukativne svrhe. Sam izgled aplikacije je minimalistički i nema potrebe za opširnijem tumačenjem korisničkog interfejsa. Samim minimalističnim izgledom gubi se mogućnost dodatnih prikaza informacije korisniku, radi opširnijeg shvatanja sistema. Dodatno unaprađenje podrazumeva ulepšavanje i proširenje korisničkog interfejsa dodatnim prikazima.

Aplikacija je urađena iz perspektive kompanije, koja upravlja samom bazom podataka. Pozitivan dodatak bi podrazumevao proširenje aplikacije za rad iz perspektive samog putnika i radnika kompanije. Proširenje za radnike bilo bi posebno za svaku vrstu radnika.

Funkcionalnosti sistema su osnovne, proširenjem skupa funkcionalnosti bi se pružilo lepše iskustvo pri korišćenju aplikacije. Proširenje skupa funkcionalnosti bi podrazumevalo uvođenje mehanizma pretrage. Radi realizacije proširenja skupa funkcionalnosti potrebna je implementacija dodatnih SQL procedura. Primer proširenja jeste mogućnost odabira stanice ulaska i silaska sa autobusa, u datom naselju, prilikom kupovine karte.

Sam dodatak i proširenje projekta ne bi narušilo trenutnu funkcionalnost sistema, jedino bi usavršilo već postojeći projekat, poboljšalo bi korisničko iskustvo i pružilo bi mogućnost boljeg razumevanja sistema.

LITERATURA

- [1] *Oracle SQL Developer Data Modeler*,
<https://www.oracle.com/database/technologies/appdev/datamodeler.html>
- [2] *SQL Server Management Studio*,
<https://docs.microsoft.com/sr-latn-rs/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15>
- [3] *ER Model*, https://sr.wikipedia.org/wiki/Модел_објекту-везе
- [4] *Relacioni model*, https://sh.wikipedia.org/wiki/Relacioni_model
- [5] *Database First*, <https://docs.microsoft.com/en-us/ef/ef6/modeling/designer/workflows/database-first>
- [6] *WPF*, <https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2019>
- [7] *DotNet*, <https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet>
- [8] *MVVM*, https://www.tutorialspoint.com/mvvm/mvvm_introduction.htm