SQL Server 2017

Sabo, Viktorija

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Viktorija Sabo

SQL SERVER 2017

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Viktorija Sabo

Matični broj: 44084/15–R

Studij: Informacijski sustavi

SQL SERVER 2017

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Kornelije Rabuzin

Varaždin, rujan 2019.

Viktorija Sabo

Izjava o izvornosti

Izjavljujem da je moj završni rad 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/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

U radu će biti opisane verzije SQL Server-a , naglasak na SQL Server 2017. Spomenut će se glavne karakteristike pojedinačne verzije te promijene s obzirom na prijašnju verziju. Nakon toga biti će riječi o sustavu za upravljanje podataka (SUBP) i SQL jeziku koji služi za manipulaciju podataka. Nakon opisa navedenih alata dalje u radu biti će opisana aplikacijska domena, model baze podataka koja je implementirana u Microsoft SQL Server-u te tehnologije koje će se koristit pri izradi aplikacije i na kraju opis funkcionalnosti same aplikacije.

Sadržaj

1. Uvod

U svijetu ne postoji institucija koja ne koristi baze podataka zbog značajnog poboljšanja u poslovanju ustanove. No, što je uopće baza podataka i koja joj je zadaća? Jedna od definicija baze podataka je baza podataka skup međusobno povezanih podataka koji su smješteni u neku vanjsku memoriju računala poput čvrstog diska (*eng. hard disk*) . Zadaća joj je da nam olakša rukovanje s podacima, pomoću nje imamo kontrolu nad podacima i svi su nam na istome mjestu te nema više zapisivanja na brdo papira. Podaci su smješteni na disk u obliku koji nije razumljiv korisnicima te se za rad s tim podacima koriste sustavi za upravljanje bazom podataka [1]. Postoje razni sustavi, nema najboljeg sve ovisi od organizacije, koje uvjete ima i koji bi više odgovarao s obzirom na cilj koji se želi ostvariti. Jedan od poznatijih sustava za upravljanje bazom podataka je MS SQL Server. U radu će biti proučene i navedene mnoge značajke SQL Server 2017 i usporedba s prijašnjim verzijama sustava. Napravit će se primjer baze podataka pomoću SQL Server te aplikacija koja će prikazivati krajnji rezultat rada s podacima. Okruženje u kojem će biti izrađena aplikacija je Microsoft Visual Studio u kojem će se pomoću ASP.NET MVC tehnologije izraditi osnovne funckije aplikacije i jednostavno korisničko sučelje. Aplikacija je namjenjena kako bi ubrzala način rada s podacima gdje će se jednim klikom vrlo lako podaci moći unositi, spremati i ažurirati. Svi potrebni koraci od izrade baze podataka do same aplikacije bit će detaljno objašnjeni u daljnjem kontekstu.

2. Sustav za upravljanje bazom podataka

Sustav za upravljanje bazom podataka je poslužitelj baze podataka. [2] On oblikuje fizički oblik baza prema traženoj logičkoj strukturi, obavlja sve operacije s podacima te se brine za sigurnost podataka i automatizira administrativne poslove s bazama. Također, može podržavati razne baze podataka od kojih svaka ima svoju logičku strukturu u skladu s istim modelom. Model podataka je skup pravila prema kojima se određuje kako će izgledati logička struktura baze podataka. On čini osnovu za oblikovanje i implementaciju baze, odnosno podaci u bazi moraju biti organizirani u skladu s onim modelom koji podržava odabrani sustav za upravljanje bazom podataka (*eng. Database Management System* - DBMS). [2] Sadašnji DBMS-ovi podržavaju više vrsta modela, a to su Relacijski model, Mrežni model, Hijerarhijski model i Objektni model podataka. U ovome radu koristit će se Relacijski model podataka koji je temeljen na matematičkom izrazu *relacija* što znači da će se podaci i veze među podacima pokazivati pomoću tablica koje su sačinjene od redaka i stupaca. Prednosti ovog modela s obzirom na ostale je da se veze mogu uspostaviti iako nisu bile predviđene u fazi modeliranja samim time se povećava fleksibilnost korištenja baze, no mana mu je što se veze svaki put moraju iznova uspostavljati i potrebno je pretraživanje podataka što oduzima vremena. U relacijskom modelu pozornost je usmjerena na dinamički aspekt, što znači više manipuliranje podacima, a manje pohranjivanja. Iz svega nevedenog može se zaključiti da ovakav način rada omogućava slobodno kombiniranje podataka iz više relacija istovremeno te jednostavne operacije sa samo jednom relacijom.

2.1. SQL Server

Glavni proizvod od tvrtke Microsoft za upravljanje bazom podataka je relacijski sustav SQL Server. On je softverski proizvod koji je namjenjen primarno za dohvaćanje i pohranjivanje podataka iz baze sukladno s zahtjevima aplikacije. Da bi dublje razumijeli SQL Server treba biti upoznat sa pojmom SQL. SQL je poseban programski jezik dizajniran za obradu podataka u sustavima za upravljanje relacijskom bazom podataka. Kako bih baza mogla drugim programima ili računalima pružati svoje usluge potreban joj je poslužitelj baze, odnosno računalni program koji omogućava pružanje njezin usluga drugim računalima. On je definiran modelom klijent-poslužitelj. [3] Stoga je SQL Server poslužitelj baze podataka koji implementira strukturirani upitni jezik SQL. Glavna prednost, odnost karakteristika je to što korisnik sam određuje što SQL treba napraviti, ali ne treba poznavati složene aktivnosti koje stoje u pozadini niti kako neka SQL naredba dođe do rezultata. U zadnjih 20-tak godina SQL

Server se jako dobro razvijao i promijenio dobivajući usput nove značajke i funkcionalnosti. Do sada je izdano desetak izdanja SQL Server-a, dalje u radu bit će obrađene neke od njih ali prioritet će biti na SQL Server 2017.

2.2. SQL Server 2008 R2

Nadovezajući se na uspjeh originalnog izdanja SQL Server 2008, SQL Server 2008 R2 je postao tada najnaprednija, pouzdana i skalabilna podatkovna platforma. Postigao je veliki utjecaj na organizacije diljem svijeta svojim novim mogućnostima, a to su osnaživanje krajnih korisnika putem Poslovne inteligencije (*eng. Business intelligence* - BI), jačanjem učinkovitosti i suradnjom između administratora baza podataka (DBA-ova) i to sve kako bi zadovoljio najzahtjevnija podatkovna opterećenja. [4] Jedna od značajki koja je uvedena u SQL Server 2008 je paralelno skladište podataka koje se sastoji od softvera i hardvera dizajniranih da se zadovolje potrebe najvećih skladišta podataka. Ovo riješenje ima mogućnost masovnog skaliranja na stotine terabajta uz korištenje nove tehnologije MPP (*eng. Massively parallel processing*). Paralelno skladište podataka dijeli velike tablice podataka na nekoliko fizičkih čvorova gdje svaki čvor ima svoj procesor (CPU), memoriju i instancu SQL Servera. Ovakvim pristupom raspodjele se uveliko poboljšavaju perfomanse te se eliminira problem s brzinom izvođenja. Integracija SQL Azure omogućila je platformu koja nudi fleksibilno i potpuno rješenje baze podataka implementirano u oblaku (*eng.cloud-base*). To za organizacije znači da ne moraju instalirati, konfigurirati ili se baviti svakodnevnim operacijama upravljanja SQL Servera kako bi podržale njihove potrebe za bazom podataka.

2.3. SQL Server 2014 i 2012.

SQL Server 2014 temelji se na poboljšanju perfomansi i ažuriranju kritičnih opcija koje su isporučene u prijašnjim verzijama. Stoga SQL Server 2014 pruža nove mogućnosti i poboljšanje postojećih, a to su pohrana podataka u radnu memoriju (*eng. In-memory*) za OLTP sustave (*eng. online transaction processing*), skladištenje podataka i poslovna inteligencija za najopsežnija rješenja baza podataka. Razlika između SQL 2012 i 2014 je u povećanju maksimalne veličine baze s 64 GB na 128GB. U 2012. godini uvedeni su indeksi stupca koji su bili od velike pomoći u uvjetnom dohvatku veće količine podataka. Njihov nedostatak je bio to što su bili ograničeni na ne-klasterirane indekse i podržavali su vrlo mali broj tipova podataka. Stoga u 2014. godine ta je značajka proširena kako bi podržala klasterirane indekse

stupaca. Isto tako, SQL Server 2014 pruža novu opciju integracije čvrstog diska (SSD) koja omogućava korištenje SSD-ova za proširenje međuspremnika SQL Servera kao trajni RAM. S ovom novom značajkom možemo proširiti međuspremnike sustavima koji su maksimizirali svoju memoriju te se mogu osigurati poboljšanja perfomansi za OLTP. U SQL Server 2014 Microsoft je unaprijedio jednu od glavnih značajki iz SQL Server 2012, a to je „Always On“ integracija i to povećanjem broja sekundarnih replika s 4 na 8. Sekundarne replike su od tad dostupne čak i kada primarna nije. Osim toga, SQL Server 2014 pruža novi „čarobnjak“ (*eng. wizard*) za dodavanje replike u Azure, koji vam pomaže u stvaranju asinkronih sekundarnih replika u sustavu Windows Azure. [5]

2.4. SQL Server 2016

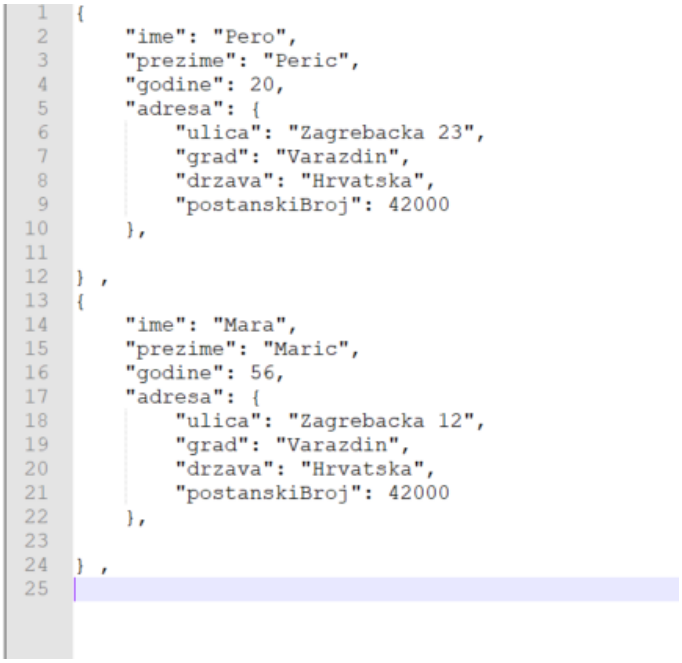
Microsoft je pri izgradnji verzije SQL Server 2016 imao mnogo stvari na umu, a to je analitika u stvarnom svijetu, bogati vizualni efekti na mobilnim uređajima, naprednu sigurnosnu tehnologiju, i slično. Novo svojstvo u SQL 2016 pod nazivom „Always Encrypted“ uvedeno je sa idejom da uvijek čuva osjetljive podatke bilo u lokalnom okruženju ili udaljenjom (*eng. Cloud/Azure*). Zadaća mu je da podatke zaštiti od neovlaštenih osoba. [6] Isto tako, SQL 2016 daje izravnu podršku JSON-u (*eng. Javascript Object Notation* - JSON), on ima mogućnost čitanja JSON format podataka, učitavati ih u tablice i primjeniti svojstvo indeksa nad JSON stupcima. Jedna od značajki sigurnosti u SQL 2016 je maskiranje podataka, smisao ove značajke je sakrivanje povjerljivih podataka. To znači da korisnik s ograničenim pravima ne bude imao pregled originalnog teksta već bude vidio samo maskirani dio podataka. Korištenje maskiranja ne predstavlja velike problem jer SQL ima unaprijed predefinirane funkcije za maskiranje koje se samo primjenjuju na različite stupce u bazi podataka. [6]

3. SQL Server 2017

SQL Server 2016 pružio je mnoga poboljšanja koja je Microsoft nazvao velikim korakom unaprijed, no iako je SQL 2016 bio veliki skok , došla je nova verzija SQL 2017 koja obećava sve to uz još mnogo toga što korporativnim korisnicima treba na dnevnoj bazi. Microsoft SQL Server 2017 trenutno je dostupan s više novih značajki koje nude bržu obradu, veću fleksibilnost korištenja i veliku uštedu troškova. S njim su perfomanse baza podataka dosegle novi vrhunac u obradi upita, novu fleksibilnost rada na više platformi, novu integraciju za statističku analizu podataka te dostupnost SQL Servera na Windowsu, Linuxu, Ubuntu i Docker. Izdanje ove tehnologije pruža mnogo novih i zanimljivih mogućnosti te nova ažuriranja koja su implementirana u postojeće značajke i usluge. Promijene koje su uvede u SQL Server 2017 će biti obrađene u daljnjem kontekstu.

3.1. JSON podrška u SQL Server 2017

Da bih išli u dublju razradu same podrške JSON-a u SQL 2017, prvo ćemo se upoznati s pojmom JSON-a. JSON (*eng. Javascript Object Notation* - JSON) je standardni format za razmjenu podataka između aplikacija i servisa. JSON objekt čini popis podataka u obliku ključ-vrijednost. [3] On je vrlo popularan zbog svoje jednostavnosti, fleksibilnosti te omogućuje njegovo korištenje bez definiranja stroge scheme za podatke. On je izvorni format za većinu programskih jezika kao što je Javascript, oni ga mogu generirati i koristiti podatke izvorno bez serijalizacije što omogućuje brzu integraciju podataka. JSON struktura kao što je već spomenuto je lista ključeva ili asocijativni niz gdje se struktura obilježava sa **{}** vitičastim zagradama, a podaci su u obliku ključ-vrijednost. Uz parove ključ-vrijednost podaci mogu biti prikazani i u uređenoj listi vrijednosti koja se obilježava kao niz s **[]** uglatim zagradama gdje su članovi niza odvojeni zarezom. Budući da vrijednost podataka može biti bilo koji tip podataka to dozvoljava da se može imati više ugniježđenih slojeva što čini JSON dobrim izborom za složene podatake. Primjer JSON objekta možemo vidjeti na slici ispod te iz nje možemo zaključiti da je razmak između ključa i vrijednosti dozvoljen. Slika ispod prikazuje JSON objekt koji opisuje osobu, objekt ima polja ime i prezime koja su prikazana pomoću *string* tipa podataka, godina je broj te objekt koji predstavlja adresu osobe čija su polja također prikazana pomoću *string-a*.



```
{
    "ime": "Pero",
    "prezime": "Peric",
    "godine": 20,
    "adresa": {
        "ulica": "Zagrebacka 23",
        "grad": "Varazdin",
        "drzava": "Hrvatska",
        "postanskiBroj": 42000
    },

} ,
{
    "ime": "Mara",
    "prezime": "Maric",
    "godine": 56,
    "adresa": {
        "ulica": "Zagrebacka 12",
        "grad": "Varazdin",
        "drzava": "Hrvatska",
        "postanskiBroj": 42000
    },

} ,
```

Slika 1. Prikaz JSON objekta (autorski rad)

Sada kad smo malo više upoznati sa JSON-om možemo objasniti kako radi dohvaćanje podataka u SQL Serveru u JSON formatu. JSON podrška u SQL Serveru dolazi uz vrlo uobičajenu akciju, a to je formatiranje tabličnih podataka kao JSON. U SQL Serveru 2017 klauzula FOR JSON se može korisiti sa SELECT naredbom u svrhu formatiranja podataka. Korištenjem FOR JSON klauzule može se birati rad između dva modula. Prvi modul je FOR JSON AUTO gdje će rezultat JSON-a automatski bit formatiran prema strukturi SELECT iskaza. Drugi modul je FOR JSON PATH u kojemu korisnik sam definirana strukturu prema kojoj će JSON biti formatiran, prednost ovog modula je da se mogu složiti vrlo kompleskni krajnji izlazi poput ugniježđenih objekata i svojstava.[6] Za korištenje oba modula potrebne su nam popunjene tablice u bazi podataka.

korisnikID	imePrezime	email	telBroj
1	MataJuric	null	null
2	KajaStok	kaja@test.com	666-0103
3	AnaAnic	anic@test.com	666-0105

Slika 2. Prikaz tablice s podacima (autorski rad)

Sada kada imamo tablice s podacima možemo koristit SELECT upit za dohvat podataka prema nekom uvjetu. [6] Kako jedan SELECT upit izgleda možemo vidjet u primjeru koji slijedi.

```
SELECT TOP (3) korisnikID, imePrezime, email, telBroj
FROM Korisnici ORDER BY korisnikID ASC;
```

Prikazani upit će kao rezultat vratiti prva 3 reda iz tablice s vrijednostima zapisanim pod svojstva koja su navedena u upitu te će naš JSON prema prvom modulu FOR JSON AUTO koji se formatira prema SELECT izrazu izgledati kao niz JSON objekata. To možemo vidjet u sljedećem prikazu. [6]

```
[
        {
            "korisnikID":1,
            "imePrezime":"MataJuric"
        },
        {
            "korisnikID":2,
            "imePrezime":"KajaStok",
            "email":"kaja@test.com",
            "telBroj":"666-0103"
        },
        {
            "korisnikID":3,
            "imePrezime":"AnaAnic",
            "email":"anic@test.com",
            "PhoneNumber":"666-0105"
        }
]
```

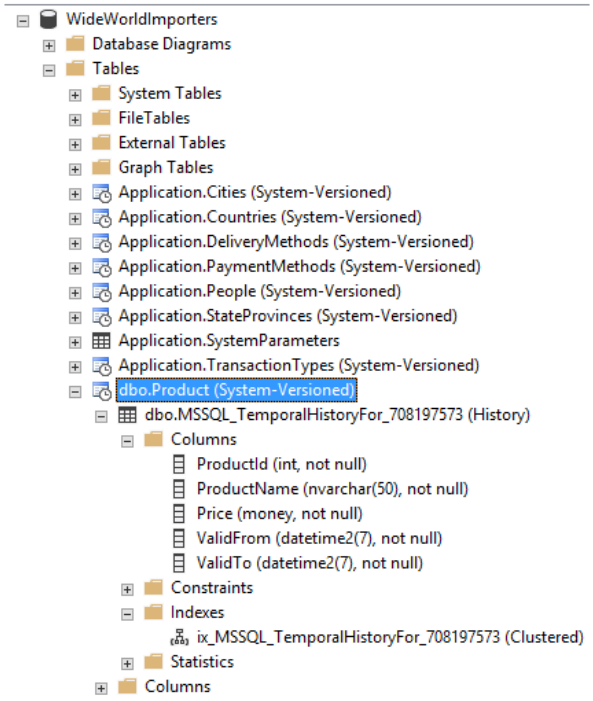
Razlika u korištenju drugog modula JSON PATH pomoću istog SELECT upita je u tome što korisnik sam definira strukturu i može unjeti promijene u izgledu JSON-a. Tako u ovom primjeru možemo uvest jedno ugniježđeno svojsto „Kontakt“ koje će u svoj objektu sadržavati e-mail adresu i telefonski broj te time promijeniti izgled JSON formata pa će njegova struktura izgledati kao sljedeći primjer.

```
[
        {
            "korisnikID":1,
            "imePrezime":"MataJuric"
        },
        {
            "korisnikID":2,
            "imePrezime":"KajaStok",
            "Kontakt": {
                "email":"kaja@test.com",
                "telBroj":"666-0103"
            }
        },
        {
            "korisnikID":3,
            "imePrezime":"AnaAnic",
            "Kontakt": {
                "email":"anic@test.com",
                "telBroj":"666-0105"
            }

        }
]
```


3.2. Vremenske tablice (*eng. Temporal Table*)

Danas postoje razne baze podataka od kojih većina vrlo često mora podržavati vremenske podatke. Pod time pojmom se misli da će se kad tad u bazu spremati podaci o nekom dokumentu, poput ugovora što je slučaj u većini organizacija. Pri tome za ugovor uvijek treba navesti trajanje ugovora zbog čega je omogućeno spremanje podataka u bazu tako da im se odredi neki vremenski interval za koji taj podatak može vrijediti. Osim što se ima potreba bilježenja takvih podataka postoji i potreba za čestim pregledom stanja ili svim promjenama koje su se izvršile u određenom vremenskom razdoblju i zbog toga dolazi do implementiranja vremenskih ograničenja. Kako bi to sve imalo smisla i preglednosti uvedene su vremenske tablice (*eng. Temporal Table*). [6] U tablicama s vremenskom podrškom zaglavlja predstavljaju predikat s najmanje jednim vremenskim parametrom koji predstavlja do kada je taj parametar ispravan, odnosno do kada vrijedi. Dva najčešća atributa koja se koriste pri vremenskim ograničenjima su od (*eng. from*) do (*eng. since*) neki parametar vrijedi. U SQL Serveru 2017

tablice sa vremenskom komponentom implementirane su kao par tablica. Postoji glavna tablica koja sadrži trenutne podatke i povijesna tablica u kojoj su pohranjeni svi povijesni podaci. Pri kreiranju tablica treba imati na umu da povijesna tablica treba sadržavati istu strukturu kao i glavna tablica, ali ne smije bit definirana kao glavna tablica. Isto tako, pri kreiranju tih tablica treba obratiti pozornost na određena svojstva. Tablica s vremenskom komponentom treba imati dva stupca sa tipom podataka DATETIME od kojih jedan sadrži informacije od kada je neki parametar dostupan, a drugi ima informaciju do kada taj isti parametar vrijedi. Osim toga treba definirati vremensko razdoblje sustava prema prethodno definiranim i opisanim stupcima i postaviti tablični atribut SYSTEM_VERSIONING na ON jer inače tablica ne bude imala vremensku komponentu. Za kreiranje tablica u bazi podataka pomaže nam SQL Server Management Studio(SSMS). SSMS nam nudi alate za konfiguriranje, nadzor i administriranje instanci SQL poslužitelja i baza podataka.

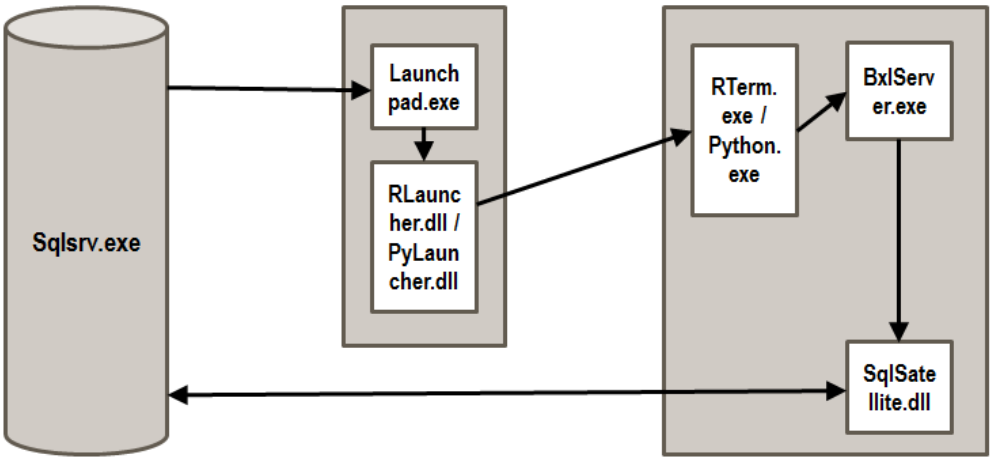


Slika 3. Prikaz kreiranih vremenskih tablica u SSMS-u(Izvor: [6], sekcija „System-versioned temporal tables in SQL Server 2017“)

Na slici 3.možemo vidjet kako izgledaju tablice sa vremenskom komponentom unutar SSMS-a. Možemo primjetiti da svaka od tih tablica ima mali sat koji ukazuje na vremenost. [6] SQL Server 2016 je bila prva verzija u koju su dodali vremenske podatke, no u SQL Serveru 2017 ta podrška ostala je ista, nisu dodana nova poboljšanja. I u jednoj i u drugoj verziji SQL Servera vremenske tablice donose povelike probleme sa vremenom aplikacije. Da bi se ti problemi riješili potrebno je razviti vlastito rješenje koje uključuje i vlastitu implementaciju svih ograničenja.

3.3. Podrška R jezika u SQL Serveru 2017

R je programski jezik najčešće korišten za statistiku, rudarenje podataka i strojno učenje. Osim što je jezik, R je i okruženje u kojem se izvršava njegov kod. R se može naučiti kao i bilo koji programski jezik. Prije nego što počnemo dublje u razradu samog R, pojasnit ćemo pojmove stastika, rudarenje podataka i strojno učenje kako bi nam bilo lakše pratiti kontekst. Statistika je način prikupljanja, proučavanja i analiza prikupljenih podataka te interpretacija rezultata analize. Rudarenje podataka je skup tehnika za analiziranje podataka kako bih pomoću njih pronašli pravila i obrasce koji bi najviše poboljšali poslovanje određene organizacije za koju se izvršila analiza. Strojno učenje je način programiranja pomoću kojeg se automatski rješavaju problemi vezani uz podatke. U SQL Serveru 2017 R servisi zbog dodane Phytonove podrške preimenovani su u servise strojnog učenja (*eng. Machine Learning Services*) koji kombiniraju snagu i fleksibilnost R jezika s alatima za pohranu podataka, razvoj tijeka rada, izvješćivanje i vizualizaciju. R dolazi u SQL Server 2016 i 2017 kao vrlo skalabilan sustav za razvoj koji ima barem jednu skalabilnu funkciju ili algoritam koji se može upotrijebiti pri analizi vrlo velikog skupa podataka. Pomoću strojnog učenja moguće je napravit model u SQL Server tablicu kojeg će se moći koristit za neka daljnja predviđanja o novim podacima. Microsoft nudi dvije vrste skalabilnog R sustava za razvoj. Prva vrsta R sustava je Machine Learning Services(In-Database). Njegova instalacija integrira R u SQL Server, uključuje usluge baze podataka koje se izvode van SQL Servera te dodaje komunikaciju između Engine Database i R runtime. Ovom instalacijom stiže skup skalabilnih R paketa. Druga vrsta je Microsoft R Server. To je samostalni R poslužitelj s otvorenim i skalabilnim paketima koji imaju mogućnost izvođenja na više platformi. Komunikacija SQL Servera i R sustava za razvoj prikazana je na slici ispod.



Slika 4. Komunikacija između SQL Servera i R sustava za razvoj.

4. Osnove SQL

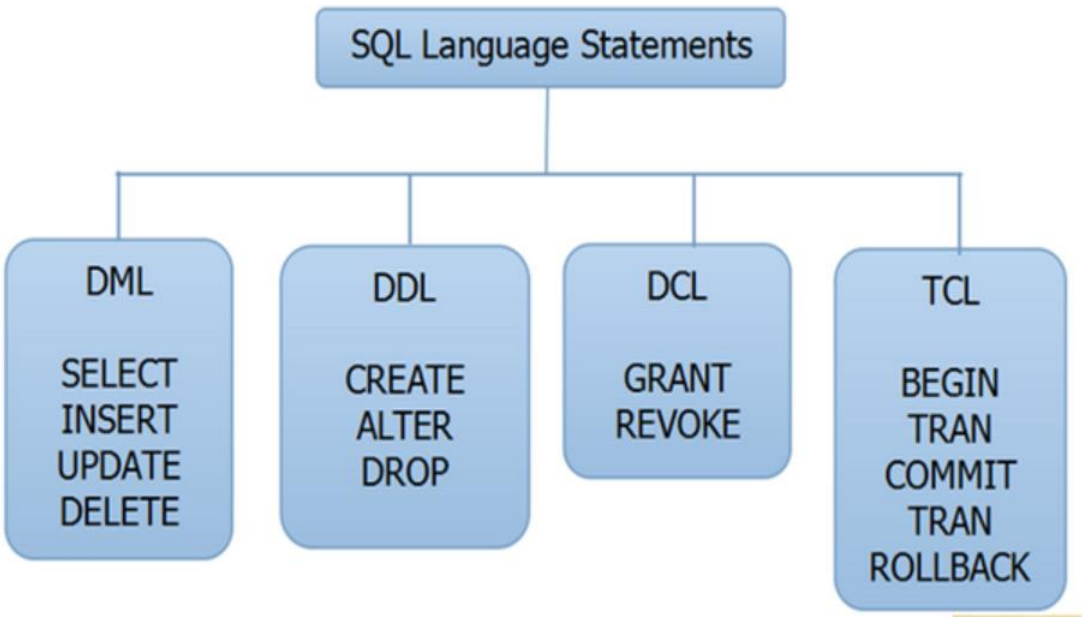
SQL je kratica za *Structured Query Language* (strukturirani upitni jezik) te služi za postavljanje upita nad bazom podataka. Razvijen je u IBM-u 70-tih godina prošlog stoljeća, a trenutno je postao glavni jezik za rad s bazama. SQL je osmišljen s namjerom da bude jednostavan i lako razumljiv za korištenje. SQL naredbe koristimo za izvršavanje zadataka kao što je ažuriranje podataka ili dohvaćanje podataka iz baze. Pomoću standardnih SQL naredbi poput *Select (dohvati), Insert (unesi), Update (ažuriraj), Delete (briši), Create (kreiraj) i Drop (vrsta brisanja)* imamo mogućnost postizanja gotovo svega što trebamo učiniti nad bazom podataka. U daljnjem tekstu biti će objašnjene neke od standardnih i naprednih naredbi SQL jezika.

4.1. SQL naredbe

Glavna podjela SQL naredbi je :

- DDL – *Data Definition Language*
- DML – *Data Manipulation Language*
- DCL – *Data Control Language*
- TCL – *Transaction Control Language*

Podjelu naredbi prema navedenim kategorijama možemo vidjeti na idućoj slici.



Slika 5. Podjela naredbi SQL jezika. (Izvor: [9])

4.2. DDL naredbe

DDL naredbe, kao i što njihov naziv govori, koristimo pri definiranju i ažuriranju objekata baze podataka, a to su tablice, pogledi, procedure i slično. Ove naredbe najviše služe za izradu sheme baze podataka, odnosno vode brigu o dizajnu i strukturi baze. DDL naredbe su *CREATE, ALTER, DROP, RENAME, TRUNCATE I COMMENT*. [9]

CREATE naredbu koristimo pri kreiranju nove tablice u čijem tijelu navodimo svojstva koja će ta tablica posjedovati. Isto tako na jako sličan način koristimo istu naredbu za kreiranje baze, pogleda, okidača, procedura i ostalo.[9] Njena sintaksa glasi :

```
CREATE TABLE ime_relacije (
Ime_atributa tip_podatka,
Ime_atributa tip_podatka,
...
);
```

ALTER naredba nam omogućava promjene nad strukturom tablice, modificira se ili nadopunjuje postojeća struktura. Pomoću ALTER naredbe možemo dodati kolonu, izbacit kolonu (ALTER TABLE ime_tablice *DROP COLUMN ime_kolone*), promijenit veličinu definiranog tipa podatka, modificirat poglede i ostalo. [9] Na primjer, naredba :

```
ALTER TABLE zaposlenik ADD COLUMN grad VARCHAR(100);
```

dodati će tablici *zaposlenik* još jednu kolonu koja se zove *grad* i predstavlja string od maksimalno 100 znakova.

DROP koristimo kada želimo izbaciti/ ukloniti objekte iz baze podataka bilo da je riječ o koloni, tablici, pogledu ili okidaču. Pri korištenju ove naredbe koristi se ključna riječ DROP te se nakon nje navodi ime objekta kojeg se želi ukloniti.

TRUNCATE naredba služi za brzo brisanje podataka iz tablice bez zaštite referencijalnog integriteta. TRUNCATE naredba je skoro ekvivalentna naredbi DELETE no razlika je u tome što uz TRUNCATE nije moguće koristiti klauzulu WHERE jer TRUNCATE briše sve podatke ili ništa iz tablice. [9] Sintaksa glasi :

```
TRUNCATE TABLE ime_tablice ;
```

Uz sve navede naredbe u DDL naredbe još pripadaju RENAME i COMMENT naredbe. Pomoću RENAME naredbe lako možemo promijeniti ime bilo kojeg objekta unutar baze podataka, a pomoću COMMENT naredbe dodajemo komentar kako bih obilježili dio koda za lakše snalaženje kasnije.

4.3. DML

DML naredbe koristimo za manipulaciju zapisa u tablici. Pomoću njih možemo dohvaćati zapise prema određenom kriteriju, možemo dodavati nove upise u tablicu, ažurirati postojeće i naravno obrisati zapise koje nam više nisu potrebni. DML naredbe su *SELECT, INSERT, UPDATE I DELETE*, iz samih naziva naredbi možemo zaključiti čemu služe u praksi.

SELECT nam služi za dohvaćanje zapisa iz određene tablice. Navodi se ključna riječ SELECT zatim asterisk (*) znak koji zamjenjuje „sve“, to znači da navođenjem tog znaka u rezultatu želimo imati prikazane sve kolone jedne tablice. U slučaju da ne želimo sve kolone tada navodimo imena pojedinih kolona koje želimo koristiti. Kako bih podatke u SELECT naredbi filtrirali koristimo WHERE klauzulu u kojoj navodimo kriterij prema kojemu će podaci biti filtrirani. [9] Primjer naredbe :

```
SELECT * FROM ime_tablice;
```

INSERT naredba služi za unos podataka u bazu podataka. INSERT naredba je vrlo jednostavna, nakon navođenja ključne riječi INSERT slijedi ime tablice, zatim se specificira koje atribute unosimo te slijedi ključna riječ VALUES u čije zagrade navodimo podatke za unos. Podaci koje unosimo moraju se poklapati sa rasporedom atributa u tablici.[9] Sintaksa, glasi :

```
INSERT INTO ime_tablice (kolona1, kolona2, ...) VALUES (podatak1, podatak
2...);
```

UPDATE naredbu koristimo ukoliko želimo raditi izmjenu podataka u tablici. Ima vrlo jednostavnu sintaksu. Nakon ključne riječi UPDATE navodi se ime tablice zatim riječ SET i naziv kolone koju želimo promijeniti i to izjednačimo sa novom vrijednošću. Ovdje je vrlo važno da se na kraju navodi WHERE klauzula pomoću koje specificiramo točan zapis u bazi nad kojim će se vršiti izmjene, pri tome se najčešće koriste identifikatori zapisa u tablici. [9]

```
UPDATE ime_tablice SET kolona1 = podatak1, ... WHERE uvjet ;
```

DELETE koristimo kada želimo izbrisati zapise iz tablice. Sintaksa je vrlo jednostavna, nakon ključne riječi DELETE navodi se ime tablice iz koje brišemo zapis te WHERE klauzula pomoću koje specificiramo točno koji zapis želimo obrisati. Važno je napomenuti da UPDATE i DELETE naredbe ukoliko se ne navede WHERE dio brišu ili ažuriraju svi podaci unutar tablice. [9]

4.4. DCL naredbe

DCL naredbe koristimo za kontrolu pristupa podacima unutar baze podataka. Pomoću njih možemo odobriti ili zabraniti određenim korisnicima pristup nekoj tablici ili zapisima unutar tablice. DCL naredbe su *GRANT, DENY i REVOKE.* [9]

GRANT naredba omogućuje da dodijelimo određenim korisnicima neka privilegije nad tablicom ili ostalim objektima. Na primjer, naredba

```
GRANT SELECT, INSERT, UPDATE ON zaposlenik TO pero;
```

dodjelu korisniku *pero* da nad tablicom *zaposlenik* može koristiti SELECT, INSERT i UPDATE naredbe za manipulaciju podataka.

Ključna riječi DENY zabranjuje korištenje navedene privilegije određenom korisniku koja mu je prethodno u nekom trenutku bila dana. Na primjer, naredba

```
DENY UPDATE ON zaposlenik TO maja;
```

korisniku pod nazivom *maja* oduzima pravo korištenja naredbe UPDATE (ažuriranja) nad tablicom *zaposlenik*.

REVOKE naredba služi za oduzimanje neke privilegije sa tablice ili ostalih objekata od korisnika i vraća na prethodno stanje.

```
REVOKE INSERT ON zaposlenik TO josip;
```

4.5. TCL naredbe

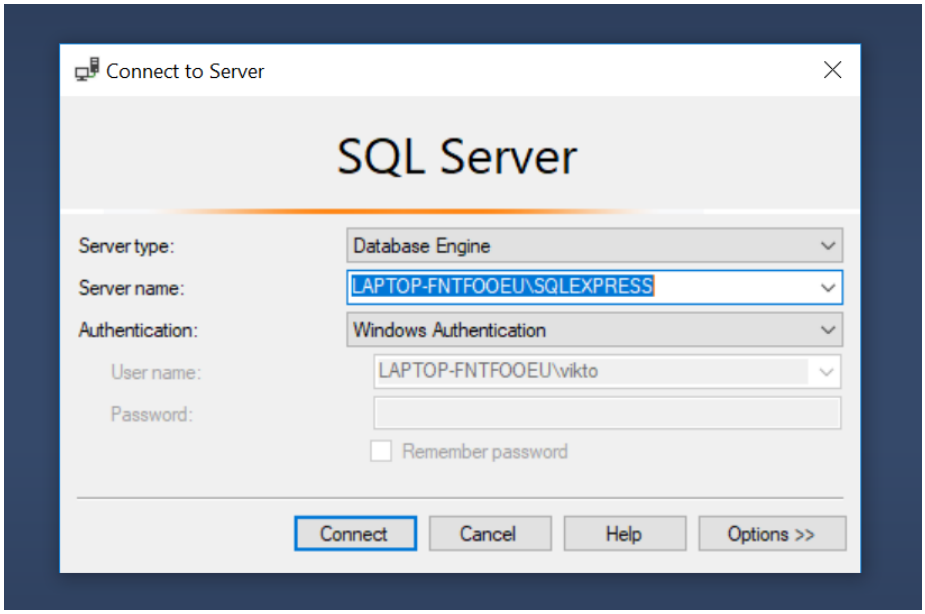
TCL naredbe nam daju mogućnost kontrole i upravljanja sa T-SQL transakcijama kako bih bili sigurni da se integritet podataka očuvao i da li se transakcija uspješno izvršila. Transakcija je jedinica rada nad bazom podataka koja se sastoji od niza logičkih izmjena, odnosno svaka SQL naredba se smatra transakcijom za sebe. [9]
TCL naredbe su :

- BEGIN TRAN – naredba za početak transakcije
- COMMIT TRAN – naredba za uspješno izvršenu transakciju
- ROLLBACK – vraćanje na početak ukoliko se transakcija nije uspješno izvršila

Svaka transakcija započinje ključnom riječi BEGIN koja služi za otvaranje transakcije. Ukoliko transakcija ima dio u kojemu se određeni kriterij provjera tada se koristi COMMIT naredba ukoliko je kriterij ispunjen ili naredba ROLLBACK koja vraća stanje transakcije na početno stanje ukoliko zadani kriterij nije ispunjen.

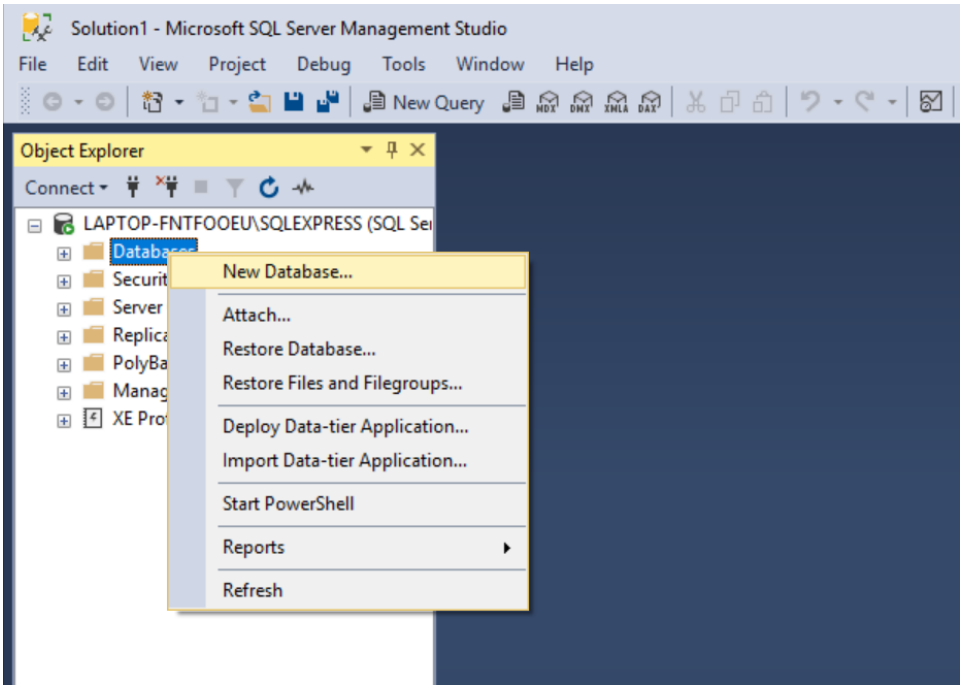
5. Aplikacijska domena

Kako bih nam kreiranje baze podataka bilo što lakše pri njenoj izradi koristit će se SQL Server Management Studio (SSMS). SSMS je program koji se koristi za upravljanje, konfiguriranje i administraciju svih komponenti MS SQL Server-a. Glavna komponenta SSMS-a je Object Explorer koji omogućuje korisnicima lakše snalaženje odnosno pretragu objekata koji se nalaze na serveru. Pri pokretanju SSMS-a otvara se iskočni prozor u kojemu se odabire server te autentifikacija na server u kojemu će se nalaziti novo kreirana baza podataka. Prozor je prikazan na slici 5.



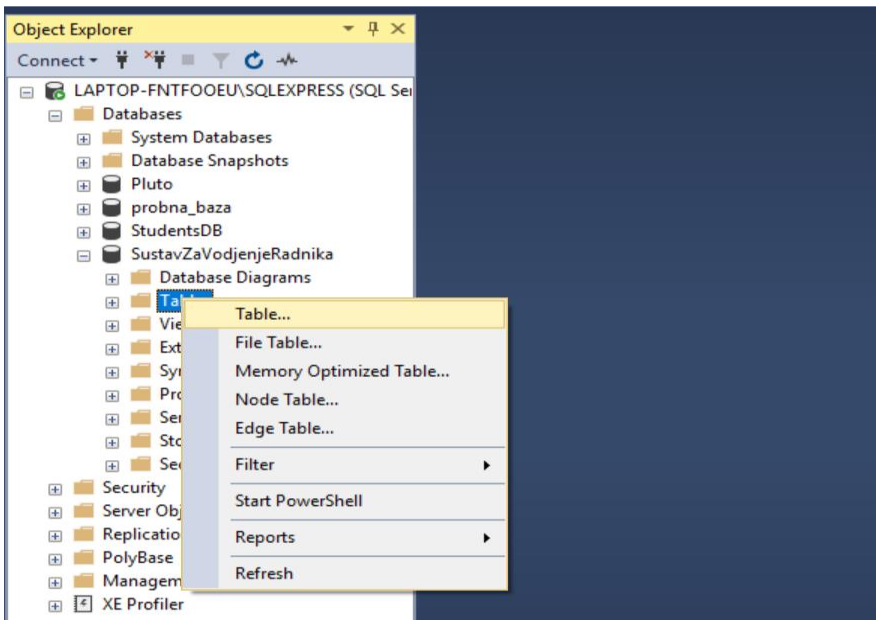
Slika 6. Spajanje na server pomoću SSMS-a.

Nakon uspješnog spajanja na server s lijeve strane se prikazuje Object Explorer u kojemu se nalaze svi objekti na serveru kojima možemo upravljati. Postoje dva načina upravljanja objekta, skriptni i grafički. Razlika između skriptnog i grafičkog je u tome da ako se odabere skriptni način znači da se pišu SQL naredbe za izvršenje neka akcije dok pomoću grafičkog u par klika možemo stvoriti bazu, tablicu, veze između tablica i slično. Ako odaberemo grafički način rada tada ćemo desnim klikom na mapu Database u Object Exploreu dobiti padajući izbornik iz kojeg ćemo odabrati New Database. To nam prikazuje slika 6.



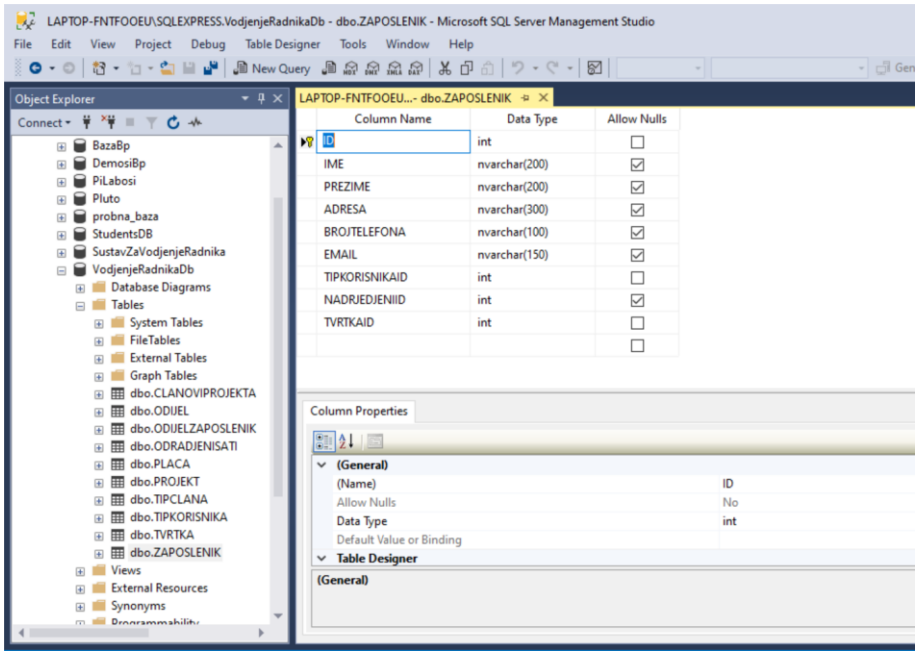
Slika 7. Stvaranje baze podataka.

Nakon odabira New Database, otvara se iskočni prozor u koji se unosi naziv i svojstva nove baze podataka. Na isti način možemo kreirat i tablicu, desnim klikom na mapu Table otvara se padajući izbornik iz kojeg odabiremo opciju Table. Prozor prikazan na slici 7.



Slika 8. Stvaranje nove tablice.

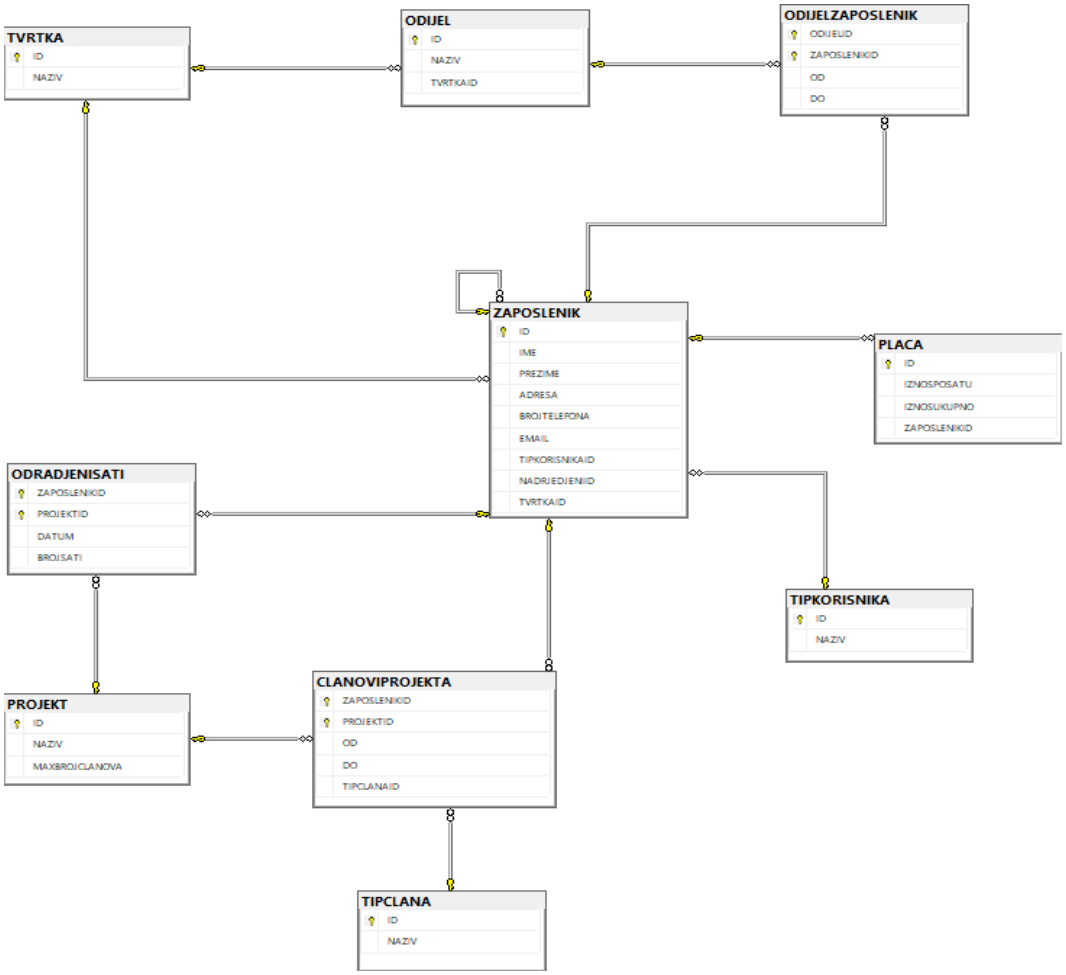
Nakon toga prikazuje se grafičko sučelje za unos naziva atributa, tip podataka i za ograničenja koja možemo stvarati nad novim atributima. Prikaz na slici 8.



Slika 9. Stvaranje tablice u grafičkom načinu rada.

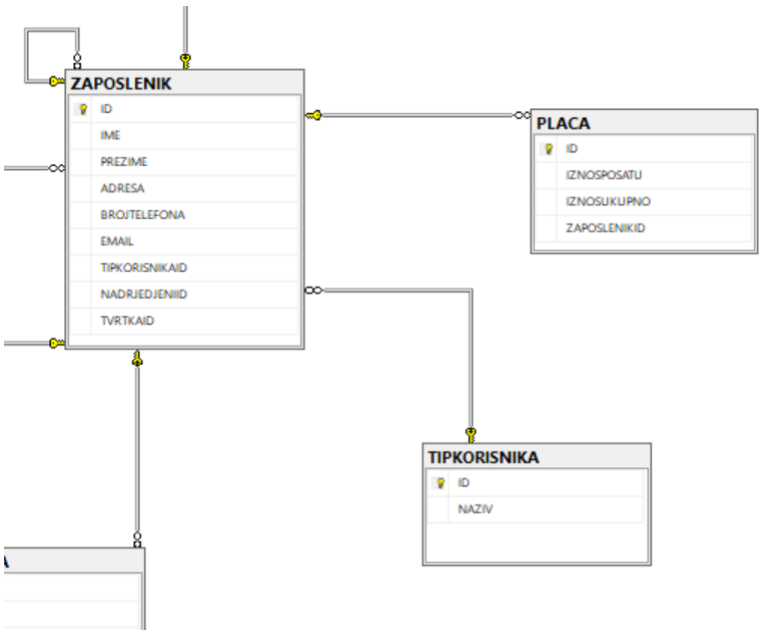
5.1. ERA Model

Za prikaz tablica i veza između tablica koristit ćemo ERA model. ERA model je metoda konceptualnog modeliranja podataka. ERA model se sastoji od entiteta, atributa, veza i ograničenja. Entitet je neki stvarni ili apstraktni predmet u kojemu se pamte podaci za taj predmet. Atribut je podatak koji klasificira stanje entiteta, odnosno njegovo jedinstveno obilježje. Veza je odnos između dva ili više entiteta, veze mogu biti dvosmjerne što znači da opisuju odnose entiteta u oba smjera. Na ERA modelu za sustav vođenja radnika biti će prikazano sve što je potrebno da bi se jedan takav sustav izgradio. Pozitivna strana ERA modela je što se lako može prilagođavati s obzirom na potrebe. Na slici 9. je prikazan početni ERA model sustava za vođenje radnika.



Slika 10. Prikaz ERA model sustav za vođenje radnika.

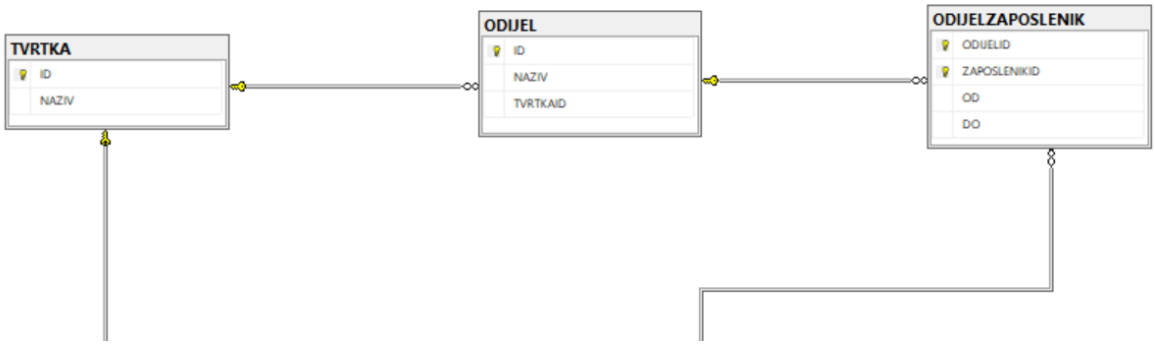
5.2. Zaposlenik



Slika 11. Prikaz cjeline Zaposlenik

Na slici 10. su prikazane tablice *Zaposlenik*, *Tip korisnika i Plaća*. U tablicu *Zaposlenik* zapisuju se osnovni podaci o zaposleniku. Ovdje je važno napomenuti atribut *NadredjeniID* koji je vanjski ključ na tablicu *Zaposlenik* i predstavlja nadređenog. On je opcionalan jer postoji mogućnost da zaposlenik nema nadređenog ukoliko je on voditelj tima ili šef odjela što je predstavljeno samom tablicom *Tip korisnika*. *Tip korisnika* je tablica šifarnika s popisom svih tipova korisnika. U tablici *Plaća* se zapisuju podaci o plaći zaposlenika.

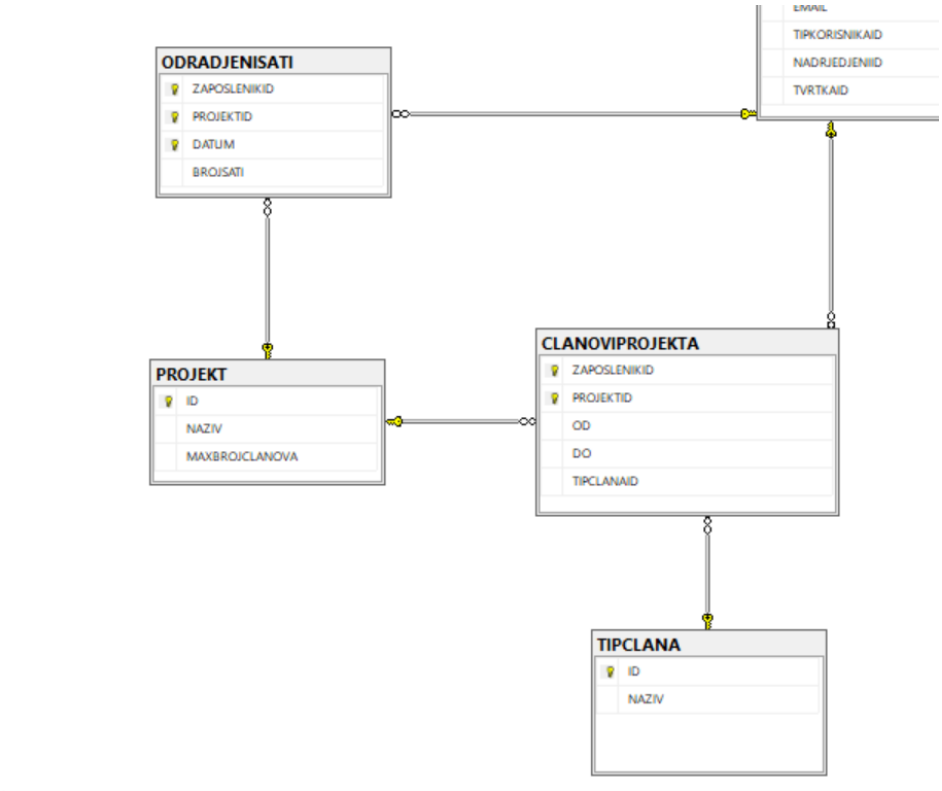
5.3. Odnos između tvrtke, odjela i zaposlenika



Slika 12. Prikaz odnosa među tvrtke, odjela i zaposlenika.

Na slici iznad možemo vidjet tablice *Tvrtka, Odjel i OdjelZaposlenik.* Pri unosu podataka u tablicu *Odjel* potrebno je navesti kojoj tvrtki on pripada. Tablica *OdjelZaposlenik* je vezna tablica između tablice *Odjel i Zaposlenik* koja nam prikazuje da svaki zaposlenik istovremeno može biti u više odjela i da jedan odjel može imati više zaposlenika. Kako bi neki zaposlenik bio dodjeljen odjelu uvjet je da je zaposlen u tvrtki kojoj odjel pripada što je riješeno okidačem objašnjenim u daljnjem tekstu.

5.4. Članovi projekta



Slika 13. Prikaz cjeline članovi projekta.

Na slici 12. su prikazane tablice *OdradjeniSati, Projekt, ClanoviProjekta i TipClana*. Svaki zaposlenik je dodjeljen projektu. Pri dodjeli zaposlenika na projekt okidačem se provjerava da li je maksimalni broj zaposlenika po projektu ispunjen. Vezna tablica *OdradjeniSati* sadrži zapise za svakog zaposlenika koliko sati na koji datum je odradio na projektu koji mu je dodjeljen. Zaposlenik može istovremeno biti na više projekata i projekt može imati više zaposlenika. Tablica *ClanoviProjekta* prikazuje podatke o tome koliko je zaposlenik vremenski proveo na nekom projektu. To je također vezna tablica između tablice *Zaposlenik* i *Projekt*. Na projektu postoji više tipova zaposlenika što je predstavljeno tablicom *TipClana*.

5.5. Funkcije

SQL Server funkcije su kao i funkcije u bilo kojem programskom jeziku. Primaju parametre, izvršavaju različite iskaze i vraćaju rezultat tih iskaza kao vrijednosti. Funkcije se mogu podijeliti na skalarne i „table valued“ funkcije. Skalarne funkcije vraćaju jednu vrijednost podataka kao rezultat funkcije. Ta vrijednost može biti bilo kojeg tipa podataka, osim teksta, slika i vremenskih tipova podataka. Table-valued funkcije predstavljaju korisnički definirane funkcije koje vraćaju tablicu kao rezultat rada funkcije. Dalje u radu korištene su skalarne funkcije.

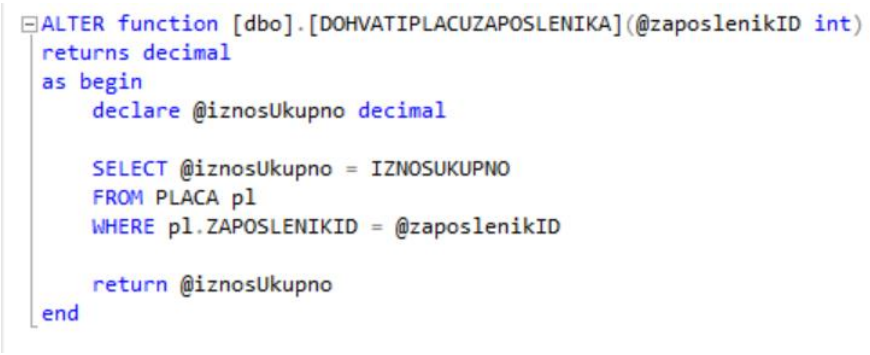
5.5.1. Dohvat odrađenih sati

```
ALTER function [dbo].[DOHVATIODRADJENESATEPOZAPOSLENIKU](@zaposlenikID int, @pocetniDatum datetime, @krajnjiDatum datetime)
returns INT

as begin
    declare @odradeniSati int;

    select @odradeniSati = SUM(BROJSATI)
    from ODRADJENISATI odj
    where odj.ZAPOSLENIKID = @zaposlenikID and odj.DATUM BETWEEN @pocetniDatum and @krajnjiDatum

    if (@odradeniSati is null)
        return 0;

    return @odradeniSati;

end
```

Slika 14. Prikaz funkcije za dohvat odrađenih sati.

Na slici 13. je navedena funkcija koja prima 3 parametra zaposlenikID, pocetni datum i krajnji datum. Kao rezultat vraća broj odrađenih sati za određenog zaposlenika unutar vremenskog razdoblja koji se definira tako da šaljemo dva datuma za koji period je neki radnik odradio sati. Sintaksa funkcije je vrlo jednostavna, prvo se navodi ime funkcije i parametri koje prima ako ih ima te se nakon toga navodi tip podatka koji će vratiti kao rezultat. Nakon toga slijedi dio u kojemu se izvršava upit koji će nam vratiti rezultat s obzirom na zahtjeve i prosljeđene parametre. Sintaksa funkcije je uvijek ista, jedino što se mijenja je upit, odnosno SQL naredba koja se izvršava tom funkcijom.

5.5.2. Dohvat plaće zaposlenika



```
ALTER function [dbo].[DOHVATIPLACUZAPOSLENIKA](@zaposlenikID int)
returns decimal
as begin
    declare @iznosUkupno decimal

    SELECT @iznosUkupno = IZNOSUKUPNO
    FROM PLACA pl
    WHERE pl.ZAPOSLENIKID = @zaposlenikID

    return @iznosUkupno
end
```

Slika 15. Prikaz funkcije za dohvat plaće zaposlenika.

Na slici 14. možemo vidjet sintaksu funkcije za dohvat plaće zaposlenika. Funkcija kao parametar prima zaposlenikID jer se kao rezultat vraća plaća samo za traženog zaposlenika. Kako bih dobili plaću za samo tog zaposlenika u SQL SELECT naredbi u WHERE dijelu filtriramo podatke tako da gledamo redak u tablici samo gdje je zaposlenikID jednak onomu koji je prosljeđen u funkciju kroz parametar.

5.6. Okidači

Okidači su vrsta pohranjenih procesa koji se automatski aktiviraju kada se odvija neki događaj koji utječe na tablicu. Okidači se najčešće koriste za realizaciju poslovnih pravila koji se koriste za provjeru integriteta podataka u bazi. Pri kreiranju okidača treba se voditi briga o tome da se navede predmet i vrsta ažuriranja koje bi taj okidač aktivirao. Pod predmet se mislim na tablicu ili pogled, a pod vrstu ažuriranja na operacije poput unos, modificiranje ili brisanje podataka. Korištenjem okidača baza podataka postaje aktivna jer se neke akcije mogu izvršavati automatizirano. Sintaksa za kreiranje okidača u SSMS-u izgleda ovako :

```
CREATE [ OR ALTER ] TRIGGER [ schema_name . ]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ] AS { sql_statement  [ ; ] [ ,...n ] | EXTERNAL NAME
<method specifier [ ; ] > }
```

Možemo vidjet da je prvo potrebno navesti ime okidača, tablicu na koju se odnosi i kako će se aktivirati okidač. Može se aktivirati nakon nekog određenog događaja, tada ćemo koristiti ključne riječi FOR ili AFTER, ili će se dogoditi prije nekog događaja i tada ćemo koristiti ključnu riječ INSTEAD OF. Dalje u radu biti će prikazani primjeri okidača korištenjem ključne riječi AFTER.

5.6.1. Članovi projekta

Prilikom upisa zaposlenika u tablicu *ClanoviProjekta* aktivira se okidač koji provjerava da li je maksimalan broj članova za taj projekt ispunjen. Ukoliko je broj ispunjen što znači da nema više mjesta za novog zaposlenika te okidač javlja grešku, a u suprotnom dozvoljava unos u tablicu. Implementaciju okidača možemo vidjeti na primjeru ispod.

```
CREATE TRIGGER [dbo].[PROVJERACLANPROJEKTA] ON [dbo].[CLANOVIPROJEKTA]
AFTER INSERT
AS BEGIN
DECLARE @brojZaposlenika int
DECLARE @max int
SELECT  @brojZaposlenika = COUNT(CLANOVIPROJEKTA.ZAPOSLENIKID)FROM
CLANOVIPROJEKTA WHERE CLANOVIPROJEKTA.PROJEKTID = (SELECT
inserted.PROJEKTID FROM inserted)
SELECT @max = PROJEKT.MAXBROJCLANOVA FROM PROJEKT WHERE PROJEKT.ID =
(SELECT inserted.PROJEKTID FROM inserted)

IF(@brojZaposlenika > @max)
      BEGIN
        RAISERROR ('Unesen je maksimalni broj članova na projekt.', 16,1)
        ROLLBACK TRANSACTION
      END
END
```

5.6.2. Provjera pripadnosti tvrtki

Prilikom dodavanja zaposlenika nekom odjelu aktivira se okidač koji vrši provjeru da li je zaposlenik zaposlen u istoj tvrtki kojoj pripada odabrani odjel. Ukoliko je unos je dozvoljen, a u suprotnom okidač ispisuje grešku. Implementaciju okidača možemo vidjeti na primjeru koji slijedi :

```
CREATE TRIGGER [dbo].[PROVJERAODIJELZAPOSLENIK] ON [dbo].[ODIJELZAPOSLENIK]
AFTER INSERT
AS BEGIN
DECLARE @odijelTvrtkaID int
```

```
DECLARE @zaposlenikTvrtkaID int
SELECT @odijelTvrtkaID = ODIJEL.TVRTKAID FROM ODIJEL WHERE ODIJEL.ID =
(SELECT inserted.ODIJELID FROM inserted)
SELECT @zaposlenikTvrtkaID = ZAPOSLENIK.TVRTKAID FROM ZAPOSLENIK WHERE
ZAPOSLENIK.ID = (SELECT inserted.ZAPOSLENIKID FROM inserted)

IF(@odijelTvrtkaID != @zaposlenikTvrtkaID)
  BEGIN
   RAISERROR ('Zaposlenik mora pripadati istoj tvrtci kao odijel', 16, 1)
   ROLLBACK TRANSACTION
  END
END
```

5.6.3. Provjera pripadnosti projektu

Pri upisu odrađenih sati aktivira se okidač koji provjerava je li zaposlenik za kojeg se upisuju sati još uvijek član tog projekta. Ukoliko nije član projekta javlja se greška, dok se u suprotnom dozvoljava upis u tablicu *OdradjeniSati*. Implementaciju okidača možemo vidjeti u primjeru koji slijedi.

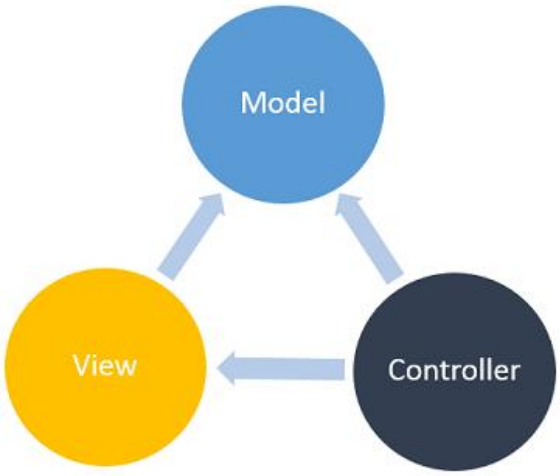
```
CREATE TRIGGER [dbo].[PROJVERAZAPOSLENIKPROJEKT] ON [dbo].[ODRADJENISATI]
AFTER INSERT
AS BEGIN
DECLARE @result int
SELECT @result = CLANOVIPROJEKTA.PROJEKTID FROM CLANOVIPROJEKTA
WHERE CLANOVIPROJEKTA.PROJEKTID = (SELECT inserted.PROJEKTID FROM inserted)
      and CLANOVIPROJEKTA.ZAPOSLENIKID = (SELECT inserted.ZAPOSLENIKID FROM
      inserted) and CLANOVIPROJEKTA.DO is NULL
IF(@result is null)
 BEGIN
      RAISERROR('Zaposlenik više nije član projekta', 16, 1)
      ROLLBACK TRANSACTION
 END
END
```

6. Opis tehnologija za izradu aplikacije

Za ranije kreiranu bazu podataka „Sustav za vođenje radnika“ kreirana je web aplikacija. Aplikacija je osmišljena kako bi ubrzala način unosa novog zaposlenika u sustav i pojednostavila praćenje rada zaposlenika. Navedena aplikacija, odnosno grafičko sučelje je implementirano pomoću Microsoft tehnologija. Koristiti će se C# programski jezik, Entity Framework i ASP .NET MVC tehnologija. U daljnjem kontekstu sve navedeno biti će detaljno objašnjeno te funkcionalnosti same aplikacije.

6.1. C# i ASP .NET MVC

C# programski jezik proizvod od tvrtke Microsoft izgrađen da bude moderan, jednostavan i da odgovor na nedostatke postojećih jezika poput C i C++. C# je potpuno objektno-orijentirani jezik što znači da svi elementi unutar njega predstavljaju objekt. Objekt je struktura koja sadrži podatkovne elemente, metode i njihove međusobne interakcije. ASP .NET MVC je okvir (*eng. framework*) za web aplikacije koji implementira uzorak Model-View-Controller odakle i dolazi naziv MVC. MVC prikaz na slici 15.



Slika 16. MVC

MVC omogućava programerima da razvijaju web aplikacije koristeći sastav od 3 uloge, model, pogledi i kontroler. Model prezentira stanje određenog aspekta aplikacije odnosno predstavlja dio u kojem je implementirana logika za aplikacijsku domenu što znači da su tablice iz baze podataka preslikane u klase modela. Kontroler održava interakcije između modela i pogleda. On upravlja podacima i predaje informacije u pogled kako bih bile dostupne

korisniku. Isto tako sve što korisnik unese preko pogleda ti se podaci šalju kontroleru kako bih se spremili u bazu podatka. Pogled je korisničko sučelje kreirano prema modelu i služi za prikaz podataka korisniku.

6.2. Entity Framework

Pri spajanju na bazu podataka koristit će se Entity Framework. Entity Framework je alat za objektno-relacijsko preslikavanje koje omogućuje rad s relacijskim bazama podataka koristeći .NET objekte i eliminirajući mnoštvo koda kojeg su programeri nekad morali unositi ručno. Microsoft Entity Framework je jedna od vodećih tehnologija za pristup podacima. Prije nego krenemo sa korištenjem Entity Framework-a potrebno je definirati koji način pristupa želimo. Postoje 3 načina korištenja Entity Framework-a :

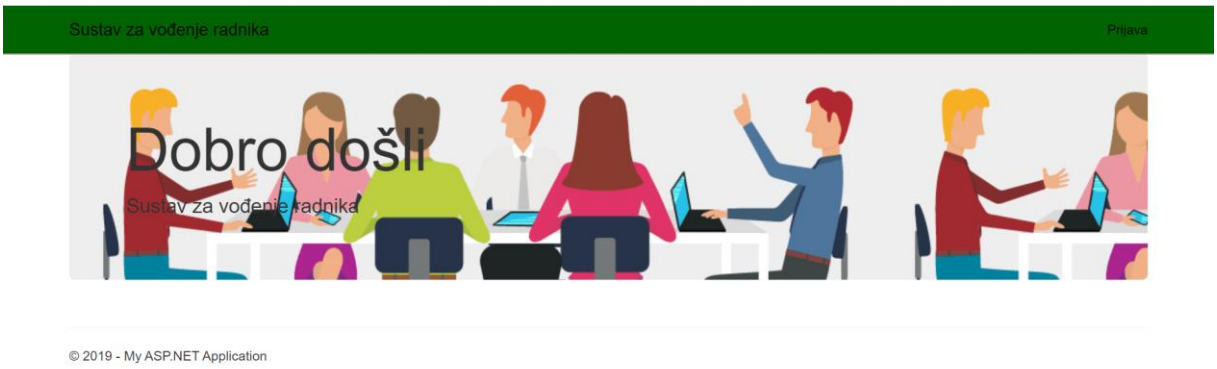
- Baza prvo (*eng. Database-first*)
- Model prvo (*eng. Model-first*)
- Kod prvo (eng. Code-first)

Pristup Baza prvo podrazumijeva kreiranje objekta klasa iz postojeće baze podataka, a pomoću alata koje sadrži Entity Framework mogu se vrlo brzo i jednostavno generirati objekti klasa na osnovu postojeće baze. Model prvo pristup omogućava kreiranje modela podataka u dizajneru te nakon toga generiranje baze podataka iz prethodno kreiranog modela. I zadnji pristup Kod prvo nam omogućava kreiranje modela korištenjem klasa. To znači da prvo kreiramo klase koje će imati strukturu istu kao i shema baze podataka. Svako svojstvo klase odgovara stupcima relacije u bazi podataka. [8]

Dalje u radu korišten je pristup Baza prvo pošto je ranije implementiran model baze i sama baza podataka, a tek onda pripadajuća aplikacija.

7. Opis funkcionalnosti aplikacije

Prilikom otvaranja aplikacije prvo se prikazuje početna stranica s pozdravnim tekstom. Na navigacijskoj traci početne stranice s desne strane se nalazi link koji vodi na stranicu za prijavu u sustav. Izgled početne stranice možemo vidjeti na slici koja slijedi.

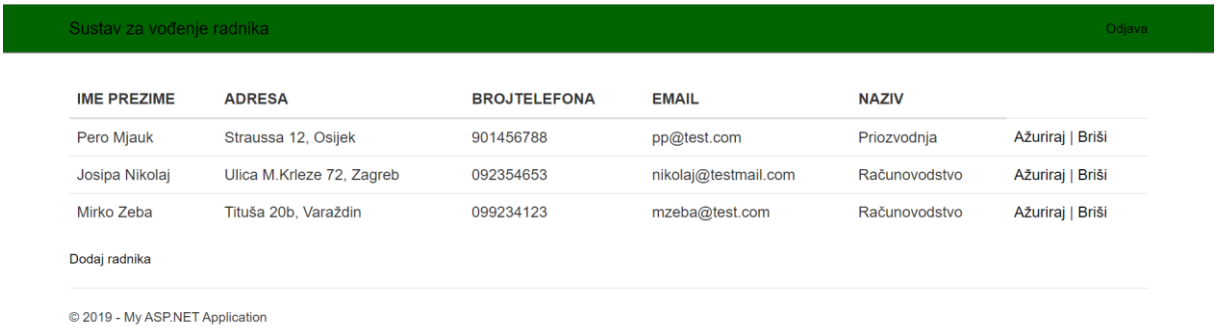


Slika 17. Prikaz početne stranice aplikacije

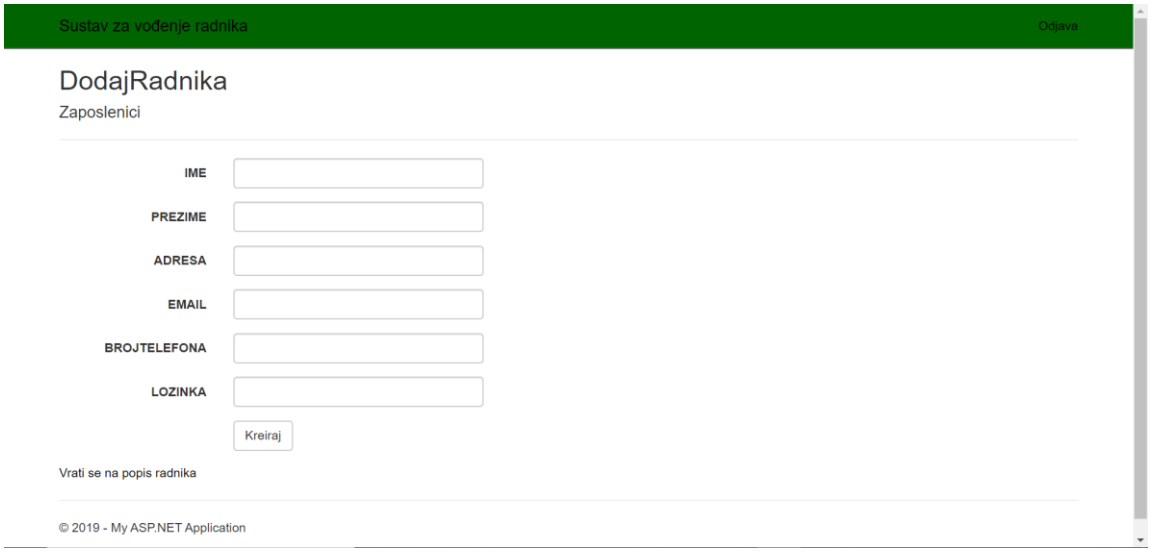
Nakon što kliknemo na link Prijava, otvara se stranica sa formom za prijavu. Ovdje korisnik upisuje svoj e-mail i lozinku kako bih pristupio administrativnom djelu aplikacije.

Slika 18. Forma za prijavu

Nakon uspješne prijave u sustav pokazuje se stranica koja sadrži popis svih radnika. Svaki zapis u tablici je moguće ažurirati ili obrisati. Niže s lijeve strane nalazi se link „Dodaj radnika“ koji vodi na stranicu sa formom za unos novog zaposlenika u sustav.

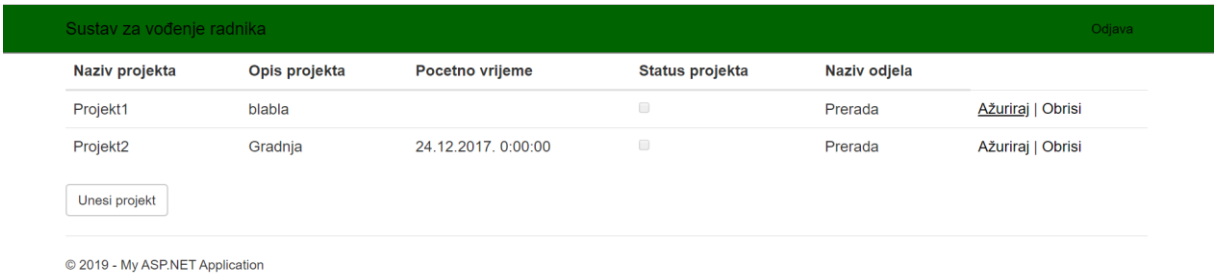


Slika 19. Popis radnika



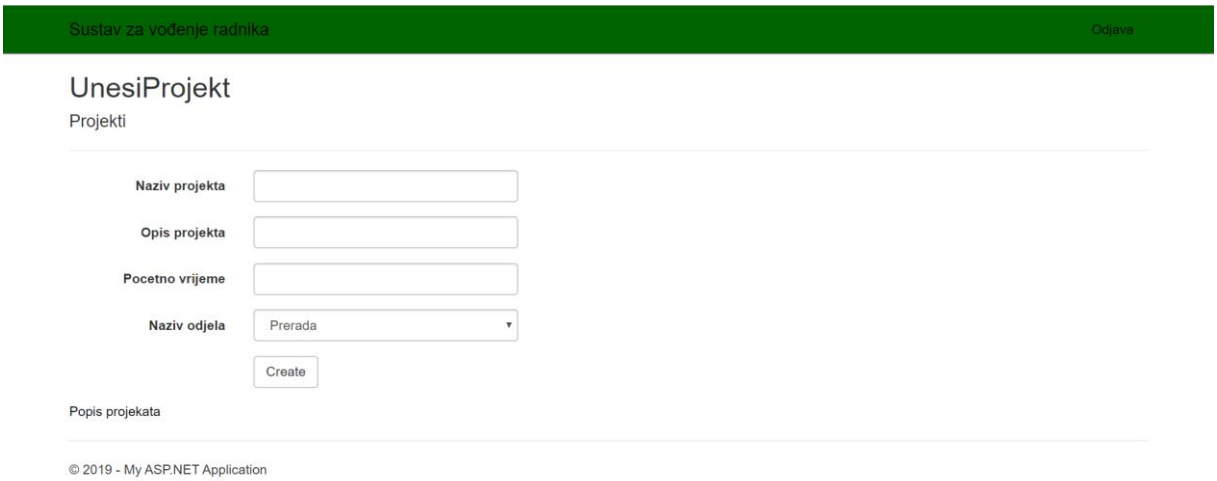
Slika 20. Forma za unos novog radnika

Isto tako imamo stranicu koja sadrži popis svih projekata koji su aktivni. Projekte unosimo na sličan način kao i zaposlenike. Imamo posebnu formu sa atributima koje treba popuniti kako bih unijeli novi projekt. Svakom projektu treba dodijeliti odjel kojemu pripada.

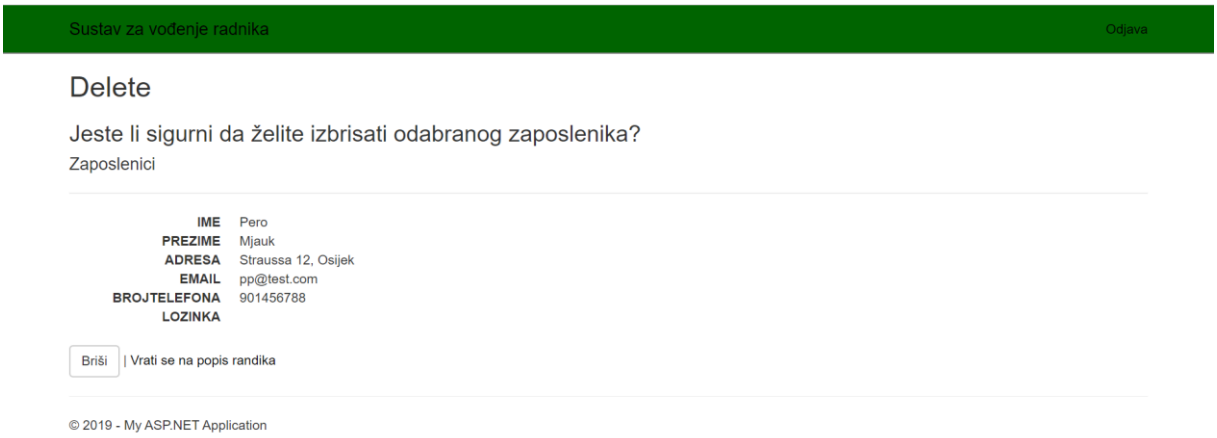


Slika 21. Popis projekata

Možemo primjetiti na slici 21.kako svaki zapis kao i kod popisa zaposlenika ima dvije akcije za ažuriranje i brisanje ne potrebnih zapisa u formi. Isto tako, s lijeve donje strane imamo link koji vodi na formu za unos novog projekta. Prikaz forme možemo vidjeti na idućoj slici.



Slika 22. Forma za unos novog projekta.



Slika 23. Prikaz forme za brisanje

Forma za brisanje zapisa za sve tipove je jednaka. Bilo da smo odabrali brisanje zaposlenika ili projekta forma izgleda kao što je prikazano na gornjoj slici, jedino što je drugačije su prikazani atributi zapisa.

8. Zaključak

Proučavajući SQL možemo zaključiti kako je SQL Server izvrstan alat za izradu modela baze podataka. Uz korištenje SSMS-a SQL nudi mnoštvo mogućnosti pomoću kojih se vrlo lako mogu ažurirati podaci i objekti. SQL Server u kombinaciji sa ASP. NET i Entity Framwork-om nam nudi kvalitetnu, jednostavnu i brzu implementaciju web aplikacija i mnoštvo toga. Uz ponešto truda s kombinacijom ovih tehnologija da se puno toga napraviti i ponuditi jako dobra rješenja. Microsoft još jednom nije razočarao. Baze podataka se razvijaju zajedno sa tehnologijama, u zadnje vrijeme su se počele veoma brzo razvijati i te se puno ulaže u njih. Možemo zaključiti da je dobro utrošiti malo vremena i još više istražiti oko baza podataka i SQL-a jer dolazi sve više novih informacijskih sustava u kojima baze podataka su neizostavna stvar.

Popis literature

[1] K. Rabuzin, *Uvod u SQL*. Varaždin: Fakultet organizacije i informatike, 2011.

[2] R. Manager, *Uvod u baze podataka*, 2. izd. Element, 2014.

[3] D. Sarka, M. Radivojević, i W. Durkin, *SQL Server 2017 Developer's Guide*. 2018.

[4] J. Kashel, T. Kent, i M. Bullerwell, *Microsoft SQL Server 2008 R2 Master Data Services*. 2011.

[5] E. Gethyn, *Getting Started with SQL Server 2014 Administration*. 2014.

[6] D. Sarka, M. Radivojević, i W. Durkin, *Introduction to SQL Server 2016 - SQL Server 2016 Developer's Guide*. 2017.

[7] K. Rabuzin, *SQL napredne teme*. Varaždin, Fakultet organizacije i informatike.2014.

[8] „Entity Framework Tutorial". [Na internetu]. Dostupno na: https://www.entityframeworktutorial.net/. [Pristupljeno: 10-ruj-2019].

[9] „SQL | DDL, DQL, DML, DCL and TCL Commands", *GeeksforGeeks*. [Na internetu]. Dostupno na:https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/. [Pristupljeno: 10-ruj-2019].

Popis slika