SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Martin Kuharić

APLIKACIJA ZA TEMPORALNO MJERENJE I OBRAČUN TROŠKOVA RADA

PROJEKT

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Martin Kuharić

Matični broj: 2604999320007

Studij: ***Organizacija poslovnih sustava***

APLIKACIJA ZA TEMPORALNO MJERENJE I OBRAČUN TROŠKOVA RADA

PROJEKT

Mentor:

dr. sc. Bogdan Okreša Đurić

mag. inf. Tomislav Peharda

Varaždin, veljača 2022.

Martin Kuharić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski 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

Projektna dokumentacija koja prati i opisuje izradu projektnog zadatka, odnosno aplikacije za temporalno i aktivno mjerenje i obračun troškova rada. U dokumentaciji su prikazane faze izrade aplikacije od konceptualne, logičke pa do implementacije aplikacije.

Ključne riječi: postgreSQL, ERA, modeliranje, Visual Studio, okidači, aktivne baze, temporalne baze

Sadržaj

1. Opis aplikacijske domene

Pored svih ostalih troškova koji e mogu pojaviti u poslovanju poduzeća, značajnu stavku čine troškovi rada, odnosno radnika koji su zaposleni u poduzeću. Troškovi rada su posebno značajni u uslužnim poduzećima u kojima čine jednu od glavnih značajki troškova jer su svojim obujmom u samom vrhu troškovne skale. Da bi se poduzećima mogao olakšati posao mjerenja i obračuna troškova, ideja je kreiranje aplikacije koja:

- automatizira obračun troškova
- minimalizira utrošak vremena za obračun troškova
- minimalizira napor zaposlenika u vidu izračuna troškova rada

Aplikacija ulazi u domenu financijskog računovodstva te implementira pojmove kao što su neto i bruto troškovi rada, davanja po zaposleniku, bonusi koje zaposlenik može ostvariti kroz dječji doplatak te ostalo. Iz navedenog razloga prilikom korištenja aplikacije dobro je poznavati osnove računovodstva.

Za izradu vizualnog modela baza podataka je korišten online alat draw.io dok se je sama implementacija baze podataka izradila preko pgAdmin programa za implementaciju i konfiguraciju baze podataka. Dakle, za izdadu baze podataka je korišten postgreSql. Za izradu grafičkog sučelja aplikacije bilo je potrebno bazu podataka povezati sa alatom Visual Studio u kojem je ista i kreirana preko Windows formi.

2. Teorijski uvod

Za izradu ovog rada, konkretnije baze podataka korišten je besplatni SUBP postgreSQL. PostgreSQL je SQL bazirani relacijski sustav, a ujedno podržava temporalnu i aktivnu komponentu baza podatak koja se koristi u svrhu izrade spomenute aplikacije. Dodatno se može proširiti te je pogodan i za primjerice prostorne baze (postGis sustav) te ostalo.

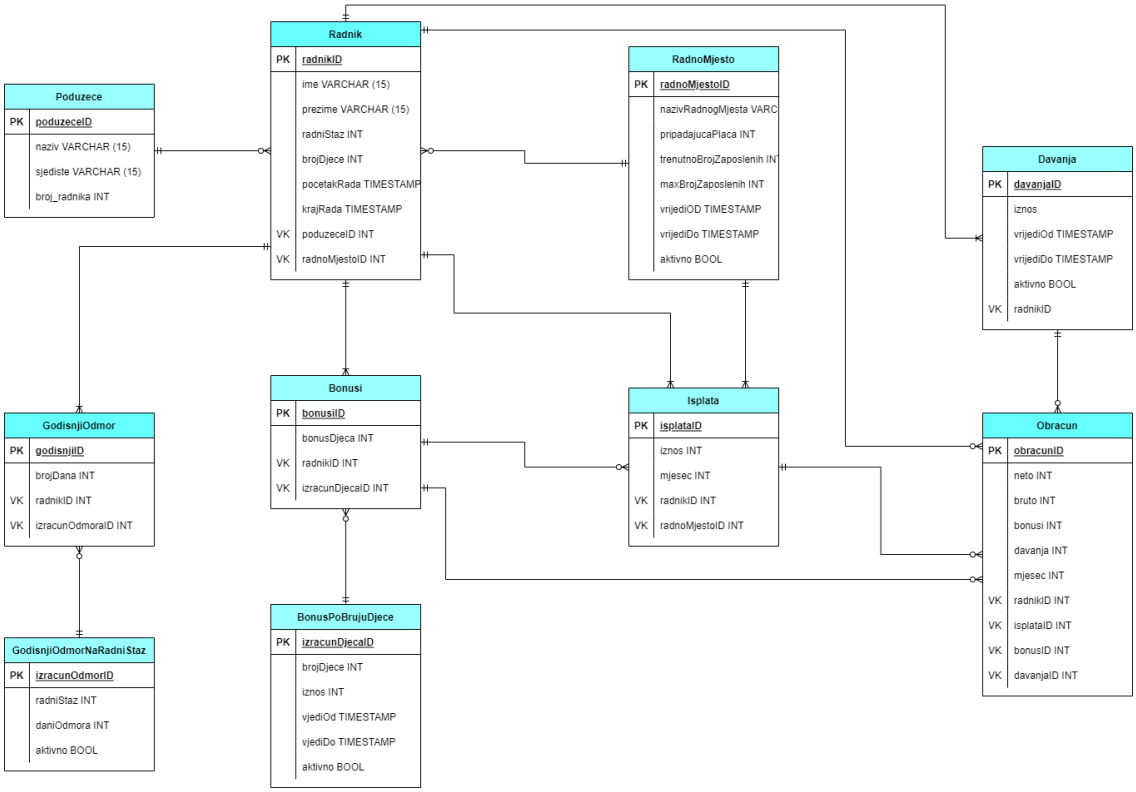
Jedna od glavnih značajki ovog rada su aktivne baze podataka koje se očituju kroz primjenu okidača. Okidače kolokvijalno možemo nazvati događajima koji nastupaju na predefinirani zahtjev kako bi kada dođe vrijeme, odnosno nastupi događaj mogli izvršiti radnju koja se od njih očekuje. Sam okidač se sastoji od 2 dijela:

- Funkcije okidača
- Samog okidača

U funkciji okidača se nalazi sama logika radnje koja će se izvršiti, dok se okidačem definira kada je vrijeme za okidanje istog. Okidač se može okinuti događajima INSERT, UPDATE te DELETE.

3. Model baze podataka

ERA modom baze podataka je vizualno lako predočiti konstrukt kreirane baze, kao i funkcionalnosti koje se žele postići izgradnjom baze podataka.



Slika 1: ERA dijagram (Izvor: vlastita izrada)

Model baze podataka se sastoji od 11 tablica, odnosno relacija pomoću kojih se pokreću funkcionalnosti same aplikacije. Relacije ugrubo možemo podijeliti na one koje se predefiniraju prije početka korištenja aplikacije, a u stvarnom sustavu se mogu realizirati preko vanjskih API-ja. Pojasnimo, način na koji se određuje koliko dana radnik ima pravo na godišnji odmor, koliku svotu novaca će dobiti po broju djece te koliki iznos davanja u smislu na mirovinski prvi i drugi stup ima te ostalo znamo prije negoli što je radnih zaposlen te su to relativno statične vrijednosti koje vrijede za svakog radnika jednako. U modelu baze su relacije koje se tiču ovakvih situacija:

- GodisnjiOdmorNaDradniStaz
- BonusPoBrojuDjece
- Davanja

Druga vrsta relacija su one koje se popunjavaju vremenu poslovanja poduzeća. Prije svega se tu ubraja relacija koja sadrži informacije o radniku. Nakon modifikacija informacija o radniku se okida mnoštvo okidača koji izmjenjuju ostale relacije u modelu.

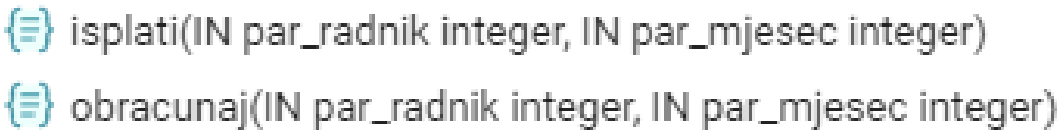
Treća vrsta relacija su relacije koje se popunjavaju automatizmom prilikom modifikacija informacija o radniku. To ubrajamo relacije bonusa te relaciju godišnjeg odmora. Vrijednosti koje se umeću u ove dvije relacije se izračunavaju automatizmom iz predefiniranih vrijednosti te vrijednosti koje su upisane prilikom unošenja radnika u sustav.

4. Implementacija

Kroz ovo poglavlje će se detaljnije pisati sama konstrukcija sustava. Prikazat će se funkcije i okidači koji su zaslužni za radnje u sustavu koje sustav odrađuje kao reakcija na korisnikovu akciju modificiranja nekih od relacija. Nakon informacija o konstruiranju same baze podataka će biti prikazano spajanje baze na okruženje u kojem je izrađeno grafičko sučelje preko kojeg je korisniku omogućen pristup bazi podataka.

4.1. Funkcije

Prva vrsta funkcija koje se koriste u radu su tzv. obične funkcije koje primaju parametre te na temelju primljenih parametara pristupaju relacijama te mijenjaju podatke u njima.



Slika 2: Prikaz funkcija (Izvor: vlastita izrada)

4.1.1. Isplati

Funkcija „Isplati“ odnosi se na prvenstveno na relaciju Isplata, a koristi se i relacijama RadnoMjesto te relacijom Bonusi iz kojih uzima iznos plaće za radno mjesto koju zbraja sa iznosom bonusa za pojedinog radnika.

Slijedi implementacijska logika funkcije:

```
declare no_radno_mjesto int;
declare placa int;
declare bonus int;
declare no_iznos int;
declare postoji bool;
begin
no_radno_mjesto := (
  select radno_mjesto_id
  from radnik
```

```
  where radnik_id = par_radnik
  );
placa := (
  select pripadajuca_placa
  from radno_mjesto
  where radno_mjesto_id = no_radno_mjesto
  );
bonus := (
  select bonus_djeca
  from bonusi
  where radnik_id = par_radnik
  );
no_iznos := (
  placa + bonus
  );
postoji := exists (
  select isplata
  from isplata
  where radnik_id = par_radnik and mjesec = par_mjesec
  );
if not postoji then
  insert into isplata (iznos, mjesec, radnik_id, radno_mjesto_id)
  values (no_iznos, par_mjesec, par_radnik, no_radno_mjesto);
  else
  raise exception 'Vec postoji isplata za zadanog radnika u tom mjesecu.
  Pokusajte ju potraziti.';
  end if;
end;
```

Iz implementacijske logike može se vidjeti na koji način funkcija odgovara kada ju korisnik pozove. Korisnik poziva funkciju predajući joj par integer vrijednosti. Na prvom mjestu se nalazi ID korisnika, dok je na drugom mjestu mjesec za koji se želi napraviti isplata. Postoje 2 slučaja koja funkcija prepoznaje. U prvom slučaju za kombinaciju ID korisnika i mjeseca ne postoji zapis u relaciji Isplata te se potom za unesenu kombinaciju kreira zapis u relaciji Isplata. U drugom slučaju za unesenu kombinaciju integera nalazi zapis u tablici te se u tom slučaju novi zapis ne kreira u relaciji već se korisniku daje do znanja da zapis postoji. Određenu akciju prati i poruka koja se vraća korisniku u slučaju postojanosti zapisa. Poruka glasi: „Već postoji isplata za zadanog radnika u tom mjesecu“. Pokušajte ju potražiti.

4.1.2.Obracunaj

Funkcija „Obracunaj“ koristi se prilikom unosa novog zapisa u relaciju Obracun. Implementacija funkcije je izvedena kako slijedi:

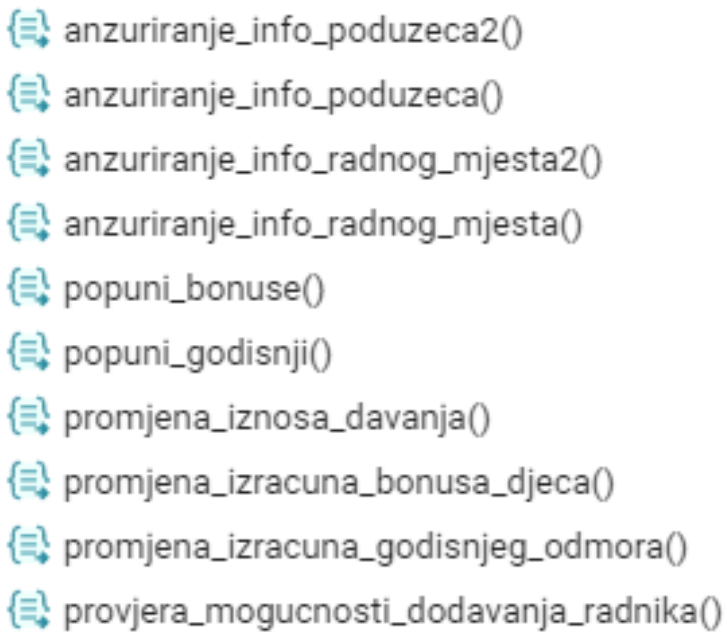
```
declare no_neto int;
declare no_bruto int;
declare no_davanja int;
declare no_bonus int;
declare no_isplata_id int;
declare no_davanja_id int;
declare postoji bool;
begin
no_davanja := (
  select iznos
  from davanja
  where radnik_id = par_radnik and aktivno = true
  );
no_neto := (
  select iznos
  from isplata
  where radnik_id = par_radnik and mjesec = par_mjesec
  );
no_bruto := (
  no_neto + no_davanja
  );
no_bonus := (
  select bonus_djeca
  from bonusi
  where radnik_id = par_radnik
  );
no_isplata_id := (
  select isplata_id
  from isplata
  where radnik_id = par_radnik and mjesec = par_mjesec
  );
no_davanja_id := (
  select davanja_id
  from davanja
  where radnik_id = par_radnik and aktivno = true
  );
```

```
postoji := exists (
  select obracun
  from obracun
  where radnik_id = par_radnik and mjesec = par_mjesec
  );
if not postoji then
  insert into obracun (neto, bruto, bonusi, davanja, mjesec, radnik_id,
isplata_id, davanja_id)
  values (no_neto, no_bruto, no_bonus, no_davanja, par_mjesec,
par_radnik, no_isplata_id, no_davanja_id);
  else
  raise exception 'Placa za trazenog radnika u navedenom mjestu je vec
obracunata.';
  end if;
end;
```

Iz implementacije vidljivo je kako funkcija koristi isključivo dvije vrijednosti, odnosno par integer vrijednosti od kojih je prva ID radnika dok je druga vrijednost mjesec za koji se želi obračunati trošak. Iz danih vrijednosti funkcija dobiva dovoljno informacija da dođe do preostalih 7 atributa koje je potrebno popuniti u relaciji Obracun. Funkcija prepoznaje dva slučaja. Prvi slučaj je da za danu kombinaciju postoji zapis u relaciji te u tom slučaju se isto daje do znanja kroz ispis poruke: „Plaća za traženog radnika u navedenom mjesecu je već obračunata“. U suprotnom slučaju, dakle ako odgovarajući zapis ne postoji u relaciji tada funkcija traži preostale potrebne vrijednosti u bazi kroz nekoliko select upita te dodaje novi zapis u relaciju Obracunaj.

4.2. Trigger funkcije

Ovom vrstom funkcija želi se postići reakcija sustava na određene akcije koje poduzima korisnik sustava. Specifičnost ovih funkcija je u tome što ne primaju argumente na klasičan način već preko okidača, odnosno izravno kod upisa i(li) modifikacije relacija nad kojima je okidač definiran.



Slika 3: Prikaz funkcija okidača (Izvor: vlastita izrada)

U omjeru na prethodno opisane funkcije, funkcija ove vrste je više što nam govori o odnosu napora radnika prema funkcionalnostima koje izvršava aplikacija automatizmom.

4.2.1. Anzuriranje_info_poduzeca

Prilikom dodavanja novog radnika u relaciju Radnik potrebno je promijeniti broj zaposlenih u poduzeću. Funkcija anzuriranje_info_poduzeca() automatizira dodavanje radnika na broj zaposlenih u poduzeće.

```
declare st_broj_radnika int;
begin
st_broj_radnika := (
  select broj_radnika
  from poduzece
  where poduzece_id = new.poduzece_id
  );
  update poduzece
  set broj_radnika = st_broj_radnika + 1
  where poduzece_id = new.poduzece_id;
  return new;
  end;
```

4.2.2. Anzuriranje_info_poduzeca2

Prilikom brisanja radnika u relaciju Radnik potrebno je promijeniti broj zaposlenih u poduzeću. Funkcija anzuriranje_info_poduzeca2() automatizira brisanje radnika iz broja zaposlenih u poduzeće, odnosno umanjuje broj zaposlenih za jedan.

```
declare st_broj_radnika int;
begin
st_broj_radnika := (
  select broj_radnika
  from poduzece
  where poduzece_id = old.poduzece_id
  );
  update poduzece
  set broj_radnika = st_broj_radnika - 1
  where poduzece_id = old.poduzece_id;
  return new;
  end;
```

4.2.3. Anzuriranje_info_radnog_mjesta

Relacija RadnoMjesto sadrži atribut koji prati trenutno broj zaposlenih na određenom radnom mjestu. Nakon dodavanja radnika na neko radno mjesto uvećava se broj zaposlenih na tom radnom mjestu.

```
declare st_trenutno_broj_zaposlenih int;
begin
st_trenutno_broj_zaposlenih := (
  select trenutno_broj_zaposlenih
  from radno_mjesto
  where radno_mjesto_id = new.radno_mjesto_id
  );
  update radno_mjesto
  set trenutno_broj_zaposlenih = st_trenutno_broj_zaposlenih + 1
  where radno_mjesto_id = new.radno_mjesto_id;
  return new;
  end;
```

4.2.4. Anzuriranje_info_radnog_mjesta2

Nakon brisanja radnika iz nekog radnog mjesta potrebno je izbrisati jednog zaposlenog. To je implementirano na način da se nad atributom „trenutnoZaposleni“ umanji iznos za jedan.

```
 declare st_trenutno_broj_zaposlenih int;
begin
st_trenutno_broj_zaposlenih := (
  select trenutno_broj_zaposlenih
  from radno_mjesto
  where radno_mjesto_id = old.radno_mjesto_id
  );
  update radno_mjesto
  set trenutno_broj_zaposlenih = st_trenutno_broj_zaposlenih - 1
  where radno_mjesto_id = old.radno_mjesto_id;
  return new;
  end;
```

4.2.5. Popuni_bonuse

Prilikom dodavanja radnika u bazu, aplikacija automatizmom izračunava bonuse na plaču za određenog zaposlenika. Bonusi se izračunavaju na temelju predefiniranih podataka iz relacije u kojoj se nalaze iznosi za određeni broj djece.

```
declare no_iznos int;
declare no_izracun_id int;
begin
no_iznos := (
  select iznos
  from izracun_bonusa_djeca
  where broj_djece = new.broj_djece and aktivno = true
  );
no_izracun_id := (
  select izracun_djeca_id
  from izracun_bonusa_djeca
  where broj_djece = new.broj_djece and aktivno = true
  );
insert into bonusi (bonus_djeca, radnik_id, izracun_djeca_id)
values (no_iznos, new.radnik_id, no_izracun_id);
return new;
end;
```

4.2.6.Popuni_godisnji

Prilikom dodavanja radnika u bazu, aplikacija automatizmom izračunava godišnji odmor na plaču za određenog zaposlenika. Dani godišnjeg odmora se izračunavaju na temelju predefiniranih podataka iz relacije u kojoj se nalaze iznosi za određene godine .

```
declare no_broj_dana int;
declare no_izracun_odmora_id int;
begin
no_broj_dana := (
  select dani_odmora
  from izracun_godisnjeg_odmora
  where radni_staz = new.radni_staz and aktivno = true
  );
no_izracun_odmora_id := (
  select izracun_odmor_id
  from izracun_godisnjeg_odmora
  where radni_staz = new.radni_staz and aktivno = true
  );
insert into godisni_odmor (broj_dana, radnik_id, izracun_odmora_id)
values (no_broj_dana, new.radnik_id, no_izracun_odmora_id);
return new;
end;
```

4.2.7.Promjena_iznosa_davanja

Prilikom dodavanja novog zapisa u relaciju Davanja stari zapis se ne briše iz baze već ga je potrebno na neki način deaktivirati. U tome je potrebno naznačiti koji zapis je najnoviji i kojeg aplikacija može koristiti u obračunu i isplati plače. Navedeno se postiže na način da se prilikom dodavanja novog zapisa u trenutno važeći zapis postavi vrijednost trenutnog vremena te se u novokreirani zapis u atribut aktivno zapiše TRUE, dok se u zapis koji je do sad vrijedio u atribut aktivno zapiše vrijednost FALSE.

```
declare st_zapis davanja;
begin
st_zapis := (
  select davanja
  from davanja
  where radnik_id = new.radnik_id and aktivno = true
```

```
  );
update davanja
set vrijedi_do = now()
where radnik_id = new.radnik_id and aktivno = true;
update davanja
set aktivno = false
where radnik_id = new.radnik_id and aktivno = true;
return new;
end;
```

4.2.8.Promjena_izracuna_bonusa_djeca

Prilikom dodavanja novog zapisa u relaciju izracun_bonusa_djeca stari zapis se ne briše iz baze već ga je potrebno na neki način deaktivirati. U tome je potrebno naznačiti koji zapis je najnoviji i kojeg aplikacija može koristiti u obračunu i isplati plače. Navedeno se postiže na način da se prilikom dodavanja novog zapisa u trenutno važeći zapis postavi vrijednost trenutnog vremena te se u novokreirani zapis u atribut aktivno zapiše TRUE, dok se u zapis koji je do sad vrijedio u atribut aktivno zapiše vrijednost FALSE.

```
declare st_zapis izracun_bonusa_djeca;
begin
st_zapis := (
  select izracun_bonusa_djeca
  from izracun_bonusa_djeca
  where broj_djece = new.broj_djece and aktivno = true
  );
update izracun_bonusa_djeca
set vrijedi_do = now()
where broj_djece = new.broj_djece and aktivno = true;
update izracun_bonusa_djeca
set aktivno = false
where broj_djece = new.broj_djece and aktivno = true;
return new;
end;
```

4.2.9.Promjena_izracuna_godisnjeg_odmora

Prilikom dodavanja novog zapisa u relaciju preko koje se izračunava vrijednost godišnjeg odmora stari zapis se ne briše iz baze već ga je potrebno na neki način deaktivirati. U tome je potrebno naznačiti koji zapis je najnoviji i kojeg aplikacija može koristiti u obračunu

i isplati plače. Navedeno se postiže na način da se u novokreirani zapis u atribut aktivno zapiše TRUE, dok se u zapis koji je do sad vrijedio u atribut aktivno zapiše vrijednost FALSE.

```
declare st_zapis izracun_godisnjeg_odmora;
begin
st_zapis := (
  select izracun_godisnjeg_odmora
  from izracun_godisnjeg_odmora
  where radni_staz = new.radni_staz and aktivno = true
  );
  update izracun_godisnjeg_odmora
  set aktivno = false
  where radni_staz = new.radni_staz and aktivno = true;
  return new;
end;
```

4.2.10. Provjera_mogucnosti_dodavanja_radnika

Prilikom dodavanja radnika u bazu, isti se dodaje na neko radno mjesto. Prilikom dodavanja radnika na radno mjesto provjerava se ima li mjesta na tom radnom mjestu, odnosno može li se dodati radnika na to radno mjesto. Navedeno je implementirano kroz dva atributa u relaciji radno mjesto. U relaciji se nalazi atribut trenutnoBrojZaposlenih koji broji trenutno zaposlene radnike na zadanom radnom mjestu te atribut maxBrojZaposlenih u kojem se nalazi vrijdnost koliko se maksimalno radnika može zaposliti na to radno mjesto. A ako je vrijednost u atributu trenutno zaposlenih manja od maksimalno dopuštenih tada se upis dozvoljava, no ako je broj jednak, odnosno ako zapošljavanje nije dozvoljeno tada se korisniku ispisuje poruka Radno mjesto je popunjeno.“, a upis se ne dopušta.

```
declare trenutno int;
declare maximalno int;
declare dopusti bool;
begin
trenutno := (
  select trenutno_broj_zaposlenih
  from radno_mjesto
  where radno_mjesto_id = new.radno_mjesto_id
  );
maximalno := (
  select max_broj_zaposlenih
```

```
    from radno_mjesto
    where radno_mjesto_id = new.radno_mjesto_id
    );
  if trenutno < maximalno then
    return new;
    else
      raise exception 'Radno mjesto je popunjeno.';
      return old;
    end if;
  end;
```

5. Aplikacija

Ovo poglavlje se bavi daljnjom primjenom konstruirane baze podataka. Nakon kreiranje baze podataka u SUBP-u za korisnika nije prikladno da se bazom koristi iz istog. Iz tog razloga pogodnije je rješenje konstruiranje aplikacije kako bi korištenje aplikacije za korisnika bilo razumljivije, prihvatljivije, lakše, a na posljetku i sigurnije. Ovu fazu možemo podijeliti na 2 koraka:

- Spajanje na bazu
- Kreiranje aplikacije

5.1. Spajanje na bazu

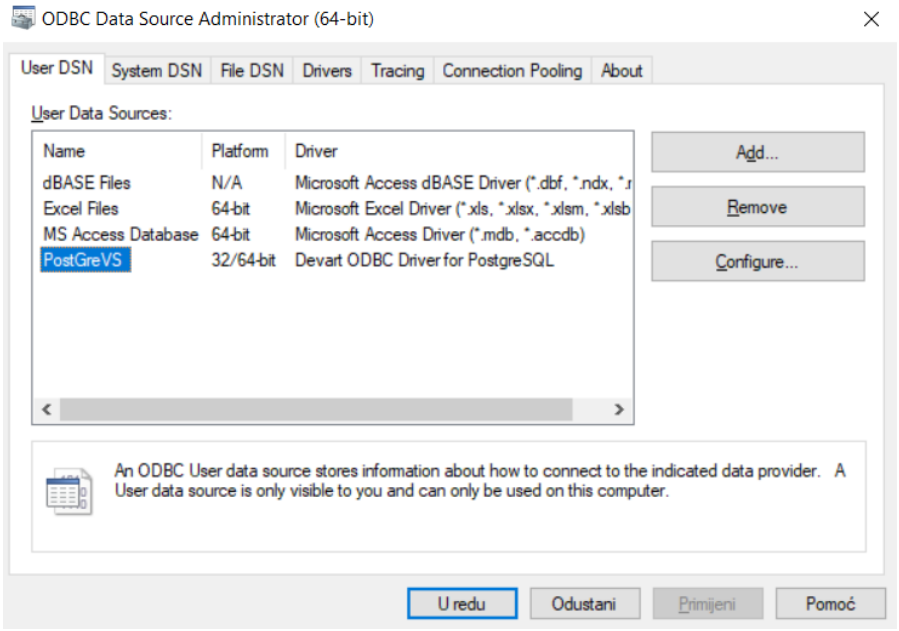
Prije bilo kakvog korištenja baze unutar aplikacije potrebno je istu spojiti na aplikaciju. U korištenom okruženju, odnosno Visual Studio 2019 okruženju se to može postići na 2 načina:

- Korištenjem connectionStringa
- Korištenjem adaptera

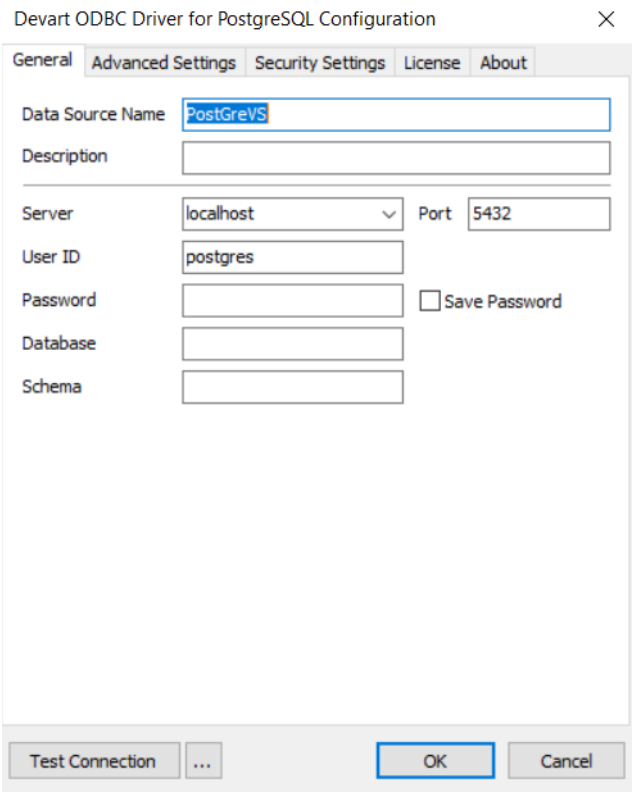
Za potrebe ovog rada, spajanje na bazu je izvedeno pomoću adaptera.

5.1.1.Koraci spajanja na bazu:

Na samom početku bilo je potrebno potražiti update za ODBC Data Source Administrator. Isti je pronađen na web stranici[1]. Te potom potrebo je kreirati novi user DSN, za ovaj projekt nazvan je PostGreVS. Isti je prikazan na slici dolje.



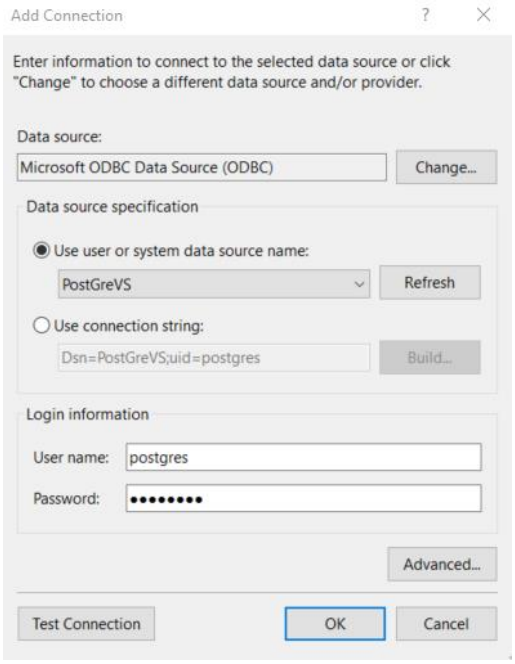
Slika 4: Prikaz ODBC Data Administratora (Izvor: vlastita izrada)



Slika 5: Prikaz kreiranja novog user DSN-a (Izvor: vlastita izrada)

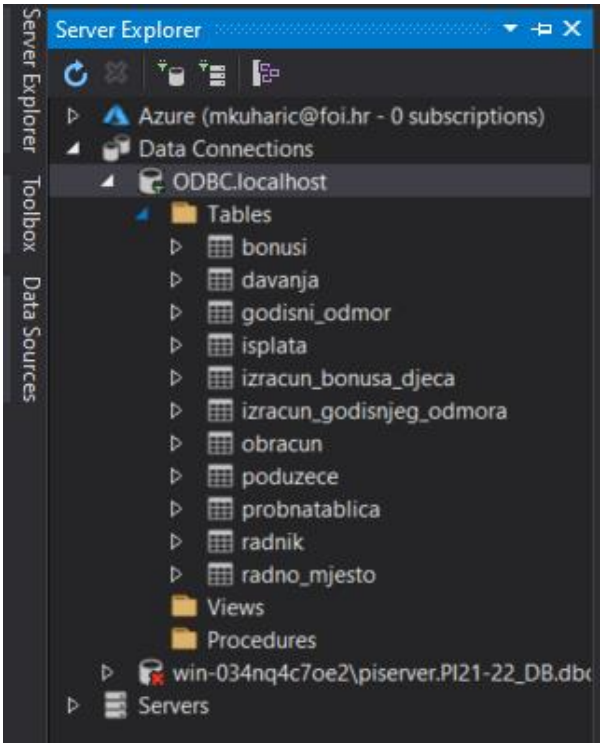
Nakon što je napravljen prethodno opisan korak možemo pristupiti spajanju baze na Visual Studio okruženje. U Visual studio okruženju najprije je potrebno kreirati konekciju. Postupak je prikazan donjom formom. U polje korisničkog imena potrebno je upisati korisničko ime kojim

je napravljena baza podataka preko sql servera, te u polje za lozinku, lozinku koja je postavljena prilikom kreiranja servera.



Slika 6: Prikaz spajanja na bazu (Izvor: vlastita izrada)

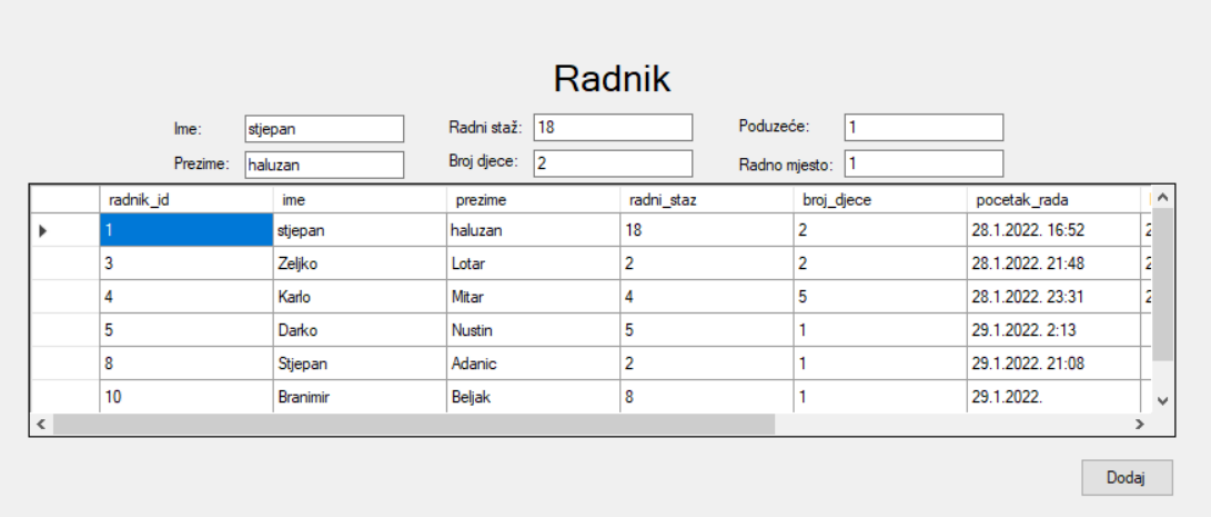
Nakon što je konekcija uspješno uspostavljena u Server Explorer-u moguće je vidjeti tablice, odnosno relacije koje su uvezene iz same spojene baze.



Slika 7: Prikaz relacija u Visual Studio 2019(Izvor: vlastita izrada)

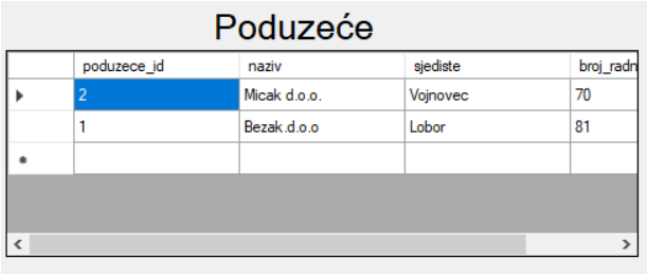
5.2. Izrada aplikacije

Aplikacija se sastoji od Windows forme na kojoj se nalaze interaktivni prikazi relacija. Prikazi relacija su interaktivnog tipa te se promjene na njima prenose neposrednu u spojenu bazu. Također, svi implementirani okidači u bazi podataka se očituju na korištenje aplikacije te kontroliraju i olakšavaju samo korištenje aplikacije. U pozadini forme se nalazi programski kod koji omogućuje korištenje aplikacije i prenošenje promjena u bazu.



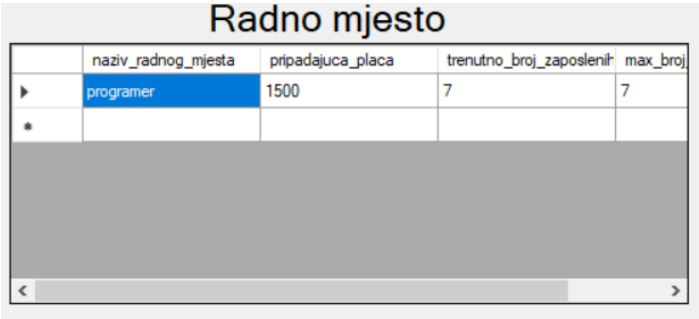
Slika 8: Prikaz tablice Radnik (Izvor: vlastita izrada)

Preko tablice Radnik moguće je unositi, brisati te modificirati informacije o radniku u poduzeću. Nakon unašanje promjena potrebno je kliknuti gumb Dodaj kako bi se promjene dodale u bazu podataka. Kod sljedeće uporabe aplikacije promjene će biti prikazane u aplikaciji.



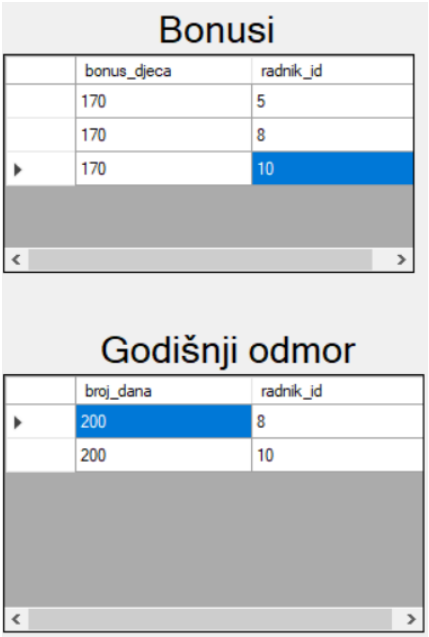
Slika 9: Prikaz tablice Poduzeće (Izvor: vlastita izrada)

Tablica Poduzeće prikazuje informacije o poduzećima koja su dodana u bazu. Na dodana poduzeća je moguće vezati radnika.



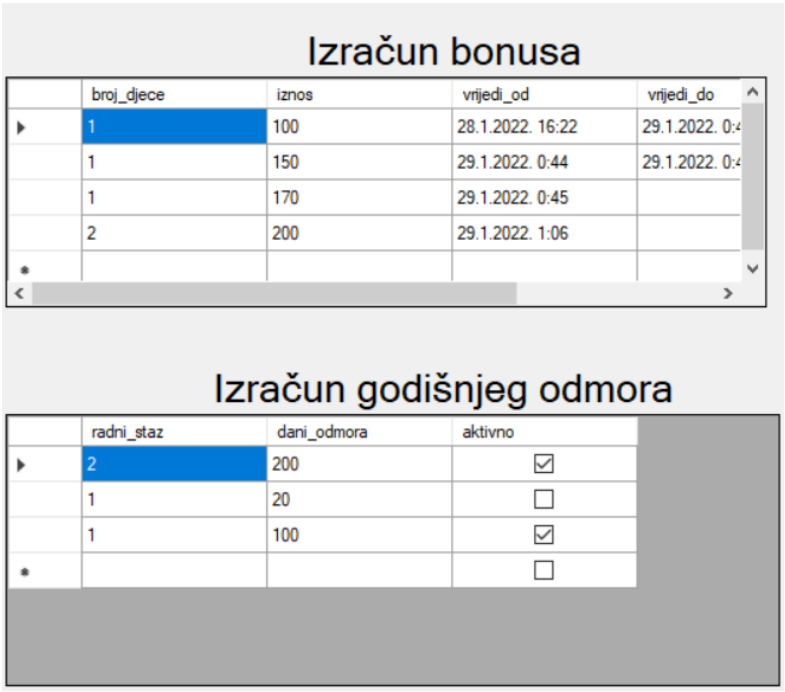
Slika 10: Prikaz tablice Radno mjesto (Izvor: vlastita izrada)

Tablica Radno mjesto prikazuje informacije o radnim mjestima koja su definirana u poduzećima. Na radno mjesto je moguće upisati radnika. Za izračun plače potrebno je pročitati iznos koji je definiran za određeno radno mjesto na koje je zaposlenik zaposlen.



Slika 11: Prikaz tablica bonusi i godišnji odmor (Izvor: vlastita izrada)

Tablice Bonusi i Godišnji odmor nam prikazuju informacije koje su generirane za pojedinog radnika prilikom unošenja radnika u sustav. Vrijednosti u navedene dvije tablice ne unaša korisnik aplikacije ručno već ih generira aplikacija automatizirano.



Slika 12: Prikaz tablica za izračun bonusa i GO(Izvor: vlastita izrada)

Tablica Izračun bonusa je definirana na samom početku korištenja aplikacije kao i tablica za izračun godišnjeg odmora. Navedene dvije tablice koriste aplikaciji kako bi aplikacija iz njih uzimala potrebne informacija kada se okinu okidači za izračun godišnjeg odmora, odnosno bonusa.

5.2.1. Programski kod

```
private void btnDodaj_Click(object sender, EventArgs e)
{
    Cursor.Current = Cursors.WaitCursor;
    try
    {
        radnikBindingSource.EndEdit();
        radnikTableAdapter.Update(this.dataSet1.radnik);
        MessageBox.Show("Uspjesno spremljeno.", "Message", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)

    {
        MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    Cursor.Current = Cursors.Default;

}
```

Slika 13: Prikaz programskog koda za unos u tablicu(Izvor: vlastita izrada)

Metoda koja nam omogućava upis novih zapisa na spojenu bazu podataka, kao i modifikaciju podataka unutar tablice Radnik. Metodom je također zadano ispisivanje poruke korisniku o uspješnosti realizirane transakcije.

6. Zaključak

Kroz rad sam se upoznao sa modeliranjem aktivnih i temporalnih baza podataka. Prva odlika navedenih vrsta baza se visi već u oblikovanju ERA modela. Temporalna komponenta je uvedena kroz dodatne vremenske atribute. Aktivna komponenta baze podataka je uvedena kroz korištenje okidača i pratećih funkcija. Aktivna komponenta baza podataka uvelike olakšava korištenje aplikacije krajnjim korisnicima, zbog toga što može automatizirati radnje koje bi inače korisnici morali ručno raditi. Napomenimo da odrađivanjem radnji ručno od strane korisnika može se potkrasti pogrešno unašanje podatak (što nenamjerno, što namjerno). Time postižemo dodatnu razinu sigurnosti ispravnog rada aplikacije.

Za izradu aplikacije je korišteno nekoliko alata. Prvenstveno je tu online alat Draw.io u kojem je izrađen model baze podataka, odnosno era model. Potom je korišten sql server preko kojeg je „dignut“ server u kojem se modelirala baza za potrebe aplikacije. Sama implementacija baze je obavljana preko PgAdmin-a te sql konzole. Aplikacija je izrađena pomoću Windows formi u Visual Studio 2019 okruženju.

U benefite ovog projekta bih prvenstveno istaknuo upoznavanje sa modeliranjem temporalne i aktivne komponente u postgreSQL-u, te konstrukciju okidača i pratećih funkcija koje vrše automatizirane radnje u aplikaciji te uvelike olakšavaju krajnjim korisnicima korištenje aplikacije.

Popis literature

[1] https://www.devart.com/odbc/postgresql/download.html

[2] https://www.postgresqltutorial.com/

[3]https://www.youtube.com/watch?v=ui91KG74gro

[4] https://www.youtube.com/watch?v=Fe9hfmnnMjo

[5]https://www.youtube.com/watch?v=KN0arGw5mWQ

[6] https://www.youtube.com/watch?v=ZU16HQDz-uM

Popis slika

Popis slika treba biti izrađen po uzoru na indeksirani sadržaj, te upućivati na broj stranice na kojoj se slika može pronaći.