

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Matej Krznarić

**APLIKACIJA ZA TEMPORALNO MJERENJE
I OBRAČUN TROŠKOVA RADA**

PROJEKT IZ KOLEGIJA TEORIJA BAZA PODATAKA

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Matej Krznarić

Matični broj: 44130/15-R

Studij: Baze podataka i baze znanja

APLIKACIJA ZA TEMPORALNO MJERENJE I OBRAĆUN TROŠKOVA RADA

PROJEKT IZ KOLEGIJA TEORIJA BAZA PODATAKA

Mentor/Mentorica:

Izv. prof. dr. sc. Markus Schatten

Varaždin, siječanj 2020.

Matej Krznarić

Izjava o izvornosti

Izjavljujem da je moj PROJEKT IZ KOLEGIJA TEORIJA BAZA PODATAKA izvorni rezultat mog 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

Tema ovog projekta iz kolegija Teorija baza podataka je Aplikacija za temporalno mjerjenje i obračun troškova rada. Kroz ovaj projekt prikazat će implementaciju aplikacije i baze podataka koja se koristi za izračunavanje troškova rada studenata u Student servisu. Također, bit će tu i teorijskog dijela, model baze podataka, primjeri korištenja aplikacije i ostalo.

Ključne riječi: aplikacija; troškovi; baza; podataka; temporalno; student; servis;

Sadržaj

1.	Opis aplikacijske domene	1
2.	Teorijski uvod	2
3.	Model baze podataka	3
4.	Implementacija	4
4.1.	Kreiranje baze podataka	4
4.2.	Okidači	5
4.3.	Aplikacija	7
5.	Primjeri korištenja	9
6.	Zaključak	13
7.	Popis literature	14

1. Opis aplikacijske domene

U ovom poglavlju opisat će aplikacijsku domenu. Za aplikacijsku domenu za koju sam napravio aplikaciju za izračun troškova odabrao sam Student servis i način rada student servisa. Student servis je posrednik između studenata i poslodavaca. U aplikaciju se unose traženi podaci i ona računa iznos koji je student zaradio, iznos provizije i ukupni iznos. Struktura aplikacijske domene sastoji se od šest entiteta, tj. koncepata, a to su:

- fakultet
- student
- partner
- zaposlenik
- račun.

Kako bi se aplikacija uopće mogla koristiti, zaposlenik se mora prijaviti sa svojim podacima. Tek tada može koristiti aplikaciju jer je namijenjena samo zaposlenicima student servisa. Dakle, svaki zaposlenik ima svoje korisničko ime.

Nadalje, zaposlenik koji se uspješno prijavio ima pristup glavnoj formi u kojoj može odabratи gumbove: Fakulteti, Studenti, Partneri i Računi. Pritisom na određeni gumb otvara se forma u kojoj su implementirani pregled,dodavanje, ažuriranje i brisanje određenih stavki. Iznimka je forma s računima na kojoj imamo i prikaz troškova. Troškovi se prikazuju na način da kada se odabra određeni račun, izlistaju se: iznos koji je student zaradio, iznos provizije koju uzima Student servis i ukupan iznos koji je potrebno platiti.

Što se tiče relacija, koncept student ima vanjski ključ koji sadrži id fakulteta, uz njega ima još i ime, prezime, jmbag i iban. Koncept student sadrži vanjski ključ fakulteta zbog toga što svaki student studira na nekom fakultetu.Koncept fakultet sadrži naziv fakulteta i adresu. Najvažniji koncept ovdje je zapravo račun koji vanjske ključeve koji sadrže id partnera koji je zapravo poslodavac. Nadalje, sadrži id studenta koji je odradio posao, ali i id zaposlenika koji je izradio račun. Također, račun još ima datum izrade, naziv posla, satnicu, datum početka obavljanja posla i datum završetka obavljanja posla. Koncept partner se sastoji od naziva, adrese, oiba i ibana.

2. Teorijski uvod

Kao što već znamo od ranije, baza podataka predstavlja skup podataka, ograničenja i operacija koji predstavljaju neki aspekt realnog svijeta. Ona se zasniva na nekom modelu podataka. U ovom projektu spominjat ćemo baze podataka kroz kontekst aktivnih baza podataka, relacijskih baza podataka i temporalnih baza podataka. Dakle, u ovom projektu bit će prikazani podaci, ograničenja i operacije koje predstavljaju rad student servisa.

Kada govorimo o aktivnim bazama podataka, moramo spomenuti pojam Active Database Management System koji se definira kao sposobnost da se automatski reagira na nastale situacije u bazi podataka i izvan nje. Svi tradicionalni sustavi za upravljanje bazom podataka su pasivni. To znači da baze podataka samo izvršavaju naredbe koje joj korisnik ili neka aplikacija zadaju. S druge strane aktivne baze podataka pravovremeno prate stanje sustava i reagiraju na situaciju. To je zbog toga jer aktivne baze podataka posjeduju mehanizme koji im omogućavaju da automatski reagiraju na događaje koji se odviju unutar, ali i izvan baze podataka. Najpoznatije aktivne baze podataka koji su jedan oblik nadogradnje relacijskih baza podataka su Postgres i Starburst.

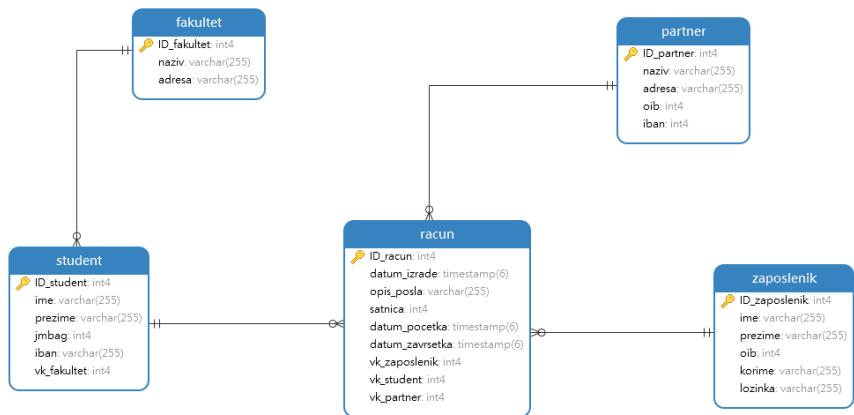
Temporalne baze podataka su nešto potpuno drugačije od ostalih baza podataka jer imaju različit jedan ključni aspekt, a to je da se svi podaci koji su zapisani u bazi podataka gledaju kroz kontekst vremena. Ova vrsta baza podataka nije isključivo vezana za relacijske baze podataka kao što su PostgreSQL, nego se mogu koristiti i kod nerelacijskih. Podaci zapisani u temporalnu bazu podataka imaju vremensku komponentu. Vremenska komponenta omogućuje prikaz kad su neki podaci spremljeni u bazu, ili od kada ti podaci vrijede.

PostgreSQL je objektno - relacijski sustav za upravljanje bazama podataka. Otvorenog je koda i svatko se može koristiti s njim. Radi na većini operacijskih sustava. Razvijen je od strane PostgreSQL Global Development Group. PostgreSQL omogućuje kreiranje i ažuriranje pogleda, pohranjenih procedura, funkcija, okidača, indexa.

Name	Storage Size	Description	Low Value	High Value	Resolution
<code>timestamp [(p)] [without time zone]</code>	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond / 14 digits
<code>timestamp [(p)] with time zone</code>	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond / 14 digits
<code>date</code>	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day
<code>time [(p)] [without time zone]</code>	8 bytes	time of day (no date)	00:00:00	24:00:00	1 microsecond / 14 digits
<code>time [(p)] with time zone</code>	12 bytes	times of day only, with time zone	00:00:00+1459	24:00:00-1459	1 microsecond / 14 digits
<code>interval [fields] [(p)]</code>	16 bytes	time interval	-178000000 years	178000000 years	1 microsecond / 14 digits

Slika 1: Datumski i vremenski tipovi u sustavu PostgreSQL

3. Model baze podataka



Slika 2: Prikaz modela baze podataka

4. Implementacija

U ovom poglavlju prikazat će implementaciju baze podataka i aplikacije. Također prikazat će i objasniti triggere, tj. okidače koje sam koristio i za što služe.

4.1. Kreiranje baze podataka

U nastavku slijedi programski kod, odnosno sadržaj skripte StudentServis.sql:

```
CREATE TABLE "fakultet" (
    "ID_fakultet" int4 NOT NULL,
    "naziv" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    "adresa" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    CONSTRAINT "fakultet_pkey" PRIMARY KEY ("ID_fakultet")
);
ALTER TABLE "fakultet" OWNER TO "postgres";

CREATE TABLE "partner" (
    "ID_partner" int4 NOT NULL,
    "naziv" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    "adresa" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    "oib" int4 NOT NULL,
    "iban" int4 NOT NULL,
    CONSTRAINT "partner_pkey" PRIMARY KEY ("ID_partner")
);
ALTER TABLE "partner" OWNER TO "postgres";

CREATE TABLE "racun" (
    "ID_racun" int4 NOT NULL,
    "datum_izrade" timestamp(6) NOT NULL,
    "opis_posla" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    "satnica" int4 NOT NULL,
    "datum_pocetka" timestamp(6) NOT NULL,
    "datum_zavrsetka" timestamp(6) NOT NULL,
    "vk_zaposlenik" int4 NOT NULL,
    "vk_student" int4 NOT NULL,
    "vk_partner" int4 NOT NULL,
    CONSTRAINT "racun_pkey" PRIMARY KEY ("ID_racun")
);
ALTER TABLE "racun" OWNER TO "postgres";

CREATE TABLE "student" (
    "ID_student" int4 NOT NULL,
    "ime" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    "prezime" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    "jmbag" int4 NOT NULL,
    "iban" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    "vk_fakultet" int4 NOT NULL,
    CONSTRAINT "student_pkey" PRIMARY KEY ("ID_student")
);
ALTER TABLE "student" OWNER TO "postgres";
```

```

CREATE TABLE "zaposlenik" (
    "ID_zaposlenik" int4 NOT NULL,
    "ime" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    "prezime" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    "oib" int4 NOT NULL,
    "korime" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    "lozinka" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
    CONSTRAINT "zaposlenik_pkey" PRIMARY KEY ("ID_zaposlenik")
);
ALTER TABLE "zaposlenik" OWNER TO "postgres";

ALTER TABLE "racun" ADD CONSTRAINT "vk_partner" FOREIGN KEY ("vk_partner")
    REFERENCES "partner" ("ID_partner") ON DELETE CASCADE ON UPDATE RESTRICT;
ALTER TABLE "racun" ADD CONSTRAINT "vk_student" FOREIGN KEY ("vk_student")
    REFERENCES "student" ("ID_student") ON DELETE CASCADE ON UPDATE RESTRICT;
ALTER TABLE "racun" ADD CONSTRAINT "vk_zaposlenik" FOREIGN KEY ("vk_zaposlenik")
    REFERENCES "zaposlenik" ("ID_zaposlenik") ON DELETE CASCADE ON UPDATE RESTRICT;
ALTER TABLE "student" ADD CONSTRAINT "vk_fakultet" FOREIGN KEY ("vk_fakultet")
    REFERENCES "fakultet" ("ID_fakultet") ON DELETE CASCADE ON UPDATE RESTRICT;

```

4.2. Okidači

U nastavku ću prikazati okidače i objasniti za što se koriste. Prvi okidač koji sam implementirao je za provjeru minimalne satnice. Naime, od 1. siječnja 2020. godine po Zakonu o obavljanju studentskih poslova studenti moraju imati minimalnu satnicu od 25,39 HRK. To provjerava sljedeći okidač:

```

CREATE OR REPLACE FUNCTION "public"."minimalna_satnica"()
RETURNS "pg_catalog"."trigger" AS $BODY$
BEGIN
    if (NEW."satnica") < 25.3 then
        RAISE EXCEPTION 'Minimalna_satnica_je_25,39_HRK.';
    else
        return new;
    end if;
$End;$BODY$
LANGUAGE plpgsql VOLATILE
COST 100

```

Nadalje, sljedeći okidač koji sam implementirao je onaj da datum početka obavljanja posla ne može biti manji od današnjeg, odnosno trenutnog datuma kada se račun izdaje. Taj okidač sam implementirao iz razloga što studenti ne mogu izdati ugovor za posao koji su ranije obavljali, nego ugovor moraju podići prije obavljanja posla.

```

CREATE OR REPLACE FUNCTION "public"."datum_pocetka"()
RETURNS "pg_catalog"."trigger" AS $BODY$
declare
var date;
BEGIN
select date(new.datum_pocetka) into var;

```

```

    if var < date(NOW())
then
    RAISE EXCEPTION 'Odaberite_datum_koji_je_veci_ili_jednak_danasnjem.';
else
    return new;
end if;
END$BODY$
LANGUAGE plpgsql VOLATILE
COST 100

```

Sljedeći okidač koji će prikazati je taj da datum završetka obavljanja posla ne može biti veći od zadnjeg dana od mjeseca u kojem je ugovor, odnosno račun u ovom slučaju podignut.

```

CREATE OR REPLACE FUNCTION "public"."datum_zavrsetka"()
RETURNS "pg_catalog"."trigger" AS $BODY$
declare
var date;
BEGIN
SELECT (date_trunc('MONTH', date(NOW())) + INTERVAL '1_MONTH')::DATE into
var;
if new.datum_zavrsetka > var
then
RAISE EXCEPTION 'Odaberite_datum_u_ovom_mjesecu!';
else
return new;
end if;
END$BODY$
LANGUAGE plpgsql VOLATILE
COST 100

```

Zadnji okidač koji sam implementirao je onaj koji provjerava da li je datum završetka manji od datuma pocetka obavljanja posla: Kod u nastavku:

```

CREATE OR REPLACE FUNCTION "public"."razlika_datumi"()
RETURNS "pg_catalog"."trigger" AS $BODY$
declare
var date;
BEGIN
SELECT date(new.datum_zavrsetka) into var;
if var < new.datum_pocetka
then
RAISE EXCEPTION 'Odabrali_ste_datum_zavrsetka_koji_je_manji_od_datuma_
pocetka!';
else
return new;
end if;
END$BODY$
LANGUAGE plpgsql VOLATILE
COST 100

```

4.3. Aplikacija

Aplikacija je izrađena u Visual Studio C# .NET Framework tehnologiji, a u nastavku će biti prikazani neki dijelovi koji su implementirani.

```
private string connectionString = String.Format("Server={0};Port={1};" +
    "User Id={2};Password={3};Database={4};",
    "localhost", 5432, "postgres",
    "Cibalial1", "StudentServis");
private NpgsqlConnection conn;
private string sql;
private NpgsqlCommand cmd;
private DataTable dt;
```

Slika 3: Povezivanje s bazom podataka

```
public void OsvjeziFakultete()
{
    conn = new NpgsqlConnection(connectionString);
    conn.Open();

    sql = @"SELECT * FROM fakultet";
    cmd = new NpgsqlCommand(sql, conn);
    dt = new DataTable();
    dt.Load(cmd.ExecuteReader());

    conn.Close();

    dgvFakulteti.DataSource = null;
    dgvFakulteti.DataSource = dt;
}
```

Slika 4: Metoda koja osvježava prikaz fakulteta

```
private void btnDodaj_Click(object sender, EventArgs e)
{
    conn = new NpgsqlConnection(connectionString);
    conn.Open();
    sql = @"INSERT INTO fakultet (naziv, adresa) VALUES ('"+txtNaziv.Text+"', '"+txtAdresa.Text+"')";
    cmd = new NpgsqlCommand(sql, conn);
    cmd.ExecuteNonQuery();
    conn.Close();

    OsvjeziFakultete();
    txtNaziv.Text = null;
    txtAdresa.Text = null;
}
```

Slika 5: Unos novog fakulteta

```

private void btnIzmjeni_Click(object sender, EventArgs e)
{
    conn = new NpgsqlConnection(connectionString);
    conn.Open();
    sql = @"UPDATE fakultet SET naziv = '" + txtNaziv.Text + "', adresa=''" + txtAdresa.Text + "' WHERE id_fakultet=" + dgvFakulteti.SelectedCells[0].Value + ";";
    cmd = new NpgsqlCommand(sql, conn);
    cmd.ExecuteNonQuery();
    conn.Close();

    OsvjeziFakultete();
    txtNaziv.Text = null;
    txtAdresa.Text = null;
}

```

Slika 6: Ažuriranje fakulteta

```

private void btnObrisi_Click(object sender, EventArgs e)
{
    conn = new NpgsqlConnection(connectionString);
    conn.Open();
    sql = @"DELETE FROM fakultet WHERE id_fakultet=" + dgvFakulteti.SelectedCells[0].Value + ";";
    cmd = new NpgsqlCommand(sql, conn);
    cmd.ExecuteNonQuery();
    conn.Close();

    OsvjeziFakultete();
    txtNaziv.Text = null;
    txtAdresa.Text = null;
}

```

Slika 7: Brisanje fakulteta

```

private void btnPrijava_Click(object sender, EventArgs e)
{
    conn = new NpgsqlConnection(connectionString);
    conn.Open();
    sql = @"SELECT * FROM zaposlenik WHERE korime ='" + txtKorime.Text + "' AND lozinka=''" + txtLozinka.Text + "'"; 
    cmd = new NpgsqlCommand(sql, conn);
    dt = new DataTable();
    dt.Load(cmd.ExecuteReader());

    if (dt.Rows.Count == 1)
    {
        frmGlavna glavna = new frmGlavna();
        this.Hide();
        glavna.Show();
    }
    else
    {
        MessageBox.Show("Neispravni korisničko ime ili lozinka!");
    }
    conn.Close();
}

```

Slika 8: Prijava u aplikaciju

5. Primjeri korištenja

Sljedeće slike prikazuju primjere korištenja aplikacije koju sam implementirao. Prikazane su sve forme, od prijave, preko forme s fakultetima, forme sa studentima, partnerima i na kraju ona najvažnija - forma s računima koja prikazuje troškove.

The screenshot shows a web browser window with the title 'Student servis'. The page contains fields for 'Korisničko ime:' (admin) and 'Lozinka:' (*****). A large blue button at the bottom right is labeled 'Prijavi se'.

Prijava

— □ ×

Student servis

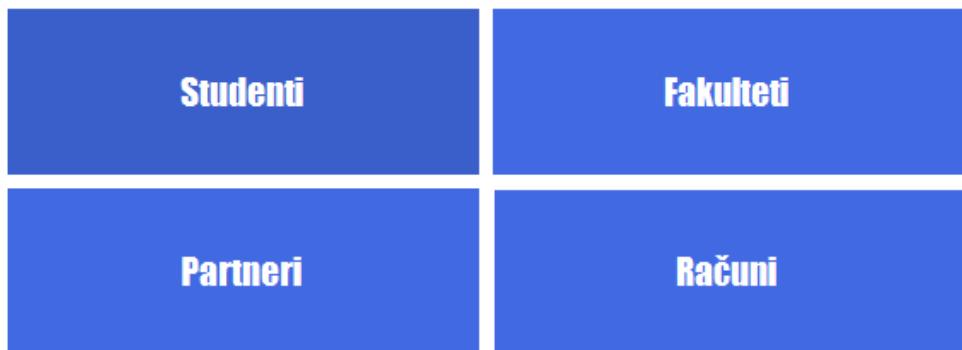
Korisničko ime:

Lozinka:

Prijavi se

Slika 9: Prijava u aplikaciju

Student servis



Slika 10: Glavna forma - izbornik

Studenti

Ime:

Prezime:

JMBAG:

IBAN:

Fakultet:
 FER

	id_student	ime	prezime	jmbag	iban	vk_fakultet
▶	1	Matej	Krznaric	12345	HR5016648542...	1
	3	Berislav	Baracic	44	HR0	2
	4	Mateo	Kidemet	77	HR55	9
	5	Matej	Kolaric	9	HR7	5

Dodaj

Izmjeni

Obrisi

Slika 11: Forma studenti

Fakulteti

Fakulteti

	id_fakultet	naziv	adresa
▶	2	FER	Zagreb
	5	MEDICINSKI	Zagreb
	1	FOI	Varazdin
	8	RGN	Zagreb
	9	ETF	Osjek
	10	PROMET	Zagreb

Naziv fakulteta:

Adresa:

Dodaj **Izmijeni**

Obriši

Slika 12: Forma fakulteti

Partneri

Partneri

	id_partner	naziv	adresa	oib	iban
▶	1	Solvis	Varazdin	12344567899	HR1234567891123...
	3	Restoran Angelus	Varazdin	12344567844	HR1234567891123...
	2	SC	Varazdin	1234567899	HR55542321466687

Naziv:

Adresa:

OIB:

IBAN:

Dodaj **Izmijeni**

Obriši

Slika 13: Forma partneri

Računi

Računi

	id_račun	datum_izrade	opis_posla	satnica	datum_pocetka	datum_zavrsetka	vk_zaposlenik	vk_student	vk_partner
	5	26.1.2020.	receppcionar	20,5	1.1.2020.	31.1.2020.	2	1	2
	2	26.1.2020.	tehnicar	25,88	1.1.2020.	31.1.2020.	1	1	1
	8	26.1.2020. 19.29	tenicar	26	1.1.2020.	31.1.2020.	1	1	1
	▶ 10	26.1.2020. 20:09	pomočni poslovi	27	26.1.2020.	31.1.2020.	1	1	1
	13	26.1.2020.20:26	kuhar	27,77	28.1.2020.	27.1.2020.	1	1	1

Opis posla:

Satnica:

Datum početka obavljanja posla:

Datum završetka obavljanja posla:

Student:

Partner:

Odgovorna osoba za izdavanje:

Troškovi

Zarada: **1.080,00** **HRK**

Provizija: **194,40** **HRK**

Ukupno: **1.274,40** **HRK**

Dodaj **Izmjeni**

Obriši

Slika 14: Forma računi

6. Zaključak

Kroz ovaj projekt u kojemu sam implementirao i kreirao aplikaciju koja obračunava troškove studenata naučio sam puno toga o temporalnim bazama podataka, ali i aktivnim bazama podataka. Velika prednost temporalnih baza podataka je to što možemo lakše manipulirati s podacima ukoliko iskoristimo taj temporalni faktor. Upravo taj faktor mi je jako olakšao izradu ove aplikacije.

PostgreSQL je tehnologija koje se izvrsno pokazala pri izradi ovoga projekta, odnosno za izradu temporalnih i aktivnih baza podataka. Alat koji se koristio za izradu svih relacija i okidača je Navicat koji se pokazao jako dobar. U njemu sam čak i uspio izgenerirati model baze podataka. Jedina mana mu je to što je besplatan jedino na probni rok od 14 dana.

Također, u ovome projektu kroz okidače sam pokušao ograničiti neke od stvari koje su propisane Zakonom o obavljanju studentskih poslova, kao što su npr. minimalna satnica, datum obavljanja posla, ali i neke logične stvari kao npr. da datum završetka obavljanja posla mora biti veći od datuma početka obavljanja posla.

7. Popis literature

- Temporalne baze podataka

https://bib.irb.hr/datoteka/895669.1-PITUPsiniaKajin-Zavrni_rad-Temporalnebazeodataka.pdf

preuzeto 26.01.2020.

- Aplikacija za upravljanje skladištem temeljena na aktivnim bazama podataka

https://bib.irb.hr/datoteka/895660.1-tomislav-bobinac-dipl-rad.pdf

preuzeto 26.01.2020.

- PostgreSQL dokumentacija

https://www.postgresql.org/docs/9.1/datatype-datetime.html

preuzeto 26.01.2020.