

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

Dominik Langer

SKLADIŠTE AUTOMOBILA

PROJEKT

Varaždin, 2020.

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

Dominik Langer

JMBAG 0016116705

Studij: Organizacija poslovnih sustava

SKLADIŠTE AUTOMOBILA

PROJEKT

Mentor:

Izv. prof. dr. sc. Markus Schatten

Varaždin, kolovoz 2020.

Dominik Langer

Izjava o izvornosti

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

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Cilj ovog projekta je kreiranje automatiziranog skladišta koje ne zahtjeve preveliku ulogu samog korisnika. Tema projekta je: "Aplikacija za vođenje statistike skladišta i planiranje zaliha (strategije upravljanja zalihama - minimalne/maksimalne količine, vremena između nabavki itd. - obavezna implementacija strategije upravljanja zalihama) - Aktivne + Temporalne baze podataka - PostgreSQL + Desktop grafičko sučelje sučelje po želji". Korištenjem aktivnih i temporalnih baza podataka korisniku sam omogućio uvid u prijašnja stanja na skladištu i promjene koje su se događale, a aktivne baze podataka služe kako bi se određene aktivnosti unutar skladišta izvršavale automatski. Na ovaj način smanjila se uloga i obveze samog korisnika prema bazi podataka, a to je glavni cilj našeg projekta, kreirati aplikaciju koja olakšava vođenje skladišta. Odlučio sam se za skladište automobila zbog interesa prema istima, također zanimalo me kako bi skladište izgledalo kada bi se automatiziralo. Za izradu projekta koristio sam alate kao što su: Microsoft Visual Studio, pgAdmin 4 i Navicat 15.

Ključne riječi: Temporalne, Aktivne, Baze, Navicat, Automobili, SKladište, Visual Studio, Okidači

Sadržaj

1. Opis aplikacijske domene	1
2. Teorijski uvod	2
2.1. Temporalne baze podataka	2
2.2. Aktivne baze podataka	2
3. Model baze podataka	3
4. Implementacija	5
4.1. Stvaranje baze podataka	5
4.2. Proces spajanja na bazu	7
4.2.1. Čitanje podataka iz baze	8
4.2.2. Zapisivanje u bazu i ažuriranje	9
4.3. Automatizacija	10
4.3.1. Dodavanje novog automobila na skladište	10
4.3.2. Vođenje evidencije skladišta	10
4.3.3. Proces naručivanja automobila	11
5. Prmimjeri korištenja	13
5.1. Početni izbornik	13
5.2. Stanje na skladištu	14
5.3. Automobili	15
5.4. Dodavanje automobila	15
5.5. Nabava vozila i naručivanje	16
5.6. Statistika proizvođača	18
5.7. Statistika gorivo	19
5.8. Statistika naručivanja	20
6. Zaključak	21
Popis literature	22
Popis slika	23
Popis popis tablica	24
1. Prilog 1	25

2. Prilog 2	26
------------------------------	-----------

1. Opis aplikacijske domene

Tema mojeg projektnog zadatka je izrada aplikacije za vođene statistike skladišta i planiranje zaliha. Izradio sam aplikaciju koja omogućuje skladištenje automobila, praćenje, promjene i dodavanje novih automobila. Kako sama mogućnost naručivanja automobila za realno skladište ne predstavlja veliku vrijednost izradio sam i statistiku koju sam grafički nastojao što više približiti korisniku. Aplikacija je izrađena uz pomoć alata **Microsoft Visual Studio** [1], programski jezik koji se koristio prilikom izrade je **C#**. Kako bi mogao voditi evidenciju skladišta bilo je potrebno izraditi bazu podataka na koju bi se mogao spojiti. Bazu sam izradio uz pomoć alata **pgAdmin 4** [2] i **Navicat 15** [3]. Navicat mi je služio kao pomoć prilikom testiranja raznih upita, prilikom kreiranja okidača i funkcija. Baza je kreirana tako da sadrži sve bitne informacije koje bi jedno skladište automobila moglo koristiti. Informacije kao što su: model automobila, proizvođač automobila, tip goriva koje se koristi u automobilu, snaga motora i mnoge druge. Nakon što su osnovne karakteristike automobila kreirane potrebno je voditi i evidenciju samog skladišta. Vrlo je važno znati koliko se automobila trenutno nalazi na skladištu. Osim informacije o količini vozila aplikacija mora voditi brigu i o minimalnim količinama koje je potrebno održavati unutar skladišta. Kod svakog automobila postoji razina ispod koje kada se dođe kreće automatiziran proces naručivanja. Od korisnika aplikacije traži se samo da vodi evidenciju automobila na skladištu i da evidentira zaprimljenu količinu automobila koje je dobio na skladište kako se kasnije ne bi dogodilo da se u procesu naručivanja naručila jedna količina automobila, a tek nakon nekog vremena ispostavilo da ta količina nije zaprimljena. Ovaj oblik aplikacije pogodan je za prodavače rabljenih automobila ili za veću automobilsku kuću koja mora imati veće količine zaliha kako bi mogla zadovoljiti potrebe svojih kupaca.

Za realizaciju ovog projekta koristio sam aktivne i temporalne baze podataka, a domena nad kojima su implementirane su: Stanje na skladištu, Evidencija skladišta, Nabava automobila i Narudžbenica. Tako sam u potpunosti zaokružio proces naručivanja i dobio potpunu automatizaciju.

2. Teorijski uvod

2.1. Temporalne baze podataka

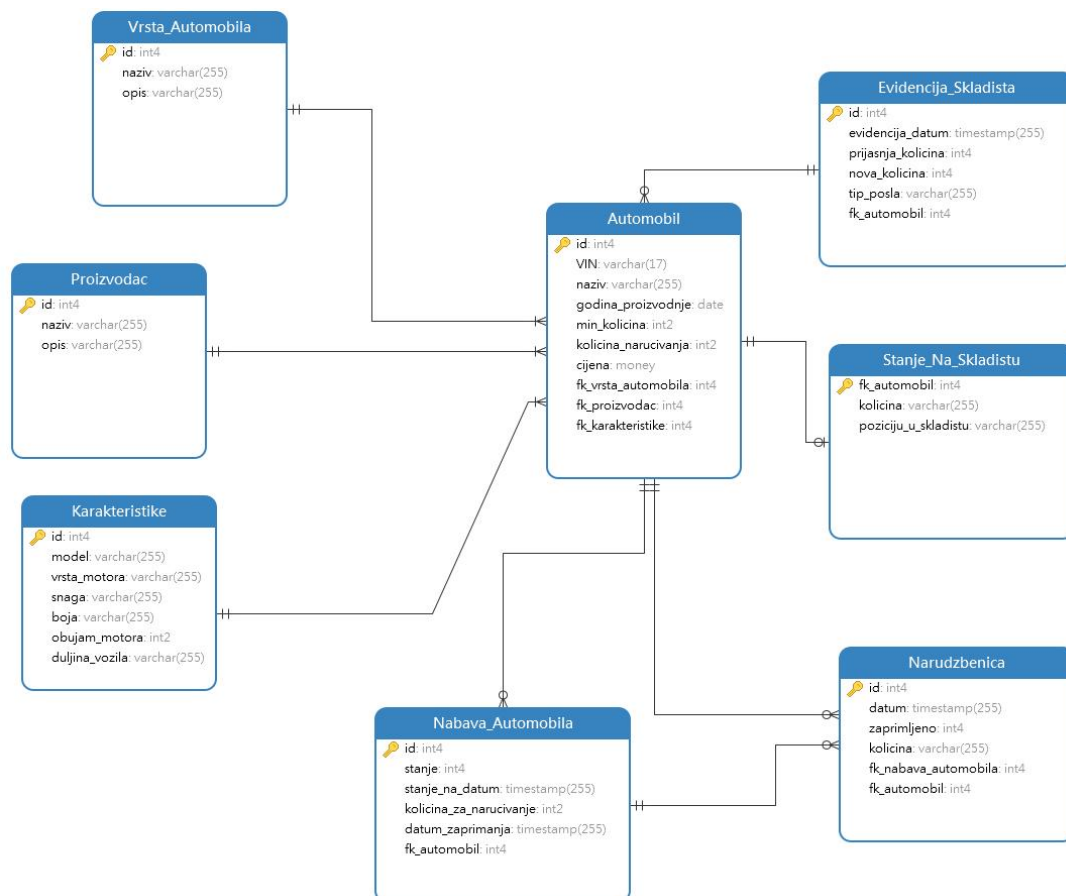
Ako gradimo bazu podataka i želimo unutar nje imati najnovije podatke tada posežemo za konvencionalnim bazama podataka. Kako u našem slučaju trebamo i prijašnje stanje podataka, tj. prijašnje stanje na skladištu automobila moramo implementirati i temporalne baze podataka. Ovaj oblik baza podataka zaslužan je za bilježenje cijelog životnog ciklusa podataka. Ovdje nije samo bitno trenutno stanje podataka, povežemo li to s našim primjerom nije nam samo bitno koliko automobila imamo sada na skladištu, nego nam je od velike važnosti mogućnost uvida u povijest tih objekata [4]. Pogledamo li temporalni relacijski model (MT) uočavamo da je strukturiran $MT = (ST, OT, IT)$. Strukturna komponenta (ST) sastoji se od temporalnih relacija, operativna komponenta (OT) sadrži temporalnu relacijsku algebru, a integritetna komponenta (IT) čini uvjete integriteta kojima se ograničavaju stanja temporalnih relacija [4].

2.2. Aktivne baze podataka

"Uvođenjem aktivne komponente u relacijske baze podataka korištenjem produkcijskih pravila omogućava se niz naprednih karakteristika baza podataka kao što su napredna integritetna ograničenja, održavanje deriviranih podataka, okidači, upozorenja, zaštita, upravljanje inačicama i sl. ali i izgradnju naprednih sustava poput baza znanja i ekspernih sustava." [4]. To je jedna vrsta konvencionalnih baza podataka koja sadrži mehanizme koji mogu reagirati na unutarnje ili vanjske podražaje te na temelju njih izvršiti određenu akciju. Unutar svojeg projekta ovakav oblik baza će iskoristiti kod kreiranja okidača. Okidači nam omogućuju da se neke instrukcije izvrše automatski. Ovakvim pristupom smanjujemo obavezu korisnika aplikacije i olakšavamo mu korištenje aplikacije [5].

3. Model baze podataka

Na slici 1 prikazan je model baze podataka (**ERA**) koji sam koristio prilikom izrade aplikacije. za izradu modela uvelike mi je pomogao već spomenuti alat Navicat koji svojom jednostavnošću i grafičkim prikazom uvelike olakšava kreiranje tablica i rad na samoj bazi. Model se sastoji od 8 entiteta i 4 okidača.



Slika 1: ERA model (Vlastita izrada)

Popis svih tablica koje se nalaze u ERA modelu:

- **Automobil** - središnja je tablica modela i sadrži sve informacije o automobilima koji se nalaze na skladištu. Tablica Automobil sadrži jedinstveni "**id**" automobila prema kojemu ga i identificiramo unutar skladišta. Dodatno su definirane informacije kao što su **VIN**, **Naziv**, **Godina proizvodnje** i nekima možda i najbitniji aspekt, a to je **cijena**. Osim informacija koje jedinstveno definiraju automobil moramo i definirati elemente koji će se odnositi na skladište i vođenje istoga. Za skladište neophodne su informacije o **minimalnim količinama** i **količini naručivanja**. Minimalne količine nam predstavljaju vrijednost ispod koje je potrebno naručiti vozila, a količina naručivanja definira koliko ustvari moramo naručiti.
- **Proizvođač** - Automobile možemo dodatno razlikovati po proizvođačima. Svaki proizvođač ima određenu reputaciju i uz njegov naziv veže se određena razina pouzdanosti u marku ili određena doza skepticizma prema marci.

- **Vrsta automobila** Kako je prošlo jako puno godina od izuma prvog automobila tako se i je i sama količina različitih vrsta automobila povećavala. Danas na tržištu postoji nebrojeno puno vrsta, a neke od najprodavanijih u zadnje vrijeme su **SUV** automobili. Nešto viši i robusniji od standardnih vozila samim korisnicima nude osjećaj sigurnosti. Neke od ostalih vrsta koje možemo pronaći na skladištu su: limuzine, karavani, kompakti, terenci i mnogi drugi.
- **Karakteristike** - Kako je svaki automobil jedinstveno definiran svojim id-jem tako se mu i jedinstvene karakteristike koje ga definiraju. Model, vrsta motora, snaga, boja, obujam motora i duljina vozila jedne su od glavnih karakteristika pomoću kojih možemo razlikovati više modela i uspoređivati ih prilikom kupnje ili prodaje.
- **Stanje na skladištu** - Svaki automobil je na skladište povezan pomoću svojeg id-a, sadrži jedinstvenu količinu na skladištu i poziciju u skladištu. Pozicija u skladištu implementirana je kako bi se zaposlenici mogli što bolje snalaziti prilikom pronalaska određenog automobila. Ovdje možemo dodavati ili smanjivati količinu koju imamo na skladištu.
- **Evidencija skladišta** - Evidencija je ovdje kako bi se moglo detaljno vidjeti kada i za koliko se dogodila promjena na skladištu. Jesu li se dodali ili oduzeli određeni automobili. Koliko je stanje bilo prije promjene i koliko je stanje nakon promjene.
- **Nabava automobila** - Padne li stanje automobila ispod minimalne količine odmah se započinje proces nabave automobila. Bilježi se stanje na skladištu i uz to stanje definira se datum, vidimo i količinu koja se treba naručiti. Datum zaprimanja se neće upisati sve dok osoba unutar narudžbenice ne unese koliko je automobila zaprimila i kada.
- **Narudžbenica** - Sadrži naziv automobila koji je naručen, korisnik definira koliko je automobila zaprimio, a u stupcu do piše kolika je količina naručena. Vidimo i ključeve prema automobilu i nabavi automobila.

4. Implementacija

Sam proces izrade baze podataka sam provodio u dva programa pgAdmin 4 i Navicat 15. Oba programa su dosta slična, ali Navicat 15 ima širi spektar funkcija koje korisnik može koristiti prilikom izrade baza podataka. Naravno sam proces kariranja početne baze mora se napraviti unutar aplikacije pgAdmin 4, gdje se nakon kreiranja baze podataka unutar alata Navicat spajamo na istu bazu i preuzimamo kontrolu nad njom. Dostupne su nam brojne funkcije kao što su kreiranje tablica, okidača, funkcija, unos podataka preko grafičkog sučelja. Sama aplikacija je intuitivno napravljena i za početnika možda nešto lakša za razumjeti nego pgAdmin 4 koji za neke funkcionalnosti traži od korisnika da koristi naredbeni redak.

4.1. Stvaranje baze podataka

Prema navedenom ERA modelu na slici 1 moguće je uz pomoć alata Navicat 15 kreirati tablice. Aplikacija sama na osnovu podataka koje smo unijeli u model kreira sql zapis na temelju kojega se zatim može kreirati baza. Svi tipovi podataka, ograničenja, veze i ključevi se generiraju na osnovu modela. U nastavku sam prikazao programski kod preko kojega sam kreirao bazu unutar alata pgAdmin4. za svaki "id" potrebno je kreirati sekvencu kako bi se vrijednosti automatski povećavale. Kreiranje sekvenci se također može provesti u oba programa, bitno je samo da se unutar default vrijednosti doda linija koja definira na koju se sekvencu referenciramo kao što je ovdje napravljeno za tablicu Proizvodac: "nextval('id_proizvodac_povecanje'::regclass)".

```
CREATE TABLE "Automobil" (  
    "id" int4 NOT NULL,  
    "VIN" varchar(17) NOT NULL,  
    "naziv" varchar(255) NOT NULL,  
    "godina_proizvodnje" date NOT NULL,  
    "min_kolicina" int2 NOT NULL,  
    "kolicina_narucivanja" int2,  
    "cijena" money,  
    "fk_vrsta_automobila" int4,  
    "fk_proizvodac" int4,  
    "fk_karakteristike" int4,  
    PRIMARY KEY ("id")  
);  
  
CREATE TABLE "Evidencija_Skladista" (  
    "id" int4 NOT NULL,  
    "evidencija_datum" timestamp(255) NOT NULL DEFAULT now(),  
    "prijasnja_kolicina" int4,  
    "nova_kolicina" int4 NOT NULL,  
    "tip_posla" varchar(255),  
    "fk_automobil" int4 NOT NULL,  
    CONSTRAINT "_copy_1" PRIMARY KEY ("id")  
);  
  
CREATE TABLE "Karakteristike" (  
    "id" int4 NOT NULL,  
    "naziv" varchar(255) NOT NULL,  
    "vrsta" int4 NOT NULL,  
    "cijena" money,  
    "min_kolicina" int2 NOT NULL,  
    "kolicina_narucivanja" int2,  
    "fk_proizvodac" int4,  
    "fk_automobil" int4,  
    "fk_karakteristike" int4,  
    PRIMARY KEY ("id")  
);
```

```

    "id" int4 NOT NULL,
    "model" varchar(255),
    "vrsta_motora" varchar(255),
    "snaga" varchar(255),
    "boja" varchar(255),
    "obujam_motora" int2,
    "duljina_vozila" varchar(255),
    PRIMARY KEY ("id")
);

CREATE TABLE "Nabava_Automobila" (
    "id" int4 NOT NULL,
    "stanje" int4 NOT NULL,
    "stanje_na_datum" timestamp(255) NOT NULL DEFAULT now(),
    "kolicina_za_narucivanje" int2 DEFAULT 0,
    "datum_zaprimanja" timestamp(255),
    "fk_automobil" int4 NOT NULL,
    CONSTRAINT "_copy_2" PRIMARY KEY ("id")
);

CREATE TABLE "Narudzbenica" (
    "id" int4 NOT NULL,
    "datum" timestamp(255),
    "zaprimljeno" int4,
    "kolicina" varchar(255),
    "fk_nabava_automobila" int4,
    "fk_automobil" int4,
    CONSTRAINT "_copy_3" PRIMARY KEY ("id")
);

CREATE TABLE "Proizvodac" (
    "id" int4 NOT NULL,
    "naziv" varchar(255),
    "opis" varchar(255),
    PRIMARY KEY ("id")
);

CREATE TABLE "Stanje_Na_Skladistu" (
    "fk_automobil" int4 NOT NULL,
    "kolicina" varchar(255),
    "poziciju_u_skladistu" varchar(255),
    PRIMARY KEY ("fk_automobil")
);

CREATE TABLE "Vrsta_Automobila" (
    "id" int4 NOT NULL,
    "naziv" varchar(255),
    "opis" varchar(255),
    PRIMARY KEY ("id"),
    CONSTRAINT "Jedinstven_naiv" UNIQUE ("naziv")
);

```

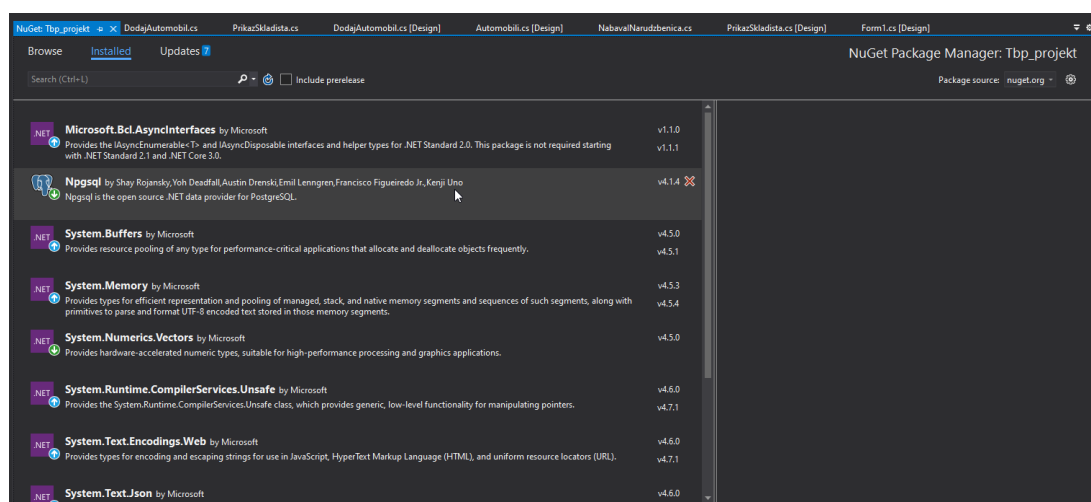
```

ALTER TABLE "Automobil" ADD CONSTRAINT "vk_vrsta_automobila" FOREIGN KEY ("
    fk_vrsta_automobila") REFERENCES "Vrsta_Automobila" ("id");
ALTER TABLE "Automobil" ADD CONSTRAINT "vk_proizvodac" FOREIGN KEY ("fk_proizvodac")
    REFERENCES "Proizvodac" ("id");
ALTER TABLE "Automobil" ADD CONSTRAINT "vk_karakteristike" FOREIGN KEY ("
    fk_karakteristike") REFERENCES "Karakteristike" ("id");
ALTER TABLE "Evidencija_Skladista" ADD CONSTRAINT "vk_automobil" FOREIGN KEY ("
    fk_automobil") REFERENCES "Automobil" ("id");
ALTER TABLE "Nabava_Automobila" ADD CONSTRAINT "vk_automobili" FOREIGN KEY ("
    fk_automobil") REFERENCES "Automobil" ("id");
ALTER TABLE "Narudzbjenica" ADD CONSTRAINT "vk_nabava_automobila" FOREIGN KEY ("
    fk_nabava_automobila") REFERENCES "Nabava_Automobila" ("id");
ALTER TABLE "Narudzbjenica" ADD CONSTRAINT "vk_automobili" FOREIGN KEY ("fk_automobil
    ") REFERENCES "Automobil" ("id");
ALTER TABLE "Stanje_Na_Skladistu" ADD CONSTRAINT "vk_automobil" FOREIGN KEY ("
    fk_automobil") REFERENCES "Automobil" ("id");

```

4.2. Proces spajanja na bazu

Nakon što smo izradili bazu podataka potrebno je napraviti konekciju na istu. Sama desktop aplikacija koja je izrađena u alatu Microsoft Visual studio, a kreirana kao Windows Forms aplikacija nema svrhu ako nije povezana na bazu. Kako bi se uspostavila komunikacija između aplikacije i baze potrebno je instalirati Npgsql NuGet pakete. Pakete možemo preuzeti tako da pritisnemo desni klik na naš "Solution" i odaberemo opciju "Manage Nuget Packagaes". Nakon ispunjenog koraka otvara nam se novi prozor unutar kojega preko pretraživača unosimo skup paketa koji nam je potreban u ovom slučaju. Aplikacija nas obavještava da su svi željeni paketi preuzeti.



Slika 2: Paket potreban za povezivanje na bazu (Vlastita izrada)

Kako smo ispunili sve preduvjete za spajanje na bazu možemo kreirati zasebnu klasu **Povezivanje.cs**. Unutar ove klase nalaze se podaci o konekciji na bazu i dvije metode koje nam služe za spajanje na bazu i prekid konekcije sa bazom.

```

public NpgsqlConnection connection;

public void Povezi()
{
    string ConnectionString = "Server=localhost;Port=5432;User_Id=postgres;
        Password=_marago747;Database=test;_";
    connection = new NpgsqlConnection(ConnectionString);
    connection.Open();

}

public void Odspoji()
{
    connection.Close();
}

```

4.2.1. Čitanje podataka iz baze

Kako je naša aplikacija povezana s bazom podataka to nam daje dodatnu mogućnost da određene informacije koje smo prije zapisali u bazu ili ćemo zapisati u istu možemo prikazati korisniku korištenjem upita. Za prikaz informacija najčešće se koristi "dataGridView" (DGV) koji se dodaje na formu. Osim upita preko kojega definiramo što želimo korisniku prikazati moramo se koristiti našim preuzetim paketom. Upit sam po sebi se neće izvršiti ako ga ne prosljedimo klasi "NpgsqlDataAdapter" za dohvaćanje podataka. U nastavku slijedi jedan primjer kako se podaci iz baze mogu prikazati unutar dataGridView-a. Dodatno su urešena zaglavlja koja se prikazuju unutar DGV-a linijama: "dgvAutomobili.Columns[0].HeaderText = "ID";"

```

string sql = "SELECT_" + "Automobil\".\"id\", \"Automobil\".\"VIN\", \"Automobil\".\"naziv\", \"Proizvodac\".\"naziv\", \"Vrsta_Automobila\".\"naziv\", \"Automobil\".\"cijena\", \"Karakteristike\".\"model\" +
    \"_\" + \"Karakteristike\".\"vrsta_motora\", \"Karakteristike\".\"snaga\"
    , \"Karakteristike\".\"boja\", \"Karakteristike\".\"obujam_motora\", \"Karakteristike\".\"duljina_vozila\"_\" +
    \"FROM_\" + \"Automobil\", \"Proizvodac\", \"Vrsta_Automobila\", \"Karakteristike\"_\"WHERE_\" + \"Automobil\".\"fk_proizvodac\" = \"Proizvodac\".\"id\"_\"AND_\" + \"Automobil\".\"fk_vrsta_automobila\"_\"_\" +
    \"_\" + \"Vrsta_Automobila\".\"id\"_\"AND_\" + \"Automobil\".\"fk_karakteristike\"_\"_\" + \"Karakteristike\".\"id\"_\"AND_\" + \"Automobil\".\"naziv\"_\"LIKE_\" +
    \"'\" + txtPretrazivanjeNaziv.Text + \"%'_\" + \"ORDER_BY_\" + \"ID\";

```

```

NpgsqlDataAdapter dataAdapter = new NpgsqlDataAdapter(sql,
    baza.connection);
DataSet automobiliDs = new DataSet();
dataAdapter.Fill(automobiliDs);
dgvAutomobili.DataSource = automobiliDs.Tables[0];
dgvAutomobili.Columns[0].HeaderText = "ID";
dgvAutomobili.Columns[1].HeaderText = "VIN";
dgvAutomobili.Columns[2].HeaderText = "Naziv_automobila";
dgvAutomobili.Columns[3].HeaderText = "đčProizvoa_automobila";

```

```

dgvAutomobili.Columns[4].HeaderText = "Vrsta_automobila";
dgvAutomobili.Columns[5].HeaderText = "Cijena_u_";
dgvAutomobili.Columns[6].HeaderText = "Model";
dgvAutomobili.Columns[7].HeaderText = "Gorivo";
dgvAutomobili.Columns[8].HeaderText = "Snaga_u_KW";
dgvAutomobili.Columns[9].HeaderText = "Boja";
dgvAutomobili.Columns[10].HeaderText = "Obujam_motora_CCM";
dgvAutomobili.Columns[11].HeaderText = "Duljina_vozila_u_m";

```

4.2.2. Zapisivanje u bazu i ažuriranje

Proces zapisivanja u bazu i ažuriranja nešto je drugačiji od procesa čitanja. Ovdje moramo posegnuti za novom klasom, a to je "NpgsqlCommand" koja nam omogućuje zapisivanje u bazu. Nakon što se zapisivanje izvršilo potrebno je osvježiti "dataGridView" kako bi se prikazale nove promjene.

```

pov.Povezi();
string sql = "INSERT INTO \"Automobil\" VALUES (DEFAULT, ' " + txtVIN
    .Text.ToString() + "', ' " + txtNazivAutomobila.Text.ToString() +
    "' , ' " + dtpGodina.Value.ToString("yyyy-MM-dd HH:mm:ss") + "' , ' "
    + int.Parse(txtMinKol.Text.ToString()) + " , " +
    " " + int.Parse(txtKolNarucivanja.Text.ToString()) + " , CAST(' " +
    txtCijena.Text.ToString() + " ' AS money) , ' " + int.Parse(
    txtVrsteAutomobilaId.Text.ToString()) + " , ' " + int.Parse(
    txtProizvodacId.Text.ToString()) + " , ' " + int.Parse(
    txtKarakteristikeId.Text.ToString()) + " );";
NpgsqlCommand command = new NpgsqlCommand(sql, pov.connection);
command.ExecuteNonQuery();
pov.Odspoji();

```

Ažuriranje baze se provodi na sličan način:

```

baza.Povezi();
string sql = "UPDATE \"Stanje_Na_Skladistu\" SET \"kolicina\" = "
    + kolicina + " , \"poziciju_u_skladistu\" = ' " + poz + "' "
    + " WHERE \"fk_automobil\" = " + kljuc + ";";
NpgsqlCommand command = new NpgsqlCommand(sql, baza.connection);
try
{
    command.ExecuteNonQuery();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "UPOZORENJE!",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
Osvjezi(baza.connection);
Osvjezi_Evidenciju(baza.connection);
baza.Odspoji();

```

4.3. Automatizacija

Kako bi se proces automatizacije mogao postići potrebno je implementirati aktivne baze podataka. Ovdje veliki naglasak dajemo na okidače koji su implementirani nad određenim tablicama. Okidači se pokreću nakon što se nad nekom tablicom izvrši upit tipa "Insert" ili "Update". Tako dobivamo jednu cjelinu gdje se od korisnika traži samo da ukoliko je potrebno unese nove automobile na skladište, ažurira stanje na skladištu nakon što su određeni automobili otpremljeni i da evidentira koliku količinu automobila je zaprimio temeljem narudžbenice.

4.3.1. Dodavanje novog automobila na skladište

Korisniku je omogućeno da ukoliko dobije novi automobil, tj. novi model automobila isti može dodati u samu bazu. Prilikom unosa potrebno je da korisnik popunio sve bitne informacije kako bi unos bio valjan. Nakon što je korisnik unio podatke iste informacije se automatski proslijeđuju na skladište unutar kojega se zapisuje stanje automobila.

```
CREATE OR REPLACE FUNCTION "public"."dodaj_auto_na_sk"()
  RETURNS "pg_catalog"."trigger" AS $BODY$
BEGIN
  INSERT INTO "Stanje_Na_Skladistu"(fk_automobil,kolicina,poziciju_u_skladistu)
  VALUES(NEW.id,NEW.min_kolicina,'Novi parking');
  RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100
```

Slika 3: Okidač za dodavanje novog automobila na skladište (Vlastita izrada)

4.3.2. Vođenje evidencije skladišta

Kako bi se svaka promjena koja se dogodila na skladištu mogla evidentirati potrebno je bilo kreirati okidač koji bilježi svaku promjenu na skladištu. Nakon što se izvrši "Insert" na tablici Stanje_Na_Skladistu ta promjena se evidentira i bilježi unutar tablice Evidencija_Na_Skladistu.

```
CREATE OR REPLACE FUNCTION "public"."unesi_novi_auto_na_skladiste"()
  RETURNS "pg_catalog"."trigger" AS $BODY$BEGIN
  INSERT INTO public."Evidencija_Skladista"(fk_automobil,nova_kolicina,tip_posla) VALUES(NEW.
  fk_automobil,NEW.kolicina,'Nova kolicina automobila');
  RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100
```

Slika 4: Okidač za vođenje evidencije skladišta (Vlastita izrada)

Osim što se automobil može dodati na skladište, tj. njegovo stanje u skladištu se može povećati ili smanjiti. Okidač se aktivira kada se izvrši "Update" nad stanjem skladišta. Pokuša li korisnik unijeti količinu koja je negativna odmah dobiva upozorenje. Ako je nova količina veća od prijašnje evidentira se da je došlo do porasta količine automobila, ako je nova količina manja

od prijašnje evidentira se da je došlo do smanjenja broja automobila. Dodatno je implementirano ako se dogodi da određena količina padne strogo ispod minimalne količine automobila kreće proces naručivanja nove količine automobila koja je definirana kao količina naručivanja koja je zasebna za svaki automobil.

```

1 CREATE OR REPLACE FUNCTION "public"."vodi_evidenciju"()
2 RETURNS "pg_catalog"."trigger" AS $BODY$DECLARE
3   min_kol INTEGER;
4   kol_nar INTEGER;
5   pomocna_var INTEGER;
6   naruci INTEGER;
7   id_nabava_auto INTEGER;
8 BEGIN
9   IF NEW.kolicina<0 THEN
10    RAISE EXCEPTION 'Kolicina ne može biti negativna';
11   ELSE
12     IF(NEW.kolicina>OLD.kolicina) THEN
13       INSERT INTO "Evidencija_Skladista"(fk_automobil,prijasnja_kolicina,nova_kolicina,
14         tip_posla) VALUES(OLD.fk_automobil,OLD.kolicina,NEW.kolicina,'Dodana je nova kolicina automobila na skladište');
15     ELSE
16       INSERT INTO "Evidencija_Skladista"(fk_automobil,prijasnja_kolicina,nova_kolicina,
17         tip_posla) VALUES(OLD.fk_automobil,OLD.kolicina,NEW.kolicina,'Oduzeta je kolicina automobila sa skladišta');
18     END IF;
19     SELECT a.min_kolicina INTO min_kol
20     FROM "Automobil" a
21     JOIN "Stanje_Na_Skladistu" sns ON a.id=sns.fk_automobil
22     WHERE sns.fk_automobil=OLD.fk_automobil;
23     SELECT a.kolicina_narucivanja INTO kol_nar
24     FROM "Automobil" a
25     JOIN "Stanje_Na_Skladistu" sns ON a.id=sns.fk_automobil
26     WHERE sns.fk_automobil=OLD.fk_automobil;
27     pomocna_var=NEW.kolicina;
28     naruci=0;
29     LOOP
30     IF pomocna_var>=min_kol THEN
31       EXIT;
32     ELSE
33       pomocna_var=pomocna_var+kol_nar;
34       naruci=naruci+kol_nar;
35     END IF;
36   END LOOP;
37   SELECT id INTO id_nabava_auto
38   FROM "Nabava_Automobila"
39   WHERE fk_automobil=OLD.fk_automobil
40   AND datum_zaprimanje IS NULL;
41   IF (naruci>0 AND id_nabava_auto IS NULL ) THEN
42     INSERT INTO "Nabava_Automobila"(fk_automobil,stanje,kolicina_za_narucivanje)
43     VALUES(OLD.fk_automobil,NEW.kolicina,naruci);
44   END IF;
45   END IF;
46   RETURN NEW;
47 END;
48 $BODY$
49 LANGUAGE plpgsql VOLATILE
50 COST 100
51

```

Slika 5: Okidač za evidenciju promjene stanja na skladištu i minimalnih količina (Vlastita izrada)

4.3.3. Proces naručivanja automobila

Kao što smo u prethodnom primjeru vidjeli ako određena količina automobila padne ispod minimalne količine automatski se pokreće proces nabave novih automobila. Prvo se popunjava tablica Nabava_Automobila, unutar koje se unosi količina koja se naručuje. Bilježi se stanje na datum kada je započet proces naručivanja. Podaci se zatim proslijeđuju prema narudžbenici i od korisnika se traži da sam evidentira datum kada je zaprimio i da definira količinu automobila koju je zaprimio. Nakon što je korisnik izvršio "Update" nad tablicom narudžbenica podaci o zaprimanju se unose u tablicu Nabava_Automobila i time se završava proces nabave. Sve izvršene promjene se evidentiraju na skladištu i ulaze u tablicu evidencije.

```

CREATE OR REPLACE FUNCTION "public"."kreiranje_narudzbe"()
RETURNS "pg_catalog"."trigger" AS $BODY$BEGIN
INSERT INTO public."Narudzbenica"(fk_nabava_automobila,kolicina,fk_automobil)
VALUES(NEW.id,NEW.kolicina_za_narucivanje,NEW.fk_automobil);
RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100

```

Slika 6: Okidač za kreiranje narudžbe (Vlastita izrada)

```

CREATE OR REPLACE FUNCTION "public"."azuriranje_nabave_automobila"()
  RETURNS "pg_catalog"."trigger" AS $BODY$DECLARE
  id_auto INTEGER;
BEGIN
  UPDATE "Nabava_Automobila" SET datum_zaprimanja=NOW() WHERE id=OLD.fk_nabava_automobila;
  SELECT fk_automobil INTO id_auto FROM "Nabava_Automobila" WHERE id=OLD.fk_nabava_automobila;
  UPDATE "Stanje_Na_Skladistu" SET kolicina = kolicina+CAST(OLD.kolicina AS INTEGER) WHERE fk_automobil=id_auto;
  RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100

```

Slika 7: Okidač za ažuriranje Nabave_Automobila (Vlastita izrada)

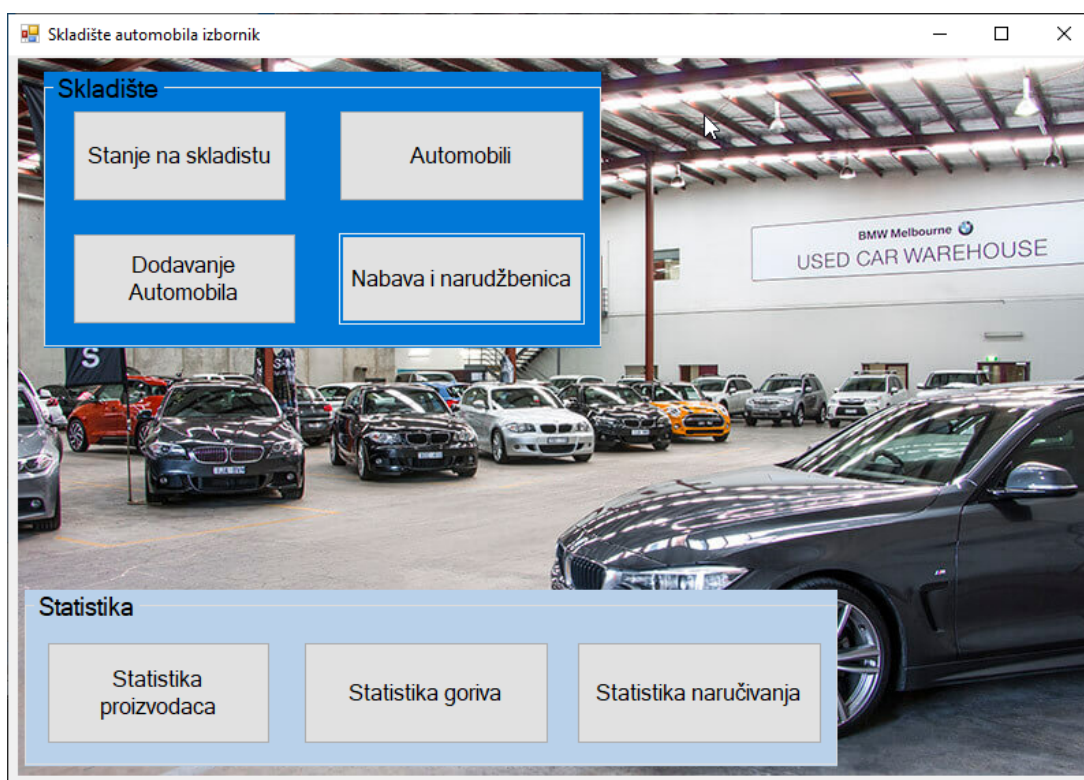
Ovakvim pristupom smo upotpunosti automatizirali bazu podataka što je bio jedan od glavnih ciljeva ovog projekta. Jednostavnost korištenja i smanjenje količine podataka o kojima se korisnik mora brinuti ili voditi evidenciju uvelike olakšava korištenje aplikacije. Kroz ova poglavlja upotpunosti smo pregledali kreiranu bazu i zaokružili cijeli proces rada skladišta, od samog dodavanja automobila na skladište, do nabave novog automobila, evidentiranja i zaprimanja istog na skladište.

5. Prmimjeri korištenja

Kroz sljedeća poglavlja razraditi ću sve mogućnosti koje desktop aplikacija nudi korisniku. Prikazati ću način korištenja aplikacije, što sve aplikacija nudi, koje mogućnosti korisnik može iskoristiti i na kraju prikazati statističke podatke korištenja skladišta.

5.1. Početni izbornik

Početni izbornik je zamišljen da korisniku omogući pregled svih funkcija koje aplikacija nudi, na korisniku je samo da pritiskom na gumb koji je implementiran unutar aplikacije odabere koju funkcionalnost želi pokrenuti. Na slici 8 vidimo da je aplikacija podijeljena u dva dijela. Prvi dio služi za interakciju sa samom bazom podataka, a drugi dio se fokusira na pružanje statistike korisniku. Interakcija između korisnika i baze treba biti što jednostavnija i ovdje sam se fokusirao da od korisnika tražim samo najbitnije informacije koje mora unijeti u aplikaciju, ostale elemente koji su već u bazi može izabrati preko DGV-a ili "comboBoxa".



Slika 8: Početni izbornik aplikacije (Vlastita izrada)

5.2. Stanje na skladištu

Stanje na skladištu je prva forma koju korisnik može otvoriti i pruža uvid u stanje na skladištu. Unutar ove forme nalaze se dva dataGridView-a. Prvi DGV prikazuje trenutno stanje automobila na skladištu. Unutar njega su prikazani naziv automobila, količina koja se nalazi na skladištu trenutno, minimalna količina automobila i pozicija na skladištu gdje se automobili nalaze. Ideja za poziciju na skladištu je ta da zaposlenici ili korisnik koji unosi podatke u aplikaciju za skladište evidentira i gdje se automobili nalaze, tj. na kojem su parkingu automobili ako se radi o velikom skladištu. Ovakvim pristupom zaposlenici mogu brže pronaći željeni automobil jer su svi automobili sortirani unutar skladišta i točno se zna gdje je koji model parkiran. Na drugom DGV koji je pod Evidencijom prikazuju se sve promjene koje su se izvodile nad određenim automobilima i datumi kada su te promjene izvršene. Postoji evidencija cijelog procesa izmjena količina na skladištu, vodi se prijašnje stanje i nova količina koja se ažurirala ili dodala preko gornjeg unosa. Korisnik može ako želi ručno dodavati novi automobil na skladište, ova funkcija nije nužna jer postoji okidač koji će automatski dodati novi automobil na skladište kada se isti unese u bazu. Od korisnika se očekuje da evidentira promjene koje se događaju unutar skladišta i da iste zapisuje u textbox-ove "Količina" i "Mjesto u skladištu". Pritiskom tipke Ažuriraj korisnik ažurira željenu količinu automobila koju je izabrao preko DGV-a. Ažuriranje se odmah bilježi i pokazuje unutar DGV-a za evidenciju tako su sve izmjene korisniku odmah vidljive. Zanimljivo je da korisnik promjena u skladištu unutar nekog intervala potrebno je samo odabrati datum od kada do kada želi vidjeti promjene u skladištu i iste će se prikazati unutar evidencije.

PrikazSkladista

Identifikacijski broj automobila: Kolicina: Mjesto u skladištu:

Dodaj **Ažuriraj**

Stanje automobila na skladištu

ID automobila	Naziv automobila	Količina raspoloživa na skladištu	Minimalna količina na skladištu	Pozicija na parkingu
1	Audi RS6	5	1	Parking 2
2	BMW X5	3	2	Parking 6
3	VW GOLF 8	3	3	Parking 5
4	Mercedes-Benz C250	4	1	Parking
5	Mercedes-Benz C350	3	1	Parking 3
15	Volkswagen GOLF 7 GTD	3	3	Parking 6

Evidencija

Datum unosa	Prijašnja količina	Nova količina na skladištu	Tip posla	Naziv automobila
15/08/2020 16:18		6	Novo stanje na skladištu	Audi RS6
15/08/2020 16:40		2	Novo stanje na skladištu	BMW X5
15/08/2020 16:41		3	Novo stanje na skladištu	BMW X5
16/08/2020 15:43		3	Novo stanje na skladištu	VW GOLF 8
16/08/2020 15:56		4	Novo stanje na skladištu	Mercedes-Benz C250
16/08/2020 16:29	6	5	Oduzeto	Audi RS6
16/08/2020 16:30	5	6	Dodano	Audi RS6
16/08/2020 16:31	3	5	Dodano	VW GOLF 8
16/08/2020 17:04	6	3	Oduzeto	Audi RS6
16/08/2020 18:47	3	2	Oduzeto	Audi RS6

Evidencija po datumu

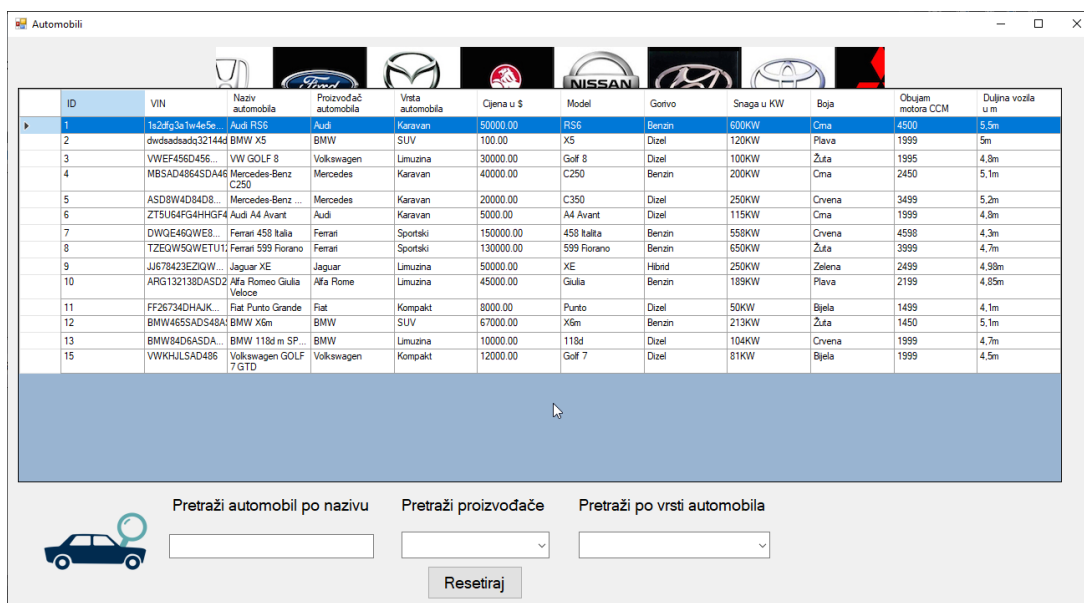
Početak evidencije: Saturday , 22 August 2020

Kraj evidencije: Saturday , 22 August 2020


Slika 9: Forma za ažuriranje i unos novog stanja na skladištu (Vlastita izrada)

5.3. Automobili

Korisniku je također omogućeno da pregleda sve automobile koje je do sada imao na svojem skladištu. Svi podaci koji su potrebni korisniku nalaze se u jednog DGV-u. DGV prikazuje sve relativne informacije vezane uz automobil. Jedinstveni "id" koji je definiran za svaki automobil, VIN identifikacijski broj šasije automobila, naziv automobila, proizvođač, vrsta automobila, cijena, snaga, boja, obujam motora i duljina vozila. Tako korisnik u bilo kojem trenutku može provjeriti koje je sve automobile imao na skladištu. Dodatno ako korisnika zanima određeni automobil njega može pretražiti putem pretraživača. Želi li vidjeti samo automobile određenog proizvođača to može jednostavnim klikom na combobox. Na isti način implementirano je i pretraživanje po vrsti automobila.



ID	VIN	Naziv automobila	Proizvođač automobila	Vrsta automobila	Cijena u \$	Model	Gorivo	Snaga u KW	Boja	Obujam motora CCM	Duljina vozila u m
1	1a2dfg3a1w4e5e	Audi RS6	Audi	Karavan	50000.00	RS6	Benzin	600KW	Crna	4500	5.5m
2	dwdsadsdq32144d	BMW X5	BMW	SUV	100.00	X5	Dizel	120KW	Plava	1999	5m
3	VWFE456D456	VW GOLF 8	Volkswagen	Limuzina	30000.00	Golf 8	Dizel	100KW	Žuta	1995	4.8m
4	MBSAD4864SDA46	Mercedes-Benz C250	Mercedes	Karavan	40000.00	C250	Benzin	200KW	Crna	2450	5.1m
5	ASD8W4D84D8	Mercedes-Benz ...	Mercedes	Karavan	20000.00	C350	Dizel	250KW	Crvena	3499	5.2m
6	ZTSU64FG4HHGF4	Audi A4 Avant	Audi	Karavan	5000.00	A4 Avant	Dizel	115KW	Crna	1999	4.8m
7	DWQE45QWE8	Ferrari 458 Italia	Ferrari	Sportski	150000.00	458 Italia	Benzin	558KW	Crvena	4598	4.3m
8	TZEQW5QWETU1	Ferrari 599 Fiorano	Ferrari	Sportski	130000.00	599 Fiorano	Benzin	650KW	Žuta	3999	4.7m
9	JJ678423EIQW	Jaguar XE	Jaguar	Limuzina	50000.00	XE	Hibrid	250KW	Zelena	2499	4.98m
10	ARG132138DASD2	Alfa Romeo Giulia Veloce	Alfa Romeo	Limuzina	45000.00	Giulia	Benzin	189KW	Plava	2199	4.85m
11	FP2673ADHAJK	Fiat Punto Grande	Fiat	Kompakt	8000.00	Punto	Dizel	50KW	Bijela	1499	4.1m
12	BMW46SSAD48A	BMW X6m	BMW	SUV	67000.00	X6m	Benzin	213KW	Žuta	1450	5.1m
13	BMW84D6ASDA	BMW 118d m SP	BMW	Limuzina	10000.00	118d	Dizel	104KW	Crvena	1999	4.7m
15	VWVKHLSAD486	Volkswagen GOLF 7 GTD	Volkswagen	Kompakt	12000.00	Golf 7	Dizel	81KW	Bijela	1999	4.5m



Pretraži automobil po nazivu

Pretraži proizvođača

Pretraži po vrsti automobila

Resetiraj

Slika 10: Forma za prikaz svih automobila (Vlastita izrada)

5.4. Dodavanje automobila

Kako se stanje skladišta vrlo često mijenja i potrebno je ažurirati asortiman kojim trenutno raspolaže skladište korisniku je omogućeno da unese novi automobil. Od korisnika se traži da unese naziv vozila, VIN, godinu proizvodnje, Minimalnu količinu, količinu naručivanja i cijenu. Proizvođača, vrstu automobila i karakteristike korisnik može izabrati putem DGV-a jednostavnim odabiranjem retka koji želimo. Minimalna količina koju smo stavili je i količina koja se odmah dodaje na skladište i evidentira.

DodajAutomobil

ID Automobila: ID Karakteristike:

Naziv Automobila: ID Proizvođač:

VIN: ID Vrsta Automobila:

Friday, 21 August 2020 Godina proizvodnje:

Minimalna količina:

Količina naručivanja:

Cijena:

Dodaj automobil

Proizvođač

id	naziv	opis
1	BMW	adssad
2	Audi	akshdskid
3	Mercedes	wdadw
4	Volkswagen	sfadfdad
5	Fiat	asdwid
6	Alfa Romeo	fsdfdf

Vrsta automobila

id	naziv	opis
1	SUV	adssad
2	Limuzina	sfadfdad
3	Karavan	gfdg
4	Kompakt	sada
5	Terenac	edfdff
6	Sportakl	kuzuk

Karakteristike

id	model	vrsta_motora	snaga	boja	obujam_motora	dužina_vozila
1	X5	Dizel	120KW	Plava	1999	5m
2	R56	Benzin	600KW	Crna	4500	5.5m
3	330d	Dizel	150KW	Bijela	2995	4.5m
4	Golf 8	Dizel	100KW	Žuta	1995	4.8m
5	C250	Benzin	200KW	Crna	2450	5.1m
6	C350	Dizel	250KW	Crvena	3499	5.2m
7	A4 Avant	Dizel	115KW	Crna	1999	4.8m
8	458 Italia	Benzin	558KW	Crvena	4598	4.3m
9	599 Fiorano	Benzin	650KW	Žuta	3999	4.7m
10	Punto	Dizel	50KW	Bijela	1499	4.1m
11	Giulia	Benzin	189KW	Plava	2199	4.85m

Slika 11: Forma za unos novog automobila na skladište (Vlastita izrada)

5.5. Nabava vozila i naručivanje

Kroz ovu formu korisniku se omogućuje da prati proces naručivanja novog automobila. Unutar prvog DGV-a Nabava automobila prikazuje se naziv i stanje automobila na određeni datum, odmah do tih informacija ispisuje se koja se količina automobila naručuje unutar narudžbenice. Datum zaprimanja je informacija koja se automatski popunjava kada korisnik uistinu i zaprimi naručene automobile. Unutar DGV-a Narudžbenica izražene su sve informacije vezane uz kreiranu narudžbenicu kao i datum, od korisnika se unutar ove forme očekuje da unese količinu automobila koju je zaprimio na skladište, a datum zaprimanja se automatski uzima kroz formu. Na taj način se popunjavaju oba DGV-a i informacije unutar baze. Postoji još mogućnost filtriranja podataka, tj. određivanja razdoblja unutar kojega smo neku robu zaprimili preko "Pretraživanja po datumu". Unese li korisnik slučajno vrijednost koja je manja ili jednaka 0 palikacija ga na to upozorava i ne dopušta mu da izvrši ažuriranja podataka. Na ovakav način se odmah spriječava unošenje grešaka u bazu.

```

private void btnUnesiZapKol_Click(object sender, EventArgs e)
{
    zapKol = int.Parse(txtZaprimljenaKol.Text.ToString());
    pozicija = int.Parse(dgvNarudzbenica.CurrentRow.Cells[0].Value.ToString());
    baza.Povezi();

    if (zapKol > 0)
    {
        string sql = "UPDATE_" + "Narudzbenica_" + "SET_" + "datum_" + "='" + DateTime.
            Now.ToString("yyyy-MM-dd_HH:mm:ss") + "',_" + "zaprimljeno_" + "='" +
            zapKol + "WHERE_" + "Narudzbenica_" + "." + "id_" + "='" + pozicija + "';";

        NpgsqlCommand command = new NpgsqlCommand(sql, baza.connection);
        try
        {
            command.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "UPOZORENJE!", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
    }
    else
    {
        MessageBox.Show("Molim_Vas_unesite_kolicinu_koja_ćnee_biti_manja_od_
            0_ili_imati_negativne_vrijednosti", "UPOZORENJE!",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    Nabava_Prikaz(baza.connection);
    Narudzbenica_Prikaz(baza.connection);
    baza.Odspoji();
}

```


NabvanaNarudzbenica

Nabava automobila

naziv	stanje	stanje_na_datum	kolicina_za_naruciv	datum_zaprimanja
Audi RS6	5	11/08/2020 09:51	5	17/08/2020 10:08
BMW X5	1	17/08/2020 10:27	2	17/08/2020 11:22
VW GOLF 8	2	17/08/2020 10:28	2	17/08/2020 11:22
Mercedes-Benz ...	0	17/08/2020 11:27	3	17/08/2020 12:13
BMW X5	1	17/08/2020 12:15	2	17/08/2020 12:16
VW GOLF 8	2	18/08/2020 18:01	2	18/08/2020 18:02
Volkswagen GOL...	1	20/08/2020 14:00	2	20/08/2020 14:01

Pretraživanje po datumu

Stanje na datum
Friday , 21 August 2020

Datum zaprimanja
Friday , 21 August 2020

Narudzbenica

id	naziv	datum	zaprimljeno	kolicina	fk_nabava_automobil	fk_automobil
1	Audi RS6	11/08/2020 09:51	4	5	1	1
2	BMW X5	17/08/2020 11:22	2	2	5	2
3	VW GOLF 8	17/08/2020 11:22	2	2	6	3
4	Mercedes-Benz ...	17/08/2020 12:13	3	3	7	5
5	BMW X5	17/08/2020 12:16	2	2	8	2
6	VW GOLF 8	18/08/2020 18:02	2	2	9	3
7	Volkswagen GOL...	20/08/2020 14:01	3	2	10	15

Zaprimanje narudzbenice

Kontrolni ID Narudzbenice
1

Unesite kolicinu koju ste zaprimili
[]

Unesi zaprimljena vozila

Slika 12: Forma za popunjavanje nabave automobila (Vlastita izrada)

PrikazSkladista

Identifikacijski broj automobila
3

Kolicina
0

Mjesto u skladistu
Parking 3

Dodaj **Ažuriraj**

Stanje automobila na skladištu

ID automobila	Naziv automobila	Količina raspoloživa na skladištu	Minimalna količina na skladištu	Pozicija na parking
1	Audi RS6	5	1	Parking 2
2	BMW X5	3	2	Parking 6
3	VW GOLF 8	3	3	Parking 5
4	Mercedes-Benz C250	4	1	Parking
5	Mercedes-Benz C350	3	1	Parking 3
15	Volkswagen GOLF 7 GTD	3	3	Parking 6

Obavijest

Molim Vas unesite količinu automobila koja neće biti 0 ili imati negativnu vrijednost

Evidencija

Datum unosa	Prijatna količina	Nova količina na skladištu	Tip posla	Naziv automobila
15/08/2020 16:18		6	Novo stanje na skladištu	Audi RS6
15/08/2020 16:40		2	Novo stanje na skladištu	BMW X5
15/08/2020 16:41		3	Novo stanje na skladištu	BMW X5
16/08/2020 15:43		3	Novo stanje na skladištu	VW GOLF 8
16/08/2020 15:56		4	Novo stanje na skladištu	Mercedes-Benz C250
16/08/2020 16:29	6	5	Oduzeto	Audi RS6
16/08/2020 16:30	5	6	Dodano	Audi RS6
16/08/2020 16:31	3	5	Dodano	VW GOLF 8

Evidencija po datumu

Početak evidencije
Saturday , 22 August 2020

Kraj evidencije
Saturday , 22 August 2020

Slika 13: Upozorenje o nepravilnom unosu (Vlastita izrada)

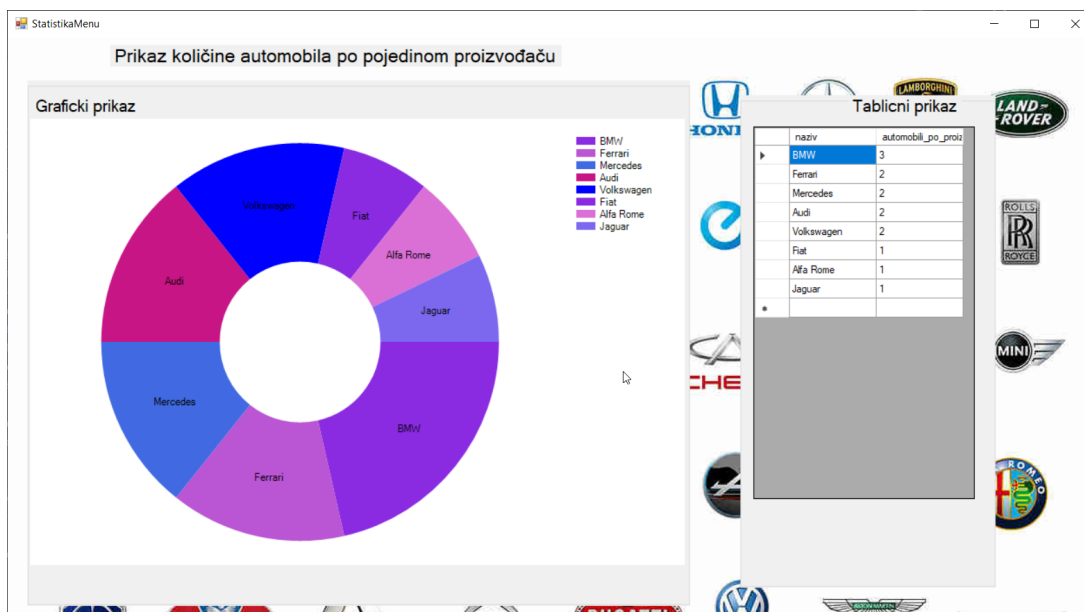
5.6. Statistika proizvođača

Statistika je još jedan element koji aplikacija nudi. Vrlo je bitno da korisnik zna po proizvođaču kojih automobila ima najviše na skladištu. Ove informacije su prikazane unutar DGV-a kako bi korisnik mogao odmah tablično vizualizirati informacije, ali i grafički. Za kreiranje grafova unutar Visual studija koristio sam element chart koji se zatim može dodatno modificirati za potrebe prikaza. Prvo se definira upit kojim definiram što želim iz baze pokazati. Nakon što sam kreirao upit potrebno je grafu prosljediti podatke iz baze. Zatim moramo definirati elemente koji se prikazuju na y osi i x osi. Nakon što smo to napravili naš grafikon se napunio

podacima i nakon pokretanja aplikacije možemo vidjeti povratnu informaciju i prikaz našeg upita u grafikonu. Ispod je prikazan jedan primjer programskog koda kako prikazati podatke preko grafikona.

```
private void Prikaz(NpgsqlConnection connection)
{
    string sql = "SELECT \"Proizvodac\".\"naziv\", COUNT(*) AS_
        automobili_po_proizvodacu FROM \"Proizvodac\", \"Automobil\" WHERE_
        Proizvodac.\"id\" = \"Automobil\".\"fk_proizvodac\" +
        \"GROUP BY 1 ORDER BY 2 DESC;";

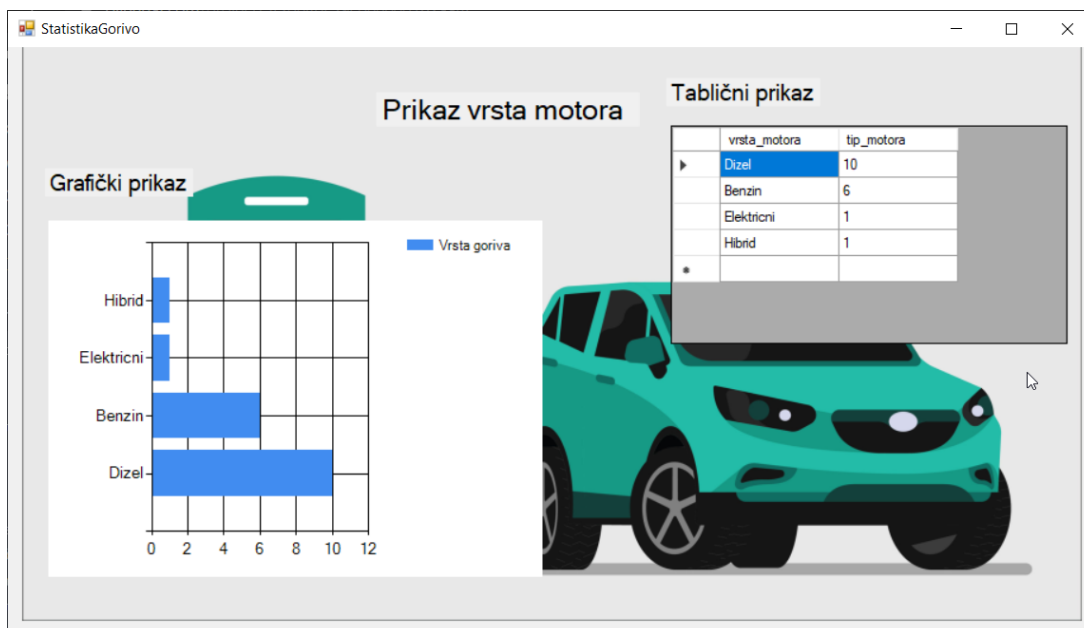
    NpgsqlDataAdapter dataAdapter = new NpgsqlDataAdapter(sql, connection);
    DataSet statistikaDs = new DataSet();
    dataAdapter.Fill(statistikaDs);
    chartProizvodaci.DataSource = statistikaDs.Tables[0];
    chartProizvodaci.Series[0].XValueMember = "Naziv";
    chartProizvodaci.Series[0].YValueMembers = "
        automobili_po_proizvodacu";
    chartProizvodaci.DataBind();
    dgvStatistikaProizvodac.DataSource = statistikaDs.Tables[0];
}
```



Slika 14: Forma za prikaz statistike proizvođača (Vlastita izrada)

5.7. Statistika gorivo

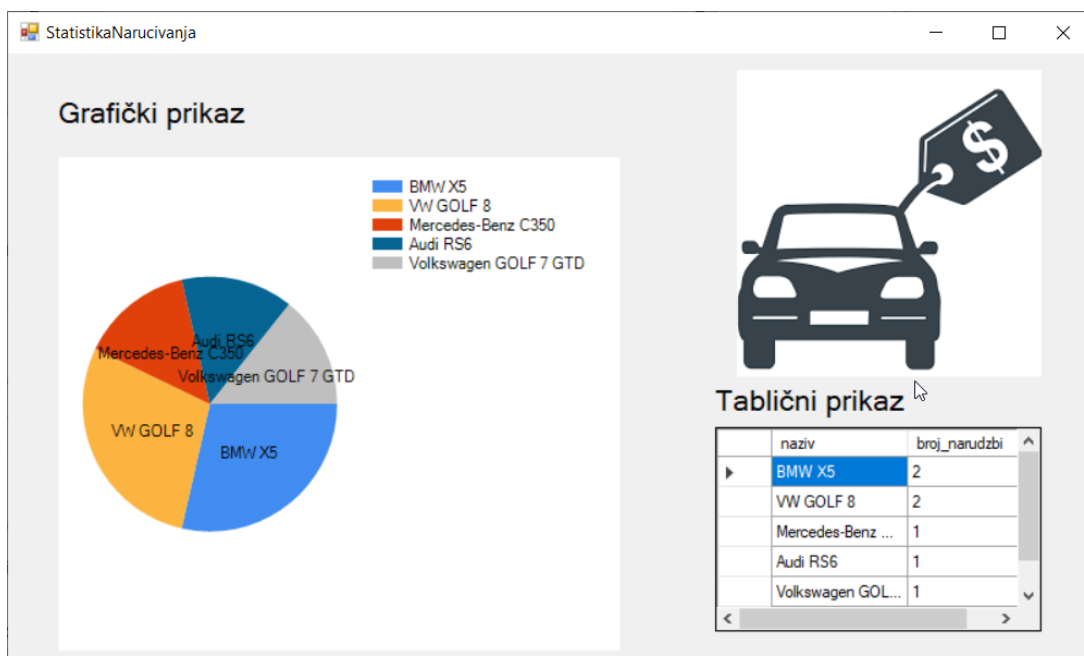
Želimo li znati koja vrsta gorivo pokreće automobile na našem skladištu za to imamo formu "Statistika goriva" koja uzima sve automobile na skladištu i provjerava koji tip goriva koriste. Ovdje možemo vidjeti da su zastupljeni i alternativni izvori energije kao što su električna energija ili kombinacija električne energije i benzina.



Slika 15: Forma za prikaz statistike goriva (Vlastita izrada)

5.8. Statistika naručivanja

Ova forma služi za vizualni prikaz količine naručivanja. Prikazuje se koliko je puta izvršena narudžba za pojedini automobil. Ovdje možemo uočiti koji nam proizvod najčešće nedostaje na skladištu i u budućnosti povećati njegovu količinu naručivanja.



Slika 16: Forma za prikaz statistike naručivanja (Vlastita izrada)

6. Zaključak

Korištenjem temporalnih i aktivnih baza podataka uvelike smo korisniku olakšali rad. Cilj nam je bio kreirati skladište koje će biti autonomno, od korisnika tražimo minimalni unos podataka kako bi skladište funkcioniralo. Želimo korisniku pružiti statističke podatke na temelju kojih se onda mogu korigirati određeni elementi naručivanja automobila ili upravljanja skladištem. Korištenje aplikacije mora biti intuitivno. Kreiranjem Glavnog izbornika korisnik može odmah odabrati i doći do formi koje su mu potrebne, kako su sve forme sortirane korisniku je lakše pronaći formu koja mu treba. Unos podataka mora biti brz i jednostavan. Zbog toga sam se odlučio na izbor DGV-a i combobox-a preko kojih korisnik samo odabire informacije o automobilu koje mu trebaju. Valjanost unosa se provjerava i ako se dogodi da korisnik unese neke netočne informacije aplikacija ga na to upozori. Unos u bazu je sveden na minimum, a sve zahvaljujući aktivnim bazama koje informacije odmah pohranjuju u predviđene tablice koristeći okidače. Temporalne baze su iskorištene kako bi korisniku pružile uvid u stanje skladišta i sve promjene koje su se radile nad njime u određenom vremenskom razdoblju.

Rad na projektu mi je bio jako zabavan, naučio sam dosta novih stvari, nekih stvari sam se morao i podsjetiti. Alati koje sam koristio nisu bili previše zahtjevni za snalaženje, ali mislim da što se tiče njihovih punih mogućnosti tek sam zagrebao površinu. Mislim da je alat Navicat pun pogodak kada govorimo o radu na bazama. Grafičko sučelje je vrlo jednostavno, a ima mogućnosti spajanja na sve poznate baze podataka. Nažalost za kvalitetan program treba izdvojiti i određenu količinu novca, ali mislim da se u konačnici takva investicija isplati.

Popis literature

- [1] *Visual Studio IDE, Code Editor, Azure DevOps, & App Center - Visual Studio*. adresa: <https://visualstudio.microsoft.com/> (pogledano 22. 8. 2020).
- [2] *PostgreSQL: The world's most advanced open source database*. adresa: <https://www.postgresql.org/> (pogledano 22. 8. 2020).
- [3] *Navicat 15 Highlights | Navicat*. adresa: <https://www.navicat.com/en/navicat-15-highlights> (pogledano 22. 8. 2020).
- [4] *Database theory and practice*. Fakultet organizacije i informatike, 2017, str. XIV+427, ISBN: 978-953-6071-62-3.
- [5] D. R. McCarthy i U. Dayal, „The Architecture Of An Active Data Base Management System*”, teh. izv.

Popis slika

1.	ERA model (Vlastita izrada)	3
2.	Paket potreban za povezivanje na bazu (Vlastita izrada)	7
3.	Okidač za dodavanje novog automobila na skladište (Vlastita izrada)	10
4.	Okidač za vođenje evidencije skladišta (Vlastita izrada)	10
5.	Okidač za evidenciju promjene stanja na skladištu i minimalnih količina (Vlastita izrada)	11
6.	Okidač za kreiranje narudžbe (Vlastita izrada)	11
7.	Okidač za ažuriranje Nabave_Automobila (Vlastita izrada)	12
8.	Početni izbornik aplikacije (Vlastita izrada)	13
9.	Forma za ažuriranje i unos novog stanja na skladištu (Vlastita izrada)	14
10.	Forma za prikaz svih automobila (Vlastita izrada)	15
11.	Forma za unos novog automobila na skladište (Vlastita izrada)	16
12.	Forma za popunjavanje nabave automobila (Vlastita izrada)	18
13.	Upozorenje o nepravilnom unosu (Vlastita izrada)	18
14.	Forma za prikaz statistike proizvođača (Vlastita izrada)	19
15.	Forma za prikaz statistike goriva (Vlastita izrada)	20
16.	Forma za prikaz statistike naručivanja (Vlastita izrada)	20

Popis tablica

1. Prilog 1

2. Prilog 2