

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

Fabijan Josip Kraljić

BAZA PODATAKA - SKLADIŠTE ODJEĆE

PROJEKT

Varaždin, 2019.

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

Fabijan Josip Kraljić

Matični broj: 0016116913

Studij: Baze podataka i baze znanja

BAZA PODATAKA - SKLADIŠTE ODJEĆE

PROJEKT

Mentor/Mentorica:

Izv. prof. dr. sc. Markus Schatten

Varaždin, siječanj 2019.

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/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovome radu je prikazana izrada baze podataka za neko skladište odjeće, točnije neke robne kuće tipa Zara ili H&M. Osim toga baza podataka bi trebala imati implementirane principe Aktivnih i temporalnih baza podataka koje služe za automatizaciju skladišta i organizaciju podataka. Krajnja baza podataka se koristi u grafičkoj aplikaciji unutar Windows okruženja.

Ključne riječi: PostgreSQL, Aktivne baze podataka, Temporalne baze podataka, Skladište, Kontinuirano naručivanje, Minimalne količine, pgAdmin, Navicat, Visual Studio

Sadržaj

1. Opis aplikacijske domene	1
2. Teorijski uvod	2
2.1. Aktivne baze podataka	2
2.2. Temporalne baze podataka	2
3. Model baze podataka	3
3.1. ERA model	3
3.2. Vrsta	4
3.3. Uzrast	4
3.4. Spol	4
3.5. Materijal	4
3.6. Odjeća	5
3.7. Veličina	5
3.8. Odjeća_vrsta	5
3.9. Stanje_na_sk	5
3.10. Evidencija_sk	6
3.11. Dobava_robe	6
3.12. Narudžbenica	6
4. Implementacija	7
4.1. Kreiranje tablica	7
4.2. Okidači	11
4.2.1. Evidentiranje promjene skladišta	11
4.2.2. Stvaranje narudžbe	12
4.2.3. Ažuriranje nove robe na skladištu	13
4.2.4. Evidentiranje nove robe	14
4.3. Rad s bazom podataka	14
4.3.1. Spajanje na bazu podataka	14
4.3.2. Čitanje iz baze podataka	15
4.3.3. Pisanje i ažuriranje baze podataka	15
4.3.4. Ispisivanje greške	16
5. Primjer korištenja	17
5.1. Početni prozor	17
5.2. Stanje skladišta	17
5.3. Asortiman	19

5.4. Veličine	19
5.5. Dodavanje na skladište	20
5.6. Evidencija skladišta	21
5.7. Statistika	21
5.8. Narudžbe	22
6. Zaključak	24
7. Literatura	25
Popis slika	26
8. Prilog 1	27

1. Opis aplikacijske domene

Tema ovog rada je izrada baze podataka za skladište nekog dućana odjeće, te izrada i povezivanje aplikacije s napravljenom bazom, gdje bi baza podataka imala implementiran aspekt automatskog naručivanja i vođenja brige o zalihama kako bi se olakšao posao krajnjeg korisnika. Bit aplikacije je unos cjelokupne robe koja se nalazi na skladištu nekog dućana i/ili robne kuće (tipa Zara-e ili H&M-a) po nekim kategorijama i/ili vrijednostima, evidentirati njezinu količinu na skladištu te popratne promjene unutar baze podataka. Nadalje, u slučaju da količina zaliha neke stavke sa skladišta padne na određenu razinu, tada baza podataka treba sama odrediti koliku količinu robe treba naručiti te na temelju toga sastavi narudžbenicu bez intervencije krajnjeg korisnika. Takva vrsta naručivanja je izvedena na temelju minimalnih količina te kontinuiranog naručivanja. Osim toga, funkcija same baze je olakšanje pronalaska određene robe na skladištu kao i evidencija ulaznih i izlaznih stavki sa samog skladišta.

Domena pomoću koje je implementirana takva aplikacija te baza podataka je sljedeća: obuhvaća Robu, Stanje skladišta, Narudžbenicu te Evidenciju skladišta te pripadajući elementi, a za realizaciju baze podataka korištene su tehnike aktivnih i temporalnih baza podataka koje će u nastavku biti objašnjene.

Razlog za takav rad je sama pomisao o tome kako uopće takvo velike robne kuće i dućani evidentiraju svoju robu, jer sigurno ne naručuju ručno svoje artikle budući da je cjelokupan asortiman dosta ogroman te je teško pratiti što je na zalihama, a što ne. Zato sam pomislio da bi aktivne baze podataka bile dobre za automatsko naručivanje robe pri nekom stanju zaliha na skladištu, a temporalne za evidenciju narudžbi koje su obrađene, a koje ne.

2. Teorijski uvod

U nastavku teksta biti će pobliže objašnjene aktivne baze podataka te temporalne baze podataka, budući da su one implementirane u samu strukturu baze podataka koja se koristi u krajnjoj aplikaciji.

2.1. Aktivne baze podataka

Aktivna baza podataka je vrsta konvencionalne baze podataka koja sadrži mehanizme koji reagiraju na nekakve unutrašnje ili vanjske podražaje te se na temelju toga izvrši neki niz akcija ovisno o tome što se želi postići.

Za takvo nešto je potrebno poznavati pravila pod imenom ECA („Event – Condition – Action“) ili jednostavno „Događaj – Uvjet – Akcija“. Spomenuto možemo interpretirati na sljedeći način, kada dođe do nekog događaja (Event) i kada su ispunjeni neki uvjeti (Condition), onda se izvrši neka akcija (Action), u prijevodu dobijemo automatiziran rad baze podataka. (Kornelije Rabuzin, Mirko Maleković, Alen Lovrenčić)

Prednosti takvog pristupa, odnosno korištenjem takve tehnike je što korisnik ne treba voditi nikakvu brigu oko održavanja nekih dijelova baze podataka već ona sama vodu tu brigu. Nedostatak je što lako dođe do nekakvih logičkih pogrešaka koje je teško ispraviti te troši dodatne resurse.

2.2. Temporalne baze podataka

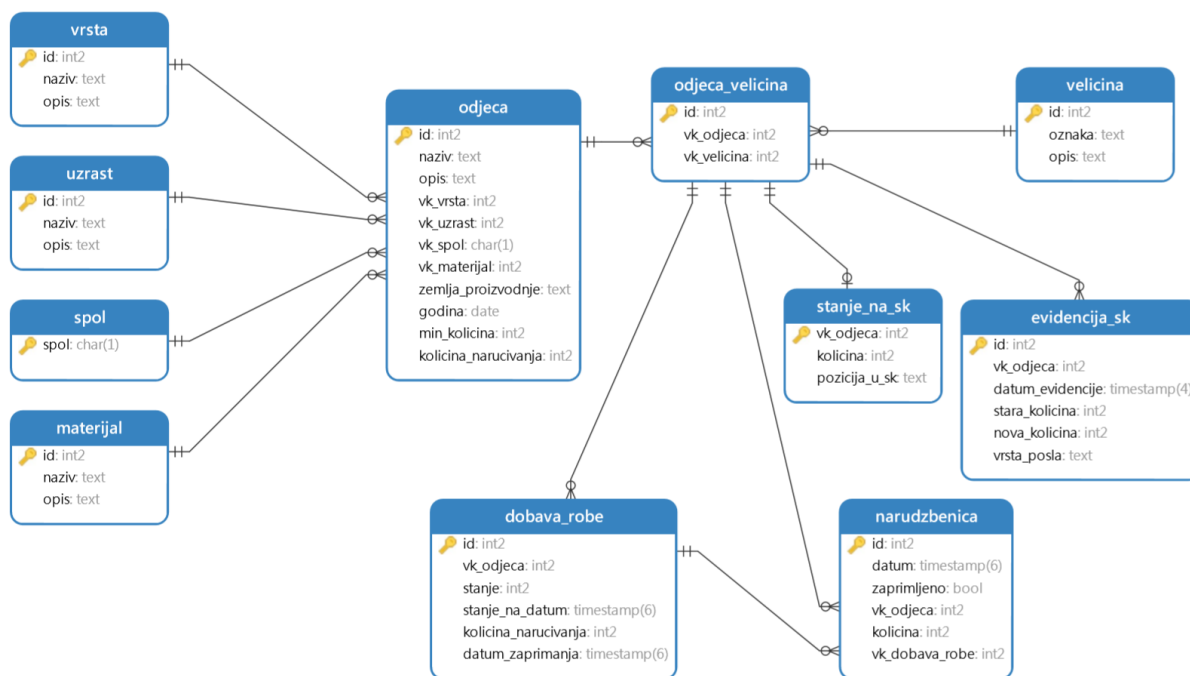
U odnosu na aktivne baze podataka, temporalne baze podataka uzimaju u obzir vremenski aspekt. Tako si možemo zamisliti da temporalne baze podataka sadrže nekakve povijesne podatke i nekakve trenutne podatke, tj. nekakav interval. Takve baze sadrže nekakvo vrijedeće vrijeme, transakcijsko vrijeme i bitemporalno vrijeme. Vrijedeće vrijeme je vrijeme tijekom koje je neka činjenica istinita, transakcijsko vrijeme je vrijeme tijekom kojeg je neka činjenica pohranjena u bazi podataka, a bitemporalno vrijeme je kombinacija oba spomenuta vremena.

Tako neki podaci imaju ugrađeni vremenski aspekt, npr. Narudžbenica je izdana na neki datum te se ona sada treba izvesti, kad se ona izvede imamo datum izvršenja Narudžbenice, tada je vrijeme izdavanja početno vrijeme, a vrijeme izvršenja završno vrijeme, odnosno Narudžbenica ima nekakav svoj vremenski aspekt koji nam je potreban.

3. Model baze podataka

Na slici ispod (Slika 1.) je prikazan model baze podataka koji je korišten prilikom izrade desktop aplikacije. Model baze podataka zovemo još i ERA model, gdje E označava entitet (tablicu), RA označava relaciju (vezu među entitetima). Prikazan model je izrađen u Navicat Premium alatu, grafičkom alatu za upravljanje bazama podataka. Alat omogućuje jednostavno kreiranje tablica i ovisnosti među njima putem grafičkog sučelja što je znatno lakše nego pisanje čistih SQL naredbi. Model baze podataka se sastoji od 11 tablica, tj. entiteta i 4 okidača, a u nastavku je sve pobliže objašnjeno.

3.1. ERA model



Slika 1: ERA model baze podataka (Vlastita izrada)

Prikaz svih tablica iz ERA modela:

- vrsta
- uzrast
- spol
- materijal
- odjeca
- odjeca_velicina

- velicina
- stanje_na_sk
- evidencija_sk
- dobava_robe
- narudzbenica

3.2. Vrsta

U tablicu *vrsta* se upisuju pojedinosti dostupnih artikala. Tipa, dali se radi od hlačama, majici, košulji i sl., jednostavna kategorizacija pojedinih odjevnih elemenata. U samoj tablici imamo attribute id, naziv te opis. Atribut id je jedinstveni id koji se sam generira unosom vrste, naziv je općenito naziv vrste, a opis dodatak nazivu ako možda naziv sam po sebi nije potpuno jasan. Na tablicu *vrsta* se veže tablica odjeće, gdje je onda id tablice *vrsta* ustvari vanjski ključ.

3.3. Uzrast

Tablica *uzrast* označava iduću kategorizaciju odjevnog elementa i to po dobnim skupinama. Naime u većim robnim kućama se pojedini odjeli dijele po dobnim skupinama, tipa za bebe, za djecu, za odrasle kako bi kasniji kupci lakše pronašli ono što traže. Na temelju tog saznanja, tablica *uzrast* se sastoji od istih atributa kao i tablica *vrsta*, a to su id, naziv te opis. Na tablicu *uzrast* se veže odjeća, te tada id služi kao vanjski ključ.

3.4. Spol

Tablica *spol* je najjednostavnija tablica u cijelom ERA modelu, te ona kako naziv kaže označava spol, budući da je u robnim kućama moguća pretraga robe po spolu, tj. postoji roba za muškarce i roba za žene. Sastoji se samo od jednog atributa, atributa *spol*. Navedena tablica se mogla i izbaciti iz samog modela i istoimeni atribut se mogao implementirati unutar tablice odjeća na koju se i *spol* veže, ali radi eventualnih kasnijih izmjena ili dodavanja unutar baze podataka je tablica na taj način implementirana.

3.5. Materijal

Zadnja tablica koja se koristi kod kategorizacije odjeće je implementirana isto kao i tablica *vrsta* i *spol*. Sadrži attribute id, naziv i opis, a u nju se unosi materijal od kojeg je pojedini odjevni element sastavljen. Tablica *materijal* se veže na tablicu odjeća gdje se tada id koristi kao vanjski ključ.

3.6. Odjeća

Tablica *odjeća* je jedna od bitniji u samoj bazi podataka. U nju se pohranjuje sav asortiman koji nudi neki dućan ili robna kuća. Na nju se vežu prijašnje četiri tablice koji upotpunjuju tablicu odjeću, tj. imam četiri vanjska ključa unutar tablice *odjeća*. Osim atributa naziv i opis, tablica *odjeća* sadrži attribute zemlja_proizvodnje, godina te min_kolicina i kolicina_narucivanja. Zadnja dva atributa označavaju minimalnu količinu zaliha robe koja se mora nalaziti na skladištu, a u slučaju da su zalihe ispod tog broja onda se naručuje nova roba i to po broju zapisanom u atributu kolicina_narucivanja. Naručivanje se vrši po principu minimalnih količina i to kontinuiranim popunjavanjem.

3.7. Veličina

Tablica *veličina* sadrži sve zapise mogućih pojavljivanja veličine, tipa S, M, L, XL ili 30/30, 30/32, 34/36 itd.. Na temelju tog saznanja, tablica *veličina* sadrži attribute id, oznaka i opis. U opis se dodaju dodatna pojašnjenja u slučaju da naziv nije dovoljno jasan.

3.8. Odjeća_vrsta

Tablica *odjeća_velicina* je tablica tipa slabog entiteta, tj. ona sadrži vanjske ključeve odjeće i veličine. U samoj tablici *odjeća_velicina* se ustvari nalazi pojedini odjevni element kategoriziran po veličini, budući da odjevni element može imati više veličina. U tom slučaju atribut id iz *odjeća_velicina* ustvari označava pojedini odjevni element, a na temelju njega možemo konkretno naručiti robu u točno određenoj veličini u točno određenom broju. Također po tom id-u vodimo i evidentiramo robu na skladištu.

3.9. Stanje_na_sk

Tablica *stanje_na_sk* služi za evidenciju pojedinog odjevnog elementa na samom skladištu dućana. Na taj način se zna koliko točno ima zaliha neke robe neke veličine na skladištu. U tablici se nalaze tri atributa, a to su vk_odjeća, kolicina i pozicija_na_sl. Atribut vk_odjeća označava vanjski ključ na tablicu *odjeća_velicina*, atribut kolicina označava koliki broj te stavke ima na skladištu, a pozicija_na_sl označava lokaciju te robe unutar skladišta, budući da velike robne kuće imaju vrlo velika skladišta pa se lakše snalaziti u nekakvom organiziranom okruženju. Nad tom tablicom su implementirani kasniji okidači koji provjeravaju količinu na skladištu te naručuju potrebu robu.

3.10. Evidencija_sk

U tablici *evidencija_sk* se bilježe sve promjene koje su se događale u samom skladištu, nešto tipa Log za skladište. U njoj se bilježi dodavanje i micanje robe s skladišta, koliki je broj robe dodan ili oduzet sa skladišta, kolika je količina robe prethodno bila na skladištu, kad se obavila ta promjena te koja se vrsta posla obavila na skladištu. Na temelju tih informacija imamo zapisanu cijelu povijest događaja nad skladištem, a navedena tablica se popunjava uz pomoć okidača.

3.11. Dobava_robe

Tablica *dobava_robe* je pomoćna tablica prije same narudžbe, ona ustvari glumi temporalnu tablicu budući da se bilježi vrijeme slanja narudžbenice i zaprimanja iste. U nju se unose podaci odjeće i količina koju treba naručiti, kao i stanje skladišta na taj dan kad se stvorio zapis. Tablica *dobava_robe* se ne puni ručno već okidačem koji je implementiran nad tablicama *stanje_na_sl* i *narudzbenica*. Na temelju te tablice vidimo kad je narudžba napravljena i kad je roba zaprimljena u skladištu, odnosno dućanu.

3.12. Narudžbenica

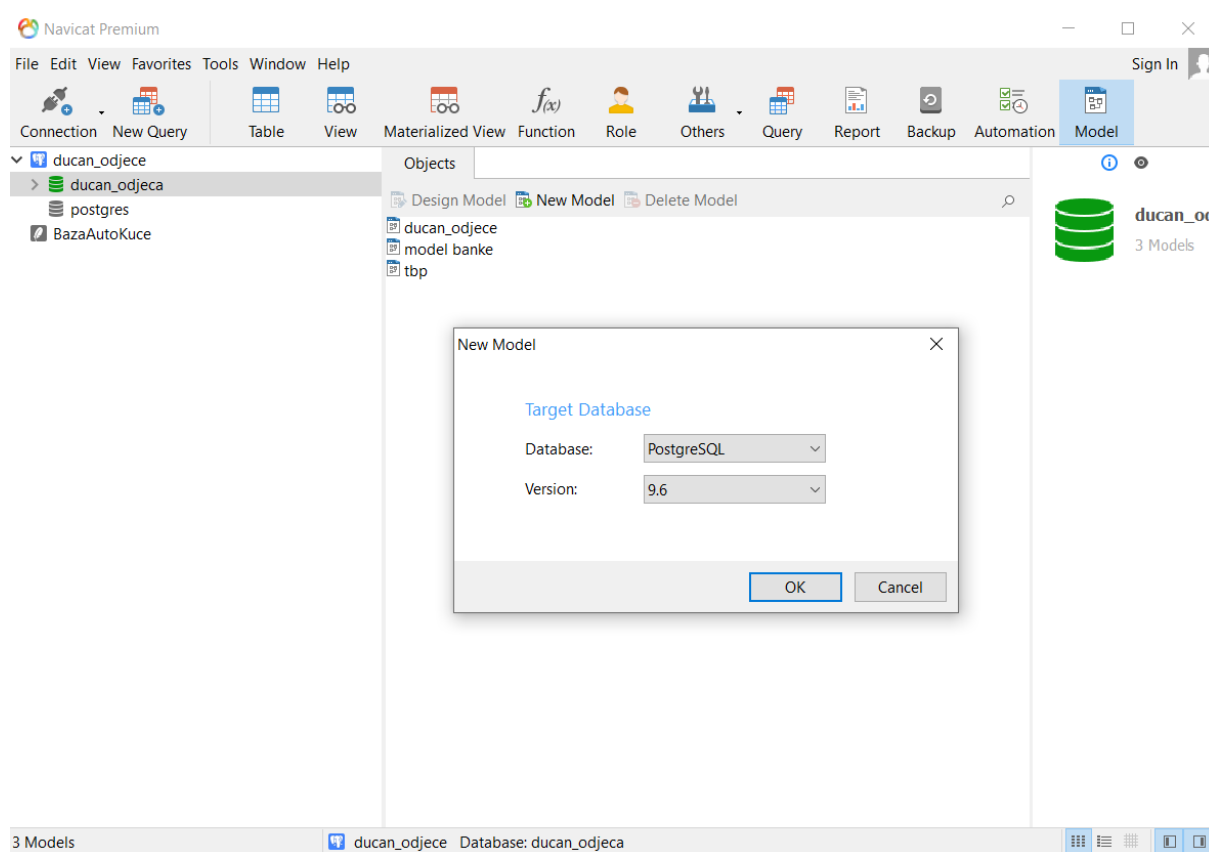
Tablica *narudzbenica* je ustvari stvarna narudzbenica, odnosno dio preslike iz tablice *dobava_robe*. Unutar nje se nalazi atribut zaprimljeno koji dok se izmjeni i postavi tako da je narudžba zaprimljena automatski zabilježi vrijeme zaprimanja robe unutar *dobava_robe* i evidentira se nova količina robe na skladištu. U samoj narudžbenici nisu sadržani ostali atributi tipa od koga se naručuje roba, na koji način se plaća ili koja je vrsta transporta jer taj dio nije presudan za vođenje i upravljanje skladišta.

4. Implementacija

U svrhu ovog rada su korištena nekoliko alata. Jedan od njih je naveden, a to je Navicat Premium koji je služio za izradu baze podataka kao i svih okidača te je omogućio upravljanje bazom podataka. Sljedeći alat je pgAdmin, aplikacija koja omogućuje stvaranje i upravljanje serverom na koji se kasnije aplikacija spaja. Sve mogućnosti koje ima Navicat Premium ima i pgAdmin jedina razlika je ta da pgAdmin nema grafički prikaz ERA modela. Zadnji alat je Visual Studio, razvojno okruženje pomoću kojeg sam izradio grafičko sučelje aplikacije i pomoću kojeg sam se povezao na bazu podataka. Bitno je napomenuti da je baza podataka PostgreSQL tipa iako se u te svrhe mogla neka druga tehnologija koristiti.

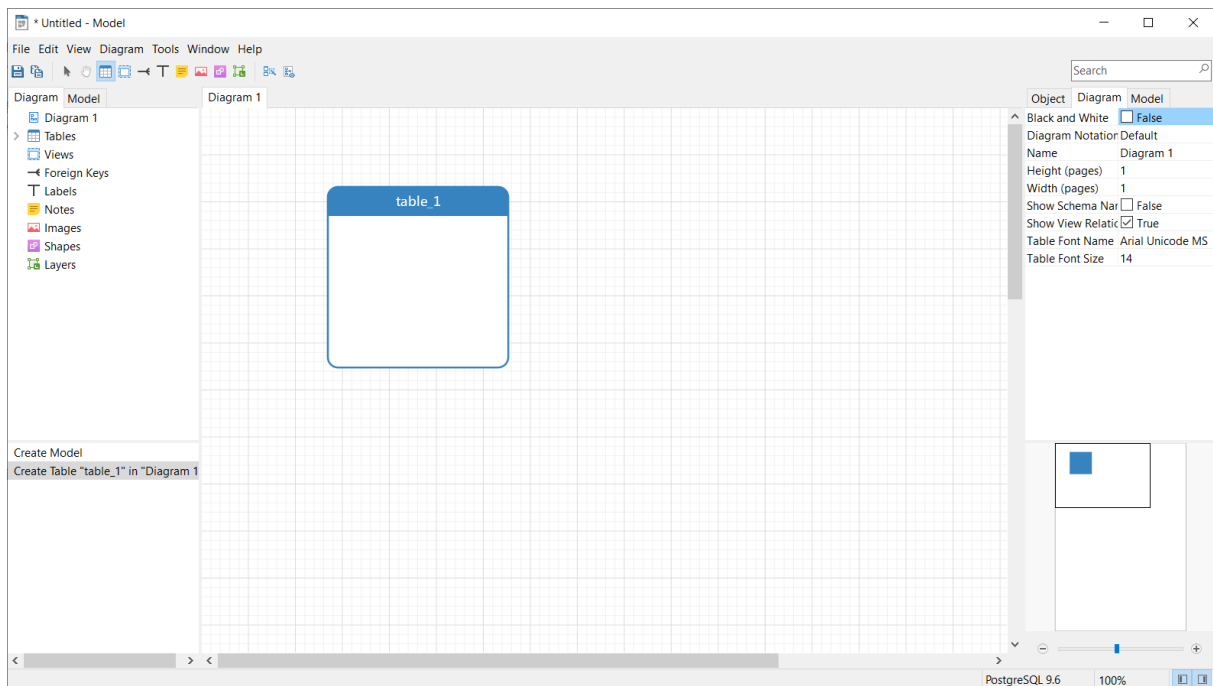
4.1. Kreiranje tablica

Kreiranje tablica je realizirano pomoću grafičkog sučelja u alatu Navicat, ali iza toga stoji SQL kod. U nastavku će biti prikazan primjer kreiranja tablice u grafičkom okruženju, a nakon toga je prikazan popratni SQL kod.



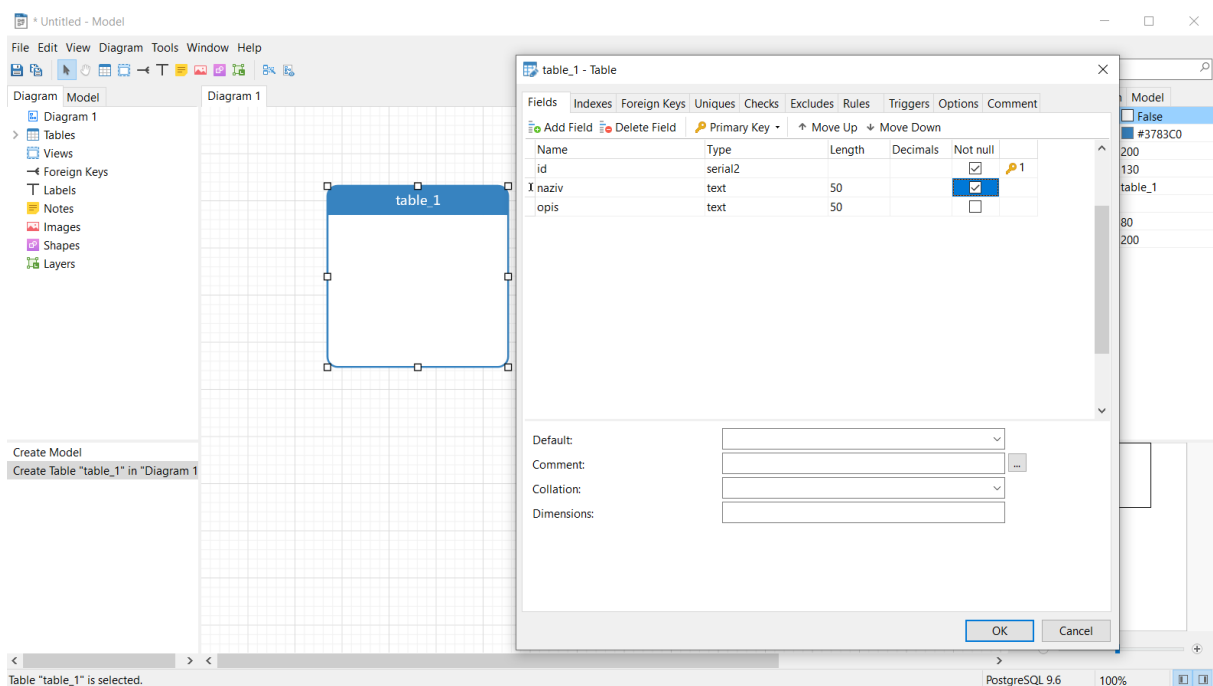
Slika 2: Stvaranje PostgreSQL baze podataka (Vlastita izrada)

Prije samog kreiranja baze u Navicatu potrebno je odabrati ikonu izrade novog modela i postaviti da je baza tipa PostgreSQL 9.6. te se otvara novi prozor. (Slika 2.)



Slika 3: Stvaranje tablice (Vlastita izrada)

Kako bi kreirali tablicu, potrebno je iz izbornika alata odabrat ikonu tablice i lijevim klikom miša pritisnutio na radnu površinu čime nam se prikaže prazna tablica. (Slika 3.)



Slika 4: Dizajniranje tablice (Vlastita izrada)

Kako bi popunili tablicu atributima, potrebno je na nju dva puta kliknuti desnim klikom miša pri čemu se otvara novi prozor (Slika 4.) u koji unosimo naziv atributa te kojeg je on tipa. Također u tom prozoru možemo podesiti sve ostale vrijednosti koje su nam potrebne, tipa vanjskih ključeva, indexa te okidača. U nastavku slijedi popratni SQL kod kreiranja tablica.

```

CREATE TABLE "dobava_robe" (
  "id" int2 NOT NULL DEFAULT nextval('dobava_robe_id_seq'::regclass),
  "vk_odjeca" int2 NOT NULL,
  "stanje" int2 NOT NULL,
  "stanje_na_datum" timestamp(6) NOT NULL DEFAULT now(),
  "kolicina_narucivanja" int2 DEFAULT 0,
  "datum_zaprimanja" timestamp(6),
  CONSTRAINT "dobava_robe_pkey" PRIMARY KEY ("id")
)

CREATE TABLE "evidencija_sk" (
  "id" int2 NOT NULL DEFAULT nextval('evidencija_sk_id_seq'::regclass),
  "vk_odjeca" int2 NOT NULL,
  "datum_evidencije" timestamp(6) NOT NULL DEFAULT now(),
  "stara_kolicina" int2,
  "nova_kolicina" int2 NOT NULL,
  "vrsta_posla" text COLLATE "default",
  CONSTRAINT "evidencija_sk_pkey" PRIMARY KEY ("id")
)

CREATE TABLE "materijal" (
  "id" int2 NOT NULL DEFAULT nextval('materijal_id_seq'::regclass),
  "naziv" text COLLATE "default" NOT NULL,
  "opis" text COLLATE "default",
  CONSTRAINT "materijal_pkey" PRIMARY KEY ("id")
)

CREATE TABLE "narudzbenica" (
  "id" int2 NOT NULL DEFAULT nextval('narudzbenica_id_seq'::regclass),
  "datum" timestamp(6) NOT NULL DEFAULT now(),
  "zaprimljeno" bool NOT NULL DEFAULT false,
  "vk_odjeca" int2 NOT NULL,
  "kolicina" int2 NOT NULL,
  "vk_dobava_robe" int2 NOT NULL,
  CONSTRAINT "narudzbenica_pkey" PRIMARY KEY ("id")
)

CREATE TABLE "odjeca" (
  "id" int2 NOT NULL DEFAULT nextval('odjeca_id_seq'::regclass),
  "naziv" text COLLATE "default" NOT NULL,
  "opis" text COLLATE "default",
  "vk_vrsta" int2 NOT NULL,
  "vk_uzrast" int2 NOT NULL,
  "vk_spol" char(1) COLLATE "default" NOT NULL,
  "vk_materijal" int2 NOT NULL,
  "zemlja_proizvodnje" text COLLATE "default",
  "godina" date,
  "min_kolicina" int2 NOT NULL,
  "kolicina_narucivanja" int2 NOT NULL,
  CONSTRAINT "odjeca_pkey" PRIMARY KEY ("id")
)

CREATE TABLE "odjeca_velicina" (

```

```
"id" int2 NOT NULL DEFAULT nextval('odjeca_velicina_id_seq'::regclass),
"vk_odjeca" int2 NOT NULL,
"vk_velicina" int2 NOT NULL,
CONSTRAINT "odjeca_velicina_pkey" PRIMARY KEY ("id")
)
```

```
CREATE TABLE "spol" (
"spol" char(1) COLLATE "default" NOT NULL,
CONSTRAINT "spol_pkey" PRIMARY KEY ("spol")
)
```

```
CREATE TABLE "stanje_na_sk" (
"vk_odjeca" int2 NOT NULL,
"kolicina" int2 NOT NULL,
"pozicija_u_sk" text COLLATE "default" NOT NULL,
CONSTRAINT "stanje_na_sk_pkey" PRIMARY KEY ("vk_odjeca")
)
```

```
CREATE TABLE "uzrast" (
"id" int2 NOT NULL DEFAULT nextval('uzrast_id_seq'::regclass),
"naziv" text COLLATE "default" NOT NULL,
"opis" text COLLATE "default",
CONSTRAINT "uzrast_pkey" PRIMARY KEY ("id")
)
```

```
CREATE TABLE "velicina" (
"id" int2 NOT NULL DEFAULT nextval('velicina_id_seq'::regclass),
"oznaka" text COLLATE "default" NOT NULL,
"opis" text COLLATE "default",
CONSTRAINT "velicina_pkey" PRIMARY KEY ("id")
)
```

```
CREATE TABLE "vrsta" (
"id" int2 NOT NULL DEFAULT nextval('vrsta_id_seq'::regclass),
"naziv" text COLLATE "default" NOT NULL,
"opis" text COLLATE "default",
CONSTRAINT "vrsta_pkey" PRIMARY KEY ("id")
)
```

Na taj način se kreiraju svih 11 tablica jednostavnim korištenjem grafičkog sučelja u Navicat-u no ovdje nedostaju relacije (veze) između samih tablica, a kako SQL kod nebi zauzimao previše mjesta ovdje će biti navedena samo dva primjera stvaranja relacija.

```
ALTER TABLE "odjeca_velicina" ADD CONSTRAINT "fk_odjeca_velicina_odjeca_1" FOREIGN
KEY ("vk_odjeca") REFERENCES "odjeca" ("id") ON DELETE NO ACTION ON UPDATE NO
ACTION;
ALTER TABLE "odjeca_velicina" ADD CONSTRAINT "fk_odjeca_velicina_velicina_1" FOREIGN
KEY ("vk_velicina") REFERENCES "velicina" ("id") ON DELETE NO ACTION ON UPDATE
NO ACTION;
```


4.2. Okidači

Nakon što su kreirane tablice potrebno je izraditi okidače koji čine aspekt Aktivnih baza podataka. U nastavku su prikazani SQL kodovi svih četiri okidača koja su korištena prilikom izrade baze podataka.

4.2.1. Evidentiranje promjene skladišta

Okidač prikazan ispod je jedan od najbitnijih u cijeloj bazi podataka. Navedeni okidač reagira na promjene, odnosno na UPDATE događaj nad tablicom *stanje_na_sk*, npr. smanjenje robe na skladištu. Prilikom svake promjene u toj tablici okidač unese u tablicu *evidencija_sk* izmijenjene podatke te kad se ta izmjena izvršila, a time dobivamo naš Log.

Nadalje u slučaju da smanjujemo zalihe i idemo ih smanjiti ispod razine 0 komada, onda nam okidač javlja grešku, tj. da količina robe ne smije negativna biti. Grešku kasnije "hvatamo" u aplikaciji te ispisujemo korisniku poruku na zaslon.

Zadnji i najbitniji aspekt je naručivanje kada zalihe padnu ispod minimalne količine. Za odjeću znamo da je potrebna neka minimalna količina, a u slučaju da ispod nje dodamo, onda je potrebno naručiti robu u količini navedenoj u tablici *odjeće*. Ako ta količina naručivanja nije dovoljna za prijelaz preko minimalne količine onda se nadoda još jednom količina naručivanja, tj. duplo se naručuje. Također je implementirana zaštita od stalnog ponovnog naručivanja u slučaju da se ispod minimalne količine nastavi roba oduzimati s skladišta dućana.

```
CREATE FUNCTION public.evidentiraj_promjene_sk()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$DECLARE
    min_kol INTEGER;
    kol_nar INTEGER;
    pomocna_var INTEGER;
    naruci INTEGER;
    id_dobave_robe INTEGER;
BEGIN
IF NEW.kolicina<0 THEN
    RAISE EXCEPTION 'Kolicina robe ne moze ici u minus (-)!';
ELSE
    IF (NEW.kolicina>OLD.kolicina) THEN
        INSERT INTO evidencija_sk(vk_odjeca,stara_kolicina,nova_kolicina,
            vrsta_posla) VALUES (OLD.vk_odjeca,OLD.kolicina,NEW.kolicina,'
            Dodano');
    ELSE
        INSERT INTO evidencija_sk(vk_odjeca,stara_kolicina,nova_kolicina,
            vrsta_posla) VALUES (OLD.vk_odjeca,OLD.kolicina,NEW.kolicina,'
            Oduzeto');
    END IF;

SELECT o.min_kolicina INTO min_kol
```

```

FROM odjeca o
JOIN odjeca_velicina ov ON o.id= ov.vk_odjeca
JOIN stanje_na_sk snsk ON ov.id=snsk.vk_odjeca
WHERE snsk.vk_odjeca=OLD.vk_odjeca;

SELECT o.kolicina_narucivanja INTO kol_nar
FROM odjeca o
JOIN odjeca_velicina ov ON o.id= ov.vk_odjeca
JOIN stanje_na_sk snsk ON ov.id=snsk.vk_odjeca
WHERE snsk.vk_odjeca=OLD.vk_odjeca;

pomocna_var=NEW.kolicina;
naruci=0;
LOOP
    IF pomocna_var>=min_kol THEN
        EXIT;
    ELSE
        pomocna_var=pomocna_var+kol_nar;
        naruci=naruci+kol_nar;
    END IF;
END LOOP;

SELECT id INTO id_dobave_robe
FROM dobava_robe
WHERE vk_odjeca=OLD.vk_odjeca
AND datum_zaprimanja IS NULL;

IF (naruci>0 AND id_dobave_robe IS NULL ) THEN
    INSERT INTO dobava_robe(vk_odjeca,stanje,kolicina_narucivanja)
    VALUES (OLD.vk_odjeca,NEW.kolicina,naruci);
END IF;

END IF;
RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public.evidentiraj_promjene_sk()
    OWNER TO postgres;

CREATE TRIGGER "evidentiraj_promjene"
AFTER UPDATE ON "stanje_na_sk"
FOR EACH ROW EXECUTE PROCEDURE "public"."evidentiraj_promjene_sk"();

```

4.2.2. Stvaranje narudžbe

Kao što je u prethodnom tekstu navedeno u tablicu *dobava_robe* se unosi naša pomoćna narudžbenica. Stvarna narudžbenica se stvara zapravo ovdje, prilikom unosa stavke u tablicu *dobava_robe* gdje se uzimaju ti novi podaci i unose se u tablicu *narudzbenica*. Okidač je implementiran nad tablicom *dobava_robe* koji reagira na INSERT događaj.

```

CREATE FUNCTION public.stvaranje_narudzbe()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN

INSERT INTO narudzbenica(vk_odjeca,kolicina,vk_dobava_robe)
VALUES (NEW.vk_odjeca,NEW.kolicina_narucivanja,NEW.id);

RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public.stvaranje_narudzbe()
    OWNER TO postgres;

CREATE TRIGGER "stvari_narudzbu"
AFTER INSERT ON "dobava_robe"
FOR EACH ROW EXECUTE PROCEDURE "public"."stvaranje_narudzbe"();

```

4.2.3. Ažuriranje nove robe na skladištu

```

CREATE FUNCTION public.azuriranje_stanja_sk()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
DECLARE
    id_odj INTEGER;
BEGIN

UPDATE dobava_robe SET datum_zaprimanja=NOW() WHERE id=OLD.vk_dobava_robe;

SELECT vk_odjeca INTO id_odj FROM dobava_robe WHERE id=OLD.vk_dobava_robe;
UPDATE stanje_na_sk SET kolicina=kolicina+OLD.kolicina WHERE vk_odjeca=id_odj;

RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public.azuriranje_stanja_sk()
    OWNER TO postgres;

CREATE TRIGGER "azuriraj_stanje_sk"
AFTER UPDATE ON "narudzbenica"
FOR EACH ROW EXECUTE PROCEDURE "public"."azuriranje_stanja_sk"();

```

Zadatak iznad prikazanog okidača je unos nove pristigle robe u skladište prilikom potvrde o zaprimanju narudžbenice, odnosno naručene robe. Okidač se izvršava nad tablicom *narudzbenica*.

4.2.4. Evidentiranje nove robe

Zadnji okidač zaslužen je za postavljanje nekog potpuno novog artikla robe na skladište. Služi za evidentiranje novog dodavanja robe na skladište, nešto što prethodno nije postojalo, budući da nemamo staru količinu već samo novu.

```
CREATE FUNCTION public.evidentiraj_novu_robu_na_sk()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    INSERT INTO evidencija_sk(vk_odjeca,nova_kolicina,vrsta_posla) VALUES(NEW.
        vk_odjeca,NEW.kolicina,'Novo na škladitu');
RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public.evidentiraj_novu_robu_na_sk()
    OWNER TO postgres;

CREATE TRIGGER "evidentiraj_novu_robu_na_sk"
AFTER INSERT ON "stanje_na_sk"
FOR EACH ROW EXECUTE PROCEDURE "public"."evidentiraj_novu_robu_na_sk"();
```

Ovim okidačima smo zapravo dobili potpunu automatizaciju baze podataka neke robne kuće, odnosno aspekt naručivanja i vođenja brige o zalihama je potpuno automatiziran.

4.3. Rad s bazom podataka

U nastavku slijede tehnike korištene prilikom rada s bazom podataka unutar aplikacije, tipa spajanje a bazu podataka, čitanje te unos i ažuriranje baze podataka. Navedeni kodovi su pisani u C# jeziku unutar Visual Studio-a.

4.3.1. Spajanje na bazu podataka

Kada smo uspješno sinkronizirali napravljenu bazu podataka s našim lokalnim serverom onda se možemo povezati iz gotovo bilo kojeg alata na našu bazu te ju koristi na način koji nam odgovara. Kako bi takvo nešto bilo izvedivo potrebna nam je konekcija na server i na bazu podataka. U nastavku je prikazan C# kod koji to omogućuje.

```

string connstring = String.Format(
    "Server=127.0.0.1;" +
    "Port=5432;" +
    "User Id=postgres;" +
    "Password=****;" +
    "Database=ducan_odjeca");
NpgsqlConnection conn = new NpgsqlConnection(connstring);
conn.Open();
...
conn.Close();

```

4.3.2. Čitanje iz baze podataka

U nastavku je prikazano čitanje iz baze podataka podataka. Dohvaćeni podaci su u aplikaciji prikazani unutar tablica korištenjem `dataGridView`-a te će na taj način biti implementiran kod. Potreban je običan SQL upit i klasa *NpgsqlDataAdapter* za dohvaćanje podataka, a ostalo je organizacija podataka.

```

DataSet ds = new DataSet();
DataTable dt = new DataTable();

string sql = "SELECT e.id AS Evidencija_ID,o.id AS Odjeca_ID," +
    "ov.id AS Velicina_ID,o.naziv AS Naziv,v.oznaka AS Velicina" +
    ",e.datum_evidencije AS Datum,e.stara_kolicina AS Stara_kolicina" +
    ",e.nova_kolicina AS Nova_kolicina, e.vrsta_posla AS Vrsta FROM " +
    "evidencija_sk e JOIN odjeca_velicina ov ON e.vk_odjeca=ov.id " +
    "JOIN odjeca o ON ov.vk_odjeca=o.id " +
    "JOIN velicina v ON v.id=ov.vk_velicina " +
    "ORDER BY 6 ASC;";

NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, conn);

ds.Reset();
da.Fill(ds);
dt = ds.Tables[0];
dataGridView1.DataSource = dt;

```

4.3.3. Pisanje i ažuriranje baze podataka

Pisanje i ažuriranje podataka u bazi je nešto drugačije u odnosu na čitanje. Koristi se druga naredba, odnosno klasa. Ne koristi se klasa *NpgsqlDataAdapter* već *NpgsqlCommand*, a da bi se konkretno naša naredba izvršila potrebna je metoda *.ExecuteReader()*. Također ako bi htjeli izbrisati neku stavku iz neke određene tablice potrebna nam je klasa *NpgsqlCommand* i metoda *.ExecuteReader()* jer jedino one omogućavaju rad nad tablicama u bazi podataka.

```

int idOdjece = int.Parse(dataGridView1.CurrentRow.Cells[1].Value.ToString());
int kolicina = int.Parse(textBoxKolicina.Text);
string pozicija = "";
pozicija = textBoxPozicija.Text;

```

```

string sql = "INSERT INTO stanje_na_sk(vk_odjeca,kolicina,pozicija_u_sk)
VALUES(" + idOdjece + "," + kolicina + "," + pozicija_u_sk + ");";

NpgsqlCommand command = new NpgsqlCommand(sql, conn);
command.ExecuteReader();

ili

string sql = "UPDATE stanje_na_sk SET kolicina="+kolicina+", pozicija_na_sk='"+
    pozicija+"' WHERE vk_odjeca="+idOdjece+";";

NpgsqlCommand command = new NpgsqlCommand(sql, conn);
command.ExecuteReader();

```

4.3.4. Ispisivanje greške

U slučaju da smo u neki okidač postavili RAISE EXCEPTION poruku tada je možemo u aplikaciji uhvatiti i prikazati te ne trebamo taj dio potpuno programski obaviti već baza podataka obavi posao za nas. Potrebno je poruku ispravno "uloviti" i prikazati, a u C# se koristi *PostgresException* i *MessageText* metoda za ispis konkretne poruke koju smo mi definirali. U programskom kodu ispod je prikazano kako se takvo nešto radi.

```

NpgsqlCommand command = new NpgsqlCommand(sql, conn);
try{
    NpgsqlDataReader dataReader = command.ExecuteReader();
}
catch (PostgresException msg) {
    MessageBox.Show(msg.MessageText, "Upozorenje");
}

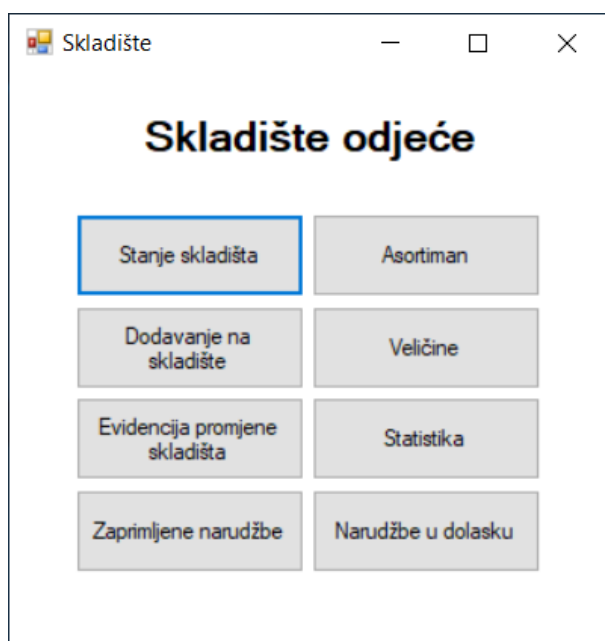
```

5. Primjer korištenja

U nastavku teksta biti će prikazana aplikacija, svi njeni prozori i njene funkcionalnosti. Uz svaki zaslon će ići slika i popratni tekst koji pobliže objašnjava pojedine funkcionalnosti koje su dostupne unutar samog prozora te općenito čemu služi pojedini prozor.

5.1. Početni prozor

Prilikom otvaranja aplikacije, otvara nam se prozor s slike ispod (Slika 5.) u kojemu imamo osam opcija. Kliknemo li na jednu od opcija otvara nam se novi prozor u kojem možemo vršiti neku funkcionalnost ili jednostavno pregledavati podatke ovisno o tome koju opciju odaberemo. U nastavku su sve funkcionalnosti navedene i objašnjene.



Slika 5: Početni prozor (Vlastita izrada)

5.2. Stanje skladišta

Na slici ispod (Slika 6.) možemo vidjeti svaku pojedinu stavku, odjeću, na skladištu i to po veličini, a za svaku veličinu imamo i broj komada te odjeće koji je dostupan na skladištu.

Također imamo mogućnost filtriranja odjevnih elemenata po šifri i to jednostavnim upisom broja, lijevo u samom prozoru, čime nam se ažurira prikaz tablice.

Odaberemo li sa zaslona neki red (odjevni element) i pritisnemo na gumb *Uredi stanje*, otvara nam se novi prozor (Slika 7.) u kojem možemo unijeti aktualno stanje skladišta, čime nam se ažurira stanje robe u bazi podataka. Ovaj dio bi ustvari trebala prodaja izvršavati, ali radi jednostavnosti projekta oduzimanje s skladišta je na takav način obavljena.

Stanje skladišta

Stanje skladišta

Uredi stanje

Filter oznake:

	oznaka	oznaka_velicine	naziv	opis	vrsta	uzrast	spol	materijal	velicina	kolicina
	1	1	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	XS	6
	1	2	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	S	6
	1	3	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	M	8
	1	4	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	L	5
	1	5	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	XL	10
	1	6	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	XXL	11
	12	7	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	XS	10
	12	8	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	S	14
	12	9	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	M	24
	12	10	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	L	13
	12	11	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	XL	20
	12	12	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	XXL	18

Slika 6: Početni prozor (Vlastita izrada)

Stanje skladišta

Stanje skladišta

Uredi stanje

Filter oznake:

	oznaka	oznaka_velicine	naziv	opis	vrsta	uzrast	spol	materijal	velicina	kolicina
	1	1	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	XS	6
	1	2	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	S	6
	1	3	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	M	8
	1	4	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	L	5
	1	5	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	XL	10
	1	6	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	pamuk	XXL	11
	12	7	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	XS	10
	12	8	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	S	14
	12	9	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	M	24
	12	10	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	L	13
	12	11	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	XL	20
	12	12	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	pamuk	XXL	18

Stanje na skladištu

1

8

d1

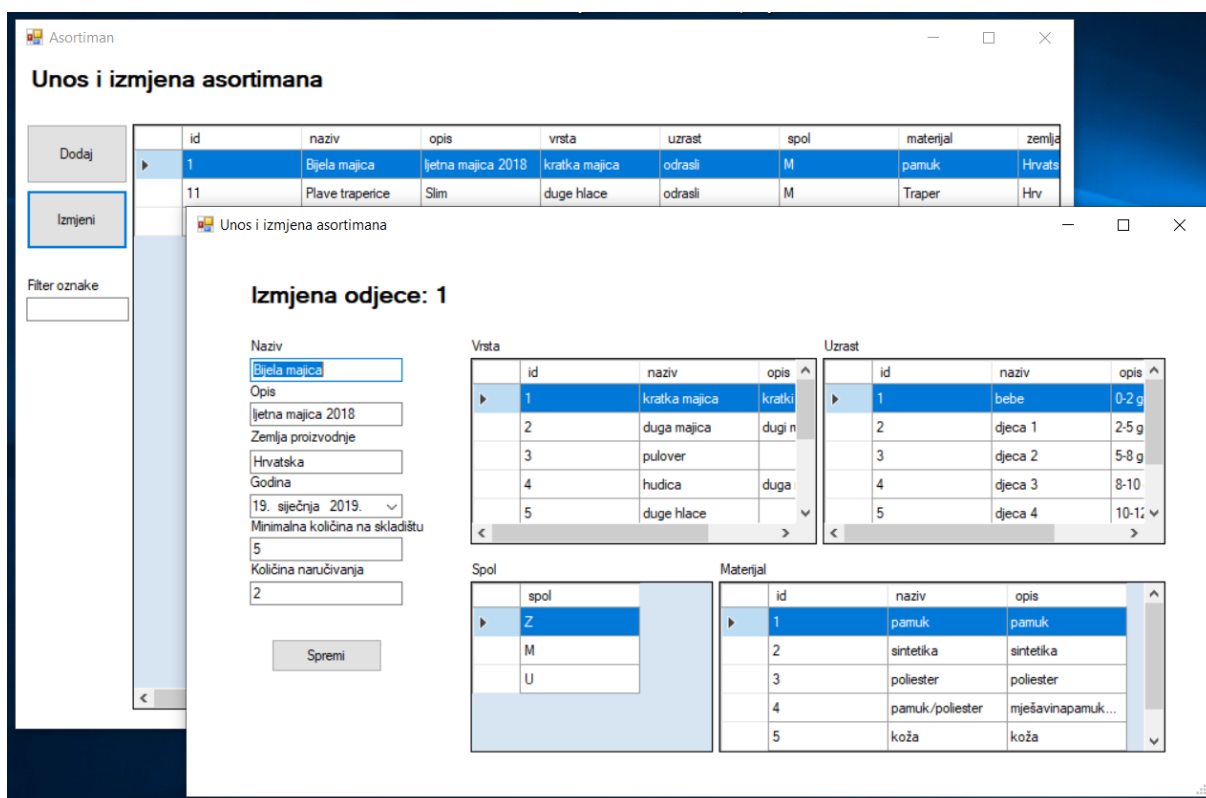
Potvrdi

Slika 7: Izmjena stanja robe na skladištu (Vlastita izrada)

5.3. Asortiman

Na sljedećoj slici (Slika 8.) je prikazan prozor asortimana te prozor unosa i izmjene asortimana, efektivno imamo prikaz tablice *odjeca*. Također je moguće filtrirati pojedine odjevne elemente po šifri, odnosno oznaci radi lakšeg snalaženja.

Označimo li neki odjevni element i odaberemo gumb *Izmjeni* otvara nam se novi prozor u kojem možemo izmijeniti podatke za već postojeću robu. U slučaju da u prodaju ulazi nova kolekcija tada je potrebno odabrati gumb *Dodaj* čime se ustvari otvara isti prozor samo potpuno prazan i u njega unosimo sve podatke nove kolekcije.



Slika 8: Unos i izmjena asortimana (Vlastita izrada)

5.4. Veličine

Novi odjevni element koji smo napravili u prethodnom dijelu je samo odjevni element bez veličine, a svaka roba ustvari ima svoju veličinu i ovdje definiramo taj dio. Tako generirani zapis zapravo predstavlja stvarni odjevi element koji je na prodaji u dućanu, a njegovu oznaku koristimo za naručivanje robe.

U otvorenom prozoru (Slika 9.) iz asortimana odaberemo odjevni element, a iz veličine odaberemo dostupnu veličinu te robe i pritisnemo gumb *Dodaj*. Tako smo dobili robu u nekoj veličini i nju možemo vidjeti u donjoj tablici. Osim toga, asortiman možemo filtrirati po njegovoj oznaci.

Dostupne veličine odjeće

Određivanje dostupnih veličina odjeće

Oznaka filter:

Veličine				Asortiman						
	id	oznaka	opis	id	naziv	opis	vrsta	uzrast	spol	
Dodaj	1	XS		1	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M	
	2	S		11	Plave traperice	Slim	duge hlace	odrasli	M	
	3	M		12	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	Z	
	4	L								
	5	XL								
	6	XXL								
	7	34-38	Stopala							
	8	38-42	Stopala							
	9	42-46	Stopala							
	10	120	Duzina za remen							
	11	130	Duzina za remen							
	12	140	Duzina za remen							
	13	150	Duzina za remen							

id	naziv	oznaka
1	Bijela majica	XS
2	Bijela majica	S
3	Bijela majica	M
4	Bijela majica	L
5	Bijela majica	XL
6	Bijela majica	XXL

Slika 9: Dodavanje veličine (Vlastita izrada)

5.5. Dodavanje na skladište

Kada smo kreirali novu kolekciju onda je potrebno tu istu novu kolekciju dodati na skladište u određenoj količini (Slika 10.). Iz tablice odaberemo odjevni element u nekoj veličini i upišemo količina koja je dostupna na skladištu te gdje se roba nalazi na skladištu. U slučaju da već imamo robu na skladištu tada dobijemo poruku upozorenja da ona već postoji.

Dodavanje robe na skladište

Količina:

Pozicija:

	oznaka	oznaka_velicine	naziv	opis	vrsta	uzrast	velicina
1	1	1	Bijela majica	ljetna majica 2018	kratka majica	odrasli	XS
1	2	2	Bijela majica	ljetna majica 2018	kratka majica	odrasli	S
1	3	3	Bijela majica	ljetna majica 2018	kratka majica	odrasli	M
1	4	4	Bijela majica	ljetna majica 2018	kratka majica	odrasli	L
1	5	5	Bijela majica	ljetna majica 2018	kratka majica	odrasli	XL
	6	6	Bijela majica	ljetna majica 2018	kratka majica	odrasli	XXL
	15	15	Plave traperice	Slim	duge hlace	odrasli	L
	13	13	Plave traperice	Slim	duge hlace	odrasli	S
	14	14	Plave traperice	Slim	duge hlace	odrasli	M
	10	10	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	L
	11	11	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	XL
	12	12	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	XXL
	12	7	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	XS
	12	8	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	S
	12	9	Zeleno cmi pulover	Pulover na crte	pulover	odrasli	M

Upozorenje!

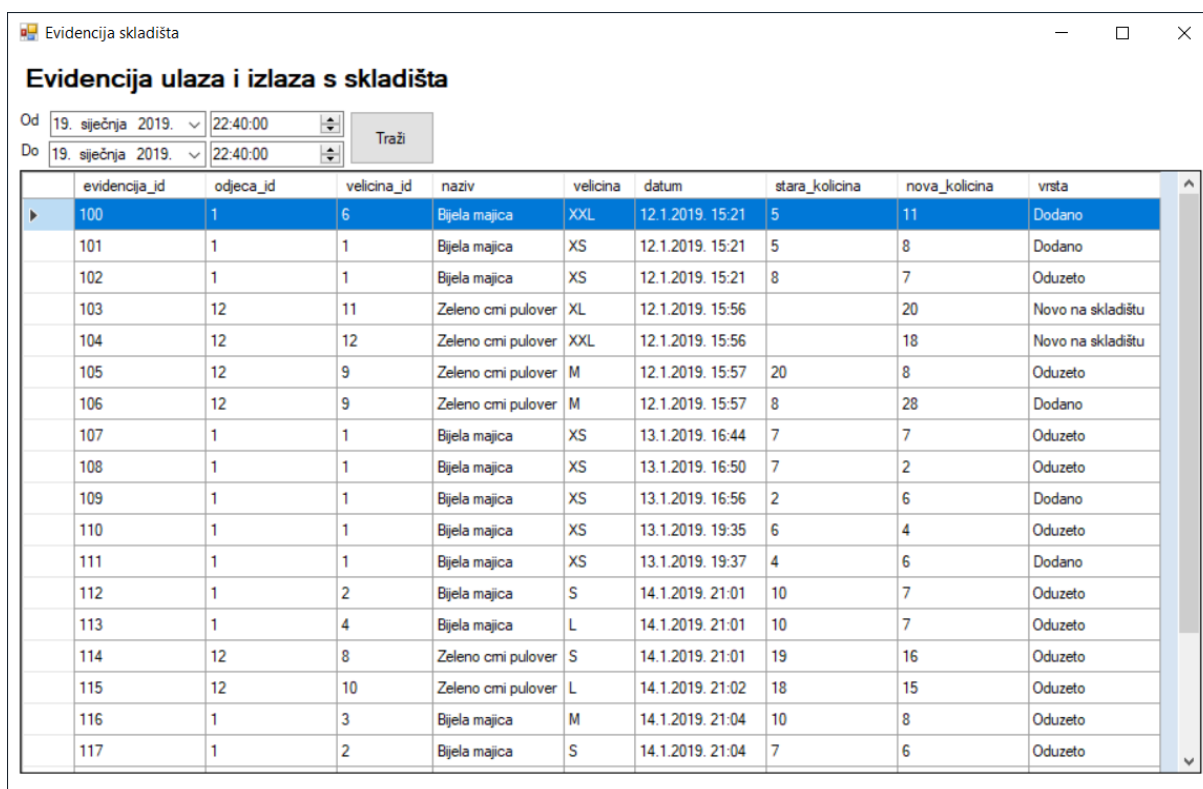
Na skladištu postoji takva stavka!

Slika 10: Dodavanje robe na skladište (Vlastita izrada)

5.6. Evidencija skladišta

Na slici ispod (Slika 11.) možemo vidjeti zaslon koji omogućuje prikaz cjelokupne povjesti događaja na skladištu, tončno koja je količina robe bila prije neke izmjene, a koja nakon izmjene. Također možemo vidjeti koja vrsta posla se na skladištu obavila, tj. jeli nešto dodano ili oduzeto s skladišta pa čak kad je dodana nova kolekcija na stanje skladišta.

Na istom zaslonu je moguće točno pretraživanje događaja po datumu i po vremenu, odnosno prikaz događaja od nekog do nekog drugog vremenskog trenutka, a pritiskom na gumb *Traži* nam se ispisuju svi zapisi koji zadovoljavaju kriterije. Takav prozor možemo smatrati Log sustavom neke baze podataka odnosno skladišta.



The screenshot shows a window titled "Evidencija skladišta" with a subtitle "Evidencija ulaza i izlaza s skladišta". It features search filters for "Od" (19. siječnja 2019.) and "Do" (22:40:00), and a "Traži" button. Below is a table with 10 columns: evidencija_id, odjeca_id, velicina_id, naziv, velicina, datum, stara_kolicina, nova_kolicina, and vrsta. The table contains 17 rows of transaction data.

evidencija_id	odjeca_id	velicina_id	naziv	velicina	datum	stara_kolicina	nova_kolicina	vrsta
100	1	6	Bijela majica	XXL	12.1.2019. 15:21	5	11	Dodano
101	1	1	Bijela majica	XS	12.1.2019. 15:21	5	8	Dodano
102	1	1	Bijela majica	XS	12.1.2019. 15:21	8	7	Oduzeto
103	12	11	Zeleno cmi pulover	XL	12.1.2019. 15:56		20	Novo na skladištu
104	12	12	Zeleno cmi pulover	XXL	12.1.2019. 15:56		18	Novo na skladištu
105	12	9	Zeleno cmi pulover	M	12.1.2019. 15:57	20	8	Oduzeto
106	12	9	Zeleno cmi pulover	M	12.1.2019. 15:57	8	28	Dodano
107	1	1	Bijela majica	XS	13.1.2019. 16:44	7	7	Oduzeto
108	1	1	Bijela majica	XS	13.1.2019. 16:50	7	2	Oduzeto
109	1	1	Bijela majica	XS	13.1.2019. 16:56	2	6	Dodano
110	1	1	Bijela majica	XS	13.1.2019. 19:35	6	4	Oduzeto
111	1	1	Bijela majica	XS	13.1.2019. 19:37	4	6	Dodano
112	1	2	Bijela majica	S	14.1.2019. 21:01	10	7	Oduzeto
113	1	4	Bijela majica	L	14.1.2019. 21:01	10	7	Oduzeto
114	12	8	Zeleno cmi pulover	S	14.1.2019. 21:01	19	16	Oduzeto
115	12	10	Zeleno cmi pulover	L	14.1.2019. 21:02	18	15	Oduzeto
116	1	3	Bijela majica	M	14.1.2019. 21:04	10	8	Oduzeto
117	1	2	Bijela majica	S	14.1.2019. 21:04	7	6	Oduzeto

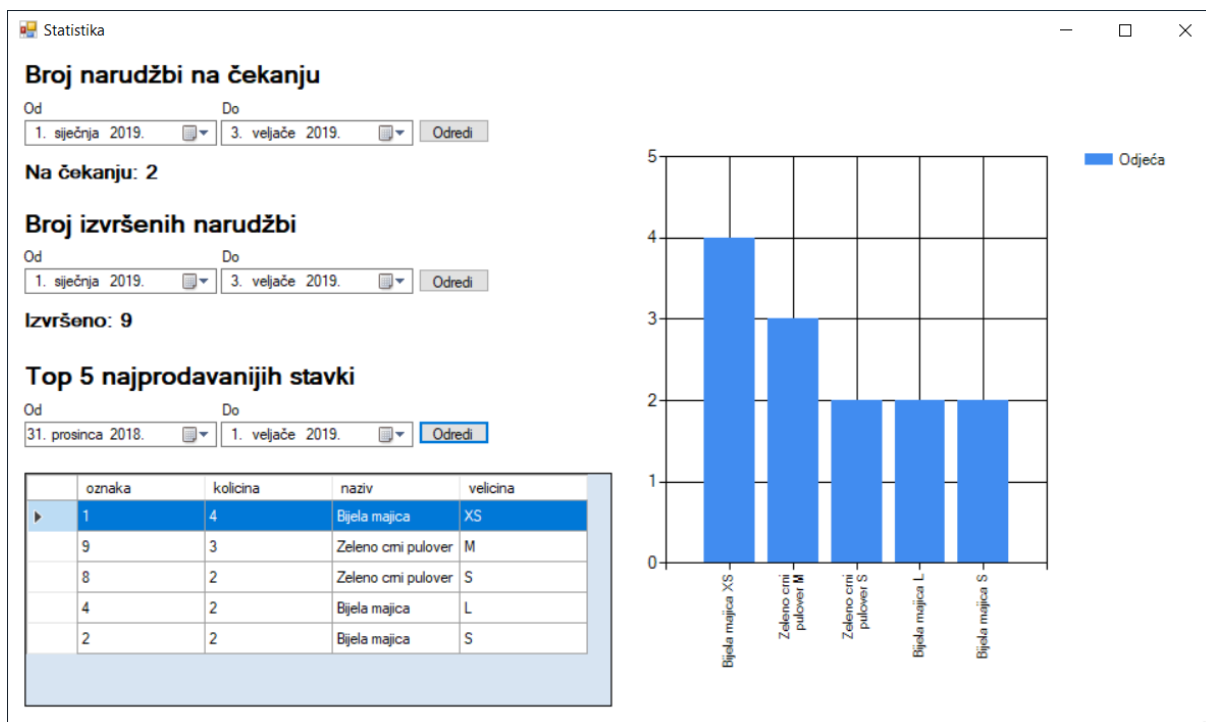
Slika 11: Prikaz evidencije (Vlastita izrada)

5.7. Statistika

Sljedeći prozor je jedan od zanimljivijih, a pogotovo menadžmentu nekog dućana i/ili skladišta, a to je statistika. Na prozoru je mali dio statistike prikazan iako se moglo znatno više implementirati, ali radi demonstracijskih razloga ovakav dio implementacije je drža jednostavnim.

U statistici je moguće pronaći sveukupan broj narudžbi koji je poslan i koji je na čekanju od nekog pa do nekog datuma, tj. omogućen je prikaz broja narudžbi koje nisu zaprimljene u skladištu. Isto tako moguće je vidjeti broj narudžbi koji se izvršio, odnosno koje su zaprimljene u nekom vremenskom intervalu.

Zadnji statistički podataka je prikaz top 5 najprodavanijih artikala. Ovakvom statistikom možemo vidjeti koji odjevni element koje veličine je koliko puta prodan, odnosno koji je najviše puta prodan u nekom vremenskom intervalu. Za takvu statistiku imamo i popratni graf. Iako u bazi podataka i aplikaciji nemamo implementiranu prodaju, oduzimanje stavki s skladišta dućana efektivno i je prodaja jer prodajom se ustvari vrši oduzimanje zaliha s skladišta.



Slika 12: Prikaz zaprimljenih narudžbi (Vlastita izrada)

5.8. Narudžbe

Zadnja dva zaslona su vezana za narudžbe, jedan za prikaz zaprimljenih narudžbi, a drugi za prikaz i potvrdu nezaprimljenih narudžbi. Prozor zaprimljenih narudžbi je ustvari prikaz podataka *dobava_robe* svugdje gdje je *narudzbenica* postavljena tako da je zaprimljena. Unutar samog prozora možemo vidjeti kad je narudžba konkretno poslana i kad je zaprimljena te s tim prozorom ustvari dobivamo temporalni prikaz nekog podataka u bazi podataka.

Prozor nezaprimljenih narudžbi je prikaz svih onih narudžbi koje nisu pristigle u skladište. U slučaju da odaberemo neku od stavki iz tablice i pritisnemo na gumb *Narudžba zaprimljena* tada nezaprimljena narudžba nestaje iz prozora nezaprimljene narudžbe i postavlja se u prozor zaprimljene narudžbe.

Zaprimljene narudžbe

	id_narudzbe	naziv_odjece	narucena_kolicina	datum_narucivanja	datum_zaprimanja
▶	1	Bijela majica	2	10.1.2019. 19:32	11.1.2019. 14:42
	2	Bijela majica	2	10.1.2019. 19:36	11.1.2019. 14:54
	3	Bijela majica	2	10.1.2019. 19:37	10.1.2019. 19:38
	5	Bijela majica	2	11.1.2019. 14:26	11.1.2019. 14:42
	6	Bijela majica	2	11.1.2019. 14:32	11.1.2019. 14:56
	7	Bijela majica	2	11.1.2019. 14:58	11.1.2019. 14:58
	8	Zeleno cmi pullover	20	12.1.2019. 15:57	12.1.2019. 15:57
	9	Bijela majica	4	13.1.2019. 16:50	13.1.2019. 16:56
	10	Bijela majica	2	13.1.2019. 19:35	13.1.2019. 19:37

Slika 13: Prikaz zaprimljenih narudžbi (Vlastita izrada)

Nezaprimljene narudžbe

Narudžba zaprimljena

Filter oznake

	id_narudzbe	naziv_odjece	narucena_kolicina	datum_narucivanja	dobava_robe
▶	11	Zeleno cmi pullover	20	14.1.2019. 21:05	10
	12	Zeleno cmi pullover	20	14.1.2019. 21:05	8

Slika 14: Prikaz narudžbi na čekanju (Vlastita izrada)

6. Zaključak

Ovakvim načinom i pristupom radu s bazama podataka, korištenjem principa aktivnih i temporalnih baza podataka, je značajno olakšan kasniji posao nekog korisnika aplikacije. Korisnik ne treba voditi brigu o tome što i koliko nečega treba naručiti već baza podataka sama odredi koje su nam zalihe potrebne i sl. Osim toga korisnik ima vremenski prikaz o primitku i izvršenju narudžbi čime se lakše može snalaziti u eventualnoj papirologiji. Također je aplikacija jasno i vizualno strukturirana čime se korisniku dodatno olakšava rad unutar baze podataka.

Moguća poboljšanja bi mogla biti sljedeća. U samoj bazi podataka nedostaje prodaja iako se radi o dućanu odjeća te bi se takav dio trebao implementirati s još nekoliko tablica no u svrhu rada i demonstracije baze podataka nije bio potreban. Također bi se baza podataka trebala nadograditi periodičkim naručivanjem robe, npr. na kraju dana ili tjedna, a ne kako je sad nakon svakog nestanka zaliha neke stavke. Za to bi se koristio pgAgent unutar pgAdmina no on iz nekog razloga nema dopuštenja u radu s stvorenom bazom podataka.

7. Literatura

1. Mirko Maleković, Kornelije Rabuzin, (2016.), Naslov: Uvod u baze podataka, Mjesto izdavanja: Varaždin, Fakultet organizacije i informatike
2. Kornelije Rabuzin, (2014.), SQL: napredne teme, Mjesto izdavanja: Varaždin, Fakultet organizacije i informatike
3. Kornelije Rabuzin, Mirko Maleković, Alen Lovrenčić, (2019.), Naslov: The Theory of Active Databases vs. The SQL Standard, Mjesto izdavanja: Faculty of organization and informatics

Popis slika

1.	ERA model baze podataka (Vlastita izrada)	3
2.	Stvaranje PostgreSQL baze podataka (Vlastita izrada)	7
3.	Stvaranje tablice (Vlastita izrada)	8
4.	Dizajniranje tablice (Vlastita izrada)	8
5.	Početni prozor (Vlastita izrada)	17
6.	Početni prozor (Vlastita izrada)	18
7.	Izmjena stanja robe na skladištu (Vlastita izrada)	18
8.	Unos i izmjena asortimana (Vlastita izrada)	19
9.	Dodavanje veličine (Vlastita izrada)	20
10.	Dodavanje robe na skladište (Vlastita izrada)	20
11.	Prikaz evidencije (Vlastita izrada)	21
12.	Prikaz zaprimljenih narudžbi (Vlastita izrada)	22
13.	Prikaz zaprimljenih narudžbi (Vlastita izrada)	23
14.	Prikaz narudžbi na čekanju (Vlastita izrada)	23

8. Prilog 1

1. <https://github.com/fkraljic/skladisteOdjece>