

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

Lucija Baličević

APLIKACIJA ZA PRAĆENJE I POMOĆ U ODRŽAVANJU RASTA BILJKE

PROJEKT

VIŠEAGENTNI SUSTAVI

Varaždin, 2024.

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

Lucija Baličević

Matični broj: 39360/18-R

Studij: Organizacija poslovnih sustava

**APLIKACIJA ZA PRAĆENJE I POMOĆ U ODRŽAVANJU RASTA
BILJKE**

PROJEKT

Mentor:

mag. inf. Tomislav Peharda,

Varaždin, siječanj 2024.

Lucija Baličević

Izjava o izvornosti

Izjavljujem da je ovaj 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.

Autorica potvrdila prihvatanjem odredbi u sustavu FOI Radovi

Sažetak

Tema ovog projektnog rada je aplikacija za praćenje i pomoć u održavanju rasta biljke. Naglasak je stavljen više na izradu i implementaciju same baze. Unutar baze implementirano je nekoliko tablica te nekoliko funkcija i okidača nad definiranim tablicama. Rad s bazom izvršava se u PgAdmin aplikaciji koja radi s PostgreSQL-om. Sama aplikacija izrađena je pomoću programskog jezika Python uz ekstenziju tkinter.

Ključne riječi: baze podataka; temporalne; aktivne; psql; PgAdmin; python

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
4. Implementacija	5
4.1. Implementacija tablica	5
4.1.1. Tablica VrstaCvijeta	5
4.1.2. Tablica VrstaLista	6
4.1.3. Tablica Zalijevanje	7
4.1.4. Tablica Rezanje	8
4.1.5. Tablica Izgled	8
4.1.6. Tablica Briga	9
4.1.7. Tablica Biljke	10
4.1.8. Tablica MojeBiljke	11
4.1.9. Tablica BrigaMojeBiljke	12
4.2. Implementacija funkcija	12
4.3. Implementacija okidača	15
5. Primjeri korištenja	18
6. Zaključak	21
Popis literature	21
Popis slika	22
Popis isječaka koda	24

1. Opis aplikacijske domene

Aplikacija za praćenje i pomoć u održavanju rasta biljke stvorena je kako bi korisniku omogućila brigu o biljkama. Ova aplikacija koristi aktivne i temporalne baze podataka na kojima su definirane funkcije i okudači.

Za izradu baze podataka korišten je PostgreSQL te je za njegovu implementaciju korištena aplikacija pgAdmin4. PgAdmin 4 je alat otvorenog koda za administraciju i upravljanje PostgreSQL bazom podataka. PostgreSQL je moćan sustav za upravljanje relacijskim bazama podataka otvorenog koda poznat po svojoj proširivosti, robusnosti i pridržavanju SQL standarda. PgAdmin 4 dizajniran je za pružanje korisničkog sučelja za upravljanje PostgreSQL bazama podataka.

U bazi je implementirano devet tablica koje će korisniku pomoći za praćenje brige o biljkama. Kako bi se više bazirali na rad nad bazama, a ne na programiranje same aplikacije, unutar baze stvorene su tri funkcije i dva okidača nad tablicama.

Za izradu same aplikacije korišten je python. Python je svestran programski jezik koji se može koristiti za razne aplikacije, uključujući razvoj Windows aplikacija. Kada je riječ o stvaranju Windows desktop aplikacija, postoji nekoliko okvira i biblioteka dostupnih za Python. Jedan od popularnih izbora je Tkinter, koji je standardni GUI (Graphical User Interface) skup alata koji dolazi s Pythonom koja je također korištena u ovom radu.

2. Teorijski uvod

Budući da aplikacija za održavanje biljke koristi aktivnu i temporalnu bazu podataka, u ovom poglavlju teorijski su ukratko opisana spomenuta dva pojma.

2.1. Aktivne baze podataka

Aktivni sustavi baze podataka mogu se definirati kao sustav baze podataka koji prati događaje vrijedne pažnje i, kada se dogode, odmah pokreće odgovarajuću reakciju. Temelje se na arhitekturi *događaj-uvjet-akcija* (eng. *Event-Condition-Action*; skr. ECA) te kao takve omogućuju specificiranje akcija koje treba poduzeti kada se određeni događaji dogode pod određenim uvjetima. Temeljene su na okidačima i kao takve nadopunjuju tradicionalne baze podataka. Također, aktivne baze podataka obuhvaćaju sve osnovne koncepte konvencionalnih baza podataka, uključujući mogućnosti modeliranja podataka i upita te kao takve pružaju podršku svim funkcijama (kao što su npr. definicija podataka, manipulacija podacima i upravljanje pohranom).

Bitna značajka aktivnih baza podataka je sposobnost detekcije pojave događaja te procjena uvjeta navedenih u pravilima, nakon čega slijedi izvršavanje odgovarajuće akcije. S obzirom da se baziraju na automatiziranom radu, aktivnosti koje u drugim slučajevima obavlja aplikacija (korisnik, API i dr.), ovdje to čini baza podataka.

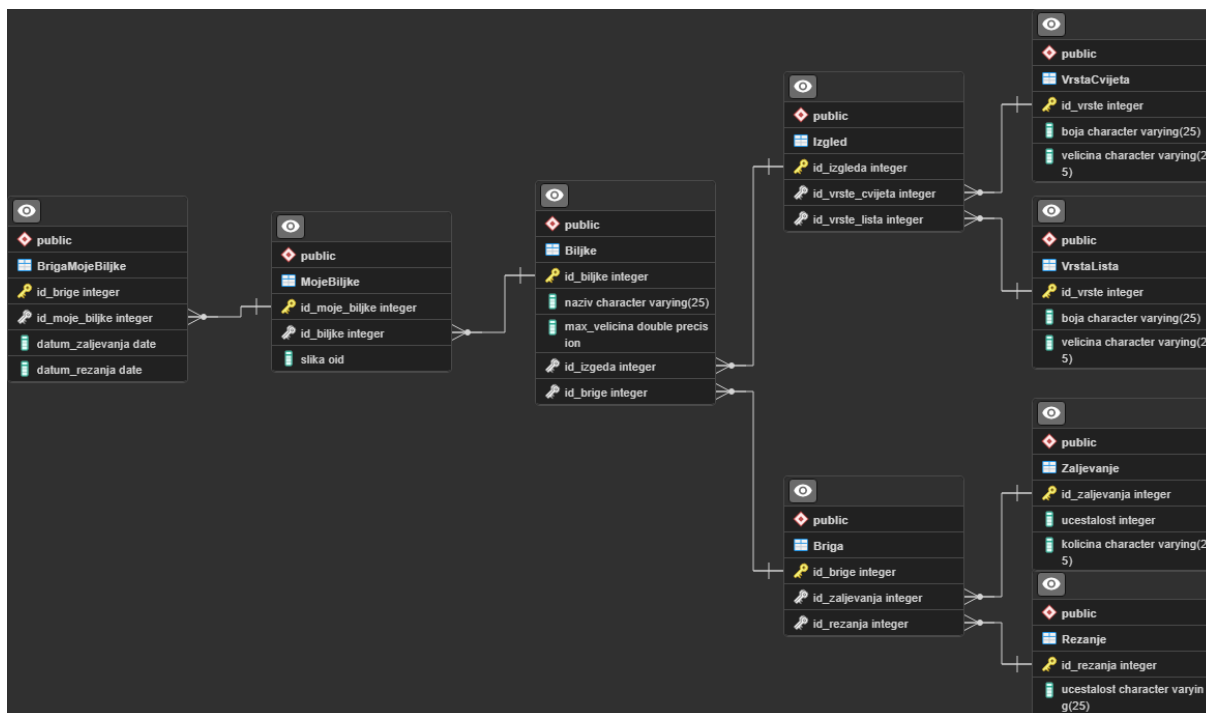
2.2. Temporalne baze podataka

Temporalne baze podataka su dizajnirane na način da uz podatke čuvaju informacije o vremenskim aspektima koji su relevantni za te podatke. Primjerice, jedan od tih vremenskih aspekata može biti vrijeme kada su podaci upisani ili kreirani, što omogućuje praćenje povijesti tih podataka. U nekim situacijama, važnije je pratiti vrijeme trajanja, tj. zapisuje se period od kada do kada su ti podaci važeći. Primjenjuju se u područjima za koje je važno pratiti promjene u podacima tijekom vremena, kao što su npr. u sustavima praćenja inventara, povijesti zaposlenika, financijskim transakcijama i slično.

Razlikuje se "vrijedeće vrijeme", koje označava period tijekom kojega je neka činjenica istinita s obzirom na aplikacijsku domenu, "transakcijsko vrijeme", koje označava period tijekom kojeg je neka činjenica pohranjena u bazi podataka, te bitemporalno vrijeme, koje predstavlja kombinaciju dva prethodno spomenuta vremena. S obzirom da svaki redak u temporalnoj bazi podataka ima vremenske oznake, ova vrsta podataka poboljšati analizu podataka, podržati donošenje odluka i omogućiti praćenje promjena kroz vrijeme, što je ključno u mnogim poslovnim i istraživačkim scenarijima.

3. Model baze podataka

Model baze podataka prikazuje se u dijagramu entiteta i odnosa koji nazivamo ERA model. Ovaj model je vizualni prikaz modela podataka koji prikazuje entitete, attribute, odnose i ograničenja unutar sustava. Obično se koristi u dizajnu baze podataka za pružanje jasnog i sažetog pregleda strukture baze podataka. ERA model izrađen u pgAdminu koji se korisni u bazi prikazan je na slici ispod.



Slika 1: ERA model baze podataka (samostalna izrada)

Sve tablice u ERA modelu su potrebne kako bi bilo moguće implementirati aplikaciju za praćenje i pomoć u održavanju rasta biljke. ERA prikazuje da se baza sastoji od devet tablica:

- VrstaCvijeta - sadrži glavne informacije o izgledu cvijeta kao što su boja i veličina cvijeta
- VrstaLista - sadrži glavne informacije o izgledu lišća kao što su boja i veličina lišća
- Zaljevanje - sadrži informacije o učestalosti i količini zaljevanja biljke
- Rezanje - sadrži informacije o učestalosti rezanja biljke
- Izgled - sadrži id-eve VrsteCvijeta i VrsteLista kojima se opisuje izgled cijele biljke
- Briga - sadrži id-eve Zaljevanja i Rezanja kojima je opisana sama briga biljke
- Biljke - osim naziva i maksimalne veličine biljke, sadrži sve informacije koje su definirane u prijašnjim tablicama

- MojeBiljke - sadrži popis svih biljki koje korisnik posjeduje kao i sliku njegove biljke
- BrigaMojeBiljke - sadrži povijest zalijevanja i rezanja pojedine korisnikove biljke

4. Implementacija

Za uspješno korištenje aplikacije, potrebno je implementirati bazu podataka koja se sastoji od tablica, funkcija i okidača. Nakon što su implementirani glavni dijelovi, moguće je preći na izradu same aplikacije koja će biti povezana na bazu te će biti omogućen rad s bazom i nad bazom.

4.1. Implementacija tablica

Kod implementacije tablica korišten je psql za implementaciju svih tablica. Svaka tablica mora sadržavati svoj id kako bi se ostale tablice mogle pozivati na nju.

4.1.1. Tablica VrstaCvijeta

Prva tablica koja je implementirana je tablica VrstaCvijeta. Ova tablica sadrži informacije o cvijetu kao što su njegova boja i veličina. Tablica je preirana na idući način:

```
1 CREATE TABLE VrstaCvijeta (  
2     id_vrste SERIAL PRIMARY KEY NOT NULL,  
3     boja VARCHAR(25),  
4     velicina VARCHAR(25)  
5 );
```

Isječak koda 1: Isječka koda implementacije tablice VrstaCvijeta

Nakon implementacije potrebno je i zapisati vrijednosti u tablicu. Svaka biljka može imati više boja cvijetova te je iz tog razloga upisano više boja. Također svaki cvijet može biti mali, srednji ili veliki, radi toga, za svaku boju definirana je i svaka veličina.

```

1  INSERT INTO "VrstaCvijeta" ("boja", "velicina")
2  VALUES
3      ('bijela','mali'),
4      ('bijela','srednji'),
5      ('bijela','veliki'),
6      ('zuti','mali'),
7      ('zuti','srednji'),
8      ('zuti','veliki'),
9      ('ljubicasti','mali'),
10     ('ljubicasti','srednji'),
11     ('ljubicasti','veliki'),
12     ('rozi','mali'),
13     ('rozi','srednji'),
14     ('rozi','veliki'),
15     ('narancasti','mali'),
16     ('narancasti','srednji'),
17     ('narancasti','veliki'),
18     ('crveni','mali'),
19     ('crveni','srednji'),
20     ('crveni','veliki');

```

Isječak koda 2: Isječak koda upisivanja u tablicu VrstaCvijeta

4.1.2. Tablica VrstaLista

Druga tablica koja je implementirana je tablica VrstaLista. Ova tablica je vrlo slična prijašnjoj, ali sadrži podatke o lisu biljke. Tablica je implementirana na način:

```

1  CREATE TABLE VrstaLista (
2      id_vrste SERIAL PRIMARY KEY NOT NULL,
3      boja VARCHAR(25),
4      velicina VARCHAR(25)
5  );

```

Isječak koda 3: Isječak koda implementacije tablice VrstaLista

Logika upisa ista je kao i za prijašnju tablicu, odnosno postoji nekoliko boja litova koji su zapisani s odgovarajućim veličinama.

```

1  INSERT INTO "VrstaLista" ("boja", "velicina")
2  VALUES
3      ('tamnosmedi','mali'),
4      ('tamnosmedi','srednji'),
5      ('tamnosmedi','veliki'),
6      ('zeleni','mali'),
7      ('zeleni','srednji'),
8      ('zeleni','veliki'),
9      ('svjetlosmedi','mali'),
10     ('svjetlosmedi','srednji'),
11     ('svjetlosmedi','veliki'),
12     ('zuti','mali'),
13     ('zuti','srednji'),
14     ('zuti','veliki');

```

Isječak koda 4: Isječka koda upisivanja u tablicu VrstaLista

4.1.3. Tablica Zalijevanje

Iduća tablica sadrži informacije o zalijevanju biljke. Za zalijevanje je potrebno zanti učestalost zalijevanja i količina. Učestalost prikazuje koliko dana treba proći između dva zalijevanja, a količina je samo definirana za korisnika kako bi znao koliko količinu vode dati biljci. Tablica je implementirana na način:

```

1  CREATE TABLE Zalijevanje (
2      id_zaljevanja SERIAL PRIMARY KEY NOT NULL,
3      ucestalost integer,
4      kolicina VARCHAR(25)
5  );

```

Isječak koda 5: Isječka koda implementacije tablice Zalijevanje

Kod upisa u tablicu provjerene su informacije o zalijevanju biljke koje će kasnije biti upisane u bazu.

```

1  INSERT INTO Zalijevanje ("ucestalost", "kolicina")
2  VALUES
3      (16, 'malo'),
4      (18, 'malo'),
5      (8, 'umjereno'),
6      (9, 'umjereno'),
7      (10, 'umjereno'),
8      (12, 'umjereno'),
9      (18, 'umjereno'),
10     (8, 'puno'),
11     (10, 'puno');

```

Isječak koda 6: Isječka koda upisivanja u tablicu Zalijevanje

4.1.4. Tablica Rezanje

Iduća tablica sadrži informacije o rezanju biljke. Za rezanje biljke potrebno je zanti učestalost rezanja kao i količinu. Količina je određena kako bi korisnik znao na koji način i što odrezati. Učestalost označava koliko dana treba proći između dva rezanja.. Tablica je implementirana na način:

```
1 CREATE TABLE Rezanje (  
2   id_rezanja SERIAL PRIMARY KEY NOT NULL,  
3   ucestalost integer,  
4   kolicina VARCHAR(25)  
5 );
```

Isječak koda 7: Isječka koda implementacije tablice Rezanje

Logika upisa u tablicu je jednaka prijašnjem upisu.

```
1 INSERT INTO Rezanje ("ucestalost", "kolicina")  
2 VALUES  
3     (20, 'suho lisce'),  
4     (20, 'suho lisce i cvijece'),  
5     (25, 'suho lisce'),  
6     (25, 'suho lisce i cvijece'),  
7     (30, 'suho lisce'),  
8     (30, 'suho lisce i cvijece');
```

Isječak koda 8: Isječka koda upisivanja u tablicu Rezanje

4.1.5. Tablica Izgled

Prva tablica koja sadrži vanjeske ključeve je tablica Izgled. U ovoj tablici je opisan izgled biljke na način da su upisani ključevi iz tablica VrstaLista i VrstaCvijeta. Tablica je implementirana kodom:

```
1 CREATE TABLE Izgled (  
2   id_izgleda SERIAL PRIMARY KEY NOT NULL,  
3   id_vrste_cvijeta INT REFERENCES VrstaCvijeta(id_vrste),  
4   id_vrste_lista INT REFERENCES VrstaLista(id_vrste) NOT NULL  
5 );
```

Isječak koda 9: Isječka koda implementacije tablice Izgled

Kod upisa vrijednosti u tablicu, ponovno su istražene informacije o biljkama koje će kasnije biti upisane u bazu te su na taj način pridružene vrijednosti izgleda cvijeta i izgleda lista. Ukoliko je vrijednosti koja prikazuje vrstu cvijeta jednaka NULL, to znači da biljka nema cvijetove.

```

1 INSERT INTO Izgled ("id_vrste_cvijeta", "id_vrste_lista")
2   VALUES
3     (1,5),
4     (2,2),
5     (4,6),
6     (3,3),
7     (2,10),
8     (1,6),
9     (7,1),
10    (1,7),
11    (8,2),
12    (NULL,11),
13    (NULL,3),
14    (NULL,2),
15    (NULL,1),
16    (NULL,7);

```

Isječak koda 10: Isječka koda upisivanja u tablicu Izgled

4.1.6. Tablica Briga

Iduća tablica koja također sadrži dva vanjska ključa je tablica Briga. Ova tablica sadrži informacije i brizi biljke na način da spaja vrijednosti zalijevanja i rezanja biljke. Implementirana je na način:

```

1 CREATE TABLE Briga (
2   id_brige SERIAL PRIMARY KEY NOT NULL,
3   id_zalijevanja INT REFERENCES Zalijevanje(id_zaljevanja) NOT NULL,
4   id_rezanja INT REFERENCES Rezanje(id_rezanja)
5 );

```

Isječak koda 11: Isječka koda implementacije tablice Briga

Prije upisa u tablicu, istražene su informacije o brizi biljke te su povezane vrijednosti iz tablica Rezanje i Zalijevanje na idući način:

```

1 INSERT INTO Briga ("id_zalijevanja", "id_rezanja")
2   VALUES
3       (3,6),
4       (7,2),
5       (3,1),
6       (3,5),
7       (7,5),
8       (7,6),
9       (5,6),
10      (7,4),
11      (9,4),
12      (8,5),
13      (6,5),
14      (8,6),
15      (3,2),
16      (4,1),
17      (2,2),
18      (1,1),
19      (6,1),
20      (9,6);

```

Isječak koda 12: Isječka koda upisivanja u tablicu Briga

4.1.7. Tablica Biljke

Tablica Biljke sadrži informacije o biljkama kao što su naziv, maksimalna veličina, izgled i briga biljke. Vrijednosti izgleda i brige pridružene su vanjskim ključevima.

```

1 CREATE TABLE Biljke (
2     id_biljke SERIAL PRIMARY KEY NOT NULL,
3     naziv VARCHAR(30),
4     maksimalnaVelicina INTEGER,
5     id_izgleda INT REFERENCES Izgled(id_izgleda),
6     id_brige INT REFERENCES Briga(id_brige)
7 );

```

Isječak koda 13: Isječka koda implementacije tablice Biljke

Tablica Biljke popunjena je vrijednostima koje su potrebne za brigu biljke, a upis u tablicu odrađen je na način:

```

1  INSERT INTO biljke ("naziv","maksimalnavelicina","id_izgleda","id_brige")
2  VALUES
3      ('Zeleni ljiljan',45,1,1),
4      ('Svekrvin jezik',120,1,1),
5      ('Spathiphyllum',90,2,6),
6      ('Zlatni puzavac',300,10,3),
7      ('Fukus gumijevac',300,11,4),
8      ('Zamioculcas',75,11,5),
9      ('Aloe vera',60,3,6),
10     ('Monstera',300,4,1),
11     ('Lirasti fikus',300,11,7),
12     ('Biljka žad',90,5,8),
13     ('Sretni bambus',150,6,9),
14     ('Bostonska paprat',60,12,10),
15     ('Sobna palma',180,12,11),
16     ('Spatifilum',90,4,12),
17     ('Afrička ljubičica',30,7,13),
18     ('Filodendron',30,13,14),
19     ('Krunica',60,8,15),
20     ('Aspidistra elatior',60,11,16),
21     ('Kineska dolar',30,14,17),
22     ('Zračna biljka',30,9,18);

```

Isječak koda 14: Isječka koda upisivanja u tablicu Biljke

4.1.8. Tablica MojeBiljke

Tablica MojeBiljke sadrži informacije o biljkama koje korisnik posjednuje. Ova tablica sadrži vanjski ključ koji se povezuje na tablicu Biljke kako bi se mogle izvuci sve informacije o biljci. Osim toga, sadrži i stupac slika u koju korisnik može unijeti svoju sliku biljke. Implementirana je tablica na način:

```

1  CREATE TABLE MojeBiljke (
2      id_moje_biljke SERIAL PRIMARY KEY NOT NULL,
3      id_biljke INT REFERENCES Biljke(id_biljke) NOT NULL,
4      slika OID
5  );

```

Isječak koda 15: Isječka koda implementacije tablice MojeBiljke

Kod dodavanja vrijednosti dodani su samo ključevi biljke:

```

1  INSERT INTO MojeBiljke ("id_biljke")
2  VALUES
3      (1), (2), (3);

```

Isječak koda 16: Isječka koda upisivanja u tablicu MojeBiljke

4.1.9. Tablica BrigaMojeBiljke

Zadnja tablica koju je potrebno implementirati je BrigaMojeBiljke. Ova tablica će sadržavati povijest brige o biljci odnosno zalijevanja i rezanja biljke. Kako bi znali o kojoj biljci se radi, potreban je i vanjski ključ iz tablice MojeBiljke. Implementirana je na način:

```
1 CREATE TABLE BrigaMojeBiljke (  
2     id_brige SERIAL PRIMARY KEY NOT NULL,  
3     id_moje_biljke INT REFERENCES MojeBiljke(id_moje_biljke),  
4     datumZalijevanja DATE,  
5     datumRezanja DATE  
6 );
```

Isječak koda 17: Isječka koda implementacije tablice BrigaMojeBiljke

4.2. Implementacija funkcija

Za lakše ispisivanje iz baze, kreirane su tri funkcije unutar baze.

Prva funkcija je DetaljiSvihBiljki. Ova funkcija je stvorena kako bi se unutar aplikacije lakše moglo doći do svih informacija svih biljki. Ova funkcija zapravo čita podatke iz nekoliko tablica te radi toga olakšava poziv i prikaz unutar aplikacije. Nakon poziva funkcije, ona će prikazati informacije o biljci kao što su njezin naziv, maksimalna veličina, učestalost zalijevanja, učestalost rezanja, boja i veličina lista te boja i veličina lista.

```

1 CREATE OR REPLACE FUNCTION DetaljiSvihBiljki()
2 RETURNS TABLE (
3     naziv_biljke VARCHAR,
4     maksimalna_velicina INTEGER,
5     ucestalost_zalijevanja INTEGER,
6     ucestalost_rezanja INTEGER,
7     list_boja VARCHAR,
8     list_velicina VARCHAR,
9     cvijet_boja VARCHAR,
10    cvijet_velicina VARCHAR
11 ) AS
12 $$
13 BEGIN
14     RETURN QUERY
15     SELECT
16         b."naziv",
17         b."maksimalnavelicina",
18         z."ucestalost" AS ucestalostzaljevanja,
19         r."ucestalost" AS ucestalostrezanja,
20         vl."boja" AS vrstalista_boja,
21         vl."velicina" AS vrstalista_velicina,
22         vc."boja" AS vrstacvijeta_boja,
23         vc."velicina" AS vrstacvijeta_velicina
24     FROM
25         "biljke" b
26     LEFT JOIN
27         "izgled" i ON b."id_izgleda" = i."id_izgleda"
28     LEFT JOIN
29         "briga" br ON b."id_brige" = br."id_brige"
30     LEFT JOIN
31         "vrstalista" vl ON i."id_vrste_lista" = vl."id_vrste"
32     LEFT JOIN
33         "vrstacvijeta" vc ON i."id_vrste_cvijeta" = vc."id_vrste"
34     LEFT JOIN
35         "zalijevanje" z ON br."id_zalijevanja" = z."id_zaljevanja"
36     LEFT JOIN
37         "rezanje" r ON br."id_rezanja" = r."id_rezanja";
38
39     RETURN;
40 END;
41 $$ LANGUAGE plpgsql;

```

Isječak koda 18: Implementacija funkcije DetaljiSvihBiljki

Iduća funkcija je vrlo slična prijašnjoj, ali od korisnika traži upis naziva biljke te prikazuje detalje samo o toj biljci. Implementirana je na način:

```

1 CREATE OR REPLACE FUNCTION DetaljiBiljke(nazivBiljke VARCHAR)
2 RETURNS TABLE (
3     maksimalna_velicina INTEGER,
4     ucestalost_zalijevanja INTEGER,
5     ucestalost_rezanja INTEGER,
6     list_boja VARCHAR,
7     list_velicina VARCHAR,
8     cvijet_boja VARCHAR,
9     cvijet_velicina VARCHAR
10 ) AS
11 $$
12 BEGIN
13     RETURN QUERY
14     SELECT
15         b."maksimalnavelicina",
16         z."ucestalost" AS ucestalostzalijevanja,
17         r."ucestalost" AS ucestalostrezanja,
18         vl."boja" AS vrstalista_boja,
19         vl."velicina" AS vrstalista_velicina,
20         vc."boja" AS vrstacvijeta_boja,
21         vc."velicina" AS vrstacvijeta_velicina
22 FROM
23     "biljke" b
24 LEFT JOIN
25     "izgled" i ON b."id_izgleda" = i."id_izgleda"
26 LEFT JOIN
27     "briga" br ON b."id_brige" = br."id_brige"
28 LEFT JOIN
29     "vrstalista" vl ON i."id_vrste_lista" = vl."id_vrste"
30 LEFT JOIN
31     "vrstacvijeta" vc ON i."id_vrste_cvijeta" = vc."id_vrste"
32 LEFT JOIN
33     "zalijevanje" z ON br."id_zalijevanja" = z."id_zalijevanja"
34 LEFT JOIN
35     "rezanje" r ON br."id_rezanja" = r."id_rezanja"
36 WHERE
37     b."naziv" = nazivBiljke;
38
39 RETURN;
40 END;
41 $$ LANGUAGE plpgsql;

```

Isječak koda 19: Isječka koda implementacije tablice DetaljiBiljke

Zadnja funkcija služi za ljepši ispis povijesti brige o svim biljkama. Tablica BrigeMojeBiljke već sadrži zapise svih briga o biljkama odnosno sve datume zalijevanja i rezanja biljki, međutim ne sadrži direktno naziv biljke. Iz toga razloga stvorena je funkcija koja će povezati tablice BrigaMojeBiljke, MojeBiljke i Biljke kako bi se mogao ispisati i naziv biljaka. Implementirana je na idući način:

```
1 CREATE OR REPLACE FUNCTION DetaljiBrigeBiljaka()
2 RETURNS TABLE (
3     naziv_biljke VARCHAR,
4     datum_zalijevanja DATE,
5     datum_rezanja DATE
6 ) AS
7 $$
8 BEGIN
9     RETURN QUERY
10    SELECT
11        b."naziv",
12        bm."datumzalijevanja",
13        bm."datumrezanja"
14    FROM
15        "brigamojebiljke" bm
16    JOIN "mojebiljke" mb ON bm."id_moje_biljke" = mb."id_moje_biljke"
17    JOIN "biljke" b ON mb."id_biljke" = b."id_biljke";
18    RETURN;
19 END;
20 $$ LANGUAGE plpgsql;
```

Isječak koda 20: Implementacija funkcije DetaljiBrigeBiljaka

4.3. Implementacija okidača

U PostgreSQL-u okidači prikazuju objekte baze podataka koji su povezani s tablicom i automatski se izvršavaju ili aktiviraju kao odgovor na određene događaje. Za ovu bazu, kreirana su dva okidača nad tablicom BrigaMojeBiljke.

Prvi okidač koristi se kod upisa u tablicu BrigaMojeBiljke je ZaliBiljkuTrigger, a poziva funkciju ZaliBiljku. Ova funkcija provjerava treba li biljku zaliti na način da pročita koji je zadnji datum zalijevanja biljke. Tu vrijednost uspoređuje s današnjim datumom, te ako je od zadnjeg datuma zalijevanja prošlo onoliko dana koliko je zapisano u tablici Zalijevanje pod učestalost, onda se biljka može ponovno zaliti, odnosno može se zapisati novi datum zalijevanja. Ukoliko to nije slučaj, funkcija će javiti korisniku da nije potrebno još zaliti biljku. Funkcija i okidač su implementirani na idući način:

```

1 CREATE OR REPLACE FUNCTION ZaliBiljku()
2 RETURNS TRIGGER AS
3 $$
4 DECLARE
5     zadnjezalijevanje DATE;
6     ucestalostzalijevanja INT;
7     danaodzalijevanja INT;
8 BEGIN
9     SELECT bm."datumzalijevanja", z."ucestalost"
10    INTO zadnjezalijevanje, ucestalostzalijevanja
11   FROM "mojebiljke" mb
12  JOIN "biljke" b ON mb."id_biljke" = b."id_biljke"
13  JOIN "brigamojebiljke" bm ON mb."id_moje_biljke" = bm."id_moje_biljke"
14  JOIN "zalijevanje" z ON z."id_zaljevanja" = z."id_zaljevanja"
15  WHERE mb."id_moje_biljke" = NEW."id_moje_biljke"
16  ORDER BY bm."datumzalijevanja" DESC
17  LIMIT 1;

18     danaodzalijevanja := CURRENT_DATE - zadnjezalijevanje;

19     IF danaodzalijevanja >= ucestalostzalijevanja THEN
20         RETURN NEW;
21     ELSE
22         RAISE EXCEPTION 'Biljku ne treba jos zaliti!';
23     END IF;
24     RETURN NEW;
25 END;
26 $$
27 LANGUAGE plpgsql;

28 CREATE TRIGGER ZaliBiljkuTrigger
29 BEFORE INSERT ON brigamojebiljke
30 FOR EACH ROW
31 EXECUTE FUNCTION ZaliBiljku();

```

Isječak koda 21: Implementacija funkcije ZaliBiljku i okidača ZaliBiljkuTrigger

Drugi okidač radi vrlo sličnu stvar. Kako se biljka češće treba zalijevati nego rezati, okidač je postavljen kod ažuriranja tablice BrigaMojeBiljke. Okidač poziva funkciju IzreziBiljku čija je logika jednaka logici prijašnje funkcije, ali u ovom slučaju provjerava učestalost iz tablice Rezanje. Okidač je nazvan IzreziBiljkuTrigger, a implementirano je na način:

```

1 CREATE OR REPLACE FUNCTION IzreziBiljku()
2 RETURNS TRIGGER AS
3 $$
4 DECLARE
5     zadnjerezanja DATE;
6     ucestalostrezanja INT;
7     danaodrezanja INT;
8 BEGIN
9     SELECT bm."datumrezanja", r."ucestalost"
10    INTO zadnjerezanja, ucestalostrezanja
11   FROM "mojebiljke" mb
12  JOIN "biljke" b ON mb."id_biljke" = b."id_biljke"
13  JOIN "brigamojebiljke" bm ON mb."id_moje_biljke" = bm."id_moje_biljke"
14  JOIN "rezanje" r ON r."id_rezanja" = r."id_rezanja"
15  WHERE mb."id_moje_biljke" = NEW."id_moje_biljke"
16  ORDER BY bm."datumrezanja" DESC
17  LIMIT 1;

18     danaodrezanja := CURRENT_DATE - zadnjerezanja;

19     IF danaodrezanja >= ucestalostrezanja OR danaodrezanja IS NULL THEN
20         RETURN NEW;
21     ELSE
22         RAISE EXCEPTION 'Biljku ne treba jos rezati!';
23     END IF;
24     RETURN NEW;
25 END;
26 $$
27 LANGUAGE plpgsql;

28 CREATE TRIGGER IzreziBiljkuTrigger
29 BEFORE UPDATE ON brigamojebiljke
30 FOR EACH ROW
31 EXECUTE FUNCTION IzreziBiljku();

```

Isječak koda 22: Implementacija funkcije IzreziBiljku i okidača IzreziBiljkuTrigger

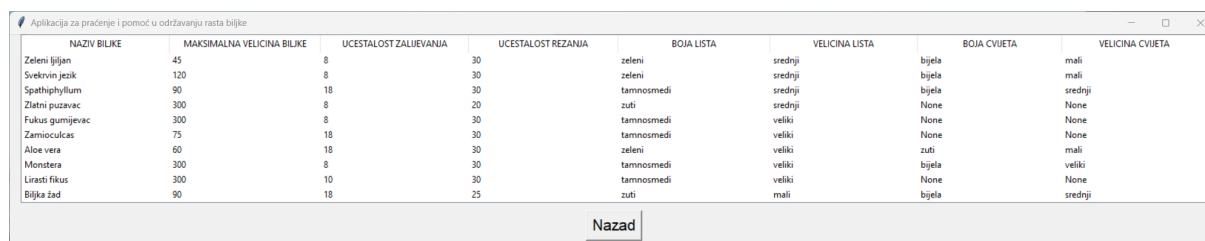
5. Primjeri korištenja

Kao što je prijašnje rečeno, za implementaciju aplikacije korišten je programski jezik python s ekstenzijom tkinter. Za početak potrebno je stvoriti konekciju s bazom na idući način:

```
1 self.db_params = {
2     'host': 'localhost',
3     'port': 5432,
4     'user': 'postgres',
5     'password': '2829',
6     'database': 'TBP-Projekt',
7 }
```

Isječak koda 23: Spajanje na bazu

Unutar aplikacije nalazi se gumb 'Sve biljke'. Prilikom klika na ovaj gumb, aplikacija ispisuje glave informacije o biljkama koje se nalaze u bazi.



NAZIV BILJKE	MAKSIMALNA VELICINA BILJKE	UČESTALOST ZALIEVANJA	UČESTALOST REZANJA	BOJA LISTA	VELICINA LISTA	BOJA CVIJETA	VELICINA CVIJETA
Zelena lilijska	45	8	30	zeleni	srednji	bijela	mali
Srebrni jezik	120	8	30	zeleni	srednji	bijela	mali
Spathiphyllum	90	18	30	tamnosmeđi	srednji	bijela	srednji
Zlatni puzavac	300	8	20	zuti	srednji	None	None
Fikus gumičevac	300	8	30	tamnosmeđi	veliki	None	None
Zamioculcas	75	18	30	tamnosmeđi	veliki	None	None
Aloe vera	60	18	30	zeleni	veliki	zuti	mali
Monstera	300	8	30	tamnosmeđi	veliki	bijela	veliki
Lirasti fikus	300	10	30	tamnosmeđi	veliki	None	None
Biljka žad	90	18	25	zuti	mali	bijela	srednji

Slika 2: Prikaz detalja svih biljki (samostalna izrada)

Za učitavanje podataka iz baze koji će se ispisati na ovo mjesto, korištena je prijašnje definirana funkcija DetaljiSvihBiljki koja se nalazi unutar baze. Kako funkcija već postoji, potrebno ju je samo pozvati iz baze.

```
1 def ucitaj_detalje_svih_biljki(self):
2     try:
3         connection = psycopg2.connect(**self.db_params)
4         with connection.cursor() as cursor:
5             cursor.callproc("DetaljiSvihBiljki")
6             detalji_svih_biljki_data = cursor.fetchall()
7
8         return detalji_svih_biljki_data
9
10    except psycopg2.Error as e:
11        simpledialog.messagebox.showerror("Error", f"\n{e}")
12        return []
13
14    finally:
15        if connection:
16            connection.close()
```

Isječak koda 24: Isječak koda učitavanja detalja biljki

U prikazu svih biljki, korisnik može kliknuti gumb 'Zali bilju' ili gumb 'Izreži biljku'. Prilikom klika na jedan od dva gumba, izvršava se upis ili ažuriranje u tablicu preko okidača koji su ranije definirani. Logika gumbova je vrlo slična, te je iz tog razloga prikazana samo logika nad gumbom 'Zali biljku'.

```

1 with connection.cursor() as cursor:
2     cursor.execute(
3         "INSERT INTO brigamojebiljke (id_moje_biljke, datumzalijevanja) VALUES (%s,
4             ↳ %s)",
5         (biljka_id, current_date)
6     )
7     connection.commit()
8     messagebox.showinfo("Info", f"Zalivena biljka s ID-em: {biljka_id}")

```

Isječak koda 25: Isječak koda zalijevanja biljke

Ovisno o tome treba li se biljka zaliti ili ne, aplikacija daje obavijest korisniku. Ako ju ne treba zaliti izbaciti će poruku 'Biljku ne treba jos zaliti!' kao što je definirano u funkciji u bazi, a ako ju treba zaliti, napisat će 'Zalivena biljka s ID-em: ' gdje će napisati točan id biljke. Ista stvar je i kod rezanja biljke.

Također u ovom dijelu nalazi se i povijest brige o biljkama. Način na koji se prikazuje povijest je prikazana na slici ispod.

----- Povijest brige biljaka -----		
NAZIV BILJKE	DATUMI ZALIJEVANJA	DATUMI REZANJA
Zeleni ljiljan	2024-01-21	None
Zeleni ljiljan	2024-01-01	None
Zeleni ljiljan	2023-12-15	2023-12-15
Spathiphyllum	2024-01-20	2024-01-20
Spathiphyllum	2023-12-15	2023-12-15
Monstera	2024-01-21	None
Monstera	2023-12-15	2023-12-15

Nazad

Slika 3: Prikaz detalja svih biljki (samostalna izrada)

Za ovaj ispis korištena je funkcija `DetaljiBrigeBiljaka` čija je implementacija ranije objašnjena. Sami poziv funkcije je vrlo jednostavan:

```
1 def ucitaj_detalji_brige_biljaka(self):
2     try:
3         connection = psycopg2.connect(**self.db_params)
4         with connection.cursor() as cursor:
5             cursor.callproc("DetaljiBrigeBiljaka")
6             detalji_svih_biljki_data = cursor.fetchall()
7
8             return detalji_svih_biljki_data
9
10    except psycopg2.Error as e:
11        simplifiedialog.messagebox.showerror("Error", f"\n{e}")
12        return []
13
14    finally:
15        if connection:
16            connection.close()
```

Isječak koda 26: Isječak koda poziva funkcije DetaljiBrigeBiljaka

6. Zaključak

U ovom projektu razvijena je sveobuhvatna aplikacija za brigu biljaka, koja koristi mogućnosti aktivnih i temporalnih baza podataka. Implementacija je provedena pomoću PostgreSQL-a u pgAdmin-u, obuhvaćajući izradu devet tablica, tri funkcije i dva okidača. Ova struktura baze podataka služi kao temelj za učinkovito upravljanje i organiziranje podataka.

Odabir PostgreSQL-a kao sustava za upravljanje relacijskom bazom podataka, zajedno s uključivanjem aktivnih i temporalnih značajki, pruža osnovu za izradu aplikacije za brigu biljaka. Sama aplikacije izrađena je uz python koji omogućava vrlo jednostavan rad s bazom i s njezinim komponentama kao što su funkcije i okidači.

Uz izradu vrlo jednostavne aplikacije, korisnik ima mogućnost brinuti se o svojim biljkama. Briga o biljci vrši se kroz zalijevanje i rezanje biljke za koje postoje implementirane funkcije unutar baze. Uz navedene funkcije korišteni su i okidači koji olakšavaju provjeru unosa i ažuriranja podataka bez dodatne implementacije logike u samoj aplikaciji.

Popis slika

1.	ERA model baze podataka (samostalna izrada)	3
2.	Prikaz detalja svih biljki (samostalna izrada)	18
3.	Prikaz detalja svih biljki (samostalna izrada)	19

Popis isječaka koda

1.	Isječka koda implementacije tablice VrstaCvijeta	5
2.	Isječka koda upisivanja u tablicu VrstaCvijeta	6
3.	Isječka koda implementacije tablice VrstaLista	6
4.	Isječka koda upisivanja u tablicu VrstaLista	7
5.	Isječka koda implementacije tablice Zalijevanje	7
6.	Isječka koda upisivanja u tablicu Zalijevanje	7
7.	Isječka koda implementacije tablice Rezanje	8
8.	Isječka koda upisivanja u tablicu Rezanje	8
9.	Isječka koda implementacije tablice Izgled	8
10.	Isječka koda upisivanja u tablicu Izgled	9
11.	Isječka koda implementacije tablice Briga	9
12.	Isječka koda upisivanja u tablicu Briga	10
13.	Isječka koda implementacije tablice Biljke	10
14.	Isječka koda upisivanja u tablicu Biljke	11
15.	Isječka koda implementacije tablice MojeBiljke	11
16.	Isječka koda upisivanja u tablicu MojeBiljke	11
17.	Isječka koda implementacije tablice BrigaMojeBiljke	12
18.	Implementacija funkcije DetaljiSvihBiljki	13
19.	Isječka koda implementacije tablice DetaljiBiljke	14
20.	Implementacija funkcije DetaljiBrigeBiljaka	15
21.	Implementacija funkcije ZaliBiljku i okidača ZaliBiljkuTrigger	16
22.	Implementacija funkcije IzreziBiljku i okidača IzreziBiljkuTrigger	17
23.	Spajanje na bazu	18
24.	Isječak koda učitavanja detalja biljki	18

25. Isječak koda zalijevanja biljke	19
26. Isječak koda poziva funkcije DetaljiBrigeBiljaka	20