Universitatea Tehnică „Gheorghe Asachi” din Iași

Facultatea de Automatică și Calculatoare

Domeniul: Calculatoare și Tehnologia Informației

Reteaua de socizlizare

„facepalm”

Proiect la disciplina

Baze de Date

Student: Petrica Petru

Anul: 3

Grupa: 1306B

Coordonator: Mironeanu Catalin

Capitolul 1. Introducere

Proiectul este realizat in scopuri didactice si are ca obiectiv imitarea unei retele de socializare reale in care utilizatorii isi pot adauga prieteni, crea postari, comentarii si grupuri pentru a comunica cu anumiti prieteni.

Utilizatorul are posibilitatea de a-si redacta o descriere, a-si adauga o poza de profil si de a crea postari. De asemenea, au posibilitatea de a cauta alti utilizatori, a le vizita paginile si a adauga comentarii la postarile acestora. Initial, utilizatorul trebuie sa-si inregistreze un cont, dupa care acesta se poate loga oricand. Parola este socata intr-o varianta criptata cu ajutorul algoritmului SHA256. Toate operatiile cu baza de date sunt validate pentru a asigura imunitatea la atacurile de SQL injection.

Utilizatorii is pot crea grupuri si invita prietenii in acestea. Acestia pot avea 3 tipuri de permisiuni in grup, READER – are doar dreptul de a vizualiza postari / comentarii, pot invita prietenii lor doar cu permisiuni de READER, WRITER – are permisiuni de a crea postari si de a scrie comentarii, poate invita prietenii cu drept de READ sau WRITE si ADMIN – are permisiuni de READ / WRITE plus permisiunea de a da afara din grup persoane cu privilegii mai mici.

Admin View

Cu exceptia utilizatorilor obisnuiti, exista utilizatorul admin, care poate accesa pagini invizibile celorlalti utilizatori si permite vizualizarea usoara a tuturor tabelelor bazei de date, inserarea datelor, modificarea datelor si stergerea datelor

Capitolul 2. Tehnologiile folosite pentru front-end și back-end

Baza de date folosită în această aplicație este postgreSQL. Pentru partea de front-end s-a folosit browser-ul impreuna cu biblioteca flask din python. Biblioteca flask ne permite crearea unor rute HTTP ce pot accepta cereri de tip GET si POST. De asemenea, biblioteca flask are incorporat motorul de generare HTML jinja care permite generarea mai usoara a structurilor HTML.

Server-ul HTTP dispune de mai multe rute:

- / - Ruta de baza, permite accesul la ruta /login si /signup
- /login – Aici utilizatorul isi poate introduce username-ul si parola pentru validare si autentificare
- /signup – Aici utilizatorul is poate crea profil
- /users/<username> - Pagina personala a utilizatorului cu username-ul <username>

Aceasta pagina permite vizualizarea profilului propriu si a altor utilizatori, in cazul ca utilizatorul se afla pe pagina personala, acesta are posibilitate de a-si edita datele si de a crea postari, vizualiza cererile de prietenie si invitatiile in grupuri. De asemenea utilizatorul isi poate vizualiza propriii prieteni si grupurile din care face parte. In cazul in care aceasta este pagina altui utilizator, se poate vedea statutul de prietenie si se poate trimite o cerere de prietenie in cazul in care utilizatorul respectiv nu este deja prieten si nu a fost deja trimisa o cerere.

- /groups/<group_name> - In cazul in care utilizatorul curent face parte din grupul cu numele <group_name>, acesta poate vizualiza aici toti membrii din grupul respectiv si toate postarile din acesta. De asemenea de aici acesta poate invita alti prieteni in acest grup.
- /admin – Pagina principala pentru utilizatorul admin, de aici se poate alege o baza de date pentru vizualizare/editare.
- /admin/<table_name> - Pagina specifica pentru una din tabelele din baza de date. Aici se permite viauzlizarea tuturor datelor dintr-o tabela, de asemenea permite inserarea, stergerea si modificarea intrarilor in aceasta.

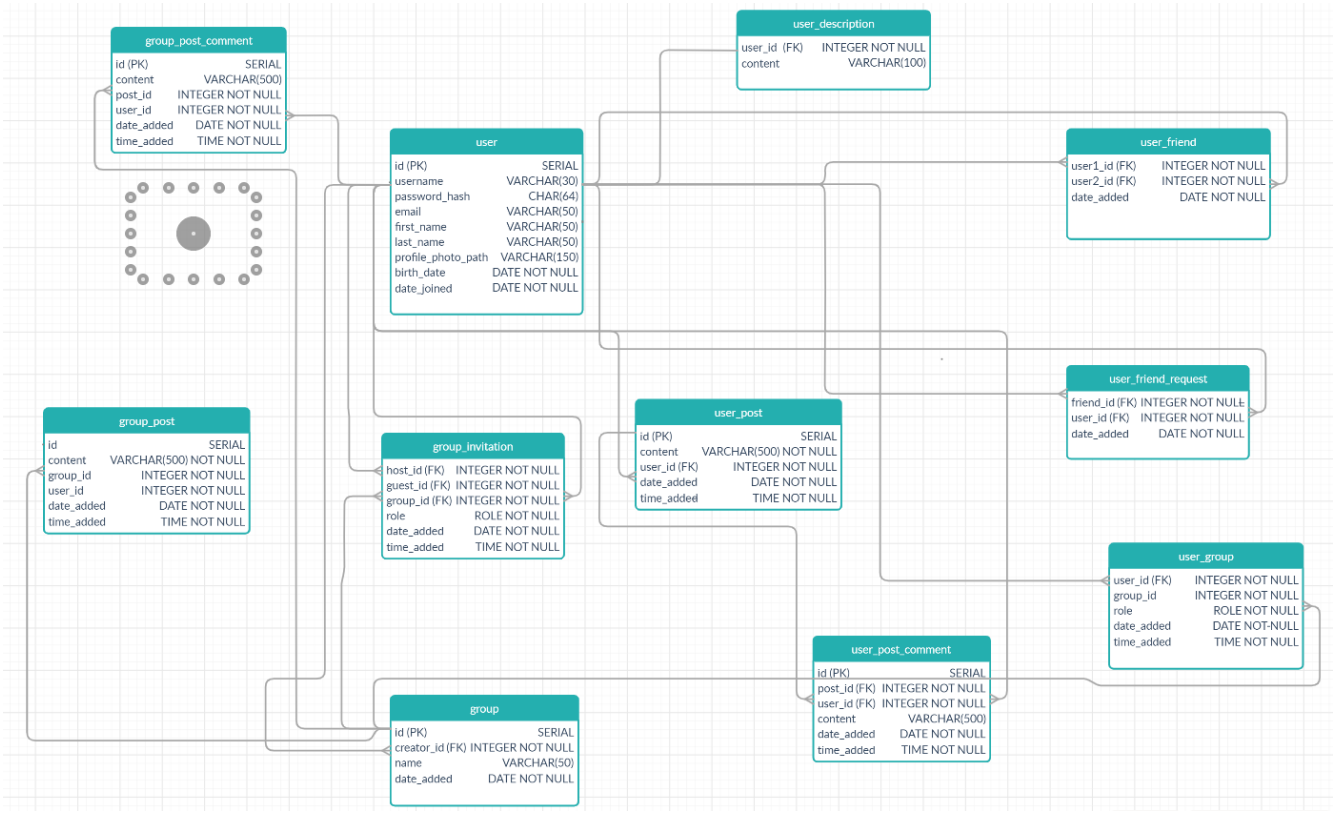
Restul rutelor sunt folosite doar pentru trimiterea datelor de pe paginile enumerate mai sus prin intermediul cererilor HTTP de tip POST.

Capitolul 3. Structura și inter-relaționarea tabelelor

Structura tabelelor a fost creată așa încât să satisfacă unul din multiplele modele real a bazelor de date pentru o retea de socizlizare.

- Utilizatorii au o descriere care este stocata intr-o tabela separata (One-to-one)
- Utilizator pot avea mai multe postari. (One-to-many)
- Utilizatorul poate fi membru ai mai multor grupuri, iar grupurile pot contine utilizatori multipli (Many-to-many
- O postare poate avea mai multe comentarii (One-to-many)
- Un prieten poate avea mai multi prieteni (One-to-many)
- Multe alte relatii de tipul one-to-many

Ținând cont de precizările de mai sus, Diagrama ER (Entity-Relation) construită după modelul Crow's Foot Notation este reprezentată mai jos.



Capitolul 4. Descrierea constrângerilor folosite

- **user**

Primary Key: id
Constrângere Unique, Not NULL: username
Constrângere Unique, Not NULL: email

- **user_description**

Foreign Key: user_id
Constrângere Check: quantity, total_amount (cantitatea și suma totală trebuie să fie pozitive

- **user_friend**

Foreign Keys: user1_id, user2_id
Primary Key: (user1_id, user2_id)
Constrângere Check: user1_id != user2_id

- **user_friend_request**

Primary Key: (friend_id, user_id)
Constrângere Unique: (shop_name, location_id)
Foreign Keys: friend_id, user_id
Check: friend_id != user_id

- **user_post**

Primary Key: id
Foreign Keys: user_id

- **user_post_comment**

Primary Key: id
Foreign Keys: user_id, post_id

- **group**

Primary Key: id
Constrângere Unique: name
Foreign Keys: creator_id

- **user_group**

Primary Key: (user_id, group_id)
Constrângere Unique: name
Foreign Keys: user_id, group_id

Toate constrangerile de tip UNIQUE sunt necesare pentru a evita introducerea acelorasi date in baza de date, evitarea conflictelor, in unele cazuri si constrangerile de tip PRIMARY KEY au acest rol.

Toate constrangerile de tip FOREIGN KEY sunt necesare pentru a stabili integritatea datelor.
Toate constrangerile de tip CHECK sunt lucruri care nu ar trebui sa se intample din motiv logic.

Normalizare

Toate tabelele sunt aduse cel puțin la a 3 formă normală.

- Un atribut conține valori atomice din domeniul său și nu grupuri de valori (I formă)
- Atributele non-cheie depind de toate cheile candidat (a II-a formă)
- Atributele non-cheie nu sunt tranzitiv dependente de cheile candidat (a III-a formă)

Forma normală Boyce-Codd este de asemenea asigurată, întrucât:

- Pentru o dependență X→Y, resultă că X – super cheie

Capitolul 5. Descrierea modalității de conectare la baza de date

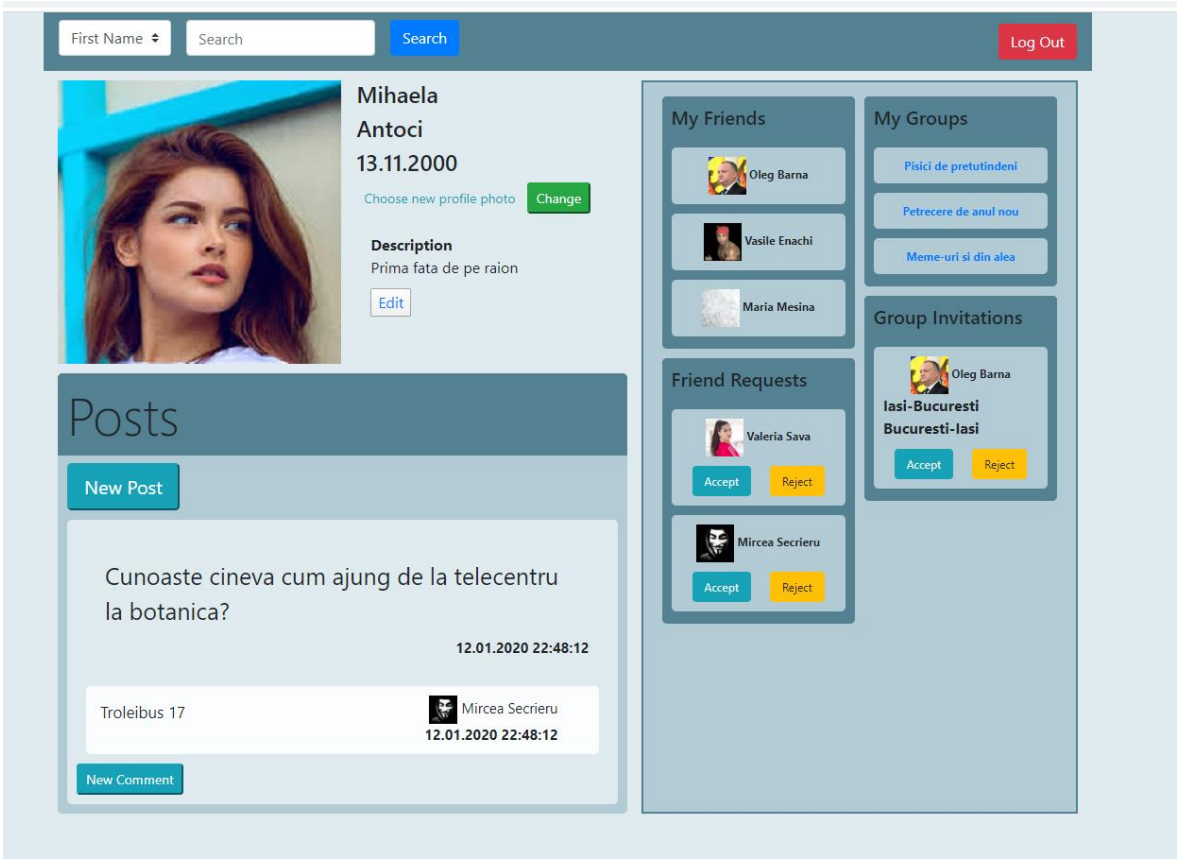
Conectarea la baza de date a fost făcută prin intermediul librăriei psy2copg.

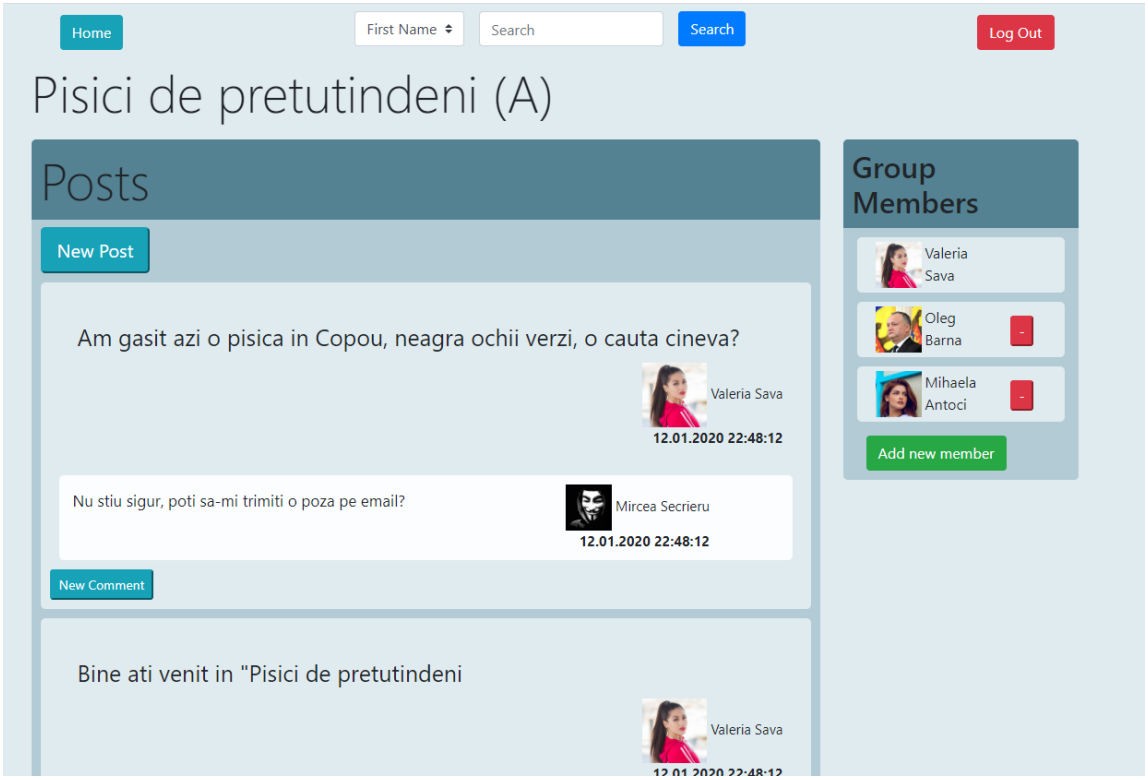
```
db_connection = psycopg2.connect(host='localhost', database='bdp', user='postgres', password='p@ssw0rd')

with open(args.script_path, 'r') as script:
    with db_connection.cursor() as cursor:
        cursor.execute(script.read())
    db_connection.commit()
```

Capitolul 6. Capturi de ecran

- **Interfața grafică**





- **Exemple cod SQL**

```sql
CREATE TABLE IF NOT EXISTS user_post_comment(
    id SERIAL,
    post_id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    content VARCHAR(500) NOT NULL,
    date_added DATE NOT NULL DEFAULT(CURRENT_DATE),
    time_added TIME NOT NULL DEFAULT(LOCALTIME),
    PRIMARY KEY(id),
    FOREIGN KEY(user_id) REFERENCES "user"(id),
    FOREIGN KEY(post_id) REFERENCES user_post(id)
);

CREATE TABLE IF NOT EXISTS "group"(
    id SERIAL,
    creator_id INTEGER NOT NULL,
    name VARCHAR(50) NOT NULL,
    date_added DATE NOT NULL DEFAULT(CURRENT_DATE),
    PRIMARY KEY(id),
    FOREIGN KEY(creator_id) REFERENCES "user"(id),
    UNIQUE(name)
);
```

```
    cursor.execute(
        'SELECT f.id, '
        '       f.first_name, '
        '       f.last_name, '
        '       f.profile_photo_path '
        'FROM "user" u, '
        '     "user" f, '
        '     user_friend uf '
        'WHERE ((u.id = uf.user1_id '
        '        AND f.id = uf.user2_id) '
        '       OR (u.id = uf.user2_id '
        '           AND f.id = uf.user1_id)) '
        '  AND u.id = %s '
        '  AND f.id NOT IN '
        '    (SELECT user_id '
        '     FROM user_group '
        '     WHERE group_id = %s) '
        '  AND f.id NOT IN '
        '    (SELECT guest_id '
        '     FROM group_invitation '
        '     WHERE group_id = %s '
        '       AND host_id = %s) ',
        (user_id, group_id, group_id, user_id))
    entries = cursor.fetchall()
```