



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DISCIPLINA
BAZE DE DATE
ANUL 2023

Proiect "Agenție imobiliară"

Nistor Dalia-Emilia
Rotariu Laura-Alexandra
30224

Cuprins

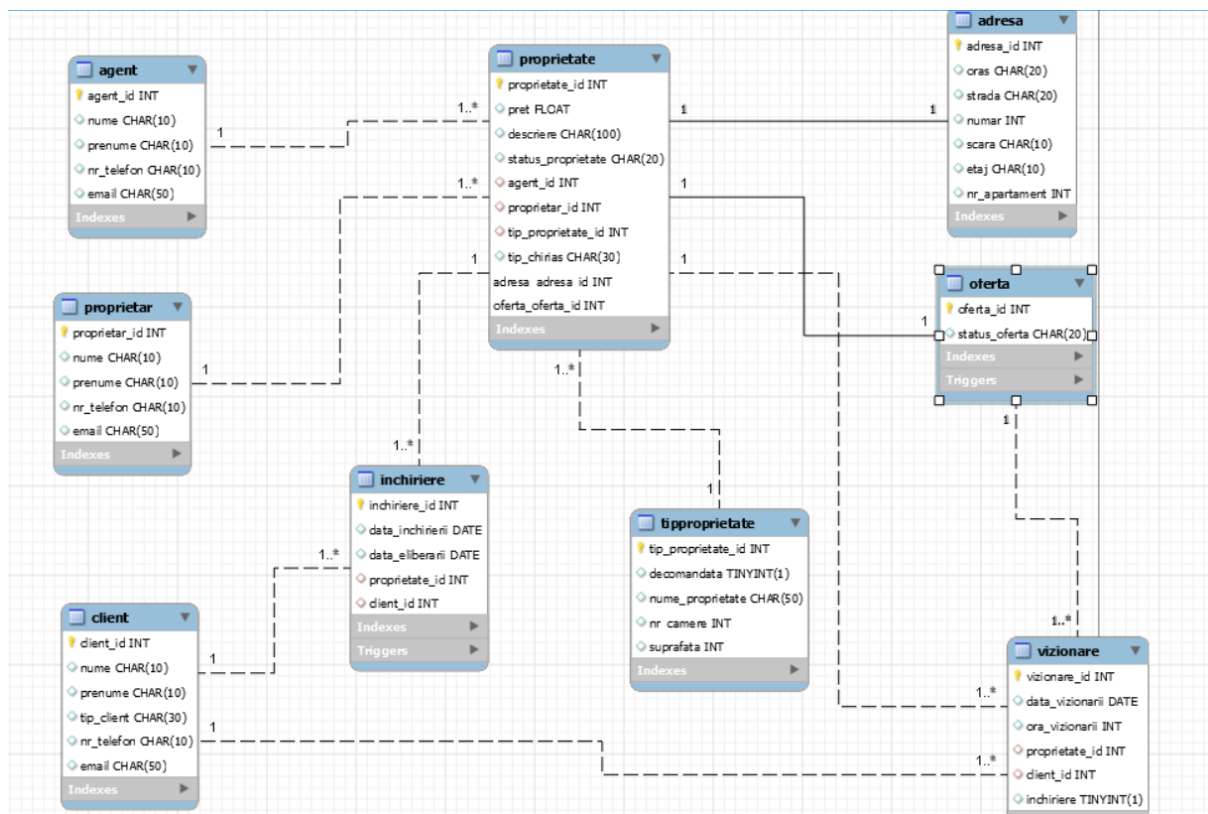
1. TEMA PROIECTULUI.....	4
2. DETALII DE PROIECTARE CONCEPTUALA A BAZEI DE DATE	5
3. SOLUTIA DE TRANSFORMARE IN RELATIONAL	6
3.1. Many to many	6
3.2. Alegerea cheii	6
4. DESCRIEREA BAZEI DE DATE RELATIONALE	7
4.1 proprietate.....	7
4.1.1. proprietate_id	7
4.1.2. pret.....	7
4.1.3. descriere	7
4.1.4. agent_id	7
4.1.6. tipproprietate_id.....	7
4.1.7. adresa_adresa_id	7
4.1.8 oferta_oferta_id	7
4.2. agent.....	8
4.3. proprietar	8
4.4.client.....	8
4.5 tipproprietate	8
4.6 adresa.....	8
4.7. oferta.....	8
4.8. inchiriere	8
4.9. vizionare	8
5. NIVELUL DE NORMALIZARE	9
6. INTEROGARI.....	10
7. OBIECTE DE TIP VEDERE, TRIGGERE, PROCEDURI STOCATE, SECURIZAREA ACCESULUI	13
7.1. Obiecte de tip vedere	13
7.2. Triggere	14
7.3. Proceduri.....	16
7.4. Securizarea accesului	22
8. DESCRIEREA APLICATIEI	22
8.1. Operatii posibile	22
8.2. Tehnologii folosite.....	23
8.2.1. Descriere scurta JavaSWING	23

8.2.2. Conectare	23
8.2.3. Vizualizare meniu.....	23
8.2.4. Clase din Java	25
9. MANUAL DE INSTALARE SI CONFIGURARI NECESARE.....	26
10. CONCLUZII SI DEZVOLTARI ULTERIOARE.....	26

1. TEMA PROIECTULUI

Proiectul consta in realizarea unei baze de date cu ajutorul careia sa se gestioneze o agentie imobiliara, cu elemente necesare: clienti, proprietari, agenti, proprietati, inchirieri, vizionari intr-o maniera organizata si automatizata.

2. DETALII DE PROIECTARE CONCEPTUALA A BAZEI DE DATE



3. SOLUTIA DE TRANSFORMARE IN RELATIONAL

3.1. Many to many

- Tabela inchiriere are rolul de a elimina nedeterminarea many to many intre tabela client si tabela proprietate, avand cheile straine proprietate_id si client_id si cheia primara inchiriere_id
- Tabela vizionare are rolul de a elimina nedeterminarea many to many intre tabela client si tabela proprietate, avand cheile straine proprietate_id si client_id si cheia primara vizionare_id

3.2. Alegerea cheii

- Fiecare din tabelele care nu sunt mentionate in sectiunea anterioara are o cheie primara care identifica celelalte tuple ale tabeli, iar numele acestor chei primare au forma [Nume tabela]_id.

4. DESCRIEREA BAZEI DE DATE RELATIONALE

In continuare este prezentata tabela cea mai importanta cu attributele ei si rolul fiecarui atribut:

4.1 proprietate

Tabela cea mai importanta este Proprietate, celelalte tabele furnizand date pentru inregistrările acestei tabele

4.1.1. proprietate_id

Atributul proprietate_id, care este cheia primara, determina celelalte attribute ale acestei tabele

4.1.2. pret

Atributul pret, reprezinta pretul cu care poate fi inchiriata proprietatea

4.1.3. descriere

Atributul descriere ofera o mica detaliere a proprietatii date spre inchiriere

4.1.4. agent_id

Atributul agent_id al proprietatii duce la tupla din tabela agent corespunzatoare agentului care se ocupa de proprietate

4.1.5. proprietar_id

Atributul proprietar_id al proprietatii duce la tupla din tabela proprietar corespunzatoare proprietarului proprietatii

4.1.6. tipproprietate_id

Atributul tipproprietate_id al proprietatii duce la tupla din tabela tipproprietate corespunzatoare tipului de proprietate

4.1.7. adresa_adresa_id

Atributul adresa_adresa_id al proprietatii duce la tupla din tabela adresa corespunzatoare adresei proprietatii respective

4.1.8 oferta_oferta_id

Atributul oferta_oferta_id al proprietatii duce la tupla din tabela oferta corespunzatoare ofertei

Mai apoi, sunt prezentate restul de 8 tabele impreuna cu attributele lor:

4.2. agent

- cu cheia primara agent_id si attributele nume, prenume, nr. tel, email care dau informatii despre agentul respectiv

4.3. proprietar

- cu cheia primara proprietar_id si attributele nume, prenume, nr. tel, email care dau informatii despre proprietarul respectiv

4.4. client

- cu cheia primara client_id si attributele nume, prenume, tip_client, nr. tel, email care dau informatii despre clientul respectiv

4.5 tipproprietate

- cu cheia primara tip_proprietate_id si attributele decomandata, nume_proprietate, nr_camere, suprafata care dau informatii despre tipul respectiv de proprietate

4.6 adresa

- cu cheia primara adresa_id si attributele oras, strada, nr, scara, etaj, nr_apartament care dau informatii despre adresa proprietatii respective

4.7. oferta

- cu cheia primara oferta_id si atributul status_oferta care da informatii despre statusul ofertei

4.8. inchiriere

- cu cheia primara inchiriere_id si attributele data_inchirierii, data_eliberarii, pripietate_id, client_id care ofera informatii despre o anumita inchiriere a unei proprietati

4.9. vizionare

- cu cheia primara vizionare_id si attributele data_vizionarii, ora_vizionarii, proprietate_id, client_id, inchiriere

5. NIVELUL DE NORMALIZARE

Baza de date se afla in nivelul de normalizare Boyce-Codd. Conform cercetatorului Chris Date, acest nivel de normalizare poate fi rezumat astfel: "(1)Fiecare atribut trebuie (2)sa reprezinte o informatie (3)despre cheie, (4)intreaga cheie, (5)si nimic in afara cheii". In cazul bazei noastre de date:

1. Fiecare atribut este atomic si nu depinde de alte attribute
2. Fiecare atribut reprezinta o singura informatie
3. Exista attribute chei, reprezentate de id-ul unic al tabelelor
4. Cheia primara este suficienta ca sa determine celelalte attribute ale tabelelor
5. Cheia primara este singurul atribut care determina celelalte attribute, nu exista un alt atribut care poate determina in mod unic o tupla.

6. INTEROGARI

In continuare se prezinta o serie de interogari ce demonstreaza cum se foloseste baza de date:

#1.Clientii care au vizionat proprietati de tipul 1?

```
SELECT COUNT(*) FROM client
JOIN vizionare on vizionare.client_id=client.client_id
JOIN proprietate on proprietate.proprietate_id=vizionare.proprietate_id
WHERE proprietate.tip_proprietate_id=1;
```

#2.Ce proprietari au proprietati in orasul Cluj si Bucuresti?

```
SELECT nume,prenume FROM proprietar
JOIN proprietate on proprietate.proprietar_id=proprietar.proprietar_id
JOIN adresa on adresa.adresa_id=proprietate.adresa_adresa_id
WHERE adresa.oras in ("Cluj","Bucuresti")
GROUP BY nume, prenume;
```

#3.Ce clienti au inchiriat proprietatile pe care le-au vizionat?

```
SELECT nume,prenume FROM client
JOIN vizionare on vizionare.client_id=client.client_id
WHERE vizionare.inchiriere = 'DA'
GROUP BY nume, prenume;
```

#4. Afisati clientii care au inchiriat proprietati in anul 2022.

```
SELECT nume,prenume FROM client
JOIN inchiriere on inchiriere.client_id=client.client_id
WHERE year(inchiriere.data_inchirierii)="2022"
ORDER BY client.client_id ASC;
```

#5. Ce agent se ocupa de proprietatea de pe strada Dorobantilor?

```
SELECT nume,prenume FROM agent
JOIN proprietate on proprietate.agent_id=agent.agent_id
JOIN adresa on adresa.adresa_id=proprietate.adresa_adresa_id
WHERE adresa.strada="Dorobantilor";
```

#6. Afisati proprietarii care au proprietati de tipul 1.

```
SELECT nume,prenume FROM proprietar
JOIN proprietate on proprietate.proprietar_id=proprietar.proprietar_id
WHERE proprietate.tip_proprietate_id=1;
```

#7. Ce clienti au vizionat in data de 20.04.2022 proprietati?

```
SELECT nume,prenume FROM client
JOIN vizionare on vizionare.client_id=client.client_id
WHERE vizionare.data_vizionarii='2022-04-20'
GROUP BY nume, prenume
ORDER BY client.client_id ASC;
```

#8. Clientii care au vizionat proprietatea 2?

```
SELECT nume,prenume FROM client
JOIN vizionare on vizionare.client_id=client.client_id
WHERE vizionare.proprietate_id=2;
```

#9. Proprietarul proprietatii vizionate de clientul 4 in data de 10.08.2022?

```
SELECT * FROM proprietar
JOIN proprietate on proprietar.proprietar_id=proprietate.proprietar_id
JOIN vizionare on vizionare.proprietate_id=proprietate.proprietate_id
JOIN client on client.client_id=vizionare.client_id
WHERE client.client_id=4 AND vizionare.data_vizionarii='2022-08-10';
```

#10. De care proprietati se ocupa agentul cu numele Rusu Ionut?

```
SELECT * FROM proprietate
LEFT JOIN agent on agent.agent_id=proprietate.agent_id
WHERE agent.nume='Rusu' AND agent.prenume = 'Ionut';
```

#11. Ce clienti au vizionat mai multe proprietati?

```
SELECT nume,prenume FROM client
JOIN vizionare on vizionare.client_id=client.client_id
GROUP BY vizionare.client_id HAVING COUNT(*)>1;
```

Interogari in algebra relationala:

1. Alegere categorii distincte de clienti:

Π tip_client(client)

2. Determinare adresa_id corespunzatoare unui ID de proprietate (1):

Π adresa_id(σ proprietate_id = 1(proprietate \bowtie adresa))

3. Proprietati a caror descriere contine 'nou' sau 'confortabil':

Π proprietate_id;pret;descriere;status_proprietate(σ descriereLIKE%nou%(proprietate))
 \cup

Π

proprietate_id;pret;descriere;status_proprietate(σ descriereLIKE%confortabil%(proprietate))

4. Proprietati a caror descriere contine 'nou' dar nu contine 'confortabil':

Π proprietate_id;pret;descriere;status_proprietate(σ descriereLIKE%nou%(proprietate))

—

Π

proprietate_id;pret;descriere;status_proprietate(σ descriereLIKE%confortabil%(proprietate))

7. OBIECTE DE TIP VEDERE, TRIGGERE, PROCEDURI STOCATE, SECURIZAREA ACCESULUI

7.1. Obiecte de tip vedere

View Clientii care au vizualizat proprietati de tipul 2

```
CREATE VIEW View_VizionariClientiProprietateaTip2 AS
SELECT nume,prenume FROM client
JOIN vizionare on vizionare.client_id=client.client_id
JOIN proprietate on proprietate.proprietate_id=vizionare.proprietate_id
WHERE proprietate.tip_proprietate_id=2;
```

View Agentii care se ocupa de proprietatea 2

```
CREATE VIEW View_AgentiiProprietatii2 AS
SELECT nume,prenume FROM agent
JOIN proprietate on proprietate.agent_id=agent.agent_id
WHERE proprietate.proprietate_id=2;
```

View Proprietate cu id inlocuite de valori

```
CREATE VIEW View_ProprietateFaraId AS
SELECT proprietate.proprietate_id,
       proprietate.pret,
       proprietate.descriere,
       proprietate.status_proprietate,
       concat_ws(' ',agent.nume, agent.prenume) AS Agent,
       concat_ws(' ',proprietar.nume, proprietar.prenume) AS Proprietar,
       tipproprietate.nume_proprietate AS TipProprietate,
       concat_ws(' ',adresa.oras,adresa.strada,adresa.numar,adresa.scara,adresa.etaj,adresa.nr_apartament) AS
Adresa,
       oferta.status_oferta AS Oferta
FROM proprietate
JOIN agent on agent.agent_id=proprietate.agent_id
JOIN proprietar on proprietar.proprietar_id=proprietate.proprietar_id
JOIN tipproprietate on tipproprietate.tip_proprietate_id=proprietate.tip_proprietate_id
JOIN adresa on adresa.adresa_id=proprietate.adresa_adresa_id
JOIN oferta on oferta.oferta_id=proprietate.oferta_oferta_id;
```

View Inchiriere cu proprietati care au fost inchiriate in data de azi

```
CREATE VIEW View_InchirieriDeAzi AS
SELECT inchiriere.data_inchirierii,
       inchiriere.data_eliberarii,
       inchiriere.proprietate_id,
       concat_ws(' ',client.numere, client.prenume) AS client
FROM inchiriere
JOIN client on client.client_id=inchiriere.client_id
WHERE inchiriere.data_inchirierii = curdate();
```

View Proprietati care nu sunt inchiriate

```
CREATE VIEW View_ProprietatiNeinchiriate AS
SELECT * FROM proprietate
WHERE proprietate.status_proprietate='DISPONIBILA';
```

7.2. Triggere

Trigger pentru modificare proprietate si oferta in momentul modificarii tabeli vizionare, deoarece tabela oferta nu are legatura directa cu tabela de vizionare, se va crea un alt trigger care modifica statusul ofertei in momentul in care se seteaza 'INDISPONIBILA' in proprietate

delimiter //

```
CREATE TRIGGER tModificareProprietate AFTER UPDATE ON vizionare
FOR EACH ROW BEGIN
    declare variabila_vizionare varchar(20);
    set variabila_vizionare = new.inchiriere;
    IF variabila_vizionare = 'DA' THEN
        UPDATE proprietate
        set status_proprietate = 'INDISPONIBILA'
        WHERE proprietate.proprietate_id = old.proprietate_id;
    END IF;
END;//
delimiter ;
```

#Trigger pentru modificarea ofertei

delimiter //

```
CREATE TRIGGER tModificareOferta AFTER UPDATE ON proprietate
```

```

FOR EACH ROW BEGIN
    declare variabila_proprietate varchar(20);
    set variabila_proprietate = new.status_proprietate;
    IF variabila_proprietate = 'INDISPONIBILA' THEN
        UPDATE oferta
        set status_oferta = 'INACTIVA'
        WHERE oferta.oferta_id = old.oferta_oferta_id;
    ELSEIF variabila_proprietate = 'DISPONIBILA' THEN
        UPDATE oferta
        set status_oferta = 'ACTIVA'
        WHERE oferta.oferta_id = old.oferta_oferta_id;
    END IF;
END;
delimiter ;

```

Trigger pentru update proprietate in momentul adaugarii unei inchirieri

```

delimiter //

CREATE TRIGGER tModificareProprietateDupaInchiriere BEFORE INSERT ON inchiriere
FOR EACH ROW BEGIN
    declare variabila_data_eliberarii varchar(20);
    set variabila_data_eliberarii = new.data_eliberarii;
    IF variabila_data_eliberarii is NULL THEN
        UPDATE proprietate
        set status_proprietate = 'INDISPONIBILA'
        WHERE proprietate.proprietate_id = new.proprietate_id;
    END IF;
END;

delimiter ;

```

Trigger pentru update proprietate in momentul modificarii data_eliberarii din inchiriere

```

delimiter //

CREATE TRIGGER tModificareProprietateDupaDataEliberarii BEFORE UPDATE ON
inchiriere
FOR EACH ROW BEGIN
    declare variabila_data_eliberarii varchar(20);
    set variabila_data_eliberarii = new.data_eliberarii;
    IF variabila_data_eliberarii is NOT NULL THEN

```

```

        UPDATE proprietate
        set status_proprietate = 'DISPONIBILA'
        WHERE proprietate.proprietate_id = new.proprietate_id;
        ELSEIF variabila_data_eliberarii is NULL THEN
            UPDATE proprietate
            set status_proprietate = 'INDISPONIBILA'
            WHERE proprietate.proprietate_id = new.proprietate_id;
            END IF;
    END;

delimiter ;

```

7.3. Proceduri

Procedura modificare data eliberare

```

DELIMITER $$

USE `proiect_agentie_imobiliara`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `modifica_data_eliberare`(id INT,
eliberare DATE)
BEGIN
    START TRANSACTION;
    UPDATE inchirieri
    SET data_eliberarii = eliberare WHERE inchiriere_id = id;
    END$$

DELIMITER ;

```

Procedura modificare data inchiriere

```

DELIMITER $$

USE `proiect_agentie_imobiliara`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `modifica_data_inchiriere`(id INT,
inchiriere DATE)
BEGIN
    START TRANSACTION;
    UPDATE inchirieri
    SET data_inchirierii = inchiriere WHERE inchiriere_id = id;
    END$$
DELIMITER ;

```


Procedura modificare status oferta

DELIMITER \$\$

USE `proiect_agentie_imobiliara`\$\$

CREATE DEFINER=`root`@`localhost` PROCEDURE `modifica_status_oferta`(id INT, status_oferta CHAR(20))

BEGIN

START TRANSACTION;

UPDATE oferta

SET oferta.status_oferta = status_oferta WHERE oferta_id = id;

END\$\$

DELIMITER ;

Procedura proprietarii cu proprietati de tipul x

DELIMITER \$\$

CREATE PROCEDURE proprietari_cu_proprietati_tip_x(IN parametru_tip int)

BEGIN

SELECT nume,prenume FROM proprietar

JOIN proprietate on proprietate.proprietar_id=proprietar.proprietar_id

WHERE proprietate.tip_proprietate_id=parametru_tip;

END;\$\$

DELIMITER;

Procedura clientii care au vizionat proprietati in data x

DELIMITER \$\$

CREATE PROCEDURE clienti_vizionari_in_data_x(IN parametru_date date)

BEGIN

SELECT nume,prenume FROM client

JOIN vizionare on vizionare.client_id=client.client_id

WHERE vizionare.data_vizionarii=parametru_date;

END;\$\$

DELIMITER;

Procedura clientii care un vizionat proprietatea x

DELIMITER \$\$

```
CREATE PROCEDURE clienti_vizionari_proprietate_x(IN parametru_id_proprietate int)
BEGIN
    SELECT nume,prenume FROM client
    JOIN vizionare on vizionare.client_id=client.client_id
    WHERE vizionare.proprietate_id=parametru_id_proprietate;
END;$$
```

DELIMITER;

Procedura proprietarul proprietatii vizionate de clientul x in data y

DELIMITER \$\$

```
CREATE PROCEDURE proprietarii_proprietatii_vizionate(IN parametru_client_id int, IN
parametru_date date)
BEGIN
    SELECT * FROM proprietar
    JOIN proprietate on proprietar.proprietar_id=proprietate.proprietar_id
    JOIN vizionare on vizionare.proprietate_id=proprietate.proprietate_id
    JOIN client on client.client_id=vizionare.client_id
    WHERE client.client_id=parametru_client_id AND
vizionare.data_vizionarii=parametru_date;
END;$$
```

DELIMITER;

Procedura de care proprietati de ocupa agentul cu numele x

DELIMITER \$\$

```
CREATE PROCEDURE proprietati_agent_x(IN parametru_agent varchar(50))
BEGIN
    SELECT * FROM proprietate
    LEFT JOIN agent on agent.agent_id=proprietate.agent_id
```

```
WHERE agent.nume=parametru_agent;
END;$$
```

```
DELIMITER;
```

```
# Procedura ce clienti au vizionat mai multe proprietati
```

```
DELIMITER $$
```

```
CREATE PROCEDURE clienti_vizionari_multiple()
BEGIN
    SELECT nume,prenume FROM client
    JOIN vizionare on vizionare.client_id=client.client_id
    GROUP BY vizionare.client_id HAVING COUNT(*)>1;
END;$$
```

```
#DELIMITER;
```

```
# procedura adauga_proprietate
```

```
DELIMITER $$
```

```
USE `proiect_agentie_imobiliara`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `adauga_proprietate`(pret FLOAT,
descriere CHAR(100), status_proprietate CHAR(20),
agent_id INT, proprietar_id INT, tip_proprietate_id INT, adresa_adresa_id INT,
oferta_oferta_id INT)
BEGIN
    START TRANSACTION;
    INSERT INTO proprietate values(proprietate_id, pret, descriere, status_proprietate, agent_id,
proprietar_id, tip_proprietate_id,
adresa_adresa_id, oferta_oferta_id);
END$$
```

```
DELIMITER ;
```

procedura adauga_adresa

DELIMITER \$\$

```
USE `proiect_agentie_imobiliara`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `adauga_adresa`(oras CHAR(20),
strada CHAR(20), numar INT, scara CHAR(11), etaj CHAR(10), nr_apartament INT)
BEGIN
    START TRANSACTION;
    INSERT INTO adresa values(adresa_id, oras, strada, numar, scara, etaj, nr_apartament);
END$$
```

DELIMITER ;

procedura adauga_oferta

DELIMITER \$\$

```
USE `proiect_agentie_imobiliara`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `adauga_oferta`(status_oferta
CHAR(20))
BEGIN
    START TRANSACTION;
    INSERT INTO oferta values(oferta_id, status_oferta);
END$$
```

DELIMITER ;

procedura adauga_proprietar

DELIMITER \$\$

```
USE `proiect_agentie_imobiliara`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `adauga_proprietar`(proprietar_id
INT, nume CHAR(10), prenume CHAR(10), tip_client CHAR(30), nr_telefon CHAR(11),
email CHAR(50))
BEGIN
    START TRANSACTION;
```

```
INSERT INTO proprietar values(proprietar_id, nume, prenume, tip_client, nr_telefon, email);  
END$$
```

```
DELIMITER ;
```

```
# procedura adauga_agent
```

```
DELIMITER $$
```

```
USE `proiect_agentie_imobiliara`$$  
CREATE DEFINER=`root`@`localhost` PROCEDURE `adauga_agent`(agent_id INT, nume  
CHAR(10), prenume CHAR(10), tip_client CHAR(30), nr_telefon CHAR(11), email  
CHAR(50))  
BEGIN  
    START TRANSACTION;  
INSERT INTO agent values(agent_id, nume, prenume, tip_client, nr_telefon, email);  
END$$
```

```
DELIMITER ;
```

```
# procedura adauga_vizionare
```

```
DELIMITER $$
```

```
USE `proiect_agentie_imobiliara`$$  
CREATE DEFINER=`root`@`localhost` PROCEDURE `adauga_vizionare`(vizionare_id  
INT, data_vizionarii DATE, ora_vizionarii INT,  
    proprietate_id INT, client_id INT, inchiriere CHAR(2))  
BEGIN  
    START TRANSACTION;  
INSERT INTO vizionare values(vizionare_id, data_vizionarii, ora_vizionarii, proprietate_id,  
client_id, inchiriere);  
END$$
```

```
DELIMITER ;
```

```
# procedura adauga_inchiriere
```

```
DELIMITER $$
```

```

USE `proiect_agentie_imobiliara`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `adauga_inchiriere`(inchiriere_id
INT, data_inchirierii DATE, data_eliberarii DATE,
    proprietate_id INT, client_id INT)
BEGIN
    START TRANSACTION;
INSERT INTO inchiriere values(inchiriere_id, data_inchirierii, data_eliberarii, proprietate_id,
client_id);
END$$

DELIMITER ;

```

7.4. Securizarea accesului

```

CREATE USER user_db@localhost
IDENTIFIED BY 'pass';

GRANT SELECT
ON proiect_agentie_imobiliara.*
TO user_db@localhost;

SHOW GRANTS FOR user_db@localhost;

```

8. DESCRIEREA APLICATIEI

8.1. Operatii posibile

In scopul utilizarii mai usoare a bazei de date am dezvoltat o aplicatie demonstrativa prin care se pot efectua urmatoarele operatii:

- Conectare la o baza de date
- Vizualizarea clientilor
- Cautarea clientilor
- Inserarea unui client
- Actualizarea unui client
- Filtrarea clientilor (in ordine alfabetica, in ordine inversa alfabetica, dupa tipul de client (persoana fizica sau firma))
- Vizualizarea agentilor
- Vizualizarea proprietarilor
- Vizualizarea ofertelor
- Adaugarea unei oferte
- Vizualizarea adreselor
- Adaugarea unei adrese
- Vizualizarea tipurilor de proprietati

- Vizualizarea proprietatilor
- Adaugarea unei proprietati
- Vizualizarea vizionarilor
- Vizualizarea inchirierilor
- Vizualizare a raspunsurilor la o serie de intrebari
- Logarea ca manager al bazei de date pentru a prelucra informatii despre agentia imobiliara

8.2. Tehnologii folosite

Pentru realizarea aplicatiei am folosit platforma JavaSWING in IDE-ul IntelliJIDEA.

8.2.1. Descriere scurta JavaSWING

Swing este un subset JFC (Java Foundation Classes) și constă dintr-o serie de componente vizuale care extind (îmbunătățesc) componentele AWT, și furnizează noi facilități precum

tabele și arbori. Structura de clase din Swing este asemănătoare cu cea din AWT, în sensul că toate componentele interfeței grafice sunt derivate dintr-un singur părinte numit JComponent (care este derivat din clasa AWT Container).

Pachetul de clase Swing reprezintă soluția furnizată de Sun pentru crearea unor interfețe utilizator grafice complet portabile pe orice platformă.

În Swing, toate numele claselor încep cu litera J, și atunci când este posibil, numele este același cu cel al clasei AWT pe care o înlocuiește.

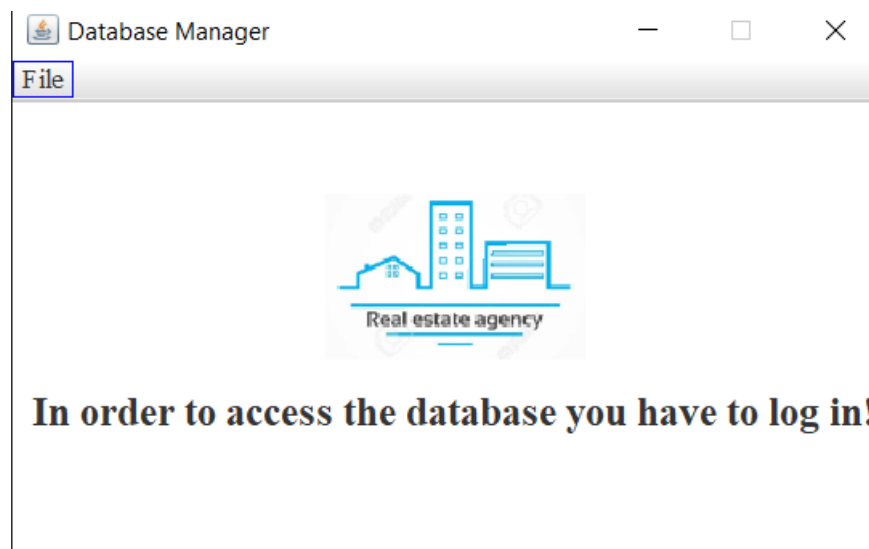
La fel ca la AWT, punctul de plecare pentru un program bazat pe Swing, este clasa JFrame sau clasa JApplet.

8.2.2. Conectare

Am creat o aplicatie JavaSWING cu conectivitate la baza de date mySQL prin API-ul JDBC. Pentru a conecta programul nostru Java cu baza de date mySQL am inclus driver-ul mySQL JDBC care este un fisier jar pe care il adaugam la calea clasei proiectului

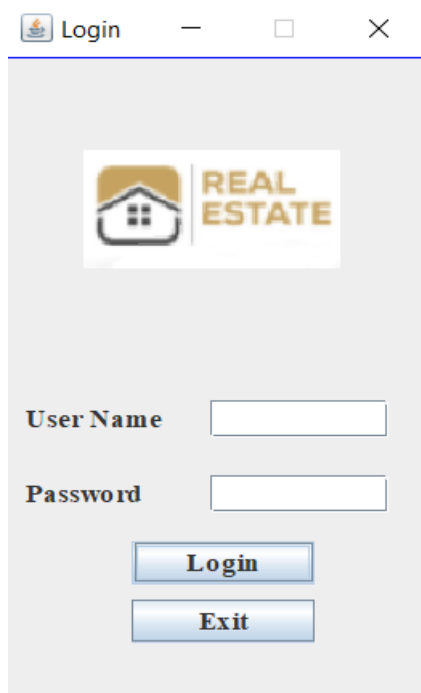
8.2.3. Vizualizare meniu

La pornirea fisierului executabil se deschide prima fereastra:



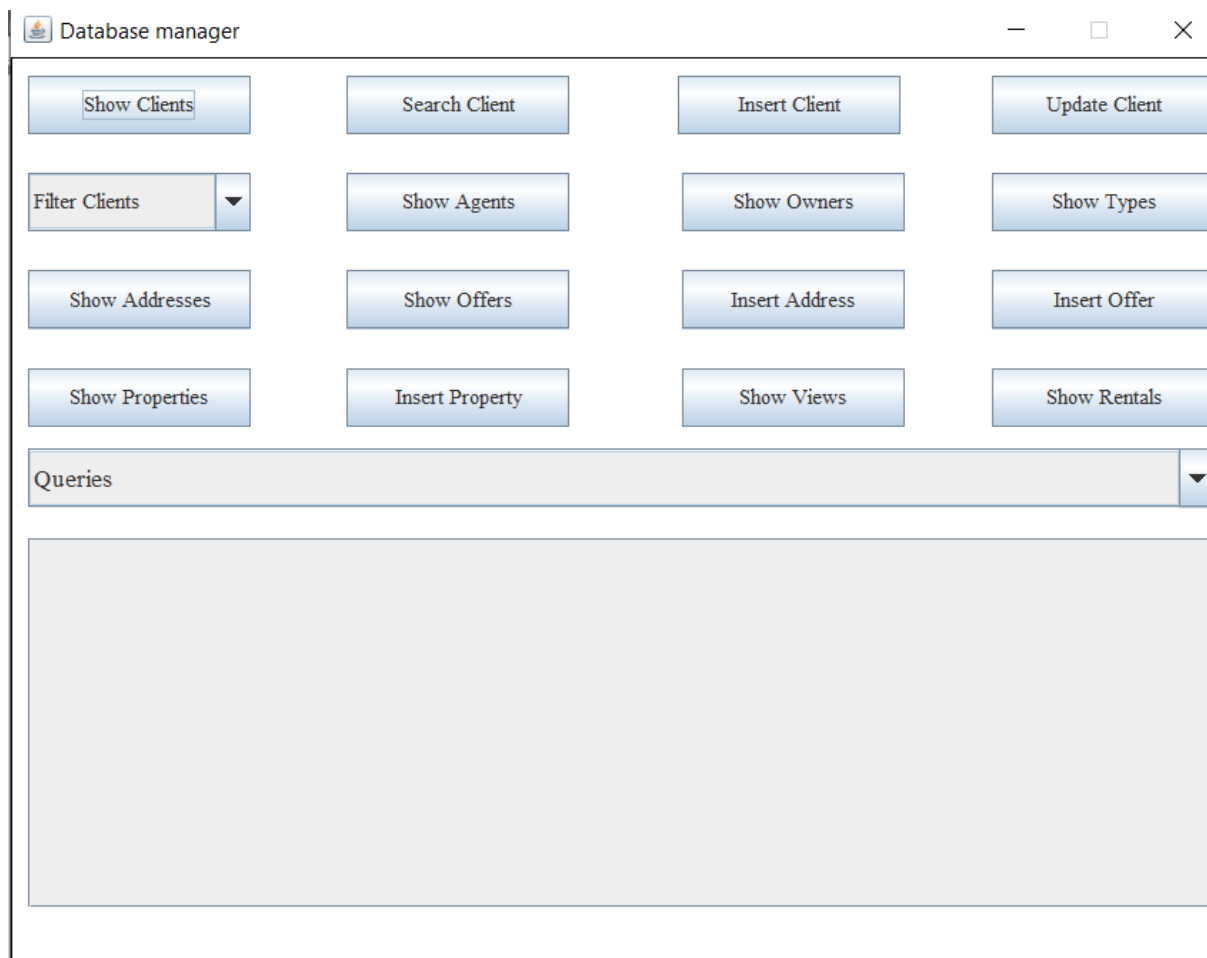
In order to access the database you have to log in!

Dupa apasarea butonului File, daca se alege optiunea Log in apare fereastra de logare:



The image shows a login window titled "Login". It features a logo with a house icon and the text "REAL ESTATE". Below the logo, there are two input fields: "User Name" and "Password". At the bottom, there are two buttons: "Login" and "Exit".

Daca datele introduse mai sus sunt corecte se deschide si meniul principal (User Name: admin, Password: admin):



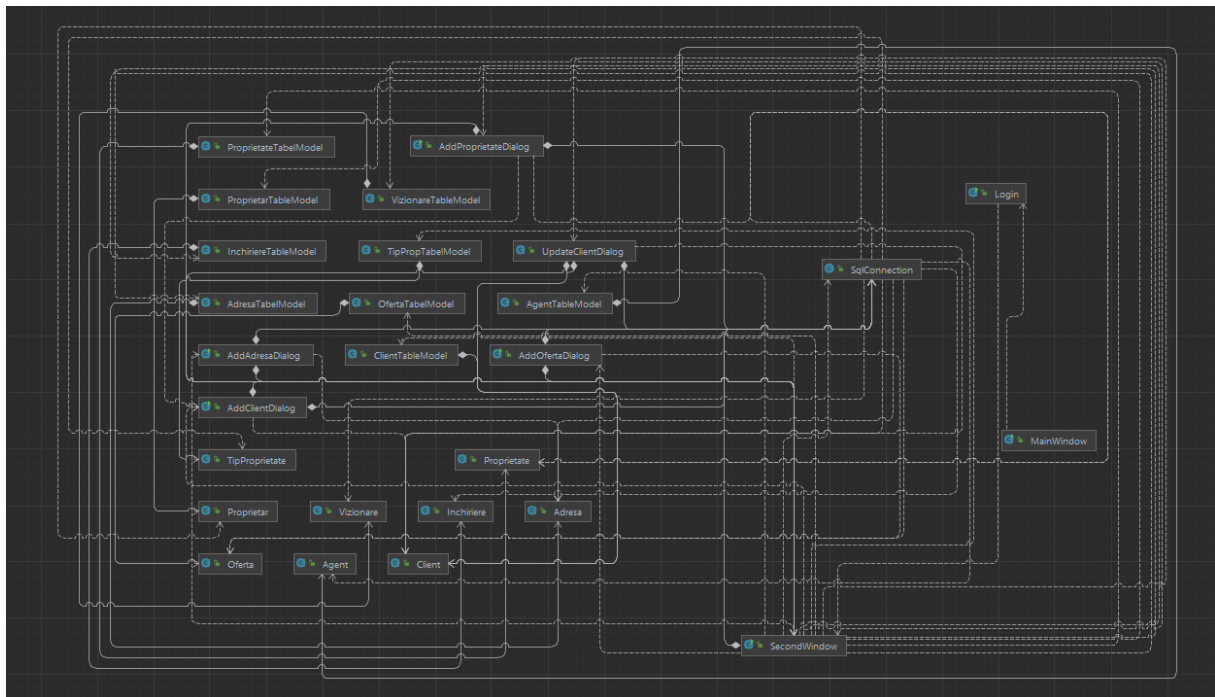
The image shows the main menu of the "Database manager" application. It has a title bar with the text "Database manager". The menu consists of several buttons arranged in a grid:

- Row 1: Show Clients, Search Client, Insert Client, Update Client
- Row 2: Filter Clients (with a dropdown arrow), Show Agents, Show Owners, Show Types
- Row 3: Show Addresses, Show Offers, Insert Address, Insert Offer
- Row 4: Show Properties, Insert Property, Show Views, Show Rentals

Below the grid, there is a "Queries" section with a dropdown arrow. The bottom part of the window is a large, empty rectangular area.

8.2.4. Clase din Java

Am implementat clase pentru fiecare element cheie din agentie spre exemplu clasele Agent, Proprietar, etc. Am folosit modelul generat pentru un tabel TableModel, implementand astfel tabele pentru toate clasele precum ProprietarTableModel. In clasa SQL Connection am realizat conectarea cu baza de date, am convertit datele din tabelele mySQL pentru Java. In clasele MainWindow, LogIn si SecondWindow am implementat toata dinamica aplicatiei, adica toate actiunile ce se pot desfasura la momentul rularii.



9. MANUAL DE INSTALARE SI CONFIGURARI NECESARE

Baza de date a fost creata in MySQLWorkbench iar codul este in fisierul SQL text file numit PROIECT_BD_AGENTIE_IMOBILIARA . Pentru a rula baza de date se acceseaza codul din MySQLWorkbench. Pentru realizarea acestui proiect am folosit :

- IntelliJIDEA+JavaSWING
- MySQL Workbench

10. CONCLUZII SI DEZVOLTARI ULTERIOARE

In concluzie, aplicatia noastra este una demonstrativa, dar metodele folosite aici se pot utiliza pentru a implementa o aplicatie reala de gestiune a unei baze de date a unei agentii imobiliare. S-ar putea lucra la imbunatatirea interfetei grupandu-se butoanele si adaugandu-se mai multe optiuni. De asemenea ar fi mult mai usor daca nu ar trebui inserate adresa si oferta inainte de inserarea unei proprietati.