MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

PROIECT

la disciplina
Introducere in Baze de Date

Sistem de gestiune a unui lanț de adăposturi de animale

Rujac Roxana, grupa 30224

Sabău Emanuela Cristina, grupa 30224

An academic: 2023 – 2024

PROIECT de SEMESTRU
Catedra de Calculatoare
Disciplina: Introducere in Baze de Date
Coordonator: s.l. ing. Cosmina IVAN
Data: 15.01.2024

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

Cuprins

Bibliografie

[1] https://www.jetbrains.com/help/idea/discover-intellij-idea.html

[2] https://docs.oracle.com/javase/tutorial/uiswing/

[3] https://ftp.utcluj.ro/pub/users/civan/IBD/

[4] http://www.elth.pub.ro/~preda/teaching/BDE/BDE_5.pdf

[5] https://www.youtube.com/watch?v=6zm8c6QFmjo&ab_channel=Intellipaat

[6] Michael Widenius, David Axmark, Kaj Arno, MySQL Reference Manual, June 2002, https://books.google.ro/books

[7] Leonte Carmen, Introducere (sau mic ghid practic) în MySQL, Universitatea „Alexandru Ioan Cuza”, Iași, https://profs.info.uaic.ro/~busaco/teach/courses/net/docs/mysql-ro.pdf

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

1. Introducere

Proiectul a presupus crearea unei aplicaţii care vine în ajutorul clientiilor, donatoriilor si proprietarului unui adăpost de animale. Scopul aplicatiei este de a facilita accessul la baza de data a adăpostului printr-o interfata grafica intuitiva. Aplicatia poate fi folosita de trei tipuri de persoane: client, donatori si proprietarul adăpostului.

Cu ajutorul acestei aplicații orice client va putea să vadă animalele pregătite să fie date la adopție, și de asemenea și istoricul lor veterinar. În cazul donatoriilor ei vor putea oferi o sumă de bani care va fi redirecționată adăpostului de animale ales. Pentru proprietar, acesta va putea urmări cu ușurință fondurile care intră din donațiile și adopțiile realizate. Se ține o evidență pentru aceasta pentru a avea în cunoștiință bugetul pe care îl are ca să poată proviziona adăpostul de animale. De asemenea, proprietarul mai ține evidența si animalelor din adăpost, cât și locurile disponibile pentru animale.

2. Analiza cerintelor utilizatorilor (Specificatiile de proiect)

2.1. Ipoteze specifice domeniului ales pentru proiect (cerinte, constrangeri)

CERINȚE
Cerinte de Adopție: Sistemul trebuie să permită înregistrarea informațiilor despre adopțiile de animale, inclusiv detalii despre noii proprietari și starea actuală a animalului.
Gestionarea Proviziilor: Există nevoie de un mod eficient de gestionare a proviziilor necesare în adăpost, cum ar fi hrana, jucăriile și alte necesități.
Servicii Veterinare: Cerințe privind evidența istoricului veterinar al fiecărui animal și programarea și gestionarea vizitelor la cabinetul veterinar.

CONSTRÂNGERI
Resurse Financiare Limitate: Adăposturile pentru animale pot avea resurse financiare limitate, ceea ce impune o gestionare eficientă a bugetului, a donațiilor și a cheltuielilor.
Capacitatea Adăpostului: Există o capacitate fizică limitată a adăpostului pentru a găzdui animale, iar acest lucru ar putea afecta numărul de animale acceptate sau adăpostite într-un anumit moment.
Legislație și Reglementări: Respectarea legilor și reglementărilor privind bunăstarea animalelor și funcționarea adăposturilor este esențială și ar putea impune constrângeri specifice.

2.2.Organizare structurată(tabelar) a cerințelor utilizator

Baza de date conține în tabele informații despre adăposturi, proprietarul unui adăpost, animalele care se află acolo cu un istoric veterinar, donațiile care se fac împreună cu informații despre donatori, tranzcțiile care se fac cu evidența care se actualizează constant a veniturilor și cheltuielior.

Baza de date modelează atât toate aceste entități, cât și relațiile dintre entități pentru a putea susține activitatea unui adăpost de animale. Modelul bazei de date este creat în așa fel încât să ia în considerare orice situație ce poate apărea în acest caz.

2.3.Determinarea si caracterizarea de profiluri de utilizatori (admin, user intern, user extern…diversi alti “actori”)

Aplicația a fost realizată în așa fel încât să fie ușor de utilizat pentru 2 opțiuni: prorietarul unui adăpost de animale și clientul care poate adopta un animal sau poate fi donatorul care are posibilitatea de a dona o sumă de bani unui adăpost de animale. Pentru toți aceștia există la început opțiunea de a se loga cu contul lor deja existent, sau de a-și crea un cont nou.

Proprietarul poate efectua urmatoarele operații:

- Poate să vizioneze informații despre animale
- Are evidența la tranzacții și la fonduri pe care poate să o vizualizeze
- Poate urmări numărul de locuri disponibile care au mai rămas pentru animale în adăpostul său.

Clientul poate efectua următoarele aplicații:

- Să aleagă să fie donator sau să adopte un animal
- Poate vizualiza istoricul veterinar al animalelor
- Poate să adopte un animal
- Poate vizualiza evidența a mai multe adăposturi de animale
- Poate dona la unul dintre acestea la alegerea sa

3. Modelul de date si descrierea acestuia

3.1.Entitati si atributele lor (descriere detaliata – implementarea fizica)

Tabela Adapost
Memorează informații despre adăpostul de animale. Are ca atribute: id_adapost, nume_adapost, adresa_adapost, numar_locuri

Tabela Proprietar
Ține minte informații despre proprietarul adăpostului de animale precum id, nume, adresa, telefon, și mai reține și id_adpost(FK) care face legătură cu adăpostul pe care îl deține.

Tabela Loc
Reține detalii despre locul unui animal într-un adăpost precum id-ul, numele locului, și starea. Dacă starea poate fi „disponibil” sau „ocupat” care ajută la determinarea locurilor disponibile rămase dintr-un adăpost de animale.

Tabela Donator
Ține evidența donatoriilor oferind informații despre ei precum id, nume, adresă, telefon.

Tabela Donatie
Reține informații despre suma donată de un anumit donator precum suma dar și id-ul(FK) celui care a realizat donația.

Tabela pentru Fonduri
Ține evidența bugetului unui adăpost de animale cu veniturile și cheltuielile realizate. Se actualizează de fiecare dată când apare o tranzacție nouă.

Tabela pentru Animal
Se ține evidența animalelor care sunt în adăpostul de animale. Se rețin informații precum id, nume, specie, id_loc(FK) care face referire la locul pe care îl ocupă în adăpost. Dacă ocupă un anumit loc înseamnă că acel loc va avea starea ocupat.

Tabela IstoricVeterinar
Reține informații referitoare la istoricul veterinar al unui animal, cum ar fi vizitele pe care le face sau tartamente speciale.

Tabela AnimalIstoric
Acesta este o tabelă intermediară pentru a gestiona relația dintre Animal și IstoricVeterinar.

Tabela CabinetVeterinar
Memorează informații despre cabinetul pe care îl frecventează animalele aflate la adăpostul de animale precum, id, nume, adresa si numărul de telefon al cabinetului.

Tabela Provizii
Reține informații despre o anumită provizie care se află în inventarul unui adăpost de animale. Conține atributele id, nume și cantitate.

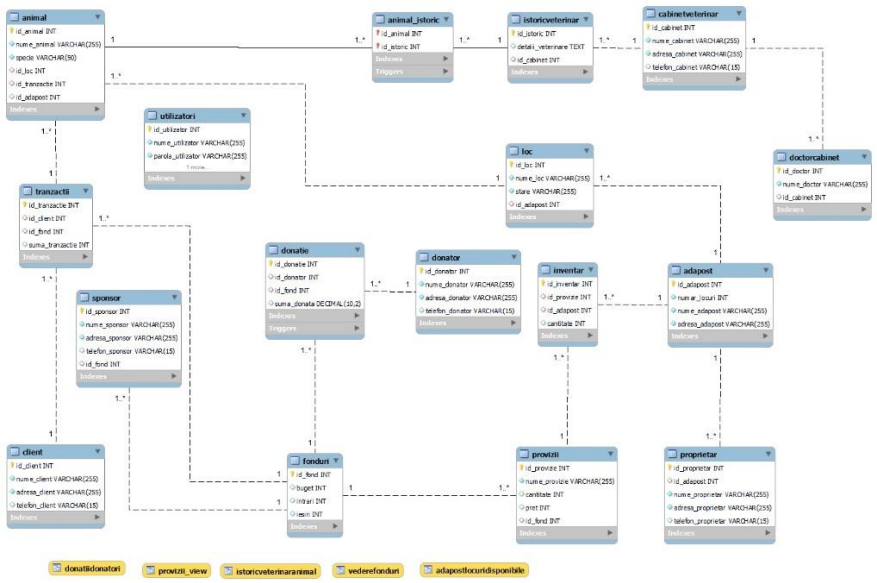
Tabela Inventar
Memorează toate proviziile făcute de un adăpost și le reține și cantitatea. Este o tabelă intermediară între adăpost și provizii.

Tabela DoctorCabinet
Pentru cabinetul veterinar există un doctor pentru care îi sunt reținute datele personale precum id, numele, id_ul cabinetului de care aparține(FK) și număr de telefon.

Tabela Client
Va ține evidența clienților unui adăpost de animale. Clienții sunt cei care adoptă animalele, despre care se reține informații precum id, nume, adresă, telefon.

Tabela Tranzactii
Este o tabelă intermediară care face legătura dintre client și animalul pe care îl adoptă. Tabela pentri tranzacție ajută pentru actualizarea fondurilor adăpostului de animale. Reține informații precum suma dată de client pentru a adopta animalul, id-ul clientului(FK) si id-ul animalului(FK).

3.2.Diagrama EER/UML pentru modelul de date complet



3.3.Normalizarea datelor

Normalizarea are ca scop eliminarea redundanței datelor și asigurarea integrității relaționale între tabele. Urmăresc fiecare tabel în parte pentru a evidenția normalizarea:

Tabelul Adapost

Este într-o formă normală de ordinul întâi (1NF) și îndeplinește cerințele pentru o bază de date relațională.

Tabelul Proprietar

Îndeplinește cerințele pentru 1NF, 2NF, și 3NF, deoarece nu există dependențe tranzitive sau redundanțe.

Tabelul Loc

Respectă cerințele pentru 1NF, 2NF și 3NF, cu o singură cheie primară și informații ne-redundante.

Tabelul Donator

Se conformează cerințelor pentru 1NF, 2NF și 3NF, fără dependențe tranzitive sau redundanțe.

Tabelul Donatie

Se încadrează în cerințele pentru 1NF, 2NF și 3NF, cu cheia primară și o cheie externă pentru a menține integritatea referențială.

Tabelul Fonduri

Îndeplinește cerințele pentru 1NF, 2NF și 3NF, cu o cheie primară și informații non-redundante.

Tabelul Animal

Respectă cerințele pentru 1NF, 2NF și 3NF, cu o cheie primară și chei externe pentru a menține coerența datelor.

Tabelul IstoricVeterinar

Este normalizat în 1NF, 2NF și 3NF, cu o cheie primară unică și fără redundanțe.

Tabela Animal_Istoric

Se încadrează în cerințele pentru 1NF, 2NF și 3NF, cu chei primare și externe corespunzătoare.

Tabelul CabinetVeterinar

Respectă cerințele pentru 1NF, 2NF și 3NF, cu o cheie primară și informații non-redundante.

Tabelul Provizii

Îndeplinește cerințele pentru 1NF, 2NF și 3NF, cu o cheie primară și informații non-redundante.

Tabelul Inventar

Este normalizat în 1NF, 2NF și 3NF, cu chei primare și externe corespunzătoare.

Tabelul DoctorCabinet

Se încadrează în cerințele pentru 1NF, 2NF și 3NF, cu o cheie primară și chei externe adecvate.

Tabelul Client

Respectă cerințele pentru 1NF, 2NF și 3NF, cu o cheie primară și informații non-redundante.

Tabelul Tranzactii
Este normalizat în 1NF, 2NF și 3NF, cu chei primare și externe pentru coerența datelor.

4. Detalii de implementare

Pentru această aplicație am folosit următoarele limbaje de programareȘ SQL pentru baza de date și Java pentru interfată.

SQL

Pentru funcționarea aplicației a fost necesar să realizăm trigerre, proceduri, vederi și am realizat și interogări pentru aplicație.

TRIGGERE
#Trigger-e

```
USE AdapostAnimale;

#Trigger pentru actualizarea automată a bugetului în tabelul Fonduri

DELIMITER //

CREATE TRIGGER ActualizareBuget
AFTER INSERT ON Donatie
FOR EACH ROW
BEGIN
   UPDATE Fonduri
   SET intrari = intrari + NEW.suma_donata,
      buget = buget + NEW.suma_donata;
END //

DELIMITER ;


#Trigger pentru gestionarea istoricului veterinar al animalelor in tabelul IstoricVeterinar
DELIMITER //

CREATE TRIGGER GestionareIstoricVeterinar
AFTER INSERT ON Animal_Istoric
FOR EACH ROW
BEGIN
   DECLARE detalii_veterinare TEXT;
```

```
   -- Obține detalii veterinarare asociate animalului
   SELECT detalii_veterinare INTO detalii_veterinare
   FROM IstoricVeterinar
   WHERE id_istoric = NEW.id_istoric
   LIMIT 1;

   -- Actualizează istoricul veterinar cu noile detalii
   UPDATE IstoricVeterinar
   SET detalii_veterinare = CONCAT(detalii_veterinare, ' | ', 'Noi detalii veterinarare: ',
NEW.id_animal)
   WHERE id_istoric = NEW.id_istoric;
END //

DELIMITER ;
```

PROCEDURI

```
#Proceduri

CREATE DEFINER=root@localhost PROCEDURE StergeAnimal(
   IN animal_id INT,
   IN id_loc INT,
   IN id_adapost INT
)
BEGIN
   -- Șterge înregistrările din tabela dependentă animal_istoric
   DELETE FROM animal_istoric WHERE id_animal = animal_id;

   -- Șterge animalul din tabelul Animal
   DELETE FROM Animal WHERE id_animal = animal_id;

   -- Restul procedurii rămâne neschimbat
   UPDATE Loc SET stare = 'Disponibil' WHERE id_loc = id_loc;
   UPDATE Adapost SET numar_locuri = numar_locuri + 1 WHERE id_adapost =
id_adapost;
END
```

```
CREATE DEFINER=root@localhost PROCEDURE EfectueazaTranzactie(
    IN p_id_client INT,
    IN p_id_fond INT,
    IN p_suma_tranzactie INT
)
BEGIN
    INSERT INTO Tranzactii (id_client, id_fond, suma_tranzactie)
    VALUES (p_id_client, p_id_fond, p_suma_tranzactie);
END


CREATE DEFINER=root@localhost PROCEDURE AdaugaDonatie(
    IN p_id_donator INT,
    IN p_id_fond INT,
    IN p_suma_donata DECIMAL(10, 2)
)
BEGIN
    INSERT INTO Donatie (id_donator, id_fond, suma_donata)
    VALUES (p_id_donator, p_id_fond, p_suma_donata);
END


CREATE DEFINER=root@localhost PROCEDURE AdaugaAnimal(
    IN p_nume_animal VARCHAR(255),
    IN p_specie VARCHAR(50),
    IN p_id_loc INT,
    IN p_id_tranzactie INT,
    IN p_id_adapost INT
)
BEGIN
    INSERT INTO Animal (nume_animal, specie, id_loc, id_tranzactie, id_adapost)
    VALUES (p_nume_animal, p_specie, p_id_loc, p_id_tranzactie, p_id_adapost);
END


CREATE DEFINER=root@localhost PROCEDURE ActualizeazaDetaliiAnimal(
    IN animal_id INT,
    IN nou_nume_animal VARCHAR(255),
    IN noua_specie VARCHAR(50)
)
BEGIN
    -- Actualizare înregistrare în tabelul Animal
    UPDATE Animal
```

```
    SET nume_animal = nou_nume_animal, specie = noua_specie
    WHERE id_animal = animal_id;
END
```

VEDERI

```
#Vederi
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = root@localhost
    SQL SECURITY DEFINER
VIEW provizii_view AS
    SELECT
        provizii.id_provizie AS id_provizie,
        provizii.nume_provizie AS nume_provizie,
        provizii.cantitate AS cantitate
    FROM
        Provizii


CREATE
    ALGORITHM = UNDEFINED
    DEFINER = root@localhost
    SQL SECURITY DEFINER
VIEW istoricveterinaranimal AS
    SELECT
        animal.id_animal AS id_animal,
        animal.nume_animal AS nume_animal,
        istoricveterinar.detalii_veterinare AS detalii_veterinare
    FROM
        ((animal
        LEFT JOIN animal_istoric ON ((animal.id_animal = animal_istoric.id_animal)))
        LEFT JOIN istoricveterinar ON ((animal_istoric.id_istoric =
istoricveterinar.id_istoric)))


CREATE
    ALGORITHM = UNDEFINED
    DEFINER = root@localhost
    SQL SECURITY DEFINER
VIEW donatiidonatori AS
    SELECT
        donatie.id_donatie AS id_donatie,
        donator.nume_donator AS nume_donator,
```

```
    donator.adresa_donator AS adresa_donator,
    donator.telefon_donator AS telefon_donator,
    donatie.suma_donata AS suma_donata
  FROM
    (donatie
    JOIN donator ON ((donatie.id_donator = donator.id_donator)))


CREATE
  ALGORITHM = UNDEFINED
  DEFINER = root@localhost
  SQL SECURITY DEFINER
VIEW adapostlocuridisponibile AS
  SELECT
    adapost.id_adapost AS id_adapost,
    adapost.nume_adapost AS nume_adapost,
    adapost.adresa_adapost AS adresa_adapost,
    (adapost.numar_locuri - COUNT(loc.id_loc)) AS locuri_disponibile
  FROM
    (adapost
    LEFT JOIN loc ON ((adapost.id_adapost = loc.id_adapost)))
  GROUP BY adapost.id_adapost , adapost.nume_adapost , adapost.adresa_adapost ,
adapost.numar_locuri
```

INTEROGĂRI

```
#Interogari

USE AdapostAnimale;

#Afișarea tuturor animalelor și adăposturilor asociate
SELECT Animal.*, Adapost.*
FROM Animal
JOIN Adapost ON Animal.id_adapost = Adapost.id_adapost;


#Afișarea informațiilor despre donații și donatori
SELECT Donatie.*, Donator.*
FROM Donatie
JOIN Donator ON Donatie.id_donator = Donator.id_donator;

#Afișarea detaliilor despre animalele care au avut tratament veterinar
SELECT Animal.*, IstoricVeterinar.*
```

```
FROM Animal
JOIN Animal_Istoric ON Animal.id_animal = Animal_Istoric.id_animal
JOIN IstoricVeterinar ON Animal_Istoric.id_istoric = IstoricVeterinar.id_istoric;


#Calcularea bugetului total și a sumei donate
SELECT SUM(buget) AS buget_total, SUM(intrari) AS sum_donate
FROM Fonduri;
```

4.1. Descrierea functionala a modulelor (organizarea logica a acestora- de exemplu structura claselor Java, cod HTML, JSP, ASP, PhP)

Clasele Java sunt:

- AdaugaAnimal
- AdoptionDialog
- AdoptionPage
- CreateAccount
- DonationDialog
- LogInCustomer
- LogInOwner
- LoginPage
- OwnerActions
- ShelterList
- ViewAnimalData
- ViewAvailableSpots
- ViewFunds
- ViewTransactions

4.2.Manual de utilizare/instalare (diferentiat pe tipuri de actori)

4.2.1.Tooluri

MySQL Workbench

MySQL Workbench este un instrument grafic pentru a lucra cu serverele și bazele de date MySQL. MySQL Workbench este prevăzut să lucreze cu versiunile de MySQL Server 5.1 și mai sus. MySQL Workbench tinde să fie un instrument ce acoperă cele mai importante activități de gestionare a bazelor de date:

SQL Development: permite să gestionezi conexiunile la serverele MySQL, la fel oferă posibilitatea de a executa interogări SQL.

Data Modeling: permite să creezi modele de baze de date în mod grafic, cât și editarea a bazelor de date deja existente. Table Editor este menit pentru editare de tabele, coloane, indici, triggere, partiționare, opțiuni, inserturi și privilegii, rutine și view-uri.

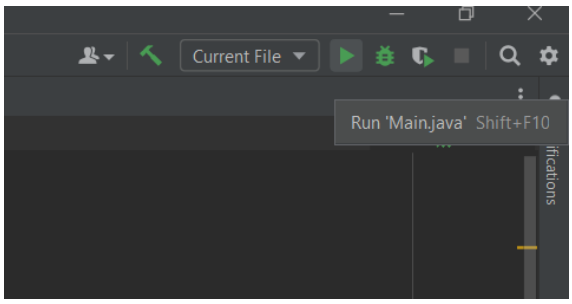
Server Administration: permite să crezi și administrezi instanțele de server.

Data Migration: permite să migrezi pe MySQL datele din Microsoft SQL Server, Sybase ASE, SQLite, SQL Anywhere, PostgreSQL.

IntelliJ IDEA

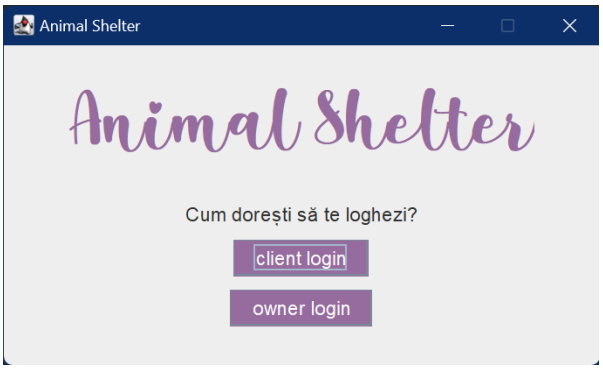
IntelliJ IDEA este un IDE pentru Java și Kotlin, conceput pentru a maximiza productivitatea dezvoltatorului. Acesta preia sarcinile rutiniere și repetitive prin furnizarea de completare inteligentă a codului, analiză statică a codului și refactoare, permițându-ți să te concentrezi asupra aspectelor pozitive ale dezvoltării software. Acest lucru nu numai că face procesul productiv, dar îl transformă și într-o experiență plăcută.

După instalarea celor două aplicații trebuie descărcat o arhivă cu tabelele din MySQL si clasele din IntelliJ IDEA și să ne asigurăm că se realizează conexiunea, iar după aceea putem rula programul din clasa LogInPage în IntelliJ IDEA.



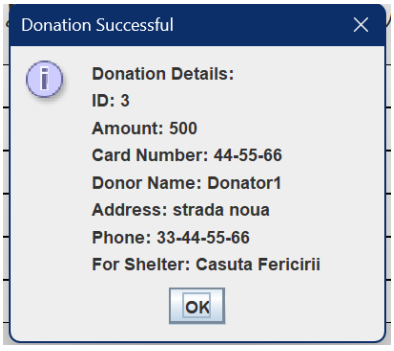
4.2.2. Interfata

Inițial ai opțiunea de a te loga pe aplicație fie ca și proprietar, fie ca și client. În momentul în care te apeși pe oricare dintre cele două butoane îți va arăta opțiunea de a te loga sau de a-ți crea un cont nou.

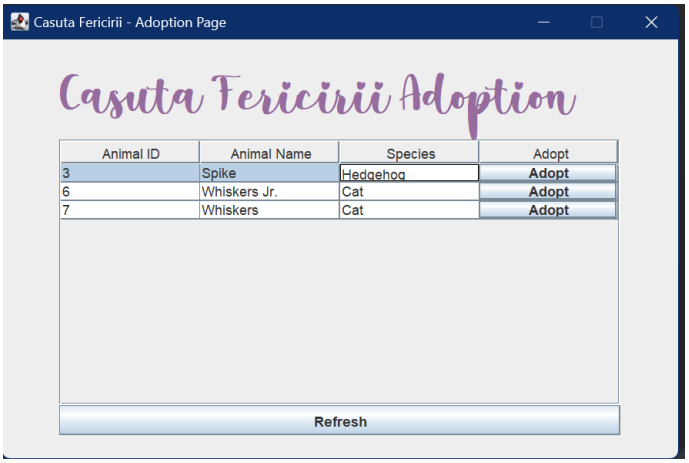


Pentru opțiunea client vei avea de ales dacă vrei să donezi sau dacă vrei să adopți un animal de la un adăpost de animale la alegere.

Pentru donator se va deschide o fereastră cu mai multe câmpuri care vor fi completate, iar la urmă va apărea un mesaj cu detaliile donației.

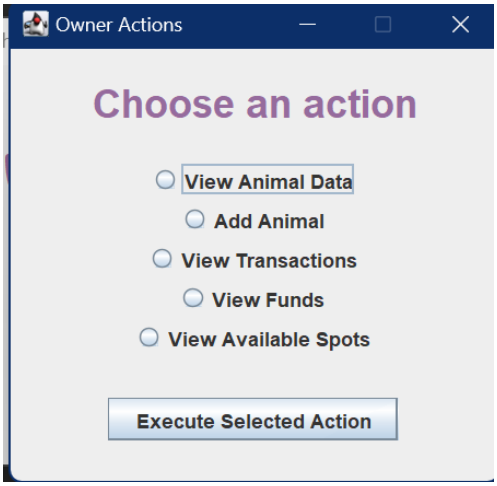


Pentru clientul care va adopta un animal, acesta va vedea animalele care sunt la adăpostul ales.



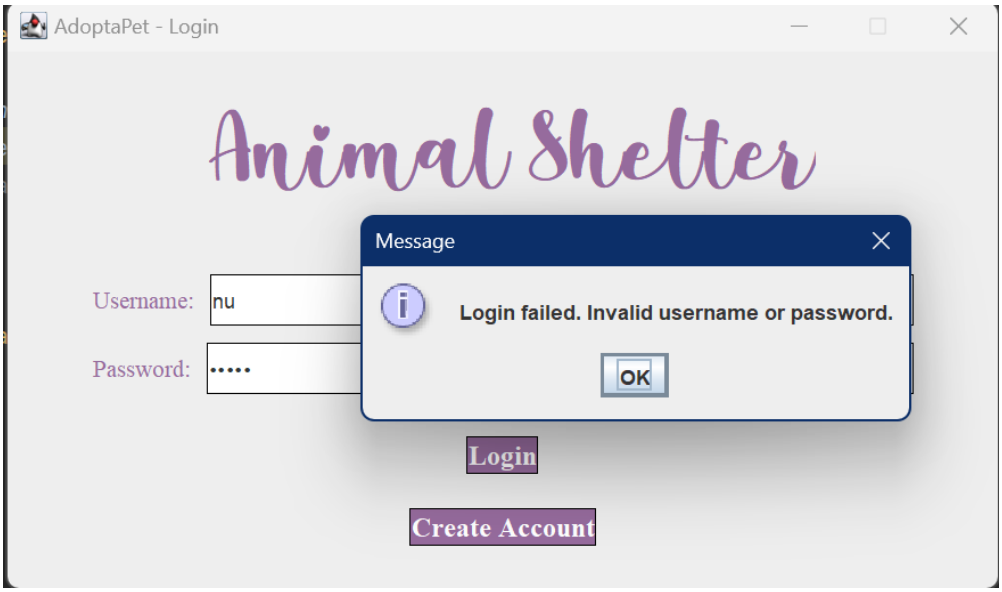
Dacă te loghezi ca și proprietar vei avea mai multe opțiuni, adică

- Să vizualizezi informații despre animale
- Să adaugi un animal în baza de date
- Să vizualizezi tranzacțiileț
- Sa vizualizezi fondurile
- Să ții evidența numarului de locuri disponibile la adăpostul pe care îl deții cât și la altele.

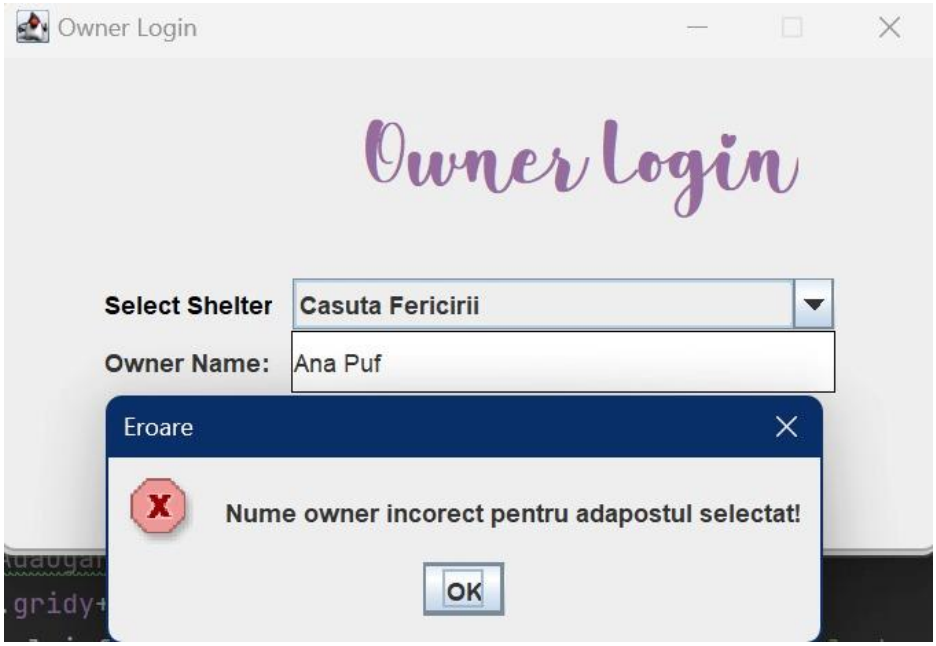


4.3.Elemente de securizare a aplicatiei

Una dintre elementele de securizare este protejarea confidențialității datelor personale ale donatorilor, proprietarilor și altor părți implicate. Aplicatia așteapta date valide, deci in cazul unei logari esuate, va apărea o fereastră care anunță utilizatorul ca logarea a eșuat.



În cazul în care vrei să te loghezi ca și owner, iar numele introdus nu corespunde cu adăpostul ales sau nu există în baza de date se va afișa un mesaqj de eroare.



5. Concluzii limitari si dezvoltari ulterioare

Baza noastră de date complexă a fost concepută cu un scopul de a optimiza și îmbunătăți modul în care gestionăm și oferim îngrijire animalelor aflate în lanțul nostru de adăposturi.

Prin crearea și implementarea acestei baze de date, am dorit să asigurăm un mediu mai eficient și mai personalizat pentru fiecare animal, respectându-i individualitatea și nevoile specifice. Faptul că acum avem acces la informații detaliate privind istoricul medical, comportamental și alte aspecte esențiale pentru fiecare animal ne permite să oferim îngrijirea adecvată și să facilităm procesul de adopție.

În ceea ce privește dezvoltarea aplicației aceasta poate fi îmbunatățită și ajustată. Spre exemplu, s-ar putea crea o nouă operație pentru proprietar în care acesta ar avea un magazin de unde să își achiziționeze proviziile adăpostului său de animale.

Proiect de Semestru