UNIVERSITATEA TEHNICĂ „GH ASACHI” IAŞI
FACULTATEA AUTOMATICĂ ŞI CALCULATOARE
SPECIALIZAREA CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI

DISCIPLINA BAZE DE DATE

Adăpost pentru câini

Coordonator,
Buțincu Cristian

Student,
Panainte Ancuța

Iași, 2019

Titlu proiect: Adăpost pentru câini

Analiza, proiectarea și implementarea unei baze de date și a aplicației aferente care să modeleze activitatea unui adăpost pentru câini. Se vor asigura elementele necesare pentru o bună funcționare a adăpostului.

Descrierea cerințelor și modul de organizare al proiectului

Numărul mare de animale salvate determină necesitatea creării unei baze de bate pentru o organizare mai eficientă a informațiilor într-un adăpost. Gestionarea implică înregistrarea persoanelor care intră în contact cu adăpostul, o evidență a angajaților și a animalelor, împreună cu specificațiile lor.

Informațiile necesare sunt:

-**persoane** care vizitează adăpostul – ne interesează să cunoaștem detaliile de contact, precum: numele, adresa și numărul de telefon;

-**angajații** – în acest caz medicii veterinari, la care ne interesează numele și numărul de telefon; aceștia vor întocmi fișele medicale;

-**animale** – informațiile despre animale sunt stocate în trei tabele urmărind:

-informații de bază – gen, data primirii, talia, numele și posibile observații;

-date despre cazare – tipul de cazare, spațiul încăperii în care au fost distribuiți; organizarea va fi efectuată în funcție de talie, astfel câinii de talie mare vor fi puși câte 2, cei de talie medie câte 3 și cei de talie mică vor fi maxim 4, considerând că animalele nu au un comportament agresiv;

-istoricul medical – constituit din fișe prin care se rețin tipurile de proceduri(medicamente, operație, vaccin, verificare), costurile și data efectuării; se vor completa fișe diferite pentru fiecare patruped și verificările vor fi efectuate ținând cont de tratamentul acordat;

-**adopțiile** – fiecărei adopții i se atribuie un tip(la sediu, la distanță sau revendicare) și o dată; ne va interesa să știm ce animal este adoptat și de către cine;

-**donațiile** – donațiile pot fi: prin transfer bancar, provizii sau oferirea a 2% din impozit; se vor cunoaște informații despre persoanele care aduc astfel de contribuții.

Descrierea funcțională a aplicației

Principalele funcţii care se pot întâlni într-un adăpost sunt:

-evidența animalelor din adăpost;

-evidența persoanelor care interacționează cu adăpostul și a acțiunilor efectuate de acestea;

-evidența angajaților.

Descrierea detaliată a entităților și a relațiilor dintre tabele

Tabelele din această aplicație sunt:

-caine;

-cazare;

-fisa_medicala;

-medic_veterinar;

-persoana;

-adoptie;
-donatie.

În proiectarea acestei baze de date s-au utilizat tipurile de relații 1:1,1:n și n:1.

Între tabelele **caine** și **adoptie** se identifică o relație one-to-one deoarece un câine poate fi adoptat o singură dată, din momentul în care a fost adoptat ne mai fiind disponibil. Legătura dintre cele două tabele se face prin **câmpul id**.

Între tabelele **caine** și **cazare** se întâlnește o relație many-to-one. Într-un spațiu de cazare pot locui mai mulți câini, însă reciproca nu este valabilă fiindcă un câine nu poate sta în mai multe încăperi. Legătura dintre tabele este dată de atributul **nr_cazare**.

O relație one-to-many este cea dintre **caine** și **fisa_medicala** întrucât câinii pot avea mai multe fise medicale, dar o fișă medicală nu este valabilă pentru mai mulți câini. Legătura dintre cele două tabele se realizează prin **câmpul id.**

Între **fisa_medicala** și **medic_veterinar** se stabilește o relație many-to-one, un medic va efectua mai multe fișe cu procedurile medicale, în timp ce o fișă poate fi completată de un singur medic. Legătura este data de **(nume_medic, id_medic**).

O altă relație de tipul many-to-one este cea determinată între tabelele **adopție** și **persoana**. O persoană are posibilitatea de a face mai multe adopții, însă o adopție poate fi realizată de o singură persoană. Legătura este **id_pers**.

De asemenea, între **persoana** și **donatie** există o relație one-to-many. O persoană poate dona de mai multe ori, dar în sens invers nu este valabil - o donație nu poate fi realizată de mai multe persoane. Legătura este **id_pers**.

Constrângerile definite în cadrul bazei de date sunt:

-not null – în cazul atributelor ce descriu informații strict necesare pentru o funcționare adecvată a adăpostului: id-uri, nume de persoane, detalii contact, date, acțiunile efectuate;
-unique – numerele de telefon sunt înregistrate ca fiind unice;
-check – pentru a asigura introducerea unor valori din opțiunile standard(gen-F/M, talie-mică/medie/mare, cazare-interior/exterior și spațiu maxim disponibil-2/3/4) și pentru a valida datele de intrare(numele să conțină doar litere mari și literi mici);
-primary key – pentru identificarea entităților; generate prin autoincrement: nr_adoptie, id, nr_donatie, id_fisa și id_pers sau introduse la intrare: nr_cazare, (id_medic, nume_medic).
-foreign key – câmpuri preluate din alte tabele, unde au rol de primary key; conectează entități diferite cu scopul de a aduce informațiile necesare.

Tehnologiile utilizate pentru implementarea bazei de date:

Modelele logic, relațional și fizic au fost realizate în programul *Data Modeler*. Scriptul rezultat a fost introdus într-o bază de date Oracle, în *Oracle SQL Developer*. Pentru construirea aplicației web s-a utilizat din *python* framework-ul *Flask*, care asigură comunicarea între partea de interfață și baza de date. Accesul la baza de date se face prin modulul *cx_Oracle*. Interfața a fost creată folosind *HTML*, iar pentru îmbunătățirea aspectului *CSS*.

Conectarea la baza de date se face prin funcția connect a modulului cx_Oracle.

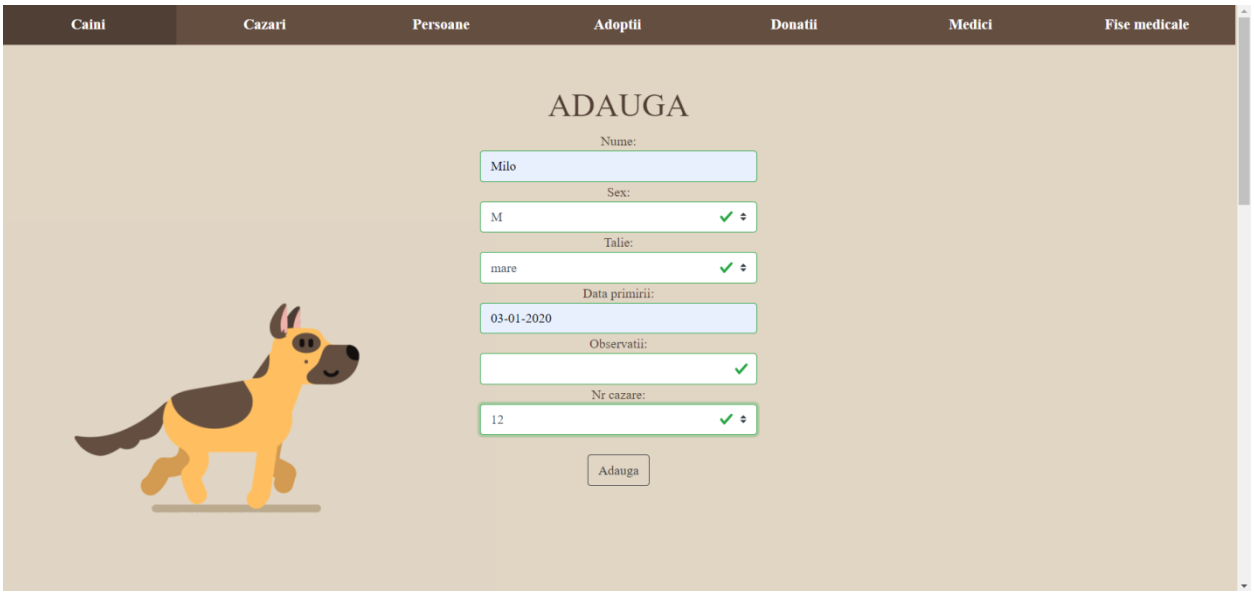
```
con = cx_Oracle.connect("student", "student", "localhost/xe")
```

Funcția INSERT – adăugarea a unei entități în tabela caine se realizează astfel:

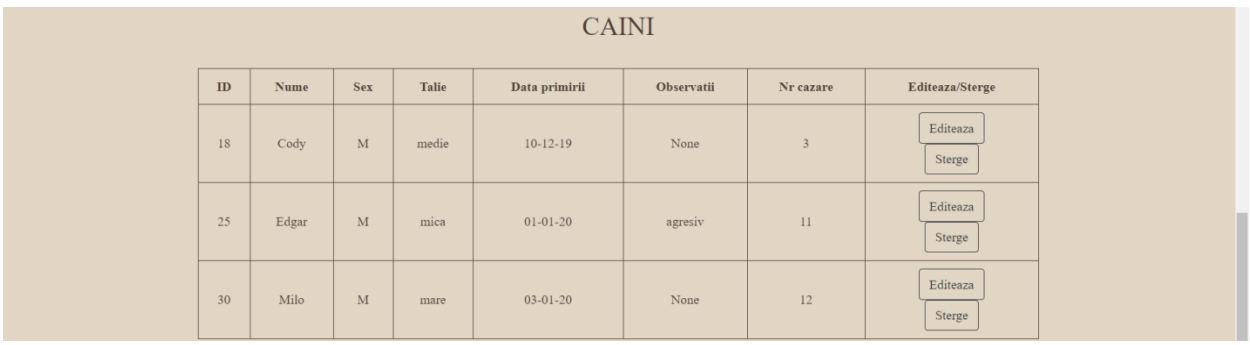
```
@app.route('/add_caine', methods=['POST'])
def add_caine_fct():
    if request.method == 'POST':
        cur = con.cursor()
        values = []
        values.append("'" + request.form['nume'] + "'")
        values.append("'" + request.form['sex'] + "'")
        values.append("'" + request.form['talie'] + "'")
        values.append(
            "'" + datetime.strptime(str(request.form['data_primirii']), '%d-%m-%Y').strftime('%d-%m-%Y') + "'")
        values.append("'" + request.form['observatii'] + "'")
        values.append("'" + request.form['nr_cazare'] + "'")
        query = 'insert into caine values(NULL,' + ', '.join(values) + ')'

        cur.execute(query)
        cur.execute('commit')
        return redirect('/caine')
```

Completarea câmpurilor:



Rezultatul:



Funcția UPDATE – editarea atributelor din cadrul entității se realizează astfel:

```
@app.route('/edit_caine', methods=['POST'])
def edit_caine_fct():
    id = "'" + request.form['id'] + "'"
    nume = "'"+request.form['nume']+"'"
    obs = "'"+request.form['observatii']+"'"
    nr = "'"+request.form['nr_cazare']+"'"

    cur = con.cursor()
    query = "update caine set nume=%s, nr_cazare=%s, observatii=%s where id=%s" % (nume, nr, obs, id)
    cur.execute(query)

    return redirect('/caine')
```

La efectuarea unui update, datele existente sunt citite din tabelă și completate automat. Astfel, se pot modifica doar câmpurile dorite, fără a fi nevoie să se insereze și cele nemodificate.

Funcția DELETE – ștergerea unei întregistrări:

```
@app.route('/del_caine', methods=['POST'])
def del_caine_fct():
    cur = con.cursor()
    cur.execute('delete from caine where id=' + request.form['id'])
    cur.execute('commit')
    return redirect('/caine')
```

Diagrama Entitate Relație