Documentație - Aplicație web pentru administrarea unui centru de adopție de animale

Proiect - Aplicații Web pentru baze de date

Atasie Oana-Andreea

Grupa 405 (IF)

Facultatea de Matematică și Informatică

Universitatea din București

Cuprins

Scopul proiectului

Aplicația își propune să faciliteze administrarea unui centru de adopție de animale. Administratorul și angajații pot vizualiza și prelucra mai ușor informațiile legate de animalele aduse în centru, precum și informațiile necesare îngrijirii acestora.

Baza de date

Schema bazei de date poate fi observată în diagrama Entitate/Relație:

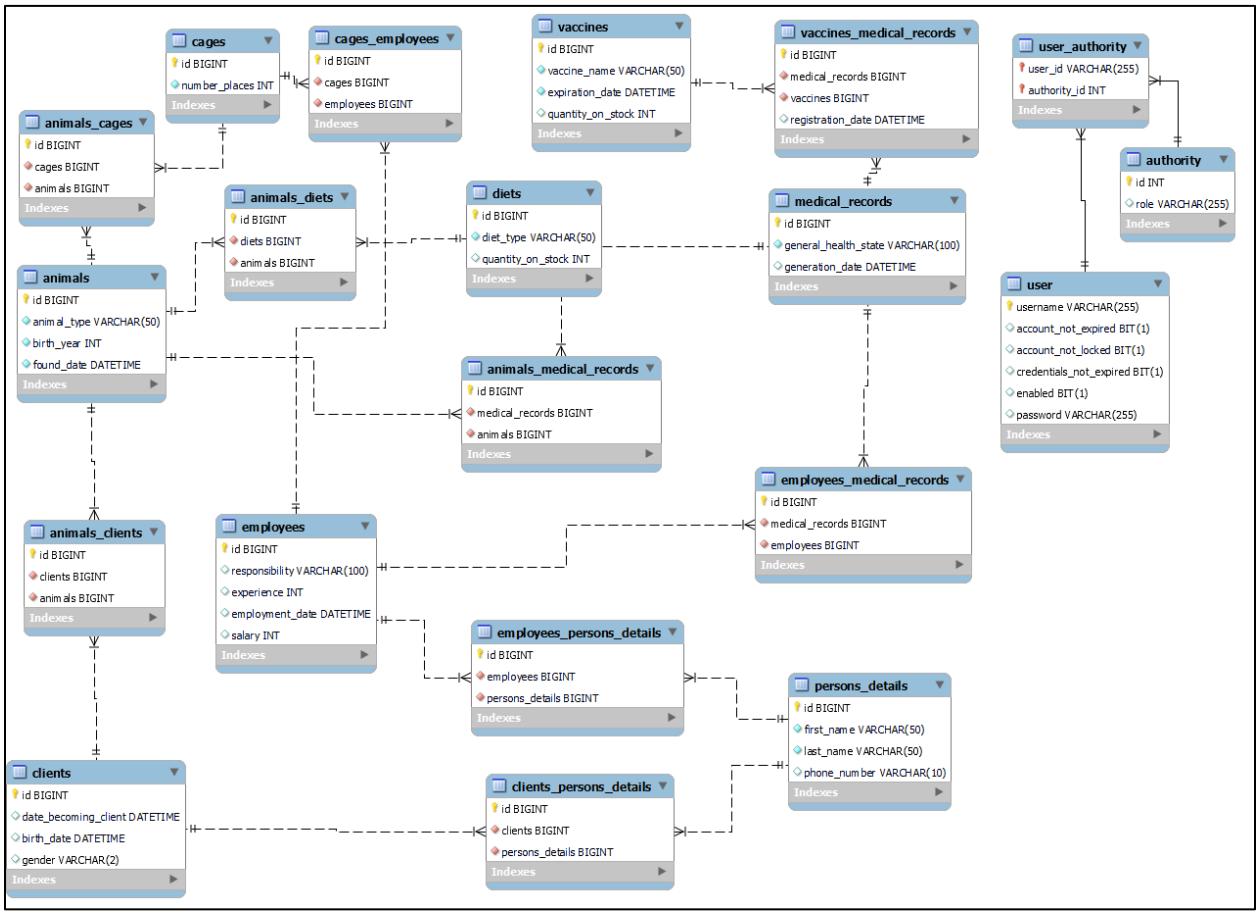


Diagrama Entitate/Relație

După cum reiese și din diagrama anterioară, între tabele există diferite tipuri de relații marcate în baza de date prin tabele de legătură. Tabelul Persons_Details se află în relație One to One atât cu Employees, cât și cu Clients. Relațiile Animals-Clients, Animals-Cages, Animals-Diets, Cages-Employees, Medical_Records-Animals, Medical_Records-Employees sunt de tip One to Many. O relație de tip Many to Many există între Medical_Records și Vaccines, această

relație fiind prelucrată ulterior drept două relații de tip One to Many printr-un tabel asociativ. Pe de altă parte, în interiorul codului, relațiile One to Many au fost mapate drept Many To One pentru o mai ușoară manipulare a entităților și a legăturilor dintre acestea.

Entități și operații CRUD

Pentru tabelele principale din baza de date au fost create următoarele entități:

- Persons Details - o persoană este caracterizată prin nume, prenume și număr de telefon;
- Employees - un angajat este caracterizat prin date personale, data angajării și salariu; centrul cuprinde două tipuri de angajați: veterinari și îngrijitori care sunt deosebiți prin responsabilitate (valoare nulă pentru veterinar) și experiență (valoare nulă pentru îngrijitor);
- Clients - prin această entitate se prelucrează informațiile despre clienții care adoptă animale; pentru fiecare client sunt reținute date personale, cât și data în care respectiva persoană a devenit client al centrului;
- Diets - se ocupă de gestionarea informațiilor despre dietele animalelor care au fost aduse în centru (numele și cantitatea existentă pe stoc);
- Cages - animalele sunt ținute în cuști îngrijite de îngrijitori, iar într-o cușcă pot încăpea un număr maxim de animale;
- Animals - se reține informații legate de animalele aduse în centru: tipul de animal, data la care a fost găsit și anul nașterii; fiecare animal are o dietă și o cușcă atribuită, iar adopția animalului poate fi marcată prin asocierea acestuia cu un client;
- Vaccines - animalele sunt vaccinate pentru a menține sănătatea acestora; există în stoc o cantitate de vaccinuri cu o anumită denumire, care expiră la o anumită dată salvată în baza de date pentru a nu folosi doze expirate;
- Medical Records - animalele sunt controlate regulat de către veterinari, aceste informații fiind consemnate în fișele medicale (starea de sănătare a animalului și data de generare);
- Registered Vaccines - fiecare fișă medicală poate conține mai multe vaccinuri, iar un vaccin poate să apară în mai multe fișe medicale.

Pentru opt dintre cele nouă entități menționate (informațiile reținute în Persons_Details au fost integrate în cele afișate pentru Employees și Clients) au fost dezvoltate toate cele patru tipuri de operații CRUD: GET, POST, PUT și DELETE.

Log-uri și aspecte

Pentru a verifica structura și logica proiectului, pentru metodele existente în Repository, Service și Controller au fost adăugate log-uri care conțin informații despre funcțiile apelate și ordinea de apelare a acestora. Pentru a afișa mai ușor log-urile, acestea au fost implementate folosind aspecte.

Pentru metodele din Repository log-urile sunt afișate după execuție, iar pentru cele din Service și Controller informațiile apar în consolă atât înainte cât și după execuție. Detaliile afișate sunt numele metodei apelate, clasa în care se regăsește funcția, data de execuție, cât și stadiul (înainte sau după rulare).

Pagini, validări și excepții

Pentru cele opt entități prezentate mai sus, operațiile CRUD au fost utilizate pentru a realiza pagini în Thymeleaf. Paginile pot fi accesate prin intermediul meniului sau prin intermediul butoanelor existente în alte pagini. Pentru fiecare tip de obiect au fost implementate patru pagini care conțin vizualizarea tuturor obiectelor de acel tip, vizualizarea detaliilor unui singur obiect, precum și adăugarea și editarea obiectelor. De asemenea, pe pagina asociată metodei GET ALL există opțiunea de ștergere a unui element.

Meniul aplicației

Pentru a putea fi vizualizate mai ușor paginile care conțin mai multe date a fost implementat un sistem de paginare. Toate elementele preluate din baza de date au fost împărțite câte trei sau cinci pe pagină în funcție de cantitatea de informații conținută de acestea. Atfel, înregistrările nu sunt afișate linear pe o singură pagină, ci sunt ilustrate în cadrul mai multor pagini numerotate.

Corectitudinea datelor a fost verificată prin intermediul excepțiilor și validărilor. Excepțiile au în principal rolul de a proteja baza de date de informații eronate. Eroile tratate în proiect sunt legate de adăugarea unor angajați, clienți, vaccine sau diete cu același și de adăugarea unui animal într-o cușcă deja plină. În ceea ce privește partea de validare, formularele de inserare și editare de date cuprind mesaje care ajută la completarea corectă a informațiilor. Pentru ca formularele să fie valide datele completate pot prezenta următoarele restricții:

- să nu fie goale (folosind adnotarea *@NotBlank*);
- să respecte formatarea unui număr de telefon (folosind o adnotare custom *@PhoneNumberMatch* care se bazează pe un format de tip regex);
- să aibă o valoare sau o dimensiune minimă (folosind adnotările *@Min* și *@Size*);
- genul unei persoane trebuie să fie M sau F (folosind o adnotare custom *@OnlyCharacterMF*);

Securitatea aplicației

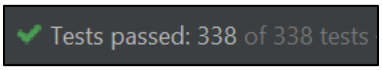
Pentru a avea o securitate a aplicației au fost create roluri în funție de care utilizatorul are acces sau nu la anumite acțiuni sau pagini. Aplicația curpinde două roluri: rol de administrator și rol de angajat. Administratorul poate executa toate acțiunile existente (interogarea, adăugarea, actualizarea, eliminarea datelor) , în timp ce angajații au doar drept de vizualizare. În momentul în care un angajat încearcă să realizeze operații de adăugare, editare sau ștergere a datelor, acesta este redirecționat către pagina de Access Denied.

Pentru a putea verifica rolul utilizatorului, în primul rând au fost create două tabele/entități. Tabelul User conține username-ul și parola utilizatorului, iar în tabelul Authority reține rolul utilizatorului. Astfel încât utilizatorul poate ajunge pe anumite pagini doar dacă deține un anumit rol, utilizatorii trebuie să se logheze folosind username-ul și parola sa. Utilizatorul neînregistrat poate doar să acceseze pagina de login, restul paginilor necesitând minim un rol din cele două.

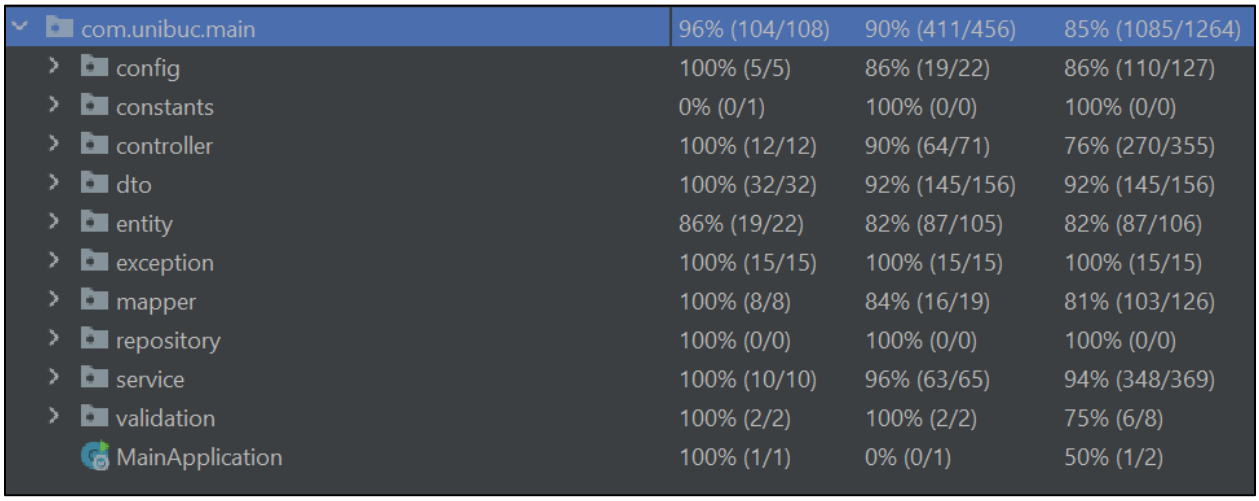
Testarea aplicației

Pentru testarea modului de funcționare al aplicației au fost scrise teste la nivel de Repository, la nivel de Service și la nivel de Controller, existând un total de 338 teste. Testele

unitare efectuate pe Service au fost realizate folosind obiecte mock, în timp ce cele de pe Repository apelează informațiile din baza de date. Pentru Controller au fost scrise teste unitare cu mock, cât și două tipuri de teste de integrare: teste mock-uite și teste care să valideze întregul șir de acțiuni care au loc atunci când utilizatorul interacționeaza cu aplicația prin frontend. Pentru testele fară mock, la nivel de Repository și Controller, a fost utilizată o bază de date H2 pentru a nu afecta baza de date reală implementată în MySQL.



Totalul testelor



Acoperirea codului