UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Proiect final

Proiectare software

Autor: Volcov Sabina
Grupa: 30231

Facultatea de Automatică
și Calculatoare

mai 2024

Cuprins

1 Problema de rezolvat

Transformați aplicația implementată la tema 3 într-o aplicație client/server astfel încât să utilizați minim 5 șabloane de proiectare (minim un șablon de proiectare creațional, un șablon de proiectare comportamental și un șablon de proiectare structural) și o arhitectură orientată pe servicii SOA pentru comunicare între aplicația server și aplicația client.

- În faza de analiză se va realiza diagrama cazurilor de utilizare și diagramele de activități pentru fiecare caz de utilizare (Observație: numărul diagramelor de activități trebuie să fie egal cu numărul de cazuri de utilizare din diagrama cazurilor de utilizare).
- În faza de proiectare se vor realiza:
  - 2 diagrame de clase corespunzătoare aplicației soft server și aplicației soft client respectând principiile DDD și folosind o arhitectură orientată pe servicii (SOA) și minim 5 șabloane de proiectare;
  - diagrama entitate-relație corespunzătoare bazei de date;
  - diagrame de secvență corespunzătoare tuturor cazurilor de utilizare (Observație: numărul diagramelor de secvență trebuie să fie cel puțin egal cu numărul de cazuri de utilizare din diagrama cazurilor de utilizare).
- În faza de implementare se va scrie cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând:
  1. proiectarea dată de diagramele de clase și diagramele de secvență;
  2. unul dintre următoarele limbaje de programare: C#, C++, Java, Python.
- Finalizarea temei va consta în predarea unui director ce va cuprinde:
  1. Un fișier cu diagramele UML realizate;
  2. Baza de date;
  3. Aplicația soft;
  4. Documentația (minim 20 pagini) - un fișier care cuprinde:
     - numele studentului, grupa;
     - enunțul problemei;
     - instrumente utilizate;
     - justificarea limbajului de programare ales;
     - descrierea diagramelor UML (inclusiv figuri cu diagramele UML realizate);
     - descrierea aplicației (inclusiv figuri reprezentând interfețele grafice ale aplicației client).

2 Faza de analiza - Diagrama cazurilor de utilizare

2.1 Enuntul problemei

Problema 11

Dezvoltați o aplicație client/server care poate fi utilizată într-o grădină botanică. Aplicația va avea 3 tipuri de utilizatori: vizitator al grădinii botanice, angajat al grădinii botanice și administrator. Utilizatorii de tip vizitator pot efectua următoarele operații fără autentificare:

- Vizualizarea listei tuturor plantelor din grădina botanică sortată după tip și specie (vizualizarea include și redarea unor imagini cu toate plantele din grădina botanică; între 1 și 3 imagini pentru fiecare plantă);
- Filtrarea listei plantelor după următoarele criterii: tip, specie, plante carnivore, zona grădină botanică;
- Căutarea unei plante după denumire.

Utilizatorii de tip angajat al grădinii botanice pot efectua următoarele operații după autentificare:

- Toate operațiile permise utilizatorilor de tip vizitator;
- Operații CRUD în ceea ce privește persistența plantelor din grădina botanică;
- Salvare liste cu plantele în mai multe formate: csv, json, xml, doc;
- Vizualizarea unor statistici legate de plantele din grădina botanică utilizând grafice (structură radială, structură inelară, de tip coloană, etc.).

Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

- Toate operațiile permise utilizatorilor de tip vizitator;
- Operații CRUD pentru informațiile legate de utilizatorii care necesită autentificare;
- Vizualizarea listei utilizatorilor care necesită autentificare și filtrarea acesteia după tipul utilizatorilor.
- Notificarea fiecărui utilizator care necesită autentificare prin cel puțin 2 variante (email, SMS, WhatsApp, Skype, etc.) la orice modificare a informațiilor de autentificare aferente acelui utilizator.

  Interfața grafică a aplicației client va fi disponibilă în cel puțin 3 limbi de circulație internațională.

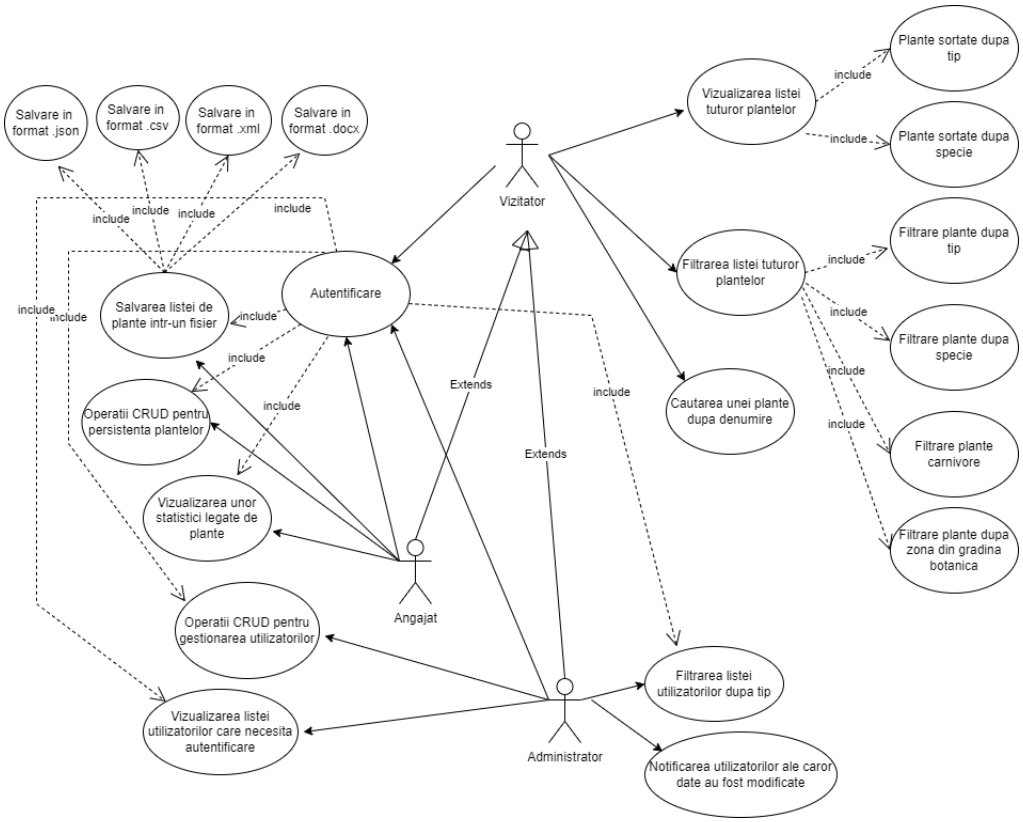

Figura 1: Diagrama cazurilor de utilizare

2.2 Diagrama cazurilor de utilizare

Din diagramă reies cei 3 actori care apar în aplicație, și anume vizitatorii anonimi, care pot vizualiza, sorta, filtra și căuta după denumire plantele din baza de date, angajații autentificați care moștenesc use-case-urile vizitatorii și pot pe deasupra realiza operații CRUD pentru persistența plantelor, salvarea listei de plante într-un fișier în 4 formate și vizualiza statistici legate de plantele din grădina botanică, și administratorii, care la fel moștenesc cazurile de utilizare ale vizitatorilor, dar spre deosebire de angajați pot efectua operații CRUD și vizualiza utilizatorii care necesită autentificare, inclusiv filtrarea acestora după tipul de utilizator. De asemenea, administratorii trimit notificări prin Whatsapp și e-mail utilizatorilor ale căror date le-au modificat.

2.3 Diagrame de activități

Fiecare caz de utilizare poate fi reprezentat printr-o diagramă de activitate care exprimă acțiunile pe care le face utilizatorul pentru a comunica cu aplicația.

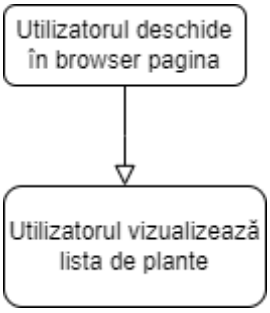

Figura 2: Diagrama de activitate - Vizualizarea plantelor

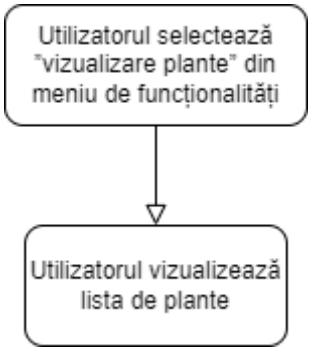

Figura 3: Diagrama de activitate - Vizualizarea plantelor din meniu

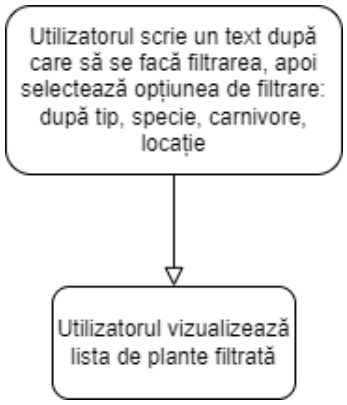

Figura 4: Diagrama de activitate - Vizualizarea listei filtrate de plante

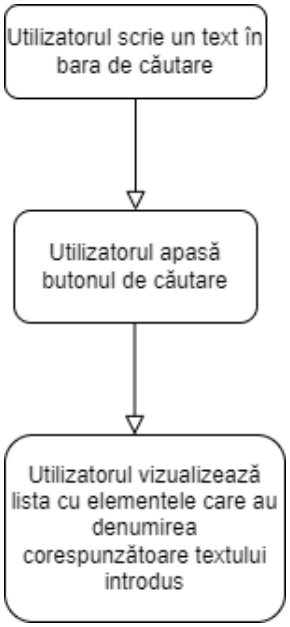

Figura 5: Diagrama de activitate - Cautarea dupa denumire

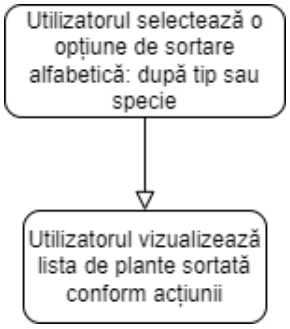

Figura 6: Diagrama de activitate - Vizualizarea listei sortate de plante

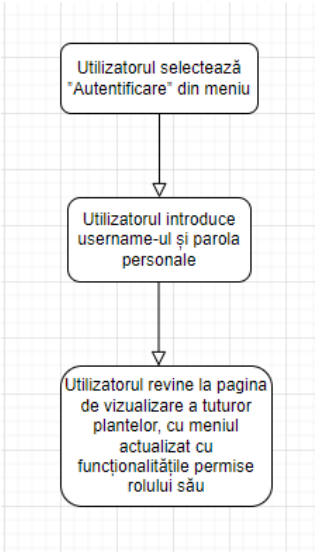

Figura 7: Diagrama de activitate - Autentificarea

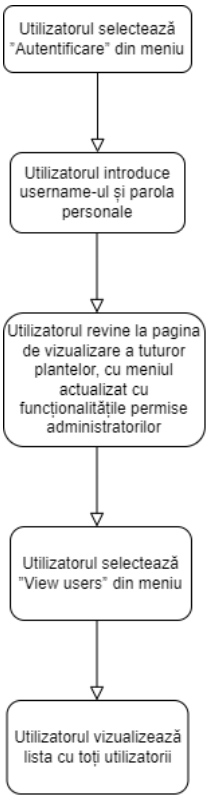

Figura 8: Diagrama de activitate - Vizualizarea utilizatorilor

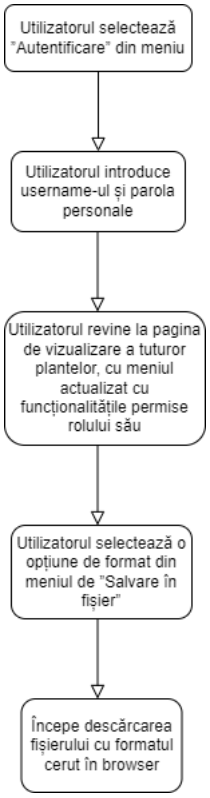

Figura 9: Diagrama de activitate - Salvarea listei de plante într-un fișier

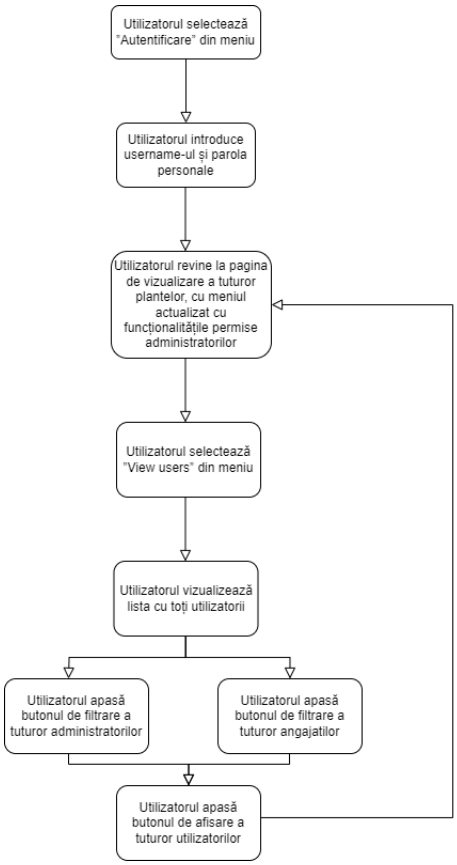

Figura 10: Diagrama de activitate - Vizualizarea listei filtrate de utilizatori

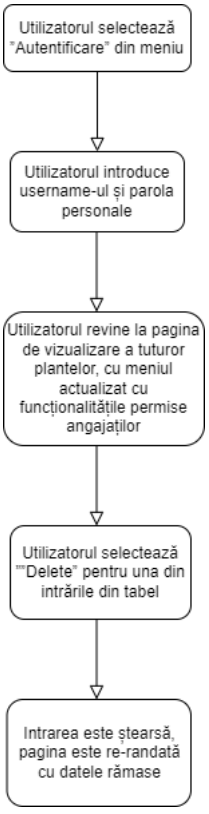

Figura 11: Diagrama de activitate - Ștergerea unei plante

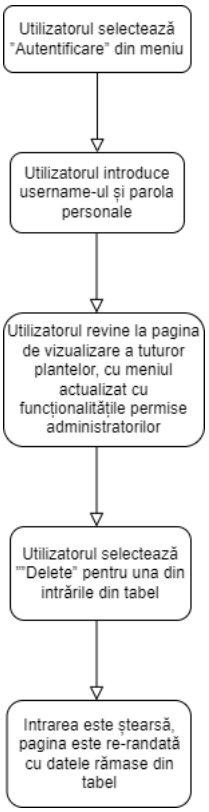

Figura 12: Diagrama de activitate - Ștergerea unui utilizator

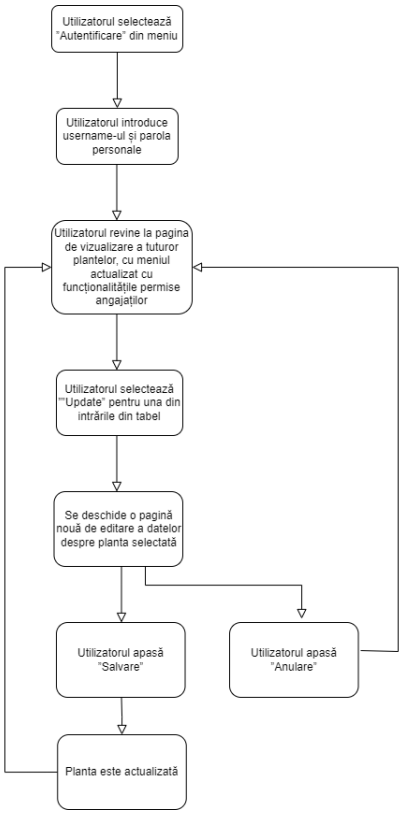

Figura 13: Diagrama de activitate - Actualizarea unei plante

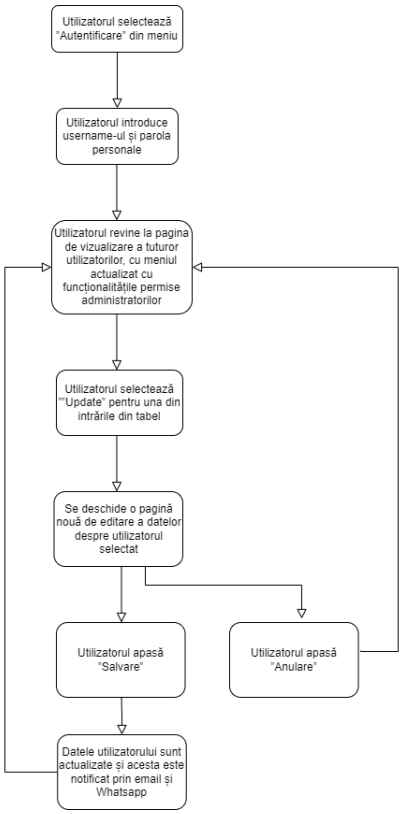

Figura 14: Diagrama de activitate - Actualizarea unui utilizator

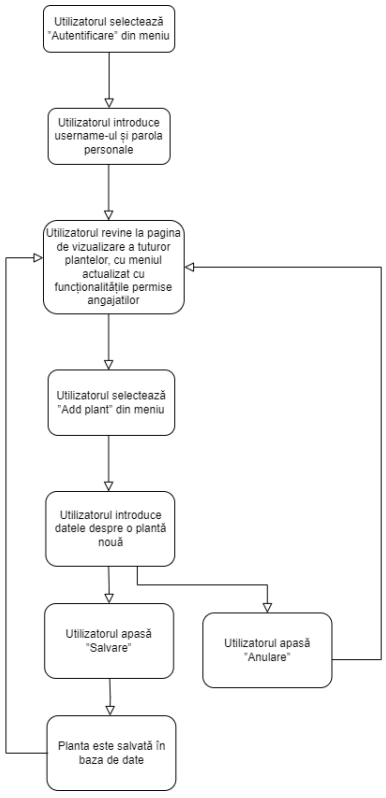

Figura 15: Diagrama de activitate - Adăugarea unei plante

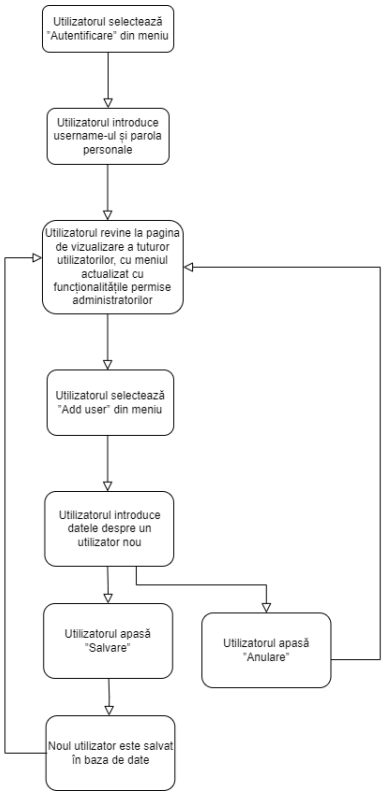

Figura 16: Diagrama de activitate - Adăugarea unui utilizator

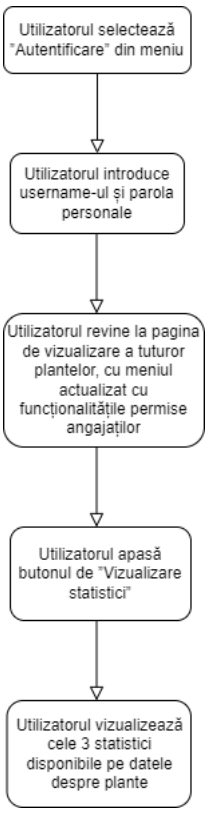

Figura 17: Diagrama de activitate - Vizualizarea statisticilor

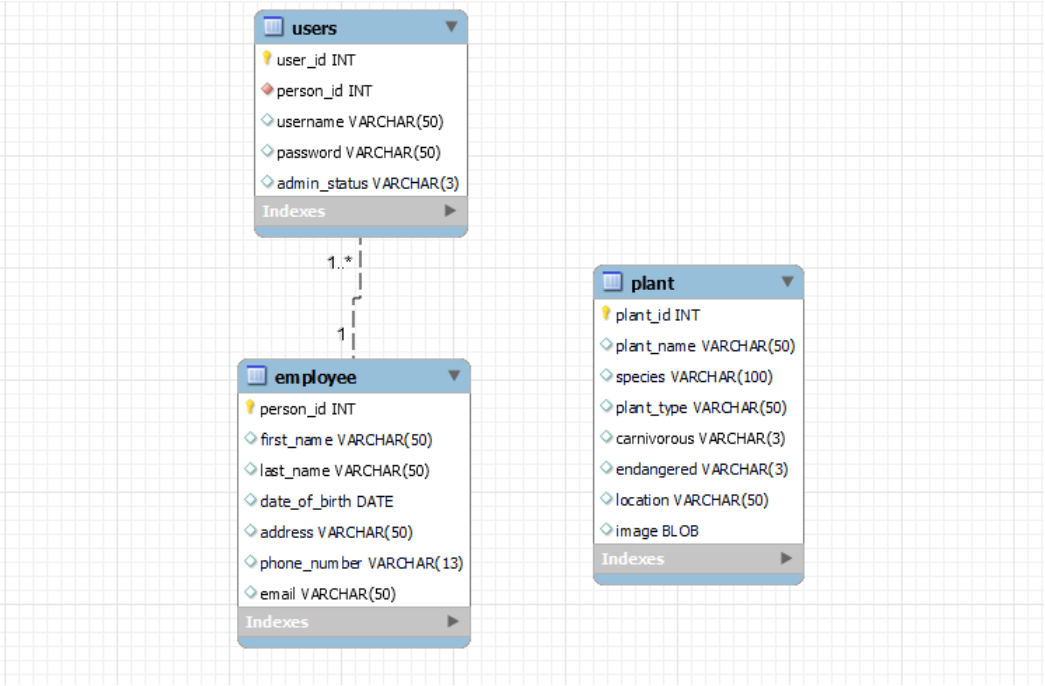

Figura 18: Diagrama entitate-relație

3 Faza de proiectare - Diagrame de clase

3.1 Structura bazei de date

Sistemul de gestionare a bazei de date folosit este MySQL, conectarea cu mediul Java realizându-se prin intermediul driver-ului jdbc.

Baza de date conține 3 tabele, și anume ”plant”, pentru persistența datelor despre plantele din grădina botanică cu toate atributele lor (denumire, specie, tip, localizare, tip de alimentație și imagine reprezentativă), ”employee”, pentru păstrarea și gestionarea datelor despre angajații din sistem (cu datele aferente: nume, prenume, data nașterii, adresă, număr de telefon, email) și ”users”, pentru datele fiecărui utilizator asociat unui angajat în parte (cu referință la persoana de care aparține, nume de utilizator, parolă și un atribut care determină rolul de administrator).

3.2 Aplicația Server

Din motive de vizibilitate, voi analiza în continuare pe rând fiecare pachet din diagrama completă:

3.2.1 Diagrama completă de clase

3.2.2 Pachetul Entity

Definirea claselor care modelează domeniul aplicației s-a efectuat în pachetul entity.

Pachetul entity conține clasele Plant, Employee și AbstractUser împreună cu clasele derivate User și UserProxy, fiecare cu propriii constructori pentru instanțele care vor corespunde intrărilor din tabelele respective din baza de date și vor modela o grădină botanică reală.

Figura 19: Diagrama de clase - aplicatia server

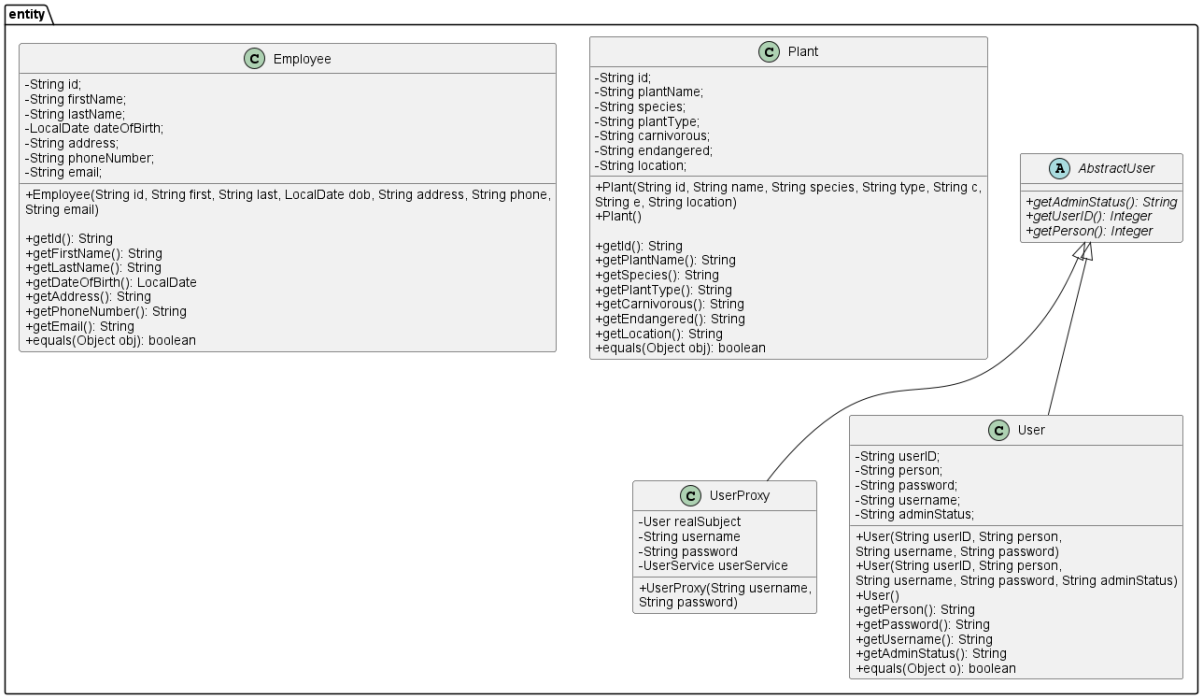

Figura 20: Diagrama de clase - pachetul entity

În afară de atribute, clasele din model au definite settere și gettere, folosite pentru popularea tabelelor din interfața grafică cu datele necesare funcționării acesteia și transmiterii obiectelor de la baza de date spre aplicația client și invers.

3.2.3 Șablonul Proxy

Clasele AbstractUser, User și UserProxy implementează șablonul structural Proxy, care presupune ușurarea fluxului de date transmise și primite controlând câmpurile obiectului transmis. Astfel, se folosește un UserProxy care include doar numele de utilizator și parola în partea inițială de rulare a aplicației și până are loc autentificarea, după care se folosește un User complet (cu datele personale ale angajatului autentificat) pentru a comunica mai departe cu client-ul.

3.2.4 Pachetul Repository

Pachetul repository din cadrul aplicației server are rolul de a crea baza de date si de a gestiona operatiile CRUD care se pot efectua pe obiectele din aplicatie.

Clasele și interfețele pe care le include acest pachet sunt: CreateDatabase (care construiește baza de date și tabelele), Repository (care gestionează legătura cu baza de date, deschizând și închizând conexiunea la cerere), PlantRepository prin intermediul căreia se realizează operațiile CRUD pentru tabelul ”plant”, UserRepository pentru operațiile CRUD caracteristice tabelului ”users” și EmployeeRepository pentru operațiile CRUD pentru tabelul ”employee”. Ultimele 3 reprezintă interfețe care moștenesc din CRUDRepository<T, U>, și nu clase, dat fiind că aplicația server folosește framework-ul Spring, iar implementările operațiilor precum save(), update(), delete() sau find() sunt adăugate automat acolo unde este nevoie de Repository, prin folosirea anotării ”@Autowired”.

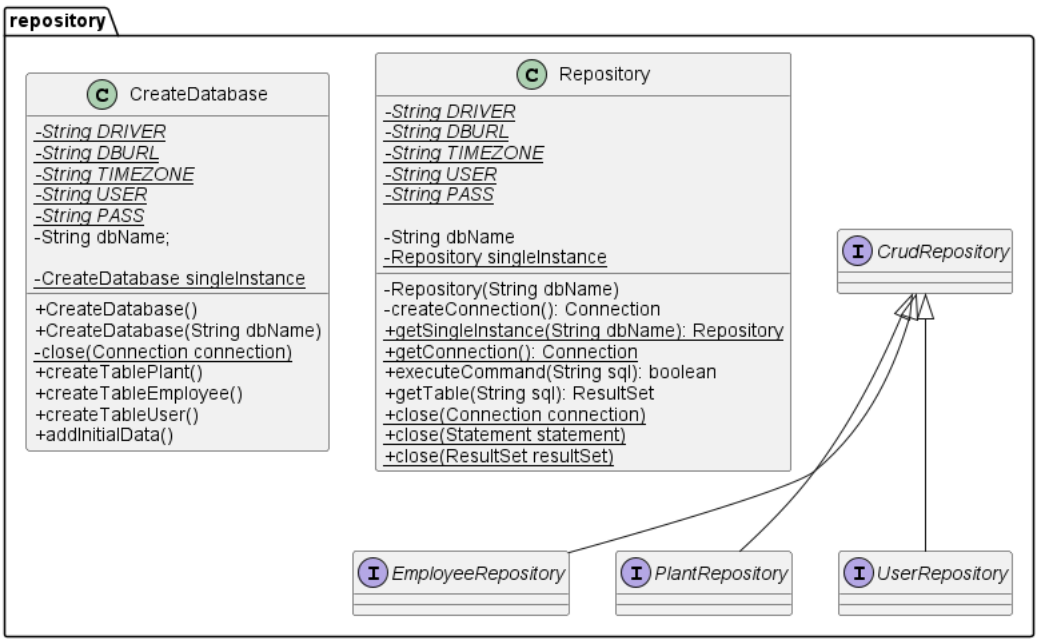

Figura 21: Diagrama de clase - pachetul repository

3.2.5 Șablonul Singleton

Clasa Repository, responsabilă de gestiunea conexiunii cu baza de date, implementează șablonul creațional Singleton, ceea ce presupune existența unei singure instanțe unice de conexiune, care poate fi folosită prin apelul static Repository.getSingleInstance(dbName). Constructorul clasei este privat, accesul la acesta făcându-se doar o singură dată, la primul apel al metodei statice publice de get.

3.2.6 Pachetul Service

Pachetul service conține clasele necesare implementării tuturor operațiilor care au loc în spatele aplicației, pe server și vor fi apelate de către aplicația client prin intermediul comenzilor HTTP definite în pachetul controller.

Pachetul conține clasele PlantService, EmployeeService, UserService pentru implementarea operațiilor pe tabelele bazei de date, dar și clasele funcționale EmailService, WhatsappService, interfața DocumentFactory (cu cele 4 clase derivate: XMLFactory, CSVFactory, DOCFactory și JSONFactory), interfața DocumentWriter (cu cele 4 clase derivate: XMLWriter, CSVWriter, JSONWriter și DOCWriter), necesare pentru a trimite notificări din partea administratorilor legate de modificarea datelor de autentificare ale unui utilizator și scrierea listei de plante într-un fișier care să poată fi apoi descărcat din aplicația client prin browser în unul din cele 4 formate. Atât EmailService, cât și WhatsappService folosesc API-uri și dependențe disponibile gratuit pe internet pentru a trimite mesaje și necesită să se cunoască atât destinația, cât și o adresă/număr de telefon de la care se trimite mesajul.

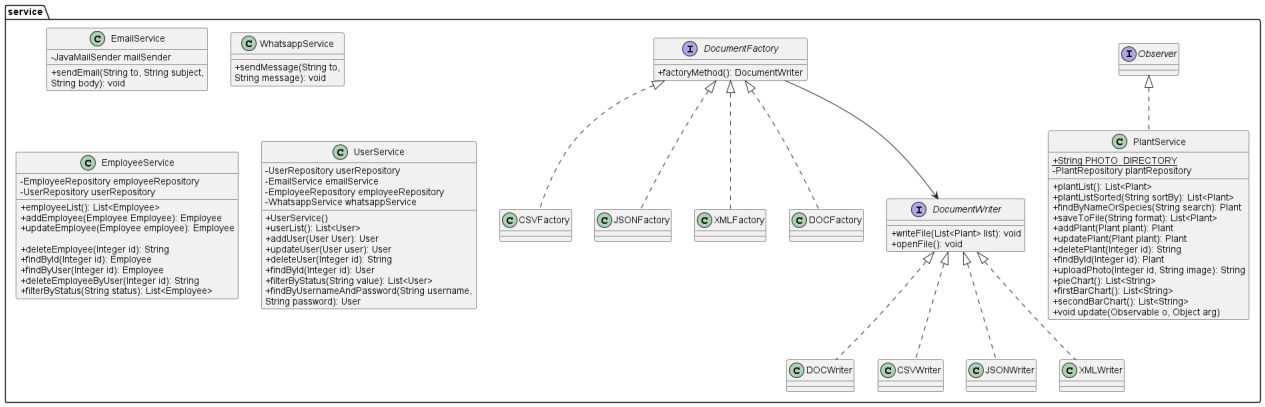

Figura 22: Diagrama de clase - pachetul service

3.2.7 Șablonul FactoryMethod

Interfața DocumentFactory și cele 4 clase derivate, împreună cu interfața DocumentWriter și cele 4 clase derivate, implementează împreună șablonul creațional FactoryMethod. DocumentFactory reprezintă partea de Factory, care creează obiecte noi de tipul produsului (DocumentWriter), iar DocumentWriter - partea de produs, cu metodele care definesc logica necesară scrierii în fișier (metoda writeFile() și openFile()).

3.2.8 Pachetul Controller

Pachetul controller este cel responsabil de definirea comenzilor HTTP prin care se realizează comunicarea dintre client și server. Pachetul constă din clasele LangController (care returnează fișierul cu traducerile pentru limba selectată în frontend), PlantController, EmployeeController și UserController (care se ocupă cu comenzile specifice fiecărui tabel din baza de date și respectiv fiecărei clase din domeniul aplicației).

Comenzile folosite sunt de tipul GET, POST, PUT și UPDATE și la apelarea acestora are loc apelul funcțiilor deja definite în pachetul service.

3.2.9 Șablonul Observer

Aplicația folosește șablonul Observer pentru a permite o interacțiune mai rapidă și mai eficientă între pachete, fără cuplare excesivă. Clasa PlantController funcționează ca clasă observabilă, care notifică clasa PlantService atunci când are loc o schimbare din partea aplicației client legată de formatul fișierului în care ar trebui să aibă loc scrierea listei de plante. Clasa PlantService este așadar un Observer, care acționează care un trigger pe FactoryMethod-ul definit prin DocumentFactory prin metoda update() și declanșează scrierea în fișierul cu formatul cerut.

Așadar, clasele responsabile de scrierea în fișier și pachetul controller sunt decuplate, dar funcționalitatea dorită rămâne disponibilă.

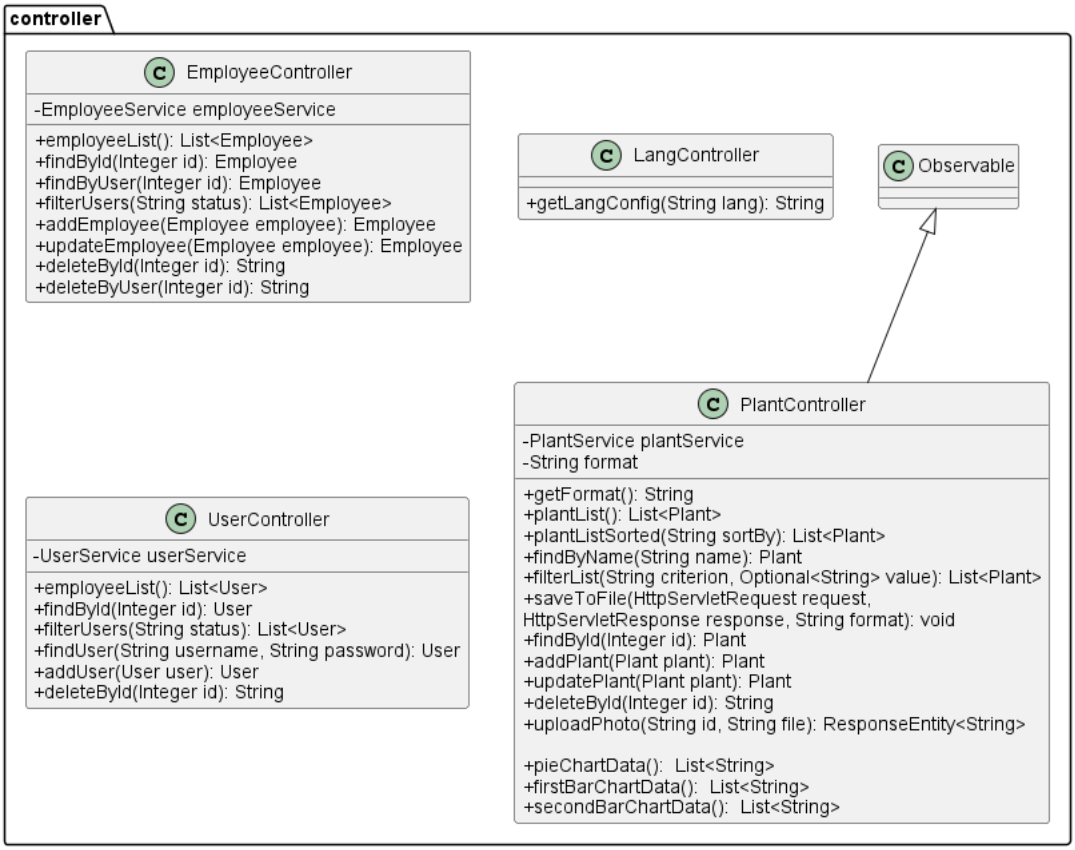

Figura 23: Diagrama de clase - pachetul controller

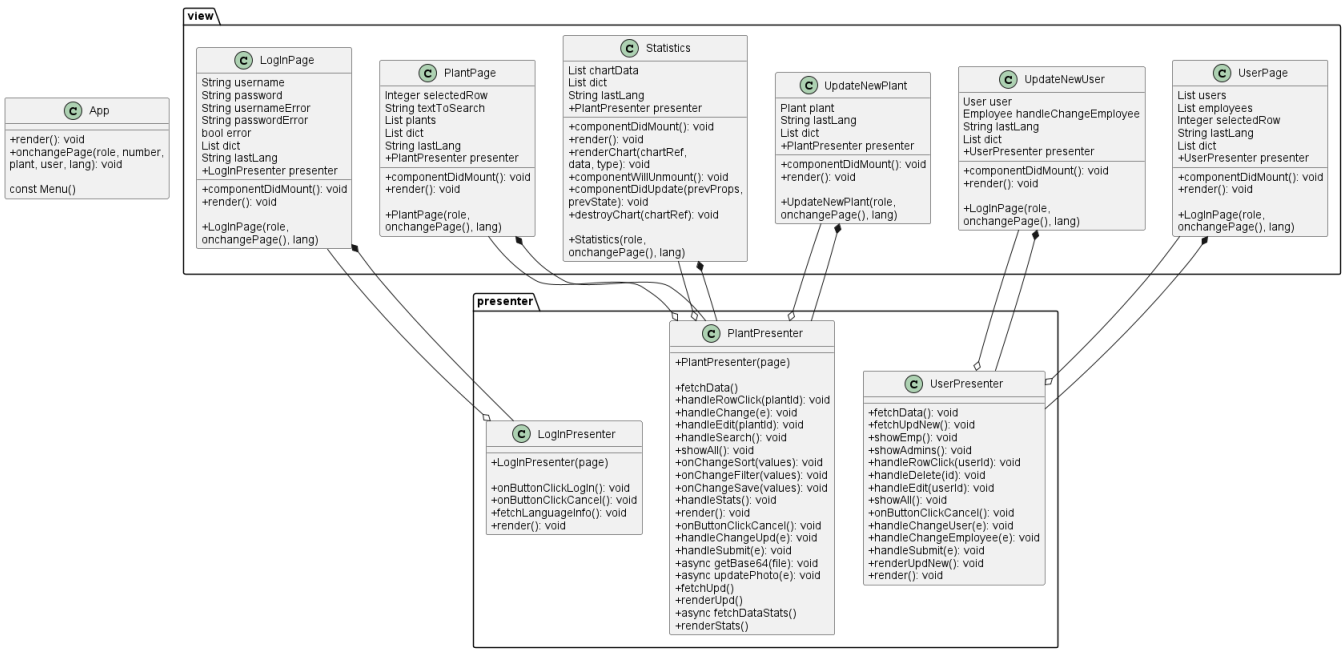

Figura 24: Diagrama de clase - aplicatia client

3.3 Aplicația Client

Aplicația client conține doar 3 pachete: view, presenter și style, ultimul doar cu rol de stilizare a aplicației, folosind fișiere .css. Pachetele view și presenter sunt cele care construiesc de fapt interfața grafică utilizator și definesc funcționalități pentru fiecare buton sau element din meniu al acesteia.

3.3.1 Pachetul View

Pachetul view conține toate componentele grafice care se randează în aplicația web, și anume clasele PlantPage, LogInPage, UserPage, Statistics, UpdateNewUser și UpdateNewPlant. Aceste clase populează câmpurile textuale cu date primite prin Axios de la aplicația server și le păstrează în starea lor, reprezentată prin liste sau obiecte JSON.

3.3.2 Pachetul Presenter

Pachetul presenter conține toate funcționalitățile aplicației client și procesarea evenimentelor care apar în timpul rulării, precum apăsarea unui buton, selectarea unei opțiuni dintr-un meniu sau introducerea unui text.

Astfel, clasa LogInPresenter include funcțiile necesare paginii LogInPage, PlantPresenter - funcțiile necesare paginilor PlantPage, Statistics și UpdateNewPlant, iar UserPresenter - funcțiile necesare paginilor UserPage și UpdateNewUser.

3.3.3 Clasa App

Dat fiind că aplicația client este implementată în JavaScript, entry point-ul aplicației îl reprezintă clasa App din fișierul "App.js". Această clasă decide ce pagină se va afișa conform cu o variabilă "currentPage" și folosește funcția onChangePage() pe care o transmite și celorlalte componente ale aplicației pentru a modifica starea curentă și a afișa corect elementele paginii web.

3.3.4 Șablonul Command

Comunicarea dintre aplicația server și aplicația client este efectuată prin protocolul HTTP și șablonul comportamental Command, folosindu-se comenzi de tip GET, POST, PUT și DELETE pentru a apela funcții din backend care să producă modificări în frontend.

Șablonul Command permite transmiterea cererilor de la client la server sub forma unui obiect HTTP în loc să facă un apel direct al unei funcții.

3.4 Diagrame de secvență

Diagramele de secvență sunt cele care redau comunicarea dintre toate componentele aplicației prin apeluri de metode și mesaje. Fiecare use-case poate avea definită o diagramă de secvență proprie.

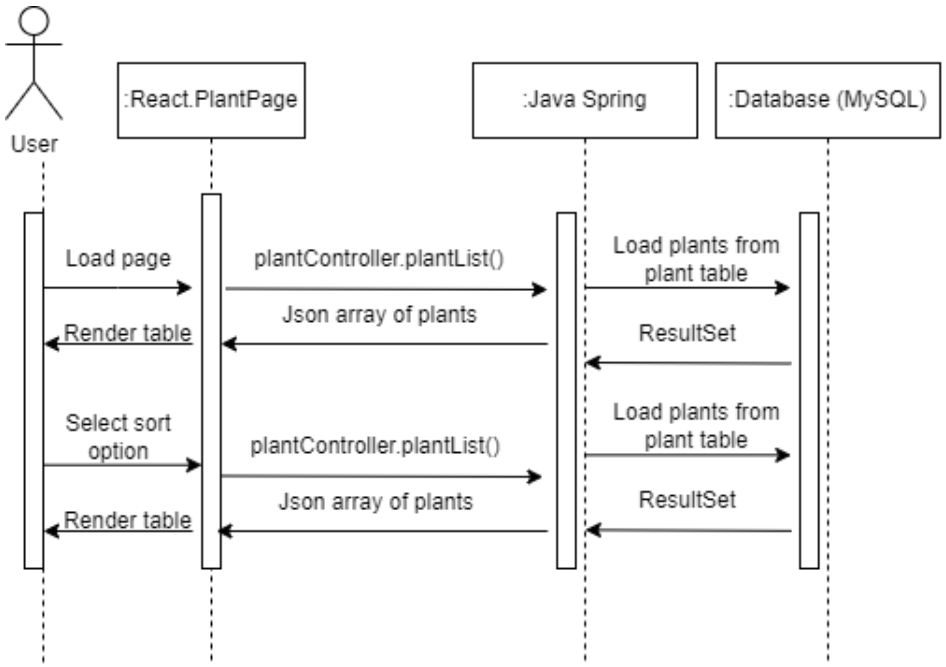

Figura 25: Diagrama de secventa - Vizualizarea plantelor

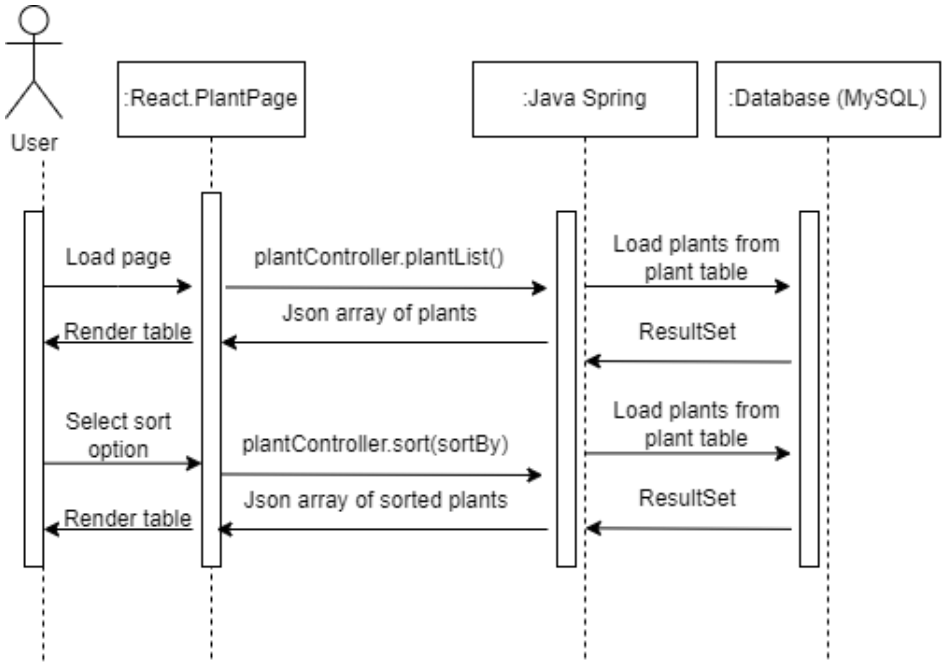

Figura 26: Diagrama de secventa - Vizualizarea plantelor sortate

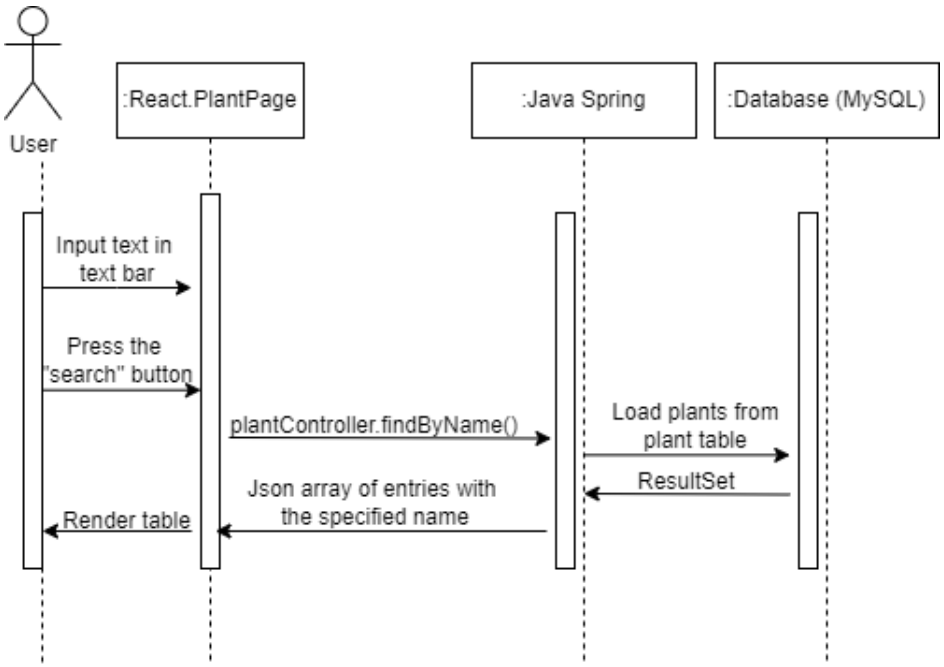

Figura 27: Diagrama de secventa - Cautarea unei plante dupa denumire

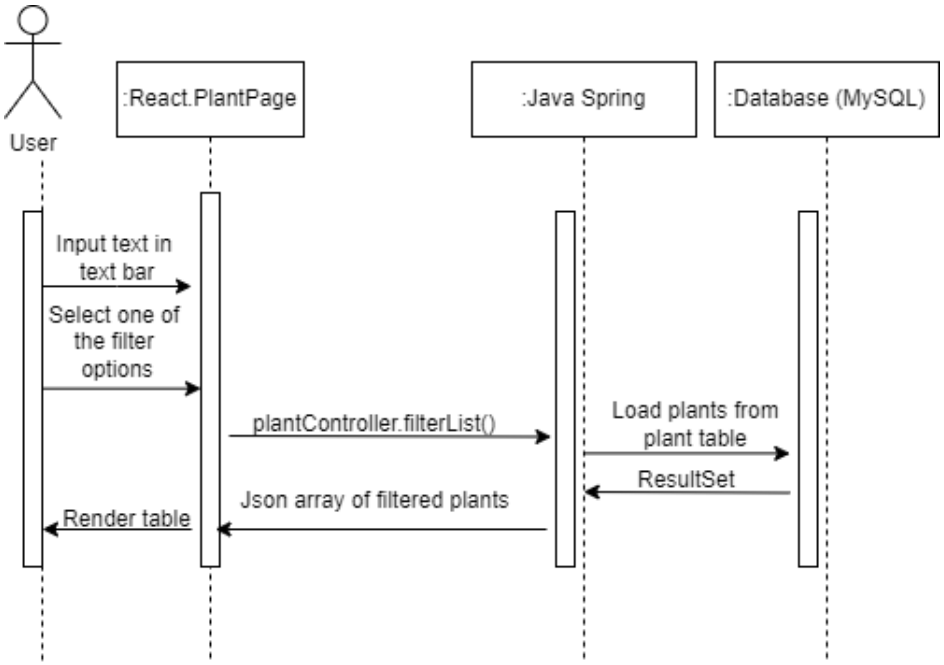

Figura 28: Diagrama de secventa - Filtrarea listei de plante

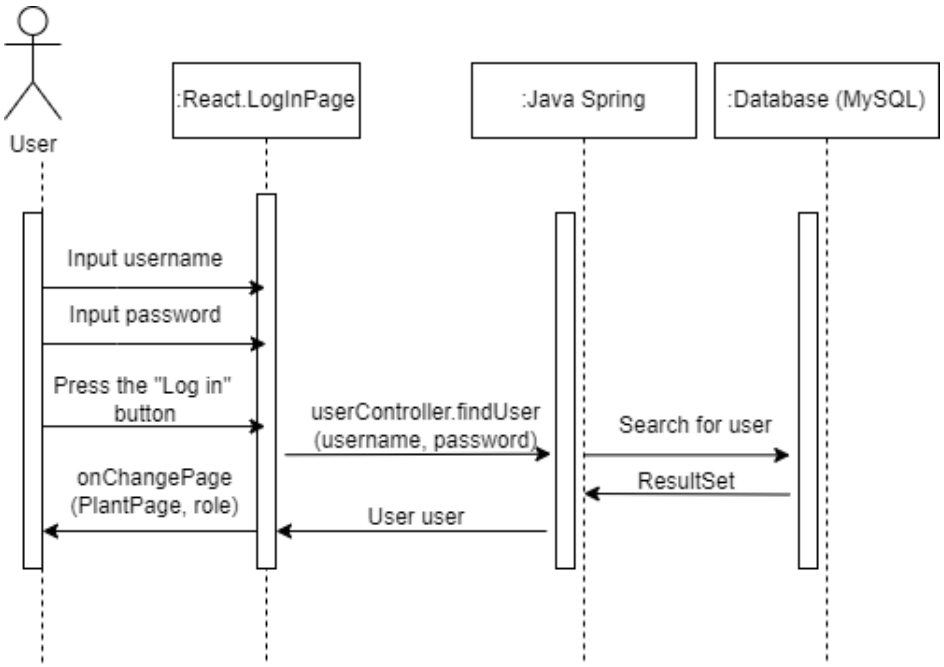

Figura 29: Diagrama de secventa - Autentificare eronata

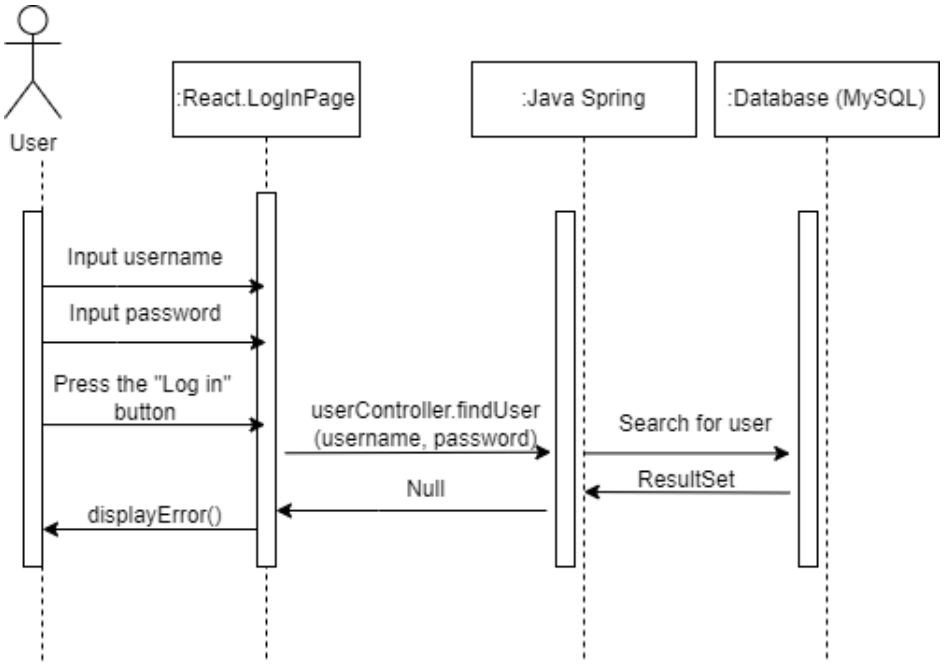

Figura 30: Diagrama de secventa - Autentificare cu succes

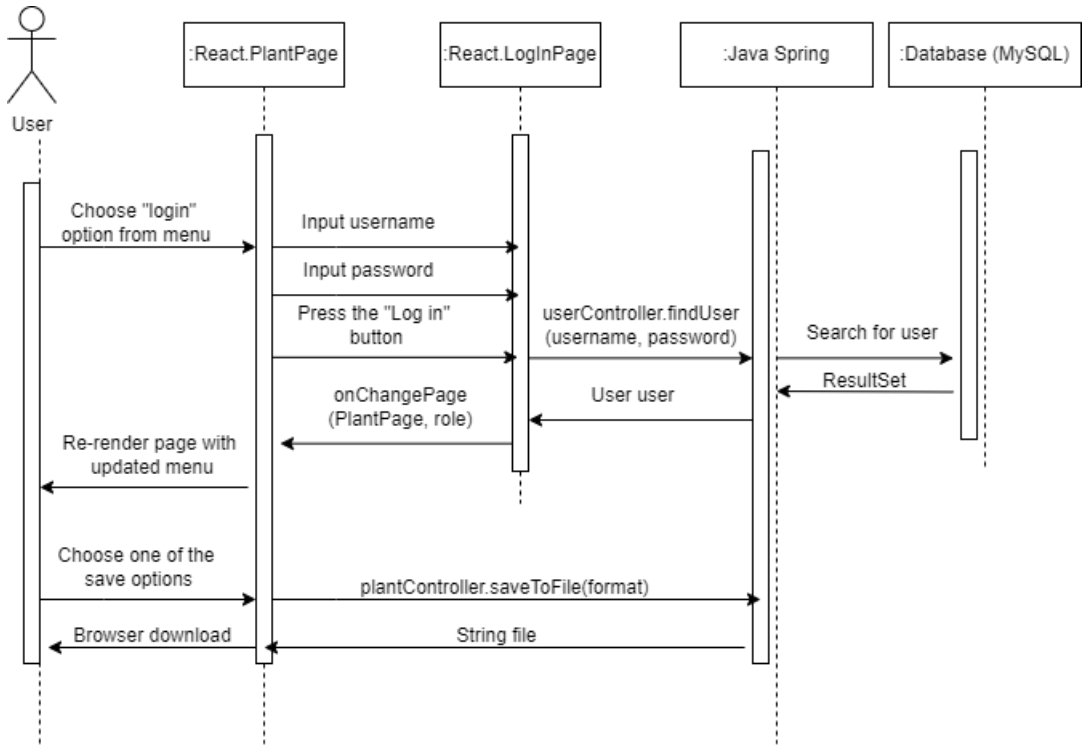

Figura 31: Diagrama de secventa - Salvarea listei de plante intr-un fisier

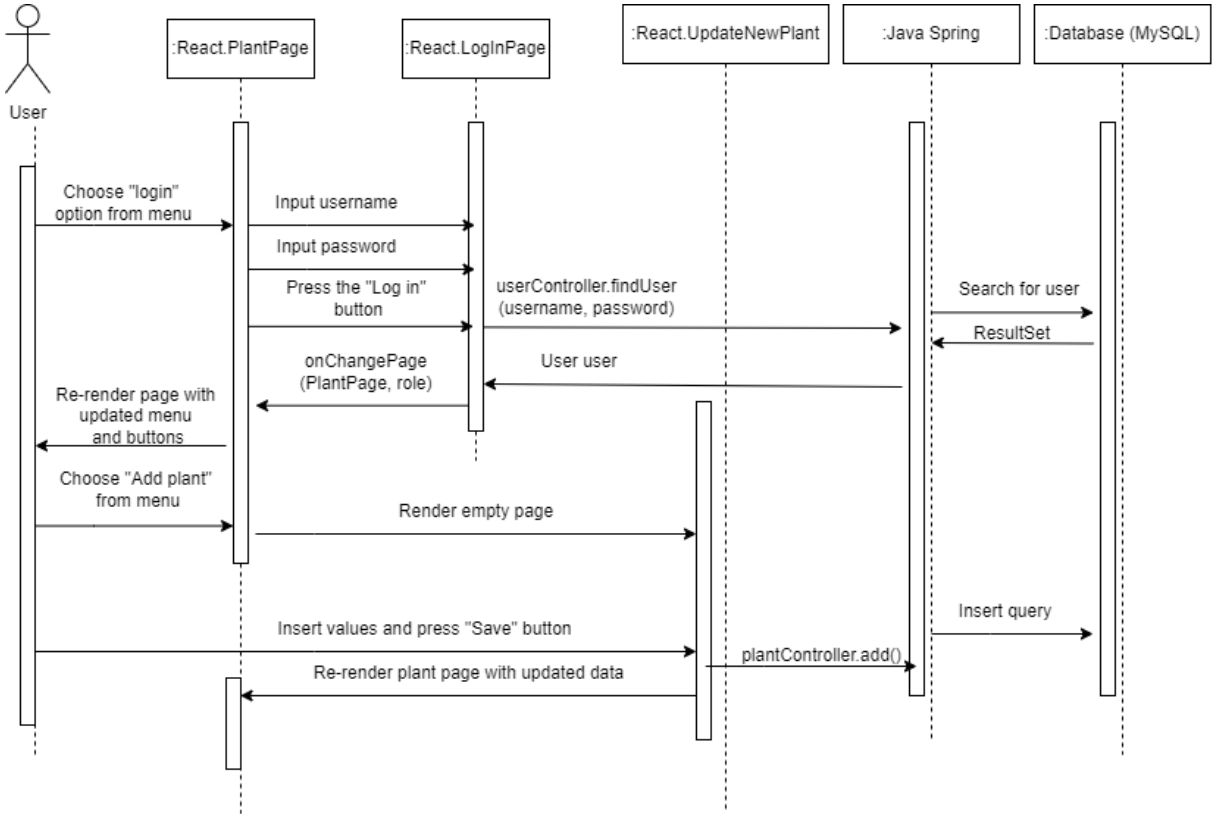

Figura 32: Diagrama de secventa - Crearea unei plante noi

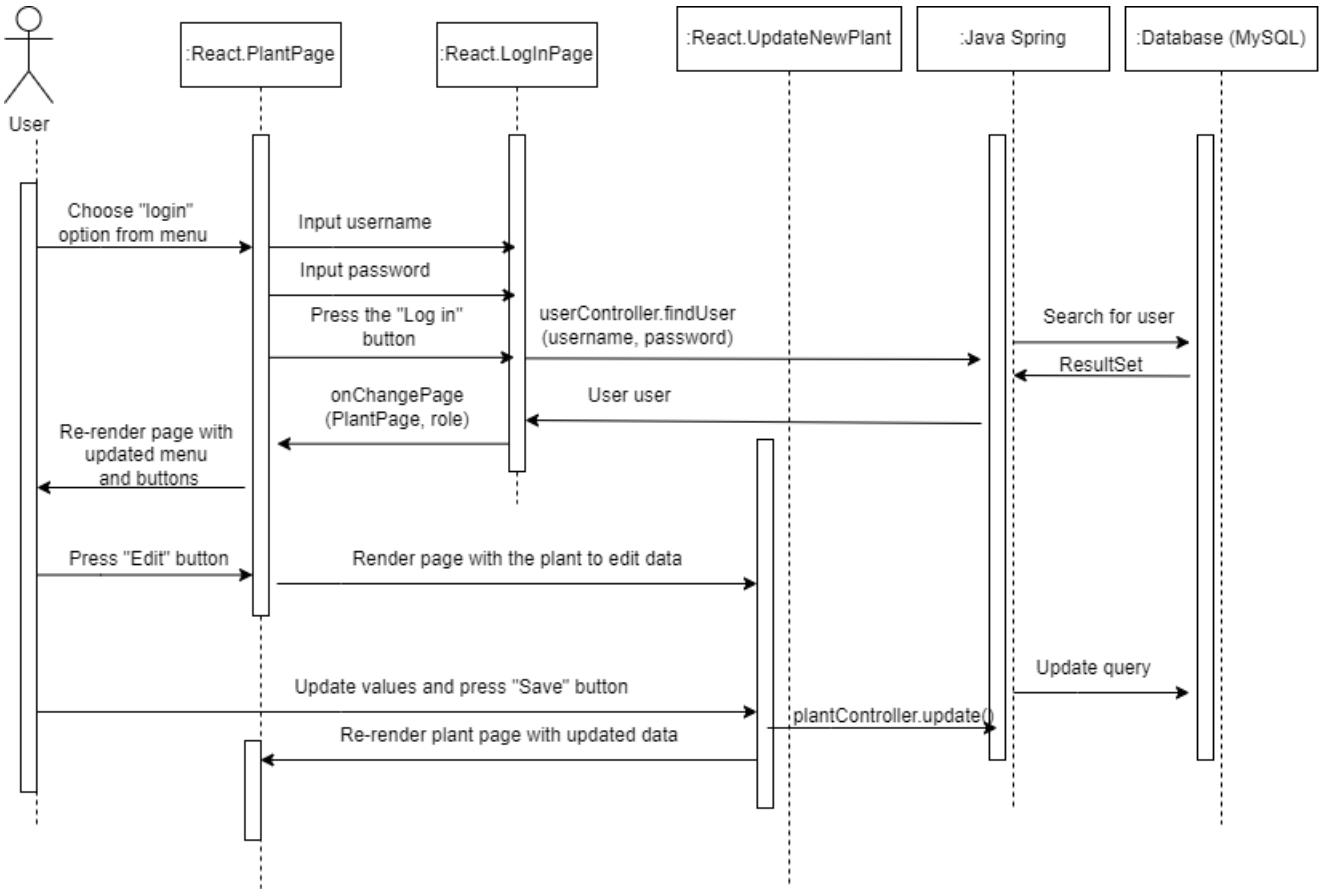

Figura 33: Diagrama de secventa - Actualizarea unei plante

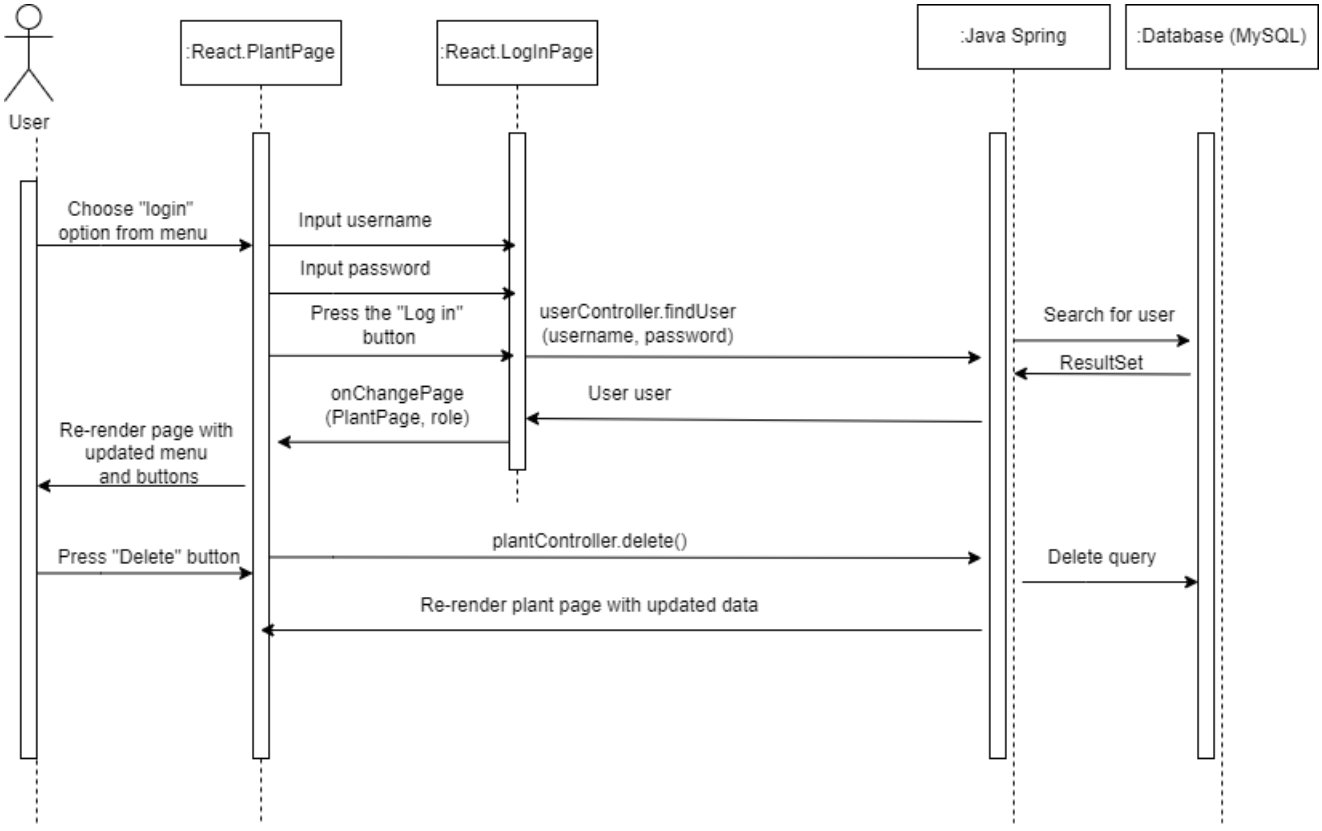

Figura 34: Diagrama de secventa - Stergerea unei plante

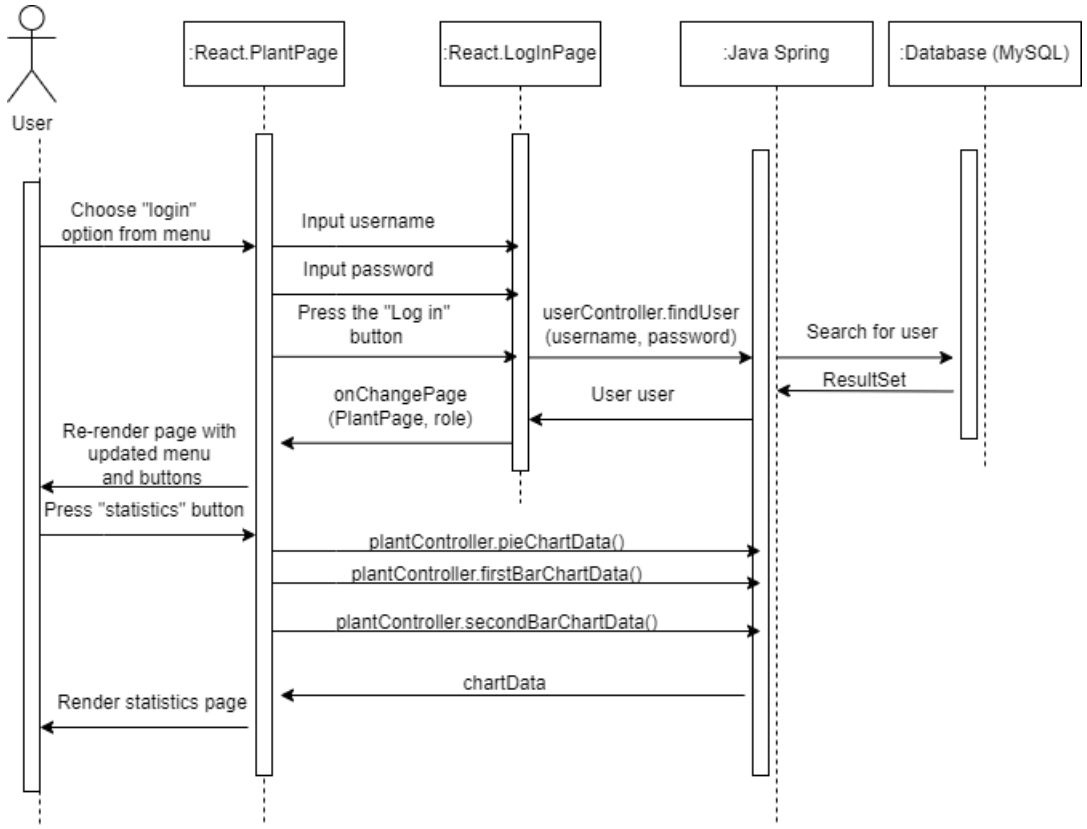

Figura 35: Diagrama de secventa - Vizualizarea statisticilor

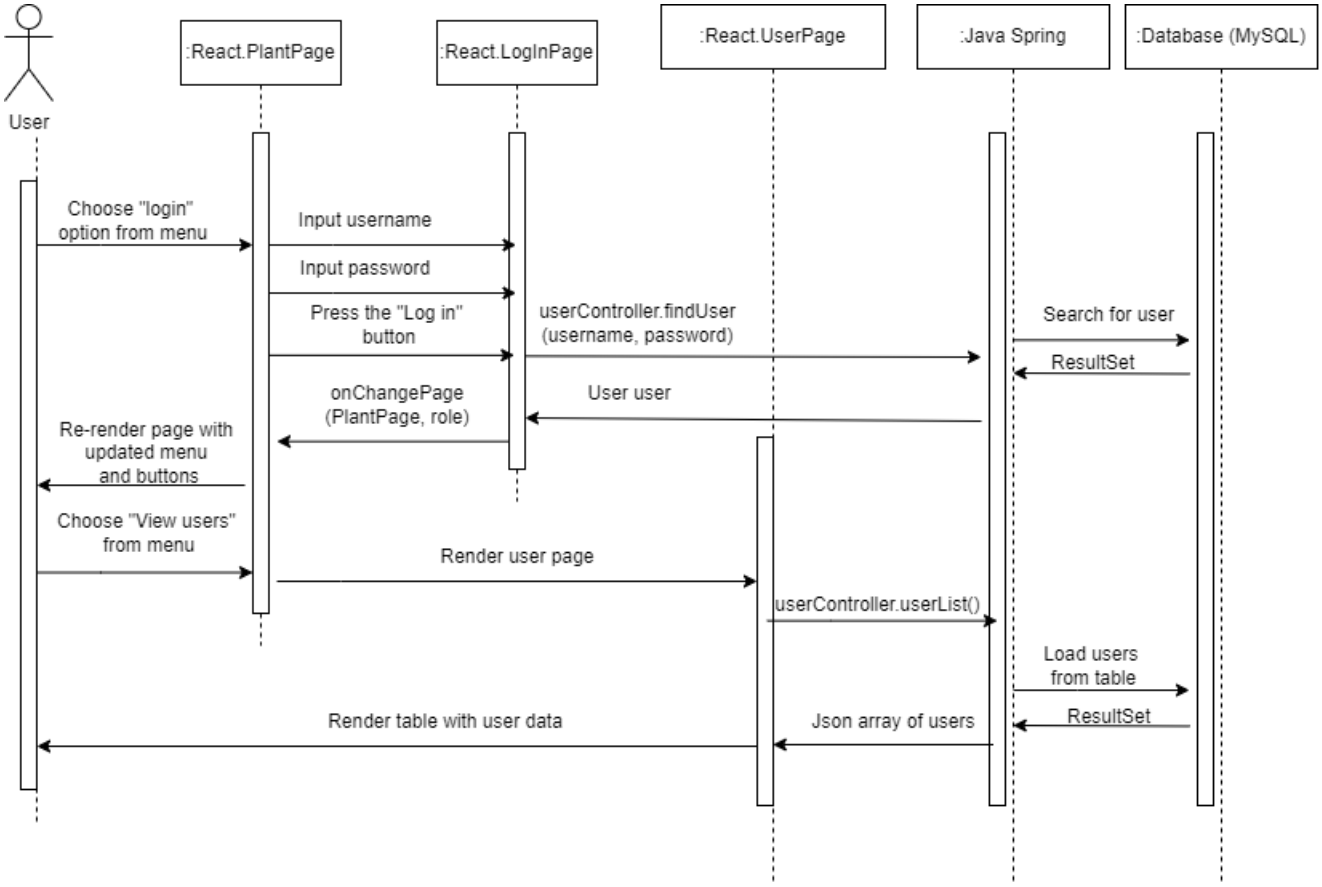

Figura 36: Diagrama de secventa - Vizualizarea utilizatorilor

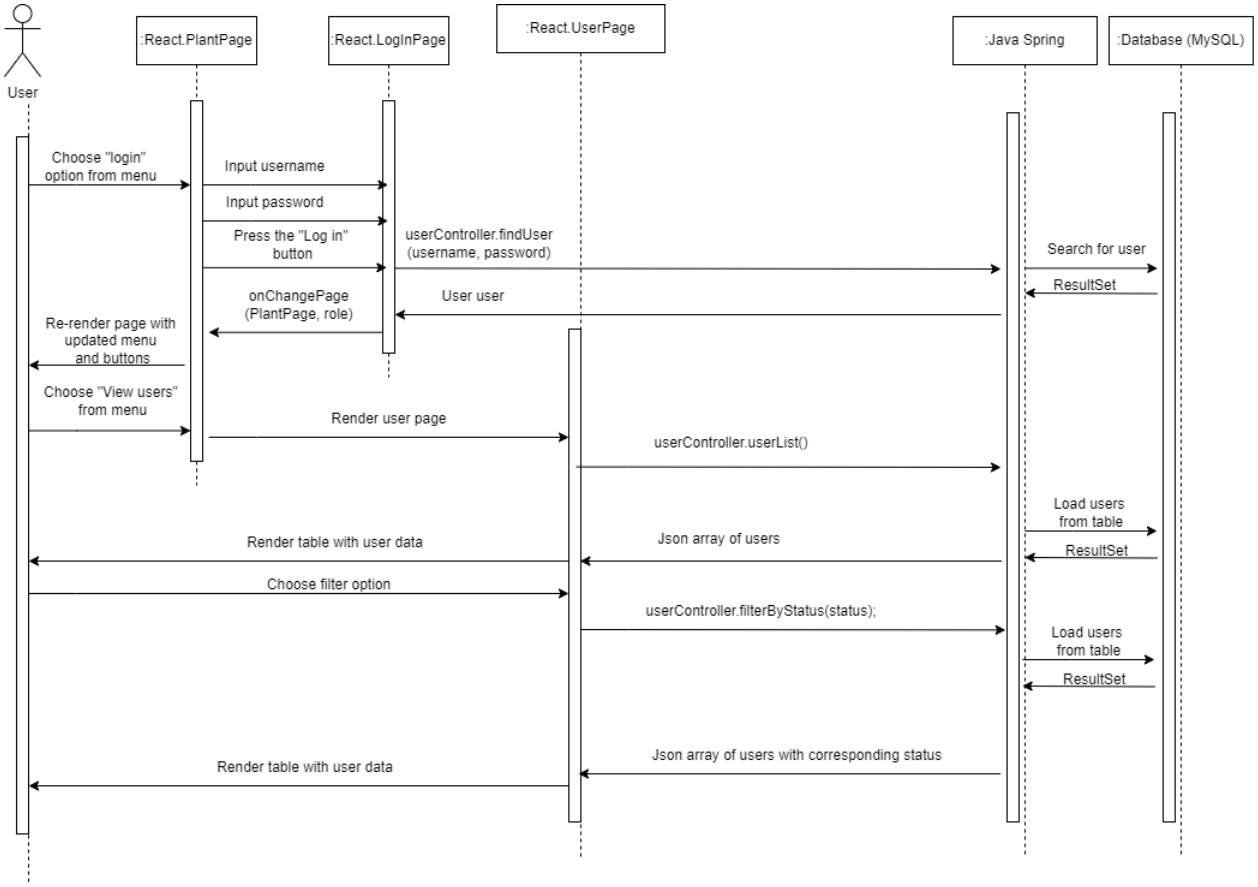

Figura 37: Diagrama de secventa - Filtrarea listei de utilizatori

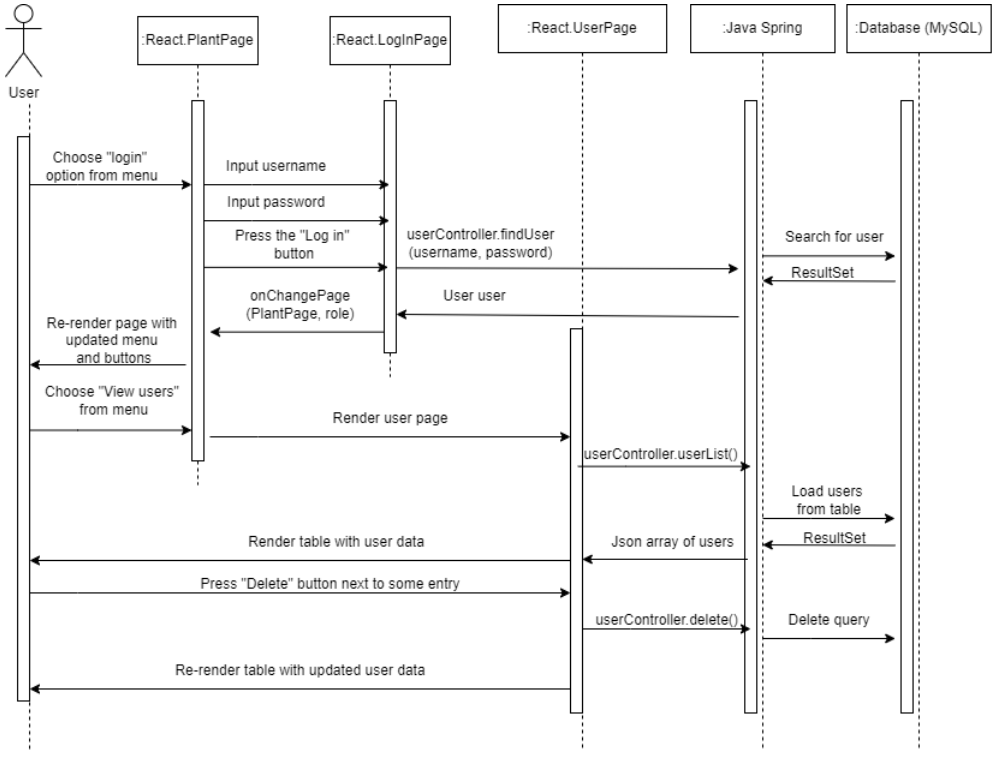

Figura 38: Diagrama de secventa - Stergerea unui utilizator

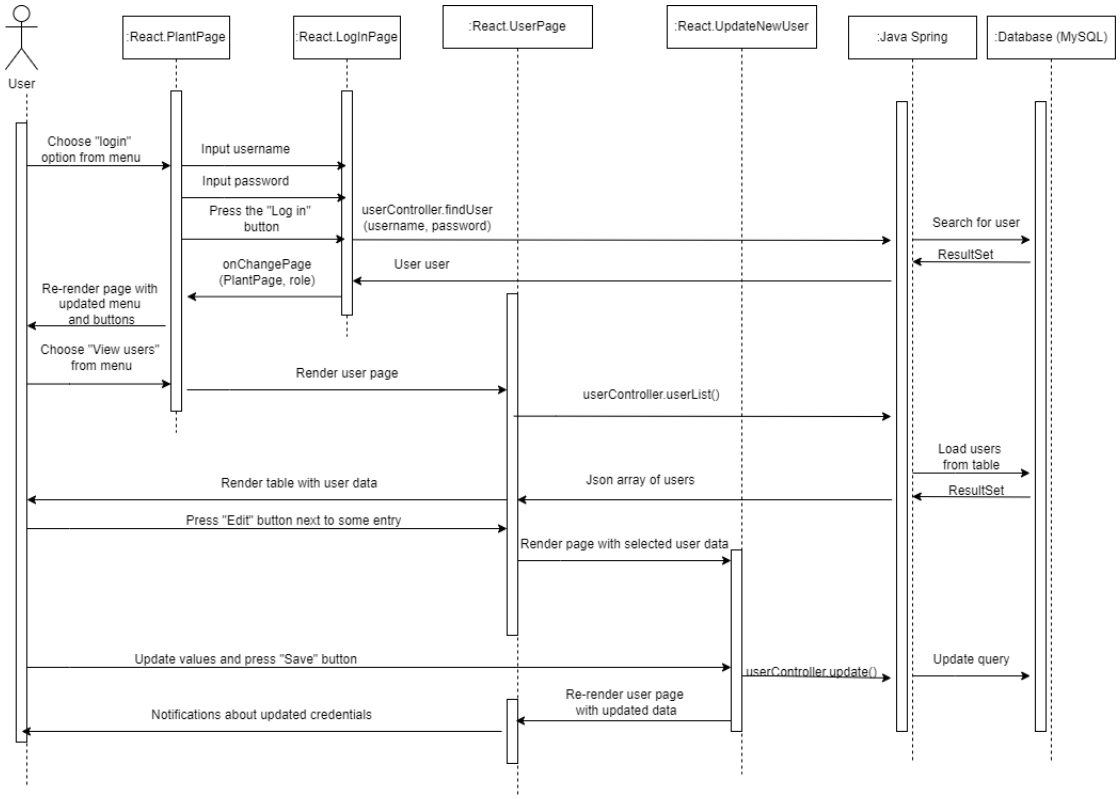

Figura 39: Diagrama de secventa - Actualizarea unui utilizator

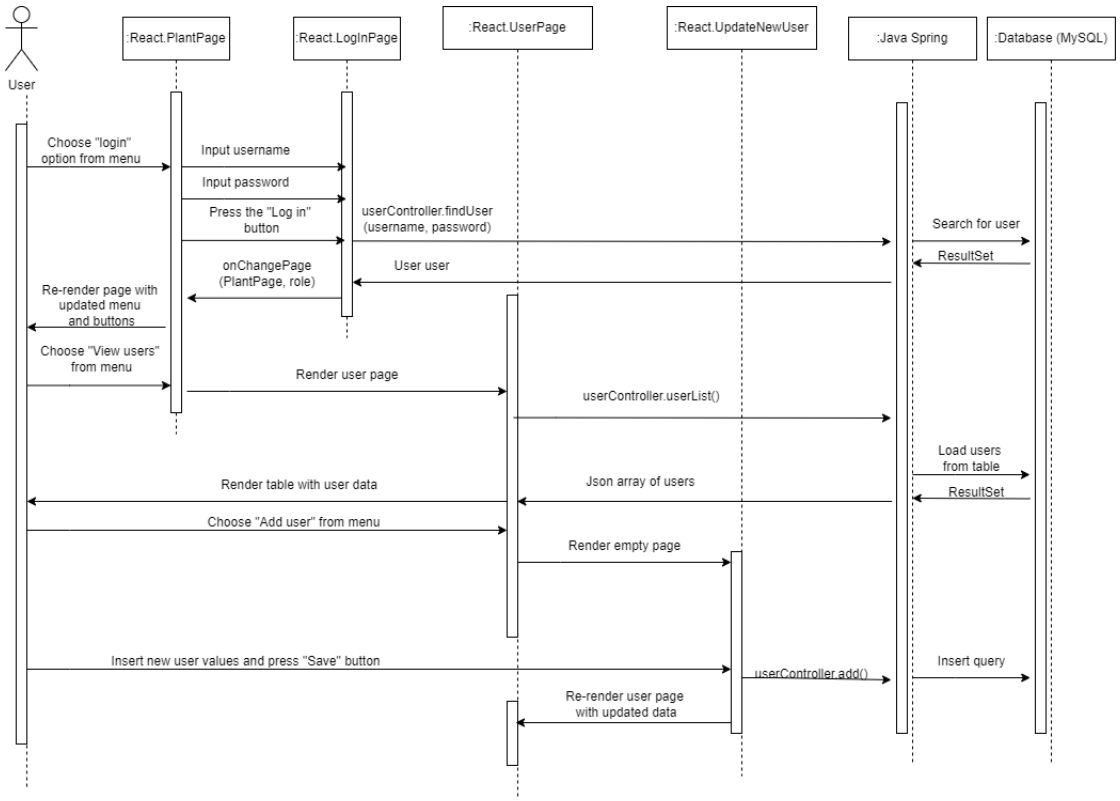

Figura 40: Diagrama de secventa - Crearea unui utilizator nou

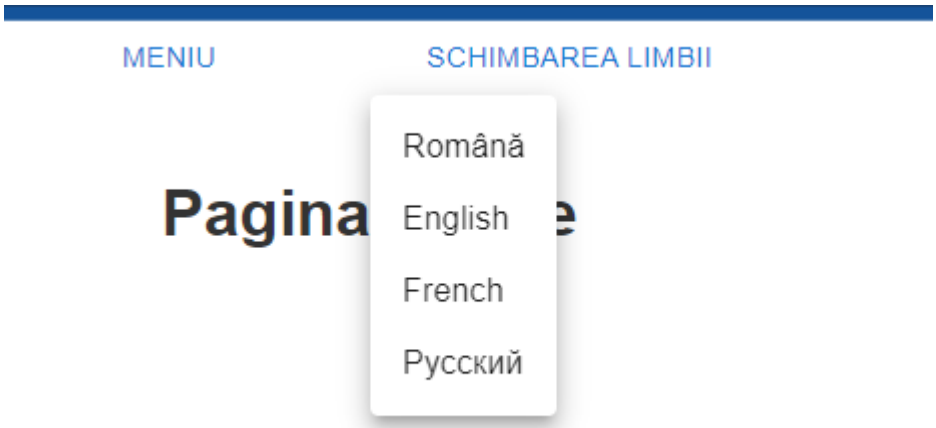

Figura 41: Meniul de limbi

4 Faza de implementare - Aplicatia

4.1 Instrumentele utilizate

Pentru crearea bazei de date am utilizat sistemul de gestionare de baze de date relaționale MySQL, cu sintaxa SQL clasică și driver-ul jdbc pentru a realiza conexiunea din Java.

Partea de cod Java pentru aplicația server am implementat-o folosind IDE-ul IntelliJ Idea, care oferă un mediu de dezvoltare intuitiv și rapid.

Am ales limbajul Java, deoarece este orientat pe obiect, ceea ce este esențial pentru a modela utilizatorii și plantele din baza de date.

Partea de cod pentru aplicația client am implementat-o folosind limbajul JavaScript și IDE-ul WebStorms, care sunt cele mai puternice instrumente pentru construirea unei interfețe grafice web funcționale și estetice.

4.2 Aplicatia

La deschiderea aplicației, utilizatorului (care încă nu are niciun rol asignat, respectiv primește eticheta de ”visitor”) îi apare pagina de prezentare a tuturor plantelor din baza de date, într-un tabel care are ca coloane atributele fiecărei intrări din baza de date: id-ul, denumirea, specia, tipul, dacă este carnivoră sau nu, dacă este pe cale de dispariție, o imagine reprezentativă și zona din grădina botanică unde se găsește.

În partea de sus a paginii există 2 meniuri, unul din care vor fi disponibile funcționalitățile permise utilizatorului curent și unul din care se poate alege limba: română, engleză, franceză sau rusă.

Pagina conține și opțiuni de sortare (după specie și după tip), opțiuni de filtrare conform cu textul introdus în bara de căutare (după tip, specie, locație etc.) și un buton de căutare care permite returnarea plantei cu denumirea sau specia indicată în bara de căutare.

Dacă utilizatorul se va autentifica conform cu username-ul și parola proprie, pe lângă datele despre plante de pe fiecare rând, va apărea o coloană nouă cu 2 butoane: ”Ștergere” și ”Editare”.

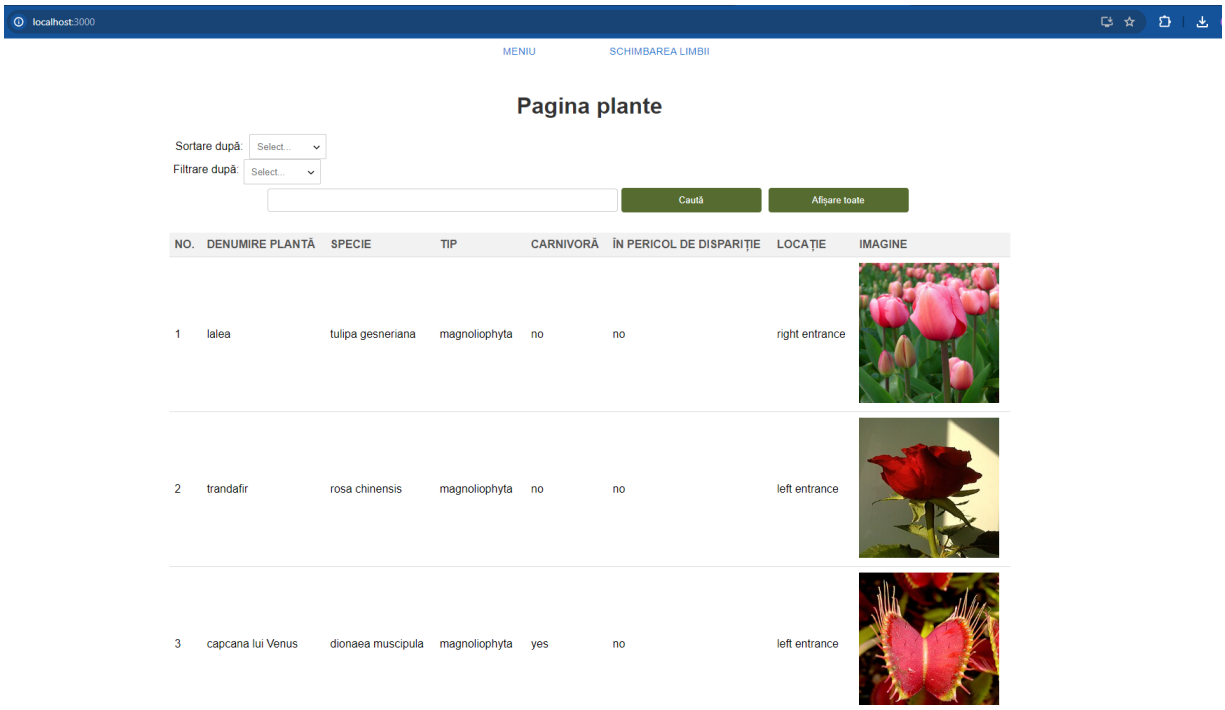

Figura 42: Pagina de vizualizare a plantelor

Dacă se apasă primul buton, intrarea selectată va fi ștearsă din tabel, iar dacă se apasă al doilea, utilizatorului i se va deschide o pagină nouă unde se vor putea modifica datele plantei.

De asemenea, după autentificare, pe pagina principală apare un nou meniu pentru salvarea listei de plante în diverse formate: .csv, .json, .xml și .doc, și un buton pentru vizualizarea statisticilor despre plantele din baza de date într-o pagină separată.

Pagina de editare și cea de creare sunt asemănătoare, permițând utilizatorului să introducă date noi peste cele existente sau ca intrare nouă, dar și să aleagă o imagine de pe dispozitiv în caz de adăugare.

De pe prima pagină, din meniul aplicației, un vizitator poate intra pe pagina de autentificare, unde își va introduce numele de utilizator și parola pentru a accesa funcționalități specifice rolului său în sistem.

Atunci când se detectează apăsarea butonului "Log in", se declanșează verificarea credențialelor. Se caută utilizatorul în baza de date, iar dacă acesta nu există sau a indicat parola greșită interfața generează un mesaj de eroare. În caz că autentificarea a avut loc cu succes, se verifică atributul de "admin_status" al utilizatorului și se identifică rolul acestuia: "employee" sau "administrator".

Se deschide iarăși pagina de prezentare a plantelor din aplicație, dar de data aceasta meniul este actualizat cu funcțiile specifice rolului utilizatorului autentificat.

Dacă utilizatorul este un angajat fără statut de administrator, apăsarea butonului "statistics" din pagina plantelor va determina deschiderea unei pagini cu 3 grafice:

1. O statistică despre numărul plantelor de fiecare tip;
2. O statistică despre numărul plantelor după tipul de alimentație;

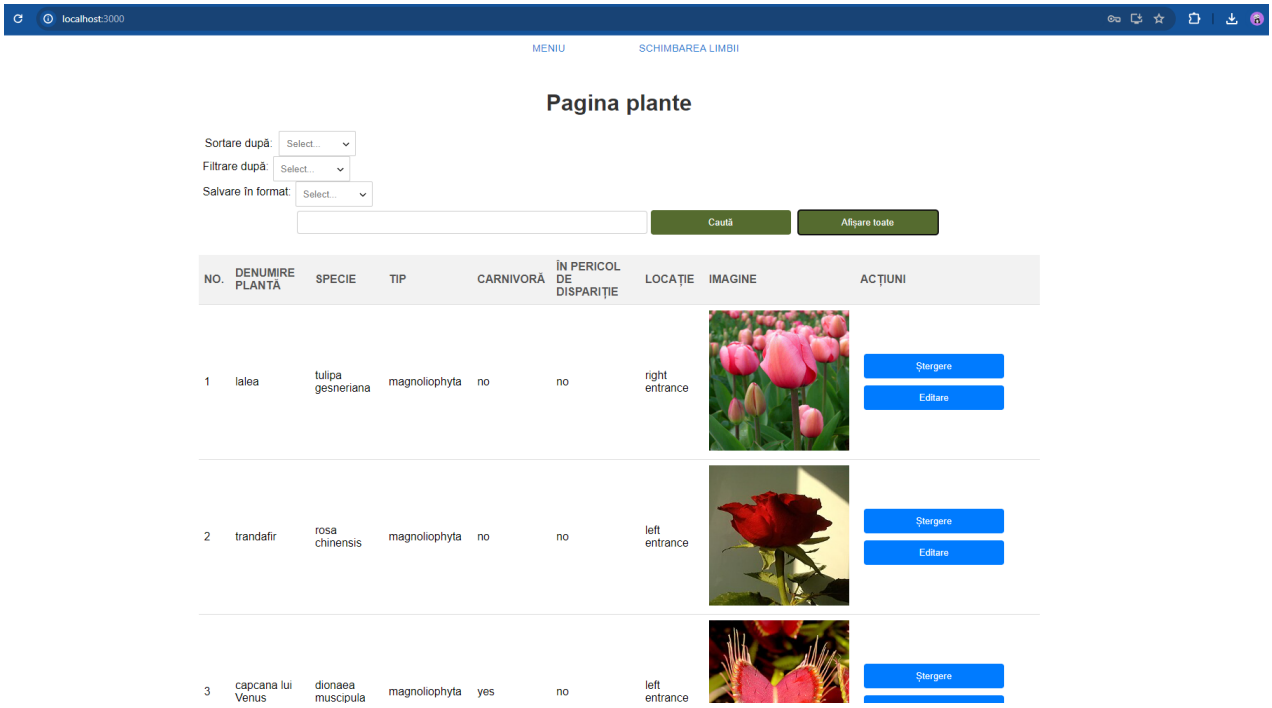

Figura 43: Pagina de vizualizare a plantelor (rol: angajat)

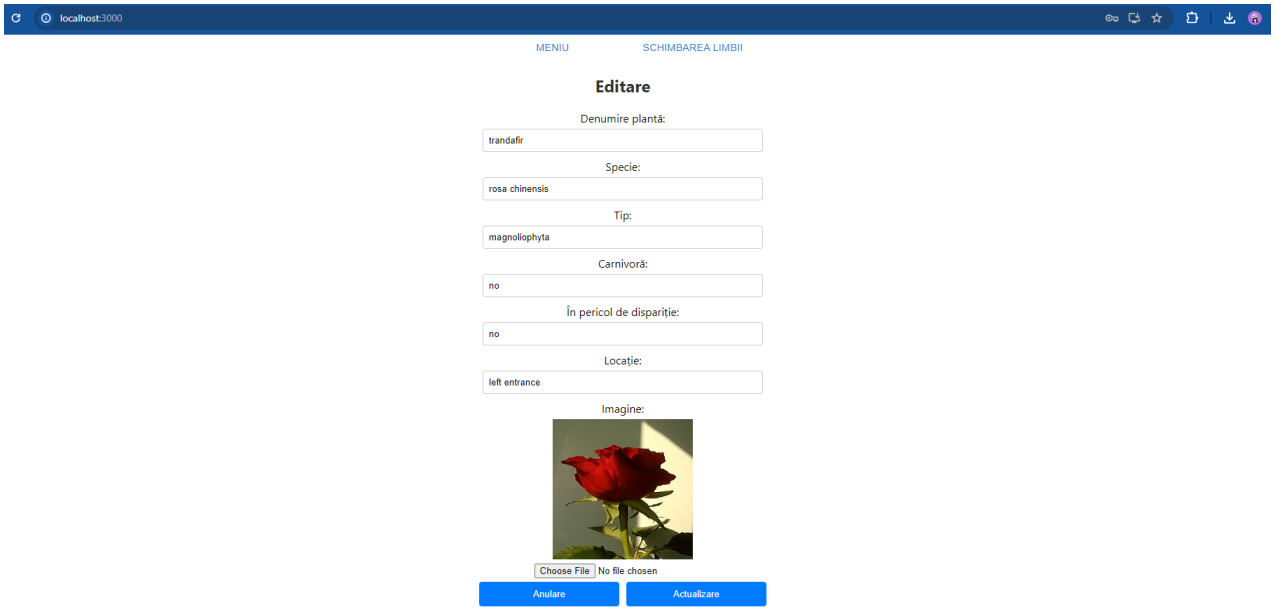

Figura 44: Pagina de editare planta

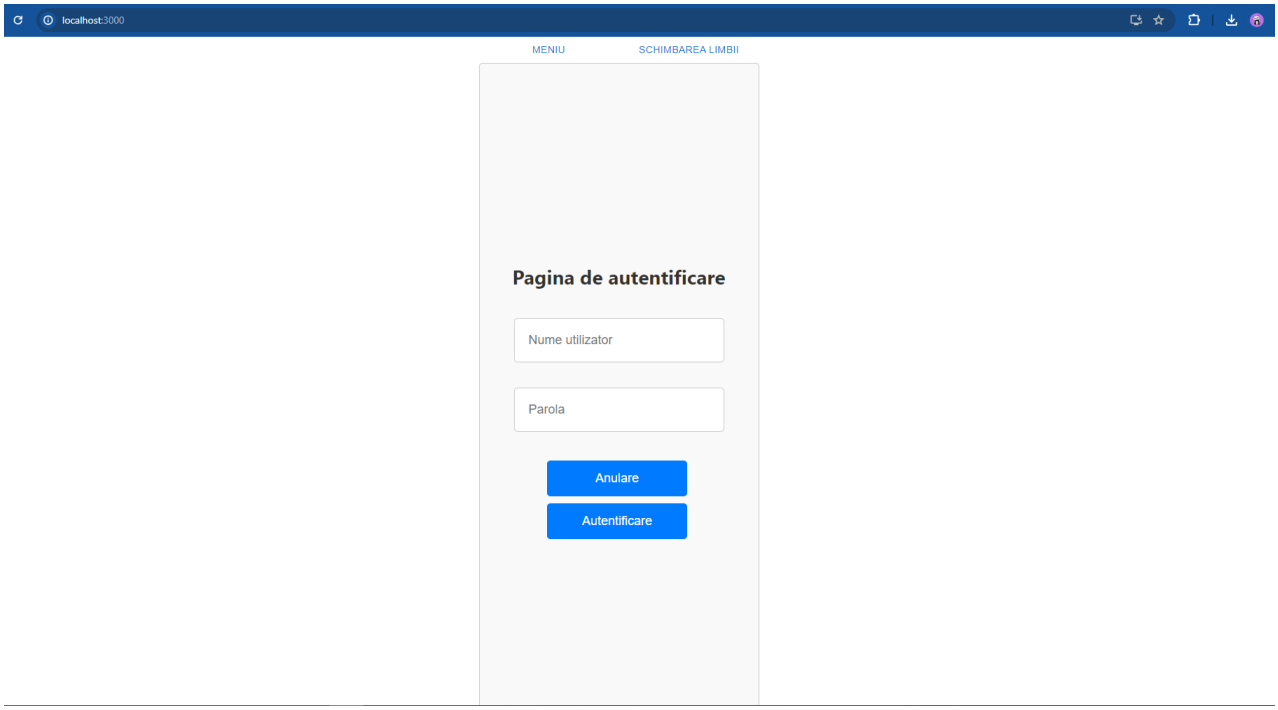

Figura 45: Pagina de autentificare

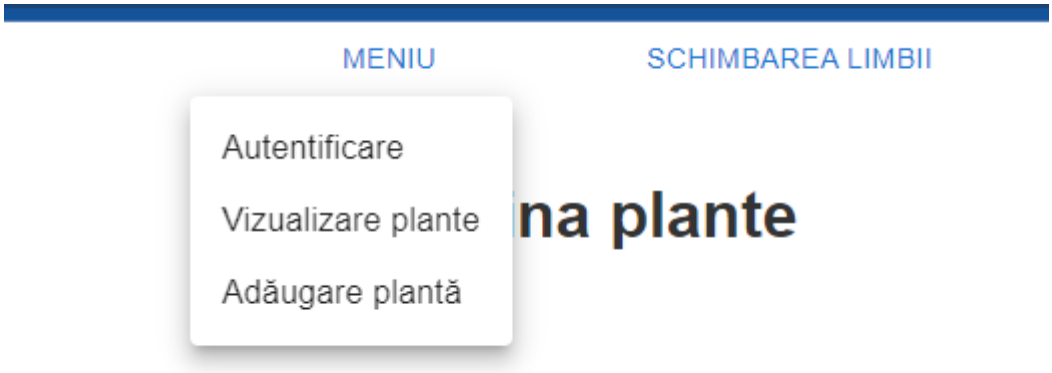

Figura 46: Meniu de functionalitati (rol: angajat)

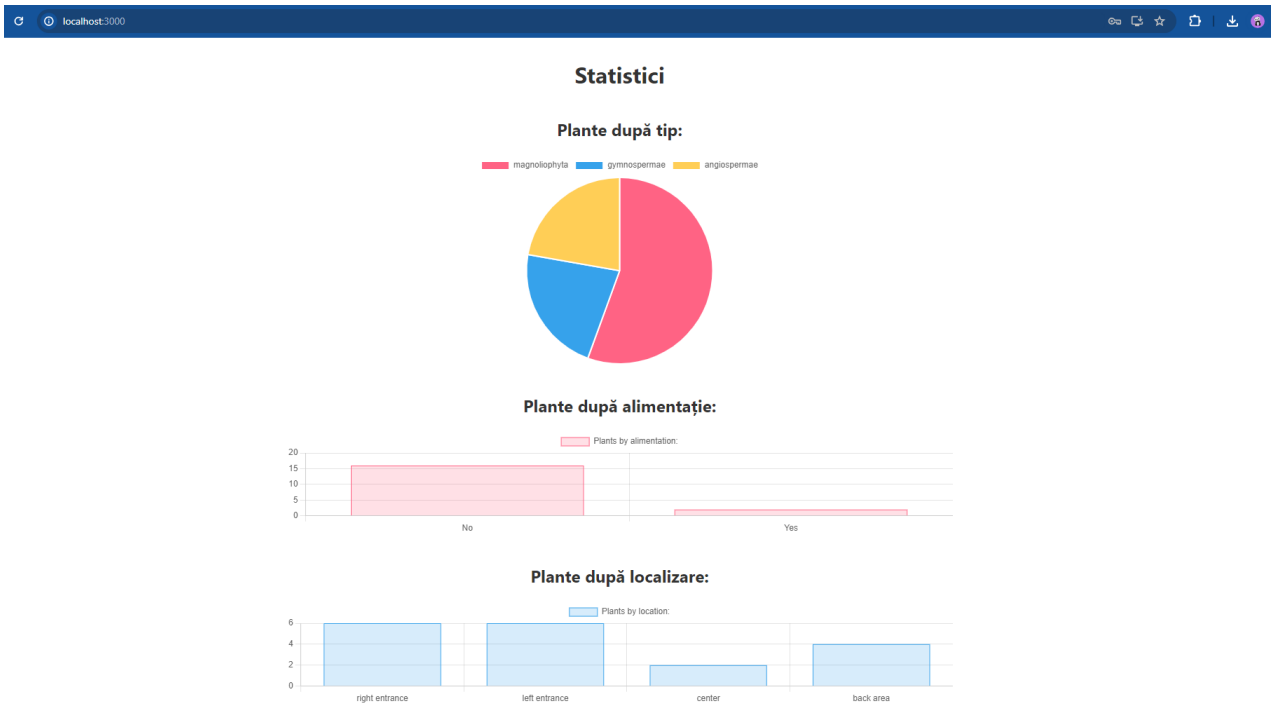

Figura 47: Pagina de vizualizare statistici

3. O statistică despre numărul plantelor după localizarea în grădina botanică;

Dacă utilizatorul autentificat are rolul de ”administrator”, în meniu apare și opțiunea de vizualizare a tuturor utilizatorilor din sistem. Pagina UserPage afișează sub forma unui nou tabel datele despre utilizatori, dar și despre angajații cărora le corespund: nume, prenume, data nașterii, adresa, numărul de telefon și email-ul.

Un administrator poate realiza editarea utilizatorilor, modificând username-ul, parola sau statutul de administrator al fiecăruia. Datele despre angajat sunt imutabile, decât dacă se creează un utilizator nou, în pagina de creare existând opțiunea de a adăuga date și pentru un angajat nou.

O altă operație CRUD care se poate face este ștergerea unui utilizator și respectiv a angajatului asociat, prin apăsarea unuia dintre butoanele corespunzătoare de lângă fiecare intrare

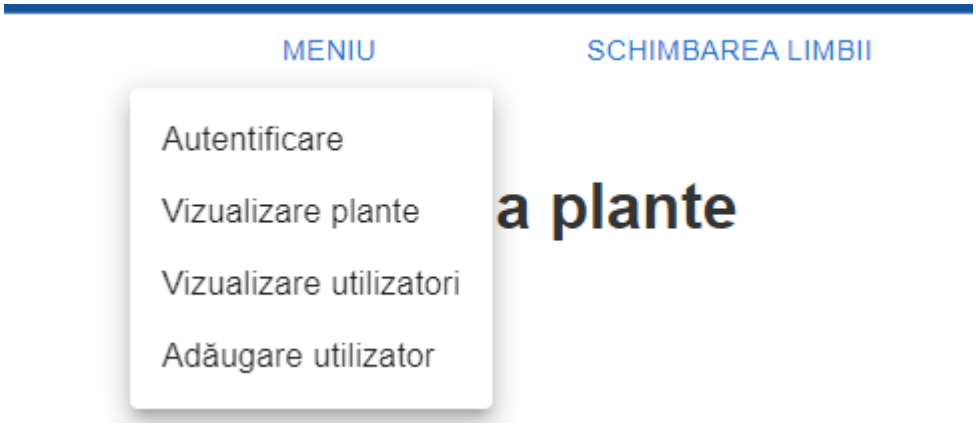

Figura 48: Meniu de functionalitati (rol: administrator)

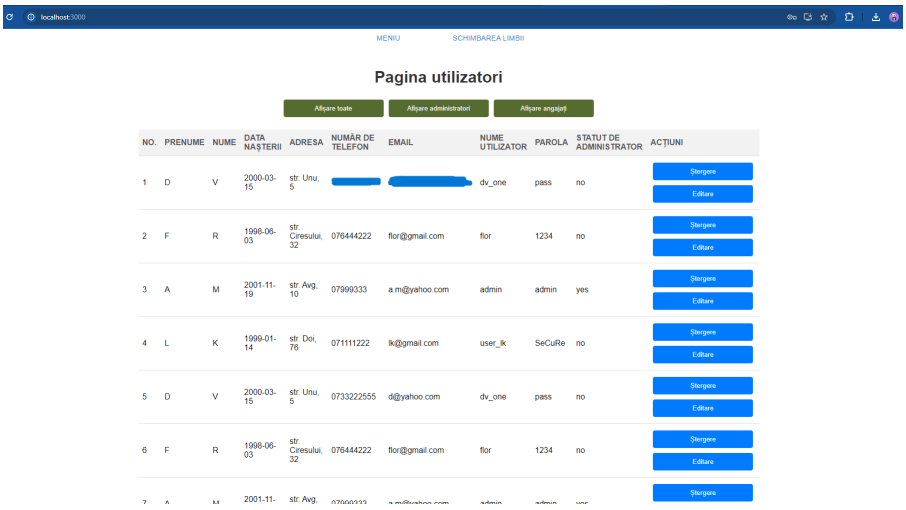

Figura 49: Pagina de vizualizare utilizatori

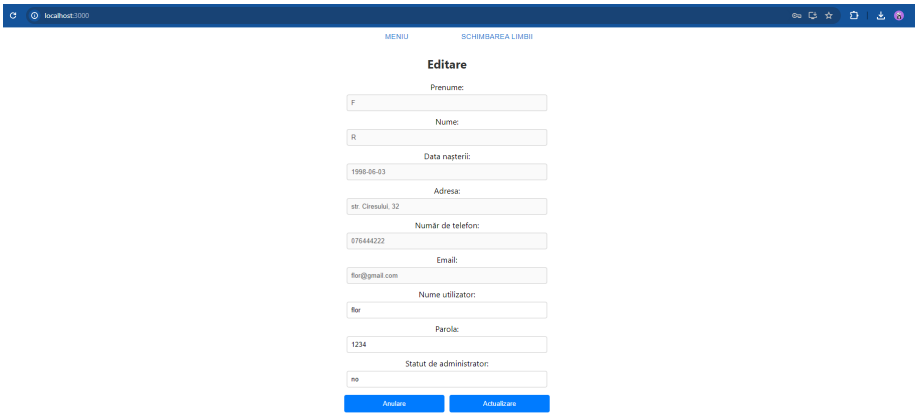

Figura 50: Pagina de editare utilizator

din tabel. Butonul de ”Editare” va duce utilizatorul pe o nouă pagine, cu datele personale și câmpurile care se pot modifica de către administrator.