UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

PROIECTARE SOFTWARE
Documentație tema 3
Grădina Botanică
Arhitectura MVC și Observer

Student: Adela Iosif
Grupa: 30233

2023-2024

Cuprins

1. Introducere 3
1.1. Obiectiv 3
1.2. Cerințe 3
1.3. Enunțul problemei 3
2. Analiza problemei 4
2.1. Diagrama cazurilor de utilizare 4
3. Proiectare 5
3.1. Diagrama de pachete 5
3.2. Diagrama entitate-relație 8
4. Implementare 9
4.1. Instrumente utilizate 9
4.2. Șablonul comportamental Observer 9
4.3. Șablonul creațional Singleton 9
4.4. Descrierea aplicației 9

1. Introducere

1.1. Obiectiv

Obiectivul acestei teme este familiarizarea cu șablonul architectural Model-View-Controller.

Pentru persistența informației se va utiliza o bază de date relațională (SQL Server, MySQL, etc.).

1.2. Cerințe

- În ***faza de analiză*** se va realiza **diagrama cazurilor de utilizare**.
- În ***faza de proiectare*** se vor realiza:
  - **diagrama de clase** respectând arhitectura **MVC** și folosind:
    - șablonul de proiectare comportamental ***Observer***;
    - un șablon de proiectare creațional.
  - **diagrama entitate-relație** corespunzătoare bazei de date.
- În ***faza de implementare*** se va scrie cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând:
  - proiectarea dată de diagrama de clase;
  - unul dintre următoarele limbaje de programare: C#, C++, Java, Python.
- Finalizarea temei va consta în predarea unui director ce va cuprinde:
  - Un fișier cu diagramele UML realizate;
  - Baza de date;
  - Aplicația soft;
  - Documentația (minim 10 pagini) - un fișier care cuprinde:
    - numele studentului, grupa;
    - enunțul problemei;
    - instrumente utilizate;
    - justificarea limbajului de programare ales;
    - descrierea diagramelor UML;
    - descrierea aplicației.

1.3. Enunțul problemei

Dezvoltați o aplicație 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:

- Operații CRUD pentru informațiile legate de utilizatorii care necesită autentificare;
- Vizualizarea listei utilizatorilor care necesită autentificare;
- Filtrarea listei utilizatorilor care necesită autentificare după tipul utilizatorilor.

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

2. Analiza problemei

2.1. Diagrama cazurilor de utilizare

În diagrama de use case (figura 1) se pot identifica cei trei actori: vizitator, angajat și admin. Astfel, vizitatorii sunt persoane nelogate care pot accesa aplicația și pot vizualiza lista plantelor din grădina botanică sortată după tip și specie (acum având ocazia să vizualizeze imagini cu fiecare plantă), dar pot opta și pentru filtrarea listei plantelor după anumite criterii: tip, specie, plante carnivore, zonă grădină, având posibilitatea și de a căuta o plantă după nume. Mai mult, actorii de tip angajat și admin se pot loga, apoi având posibilitatea de efectuare a operațiilor corespunzătoare vizitatorului (doar angajatul), alături de altele suplimentare specifice fiecărui rol. Astfel, utilizatorul de tip angajat are ca operații specifice salvarea listei cu plante în diferite formate (csv, json, xml, doc), operațiile CRUD pe plantele din grădina botanică, dar și posibilitatea vizualizării unor statistici despre plante (proporția plantelor carnivore din grădina botanică, numărul de plante din fiecare zonă a grădinii). Administratorul are ca acțiuni specifice rolului operațiile CRUD pe utilizatori, vizualizarea listei acestora și filtrarea listei după tipul de utilizator.

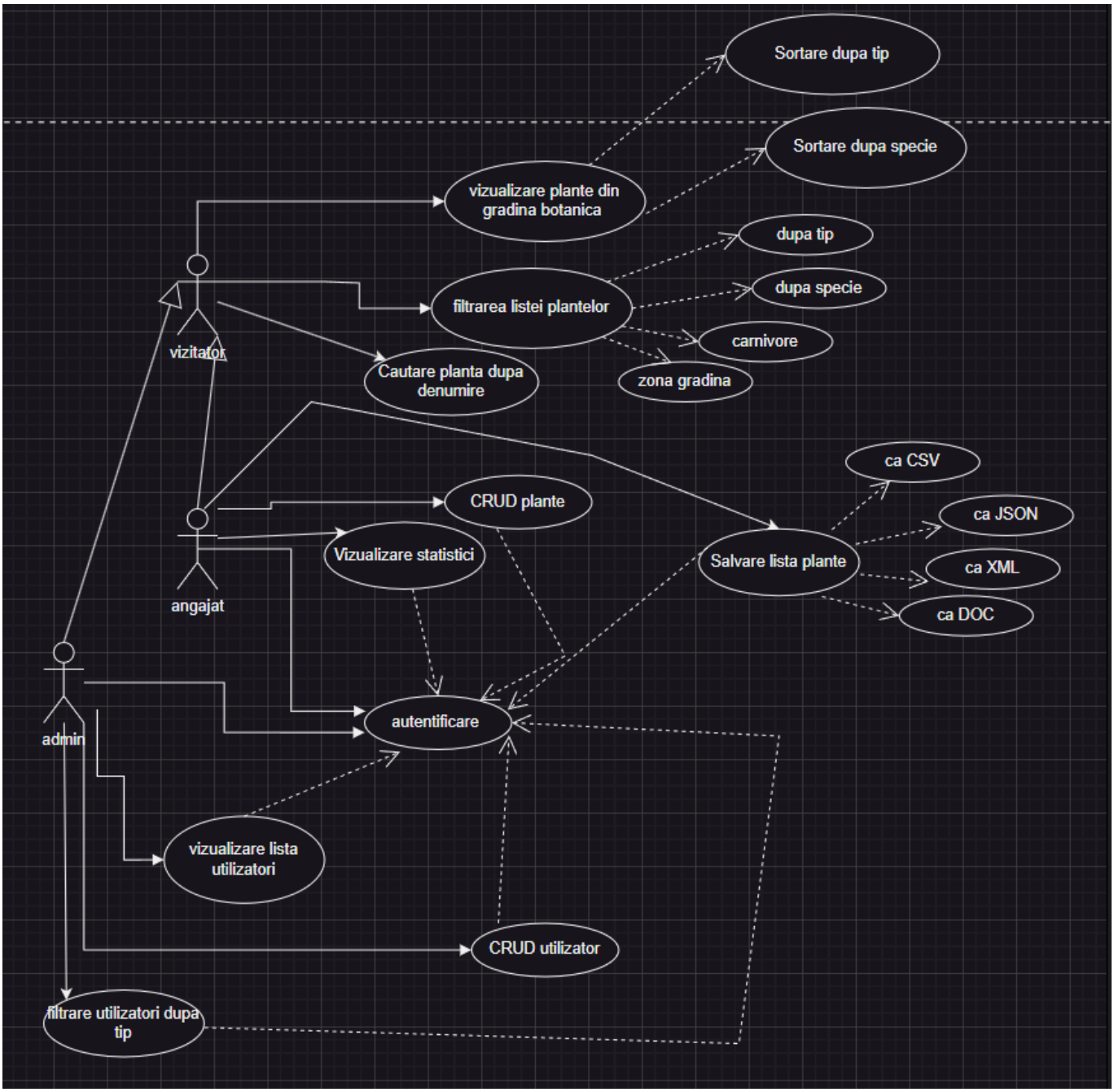

Figura 1. Diagrama de use case

3. Proiectare

Aplicația desktop proiectată de mine respectă arhitectura **MVC**; de asemenea, am folosit șablonul de proiectare comportamental Observer și șablonul de proiectare creațional Singleton.

3.1. Diagrama de pachete

Soluția prezentată (figura 2) utilizează varianta șablonului arhitectural MVC în care pachetele ***View*** și ***Model*** sunt decuplate.

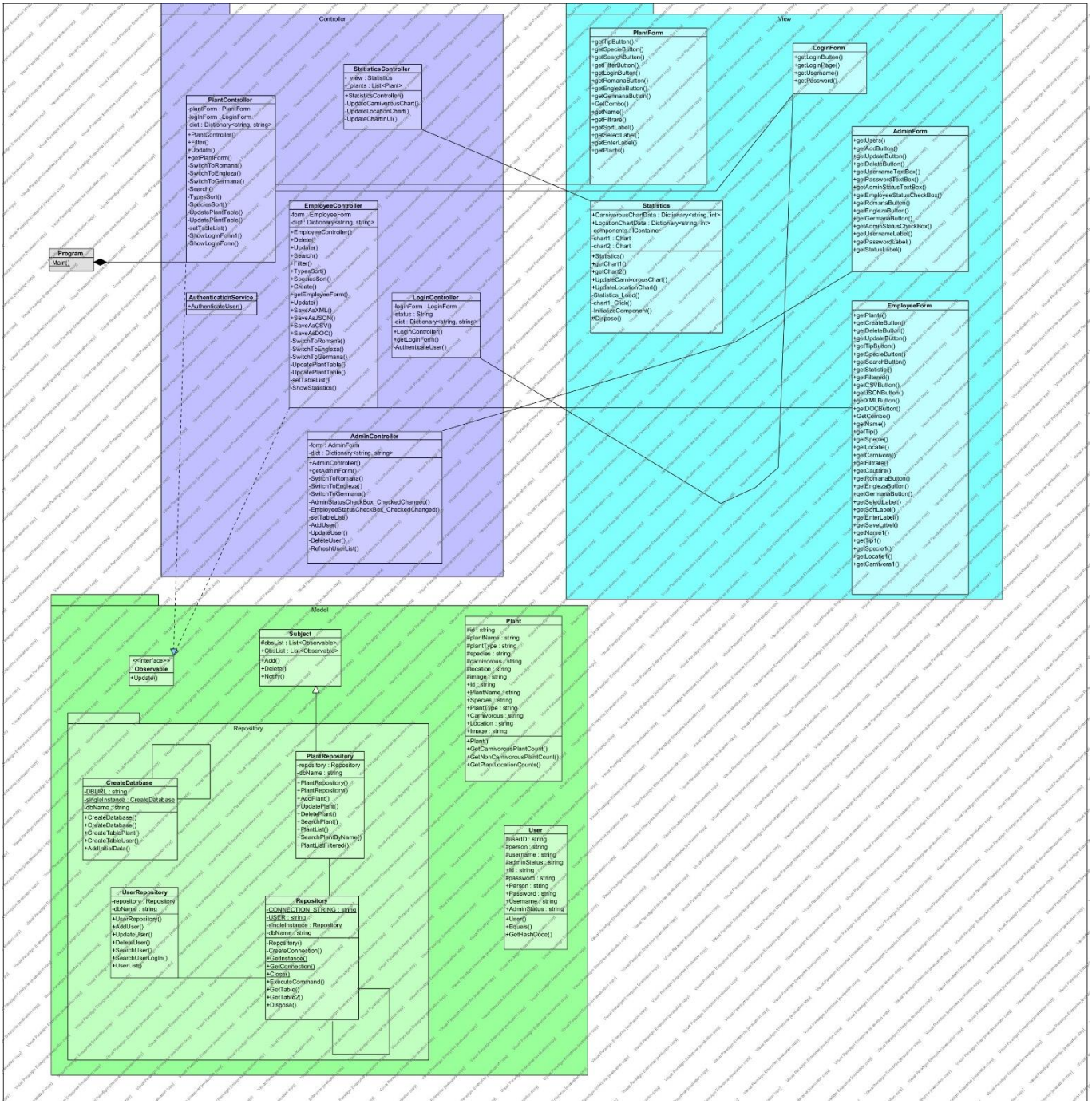

Figura 2. Diagramă de clase restrânsă a arhitecturii MVC

Pachetul ***Model*** cuprinde 4 clase și un subpachet (figura 3): ***Plant, User, Subject, Observable*** și pachetul ***Repository*** (figura 4; acesta având patru clase: ***CreateDatabase***, ***PlantRepository***, ***Repository***, ***UserRepository***). Astfel, pachetul ***Model*** stochează date și logica aferentă.

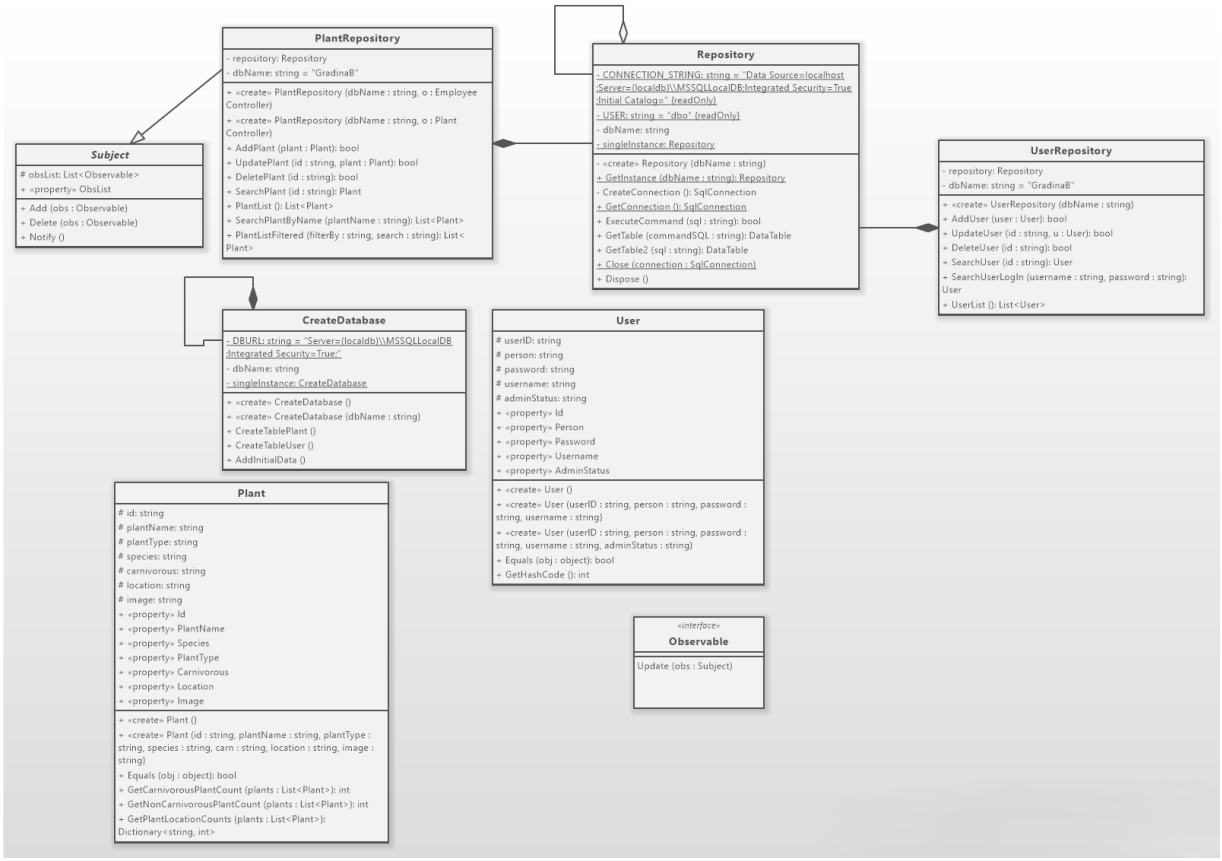

Figura 3. Pachetul ***Model***

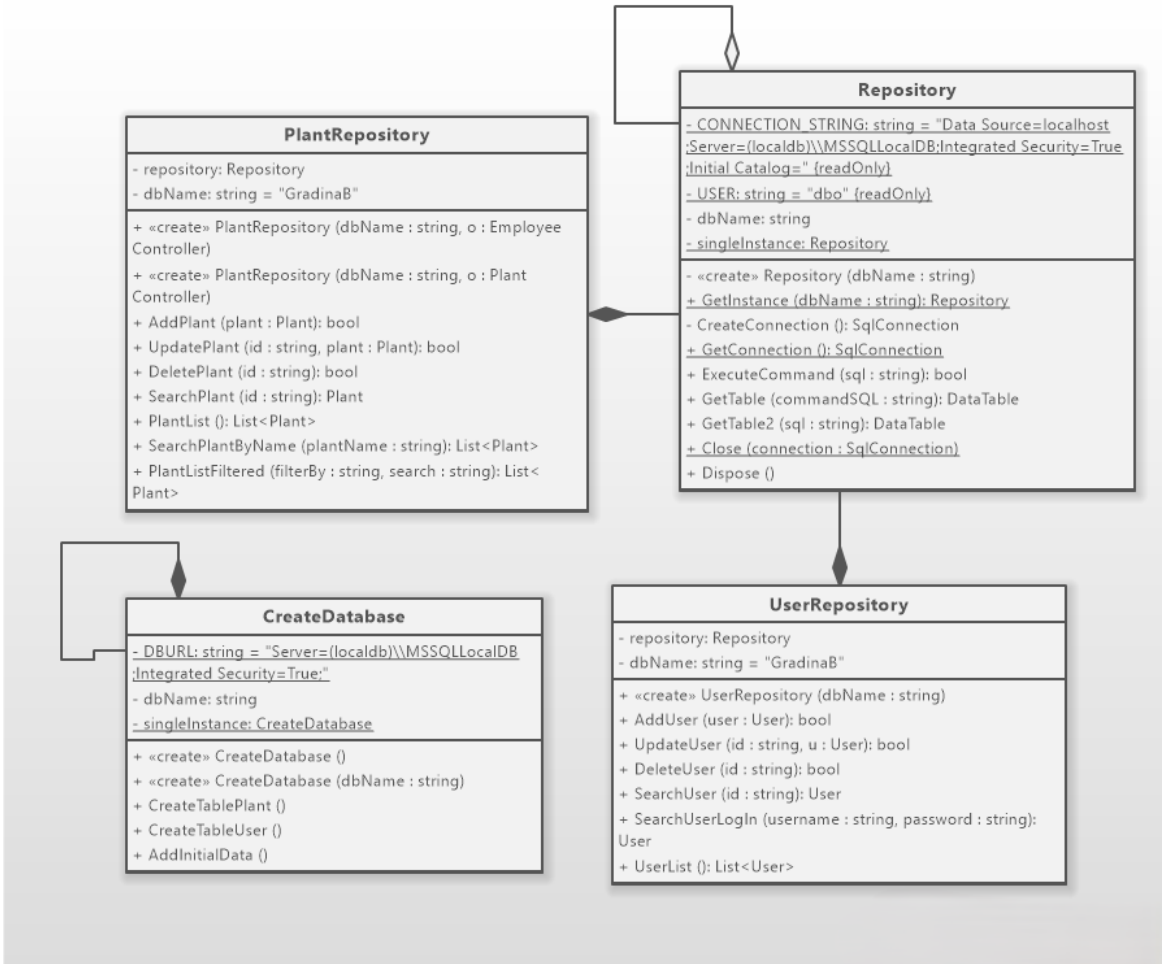

Figura 4. Subpachetul ***Repository*** din pachetul ***Model***

Pachetul ***View*** (figura 5) cuprinde 5 clase (***AdminForm***, ***EmployeeForm***, ***LogInForm***, ***PlantForm, Statistics***) care sunt responsabile cu afișarea unei interfețe prietenoase și cu gestionarea acțiunilor făcute de utilizator.

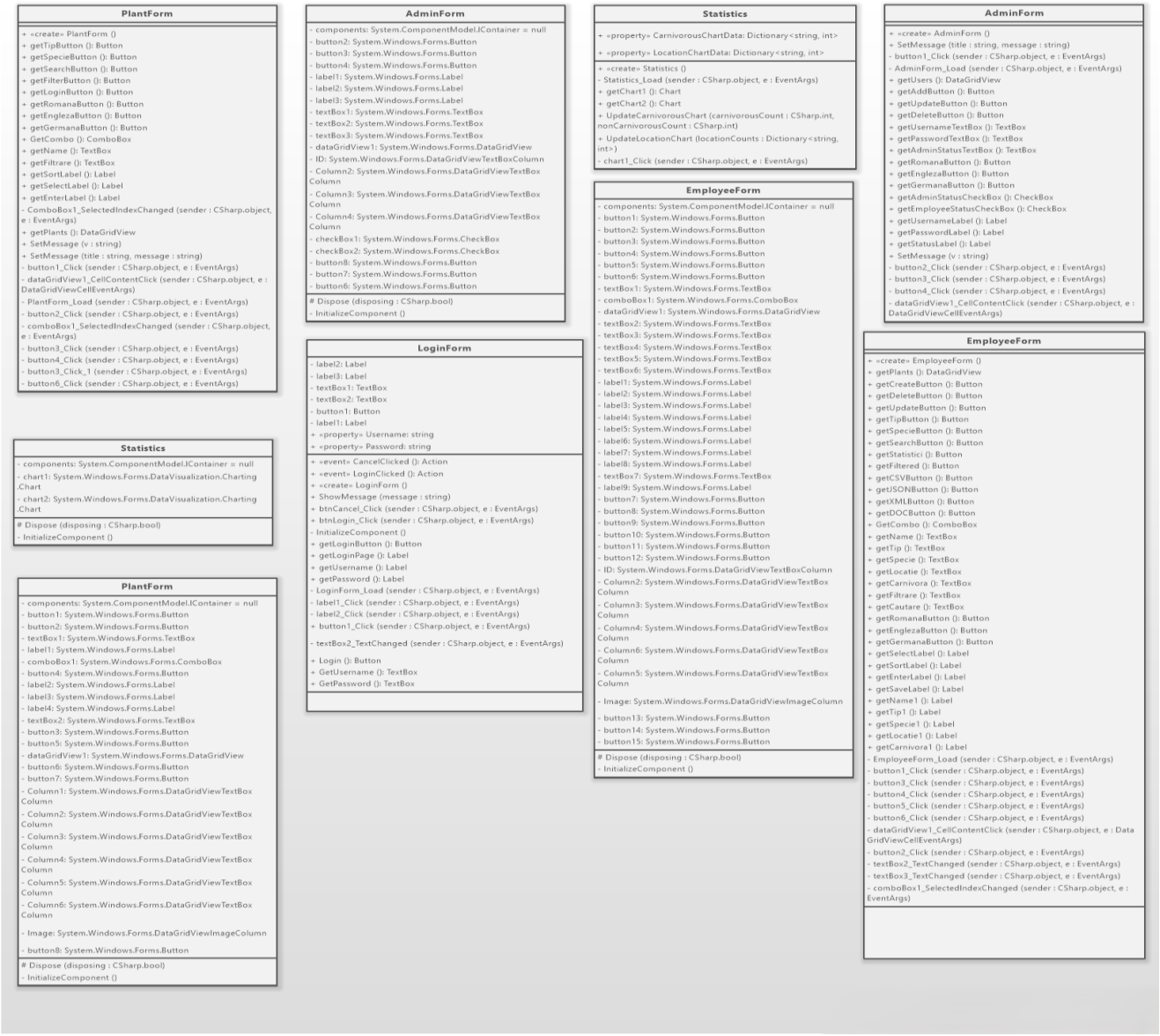

Figura 5. Pachetul ***View***

Pachetul ***Controller*** cuprinde 6 clase (figura 6): ***AdminController***, ***AuthenticationService***, ***EmployeeController***, ***LoginController***, ***PlantController***, ***StatisticsController***. Astfel, pachetul ***Controller*** este responsabil cu implementarea propriu-zisă a acțiunilor la care are acces un utilizator; prin intermediul acestui pachet se face legătura între ***View*** și ***Model***.

Șablonul arhitectural **MVC** (figura 6) ales, presupune următoarele: Utilizatorul face o cerere către Controller, acesta cerând informații de la Model. Modelul va răspunde, iar Controllerul va trimite informații către View, iar View aduce răspunsul utilizatorului.

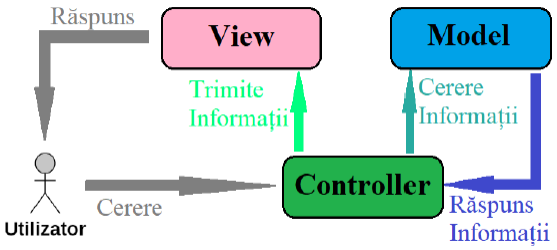

Figura 6. Arhitectura **MVC**

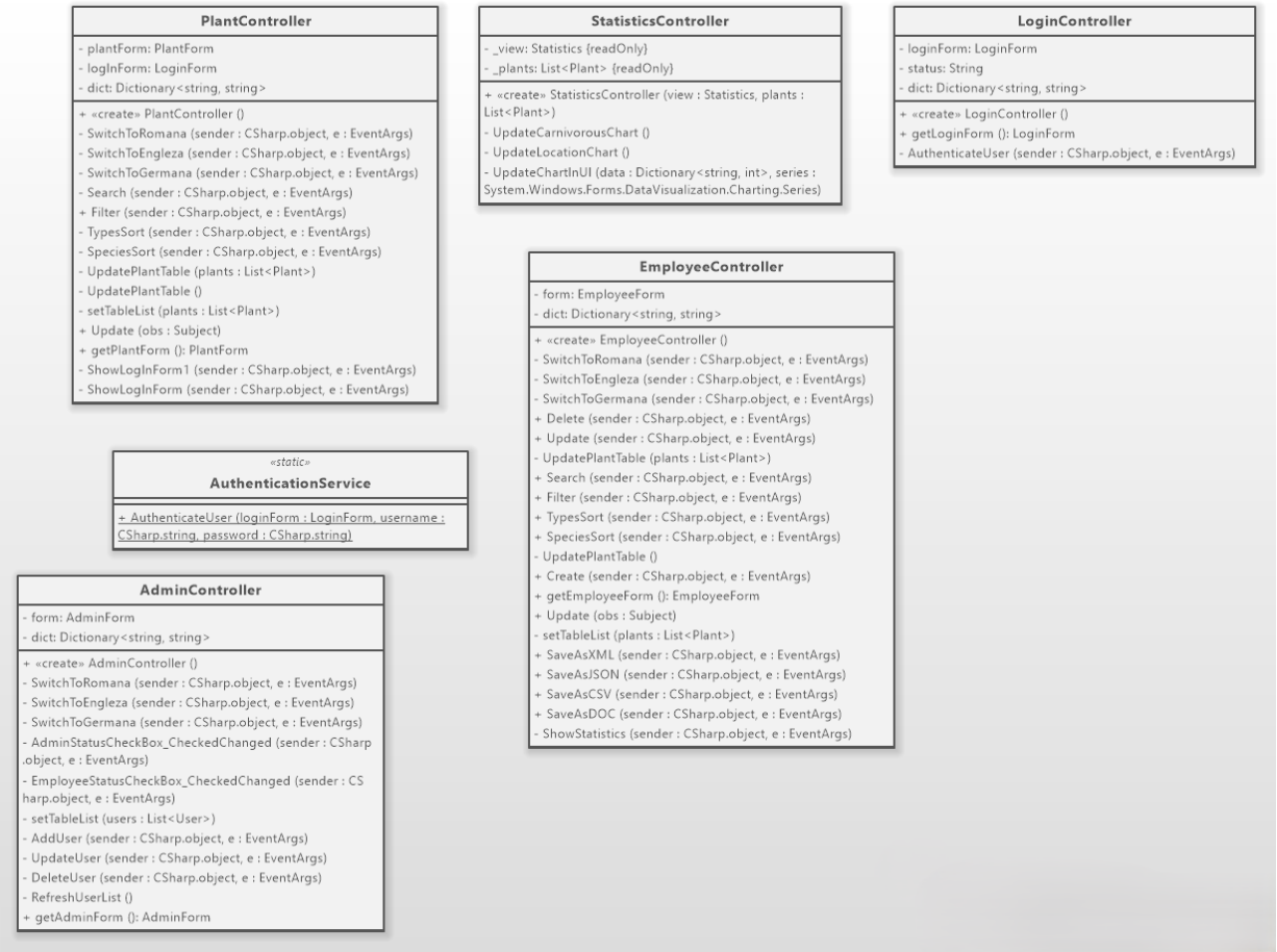

Figura 7. Pachetul ***Controller***

3.2. Diagrama entitate-relație

Diagrama bazei de date este similară cu cea de la temele precedente, dar a fost necesară adăugarea unei noi coloane, unde voi reține calea (locală) către imagine. Acest tip de diagramă (figura 8) este un model vizual care arată conexiunile entităților din baza de date. Astfel, baza de date este formată din două tabele: Users și Plant. Tabela Users este folosită pentru păstrarea datelor fiecărui utilizator (nume de utilizator, parolă, atribut pentru verificarea statutului de administrator). De asemenea, tabela Plant stochează toate plantele prezente în grădina botanică, cu atributele lor: nume, specie, tipul de plantă, atribut care reține dacă planta este carnivoră sau nu, locația plantei în cadrul grădinii, dar și câte o imagine pentru fiecare plantă.

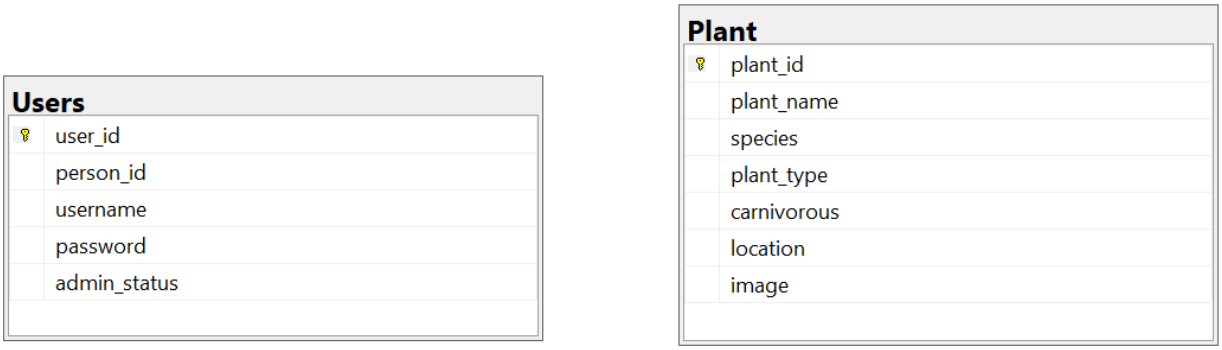

Figura 8. Diagrama entitate-relație

4. Implementare

4.1. Instrumente utilizate

În soluția prezentată se utilizează:

- Limbajul de programare C# (în IDE-ul Visual Studio)
- Sistemul de gestionare de baze de date relaționale SQL Server
- Visual Paradigm și Software Ideas Modeler (pentru crearea diagramei de pachete, clase)
- draw.io (pentru diagrama cazurilor de utilizare)

Am ales limbajul C#, deoarece este un limbaj de programare orientat-obiect care simplifică crearea interfeței grafice prin folosirea bibliotecii de clase grafice Windows Forms. Mai mult, C# este un limbaj de programare modern, de uz general, regăsindu-se pe primele poziții ale topurilor celor mai folosite limbaje de programare.

4.2. Șablonul comportamental Observer

Observer este un șablon de proiectare comportamental care definește o relație de dependență 1..* între obiecte, astfel încât când un obiect își schimbă starea, toți dependenții lui sunt notificați și actualizați automat; se evită cuplarea strânsă.

Astfel, în proiectul meu, ***EmployeeController*** este observatorul (Observable în C#), iar dacă persoana autentificată este angajat, va trimite un semnal de crud la ***PlantRepository***. ***PlantRepository*** este clasa observată care va face operația și va trimite semnalul înapoi (prin Notify) la ***EmployeeController***. Mai mult, acesta din urmă conține metoda Update prin care va face refresh la tabelul din interfața grafică.

4.3. Șablonul creațional Singleton

Singleton este un șablon de proiectare creațional care asigură existența unei singure instanțe a unei clase în cadrul unei aplicații și oferă acces la acea instanță de oriunde din cod.

În proiectul meu, am implementat acest design creațional în clasa ***Repository***. Clasa are un constructor privat (private Repository(string dbName)) și o metodă statică (public static Repository GetInstance(string dbName)) care returnează o instanță a clasei ***Repository***. De asemenea, am o variabilă statică privată (private static Repository singleInstance) care este inițializată cu valoarea null. Metoda GetInstance este utilizată pentru a obține o instanță a clasei ***Repository*** (dacă variabila singleInstance este null, atunci este creată o nouă instanță a clasei ***Repository*** cu numele bazei de date specificate, altfel este returnată instanța existentă). În esență, atunci când se dorește folosirea clasei ***Repository***, trebuie apelată metoda GetInstance(string dbName) și furnizat numele bazei de date. Aceasta va garanta că se va primi întotdeauna aceeași instanță a clasei ***Repository***, indiferent de câte ori se va folosi în aplicație.

4.4. Descrierea aplicației

La deschiderea aplicației se va afișa pagina de start (figura 9), mai exact cea în care utilizatorul este doar vizitator, acesta nefiind înregistrat în cont (nu îi este atribuit niciun rol). Astfel, un vizitator poate să sorteze plantele după tip sau specie, apăsând unul din butoanele corespunzătoare. Mai mult, plantele din grădina botanică pot fi vizualizate în listă, împreună cu atributele lor: id, nume, tip, specie, daca este carnivoră sau nu, zona din grădina botanică unde se găsește, o imagine reprezentativă plantei.

De asemenea, pagina se va deschide implicit în limba engleză, dar există opțiunea de a alege limba preferată, prin apăsarea unuia dintre butoanele specifice, localizate în dreapta-sus.

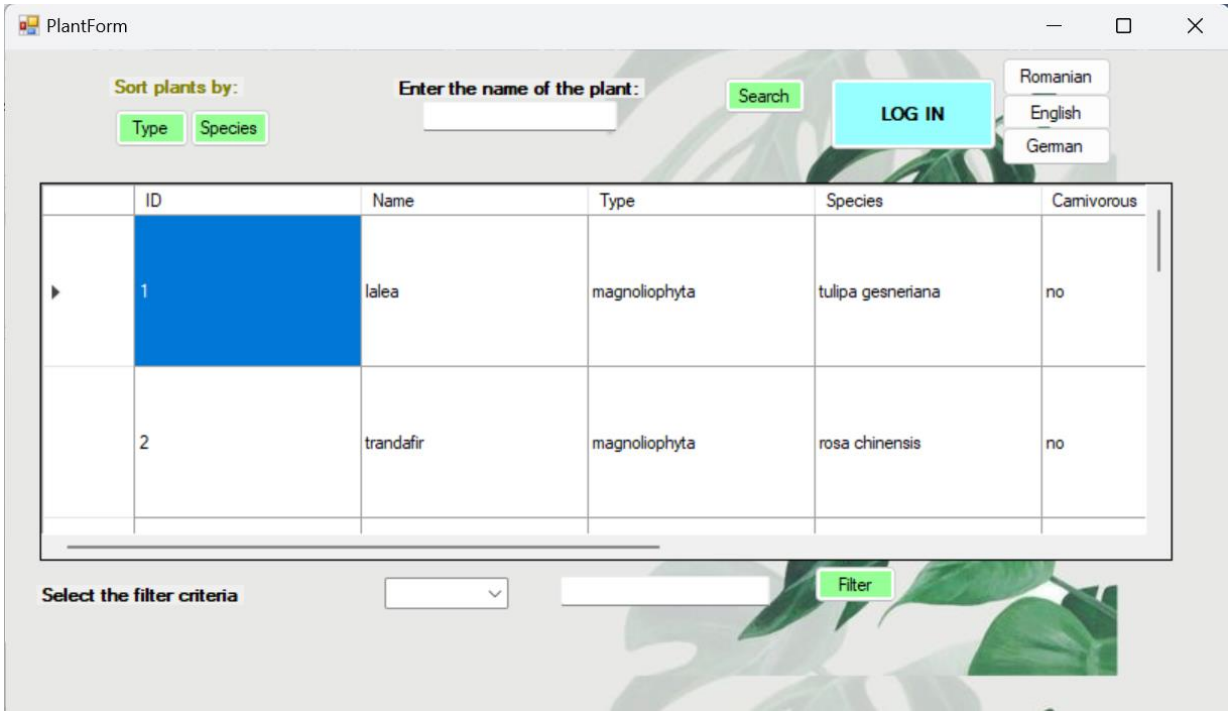

Figura 9. Pagina de vizitator

Pagina de vizitator conține și opțiunile de căutare a unei plante după nume sau filtrare, după anumite criterii (tip, specie, zonă, carnivore – trebuie selectat criteriul general, apoi introdus explicit în textbox-ul destinat; figura 10). De asemenea, în meniul unde se găsesc denumirile filtrelor, există și opțiunea de resetare a listei.

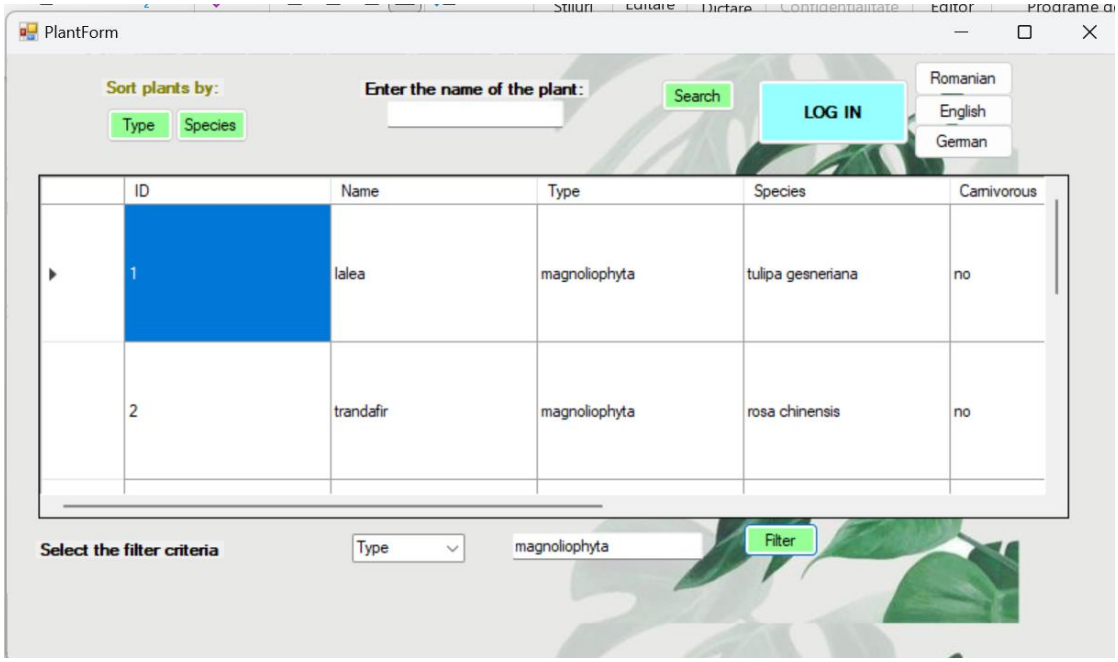

Figura 10. Filtrarea plantelor după un anumit criteriu

În pagina de start a aplicației desktop există și opțiunea de Login, care va deschide o filă nouă (figura 11), unde se poate face autentificarea (în rolul de angajat sau administrator).

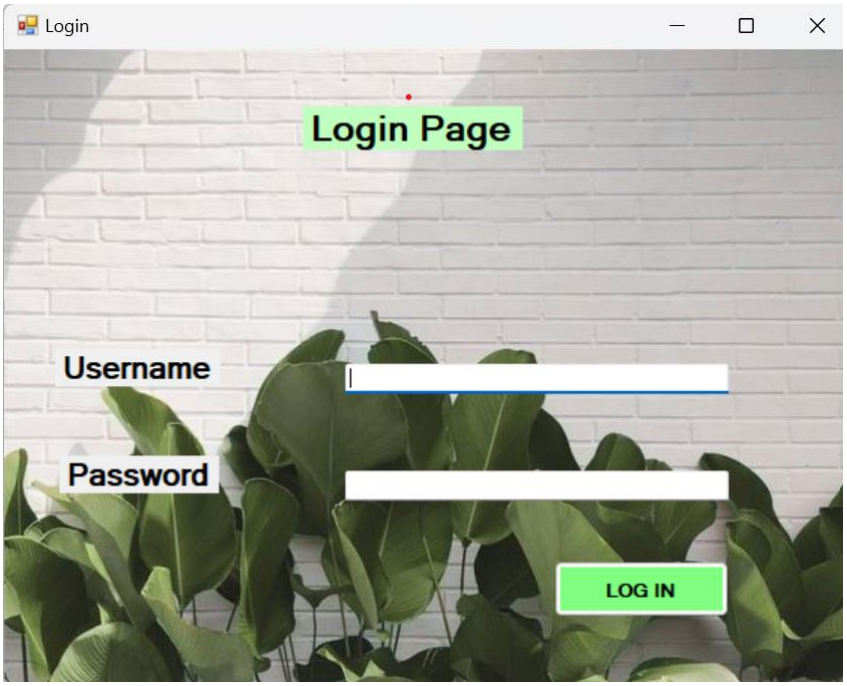

Figura 11. Pagina de autentificare

Logarea ca angajat (figura 12) oferă toate funcționalitățile disponibile și unui vizitator, la care se adaugă operații de creare, citire, actualizare, ștergere a plantelor din grădina botanică, posibilitatea de a salva listele cu plante în mai multe formate (csv, json, xml, doc) și vizualizarea unor statistici despre plantele din grădina botanică.

De asemenea, există opțiunea de selectare a limbii preferate: română, engleză sau germană (butoanele din partea dreaptă-sus a paginii).

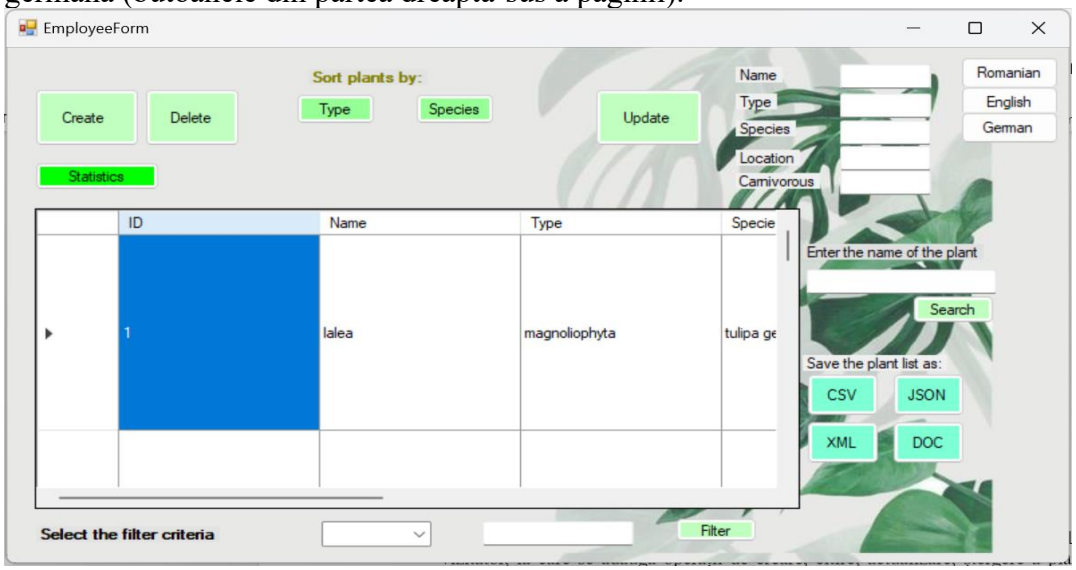

Figura 12. Pagina de angajat

Pentru operațiile de salvare a listei cu plante (figura 13), se alege formatul fișierului dorit la ieșire (prin click pe unul din butoanele: CSV, JSON, XML, DOC); apoi, se va deschide File Explorer, oferind posibilitatea salvării fișierului generat, în locația dorită.

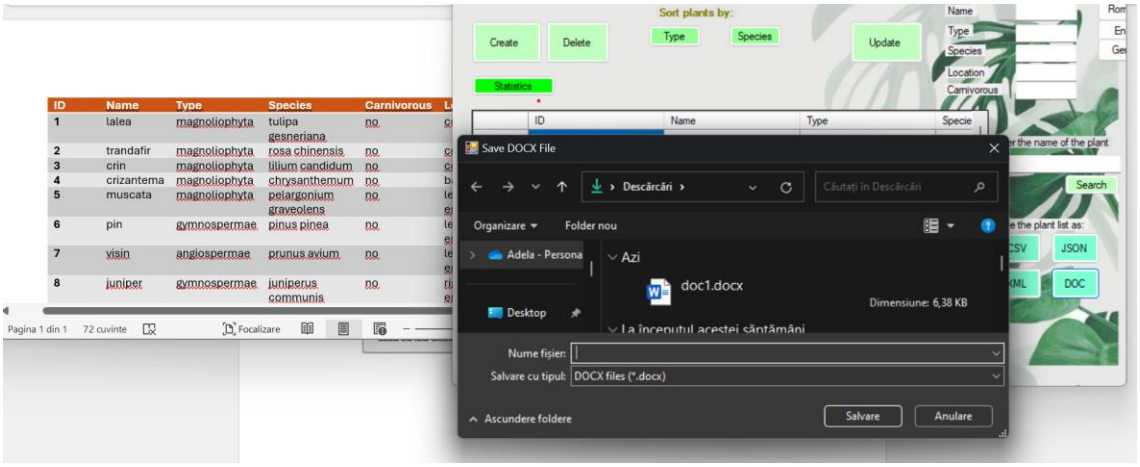

Figura 13. Salvarea listei cu plante în format doc

De asemenea, la adăugarea (figura 14) și la actualizarea unei plante, pentru a oferi și posibilitatea de adăugare/actualizare a imaginii specifice plantei, se va deschide File Explorer.

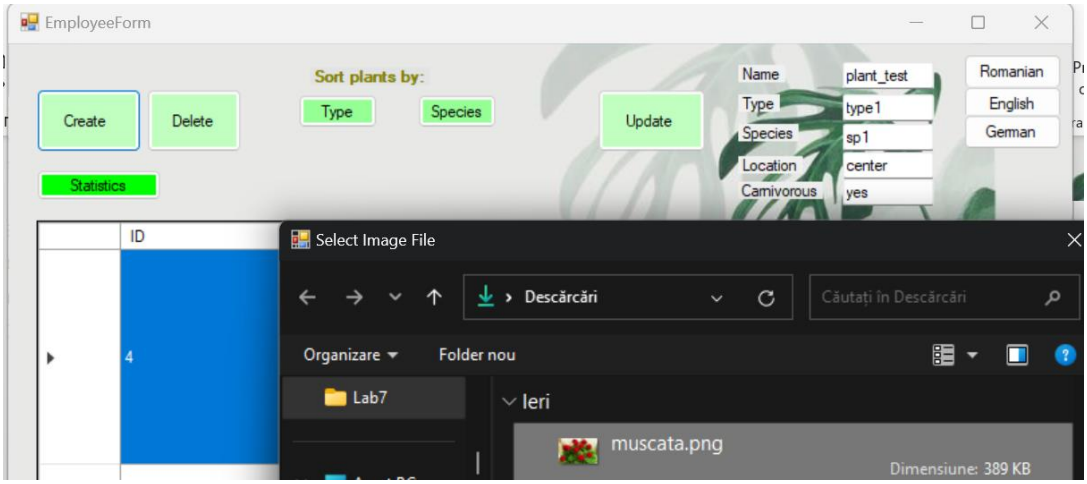

Figura 14. Adăugarea unei plante în listă

Mai mult, angajatul are opțiunea de vizualizare a unor statistici în ceea ce privește plantele din grădina botanică (figura 15). Astfel, am realizat două reprezentări; în prima se vede distribuția plantelor carnivore în grădina botanică, iar în cea de-a doua, numărul de plante din fiecare zonă a grădinii botanice.

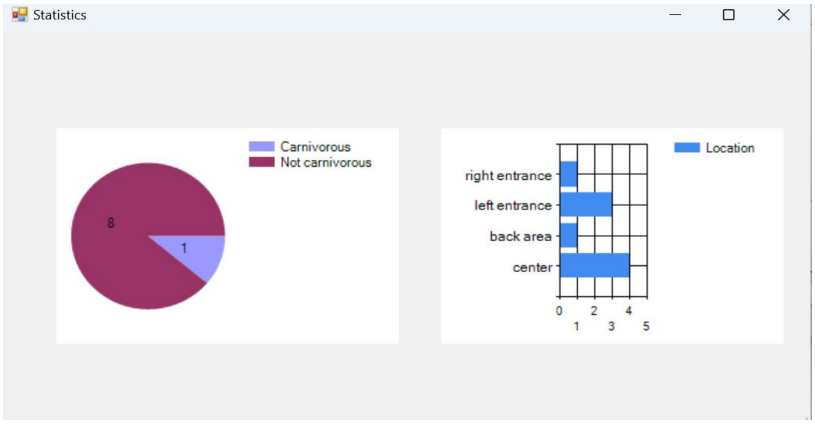

Figura 15. Statistici despre plantele din grădina botanică

Dacă ne înregistrăm ca admin (figura 16) vom putea efectua operații de creare, citire, actualizare și ștergere a utilizatorilor din baza de date. De asemenea, administratorul va putea vizualiza lista tuturor utilizatorilor și îi va putea filtra după tipul lor (angajați/administratori).

Limba standard în care se va deschide pagina este engleza, dar similar și celorlalte pagini ale aplicației, utilizatorul are posibilitatea de a selecta limba favorită (apăsând unul din butoanele specifice din dreapta-sus).

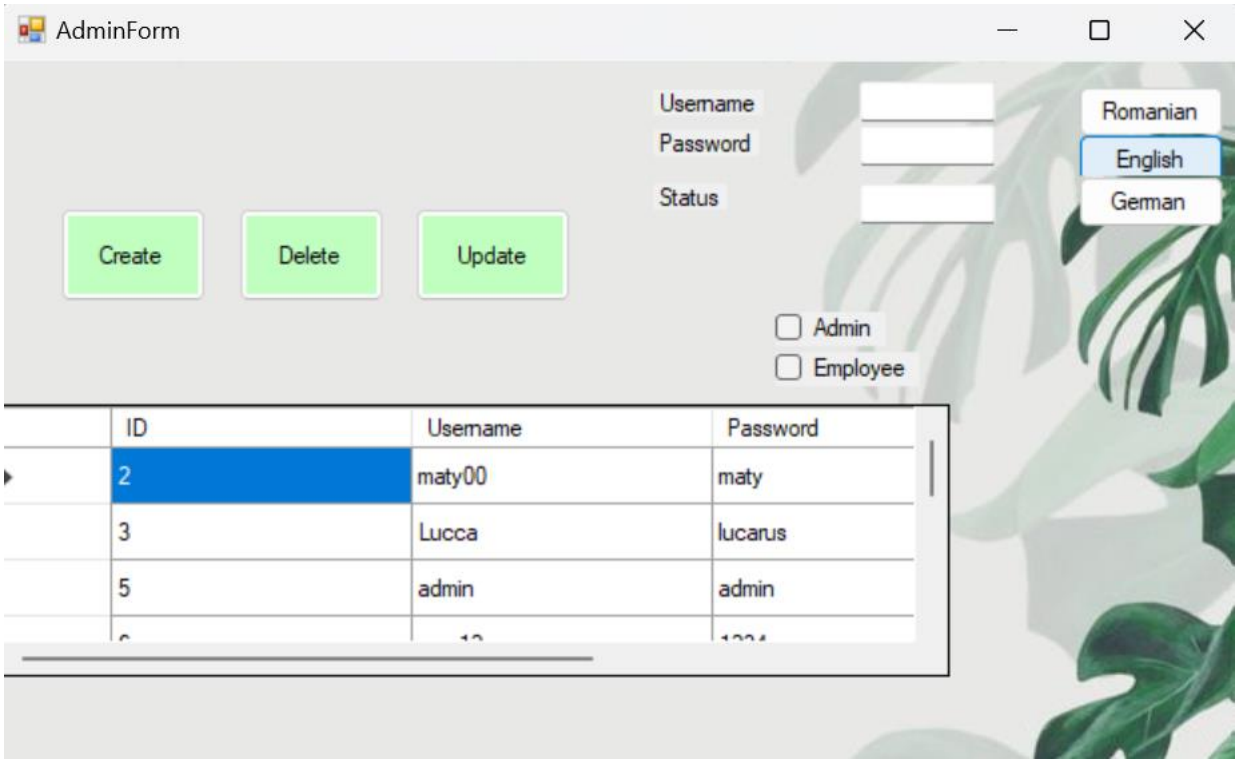

Figura 16. Pagina de administrator

Pentru a filtra doar tipul dorit de angajat (figura 17), se va bifa preferința, utilizând unul dintre cele 2 checkbox-uri (Admin/Employee).

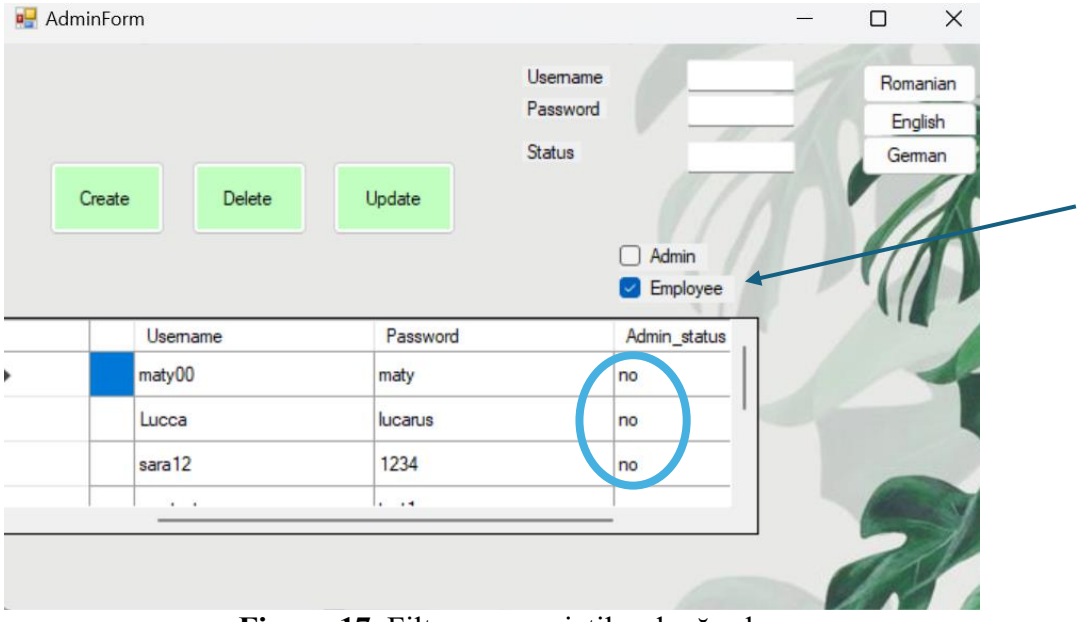

Figura 17. Filtrarea angajaților după rol

Operațiile de adăugare, ștergere și actualizare sunt similare cu cele corespunzătoare plantelor. La adăugare se vor introduce utilizatorul, parola și statusul angajatului (dacă este sau nu administrator). Pentru ștergere și actualizare se va selecta un angajat din bara din stânga a tabelului și se va efectua operația apăsând butonul specific.