

PROIECT: APLICAȚIE GRĂDINĂ ZOOLOGICĂ

PENTRU: **PROIECTARE SOFTWARE**

STUDENT: MORAR TIMOTEI CRISTIAN

GRUPA: 30235

1



# CUPRINS

1. SPECIFICAȚIE 3
2. INSTRUMENTE UTILIZATE SI JUSTIFICARE LIMBAJ 4
3. [DIAGRAME UML 5](#_TOC_250000)
4. DESCRIEREA APLICAȚIEI 10
5. POSIBILITĂȚI DE DEZVOLTARE ULTERIOARĂ 13



1. SPECIFICAȚIE

Această aplicație este concepută pentru a facilita experiența vizitatorilor, angajaților și administratorilor unei grădini zoologice. Cu trei tipuri distincte de utilizatori - vizitatori, angajați și administratori - fiecare cu privilegii specifice, aplicația oferă funcționalități adaptate nevoilor și rolurilor lor în grădina zoologică.

Pentru *vizitatori*, aplicația furnizează o platformă accesibilă și informativă. Aceștia pot explora lista completă a animalelor din grădină, sortate în funcție de alimentație și specie. De asemenea, au posibilitatea de a filtra această listă conform criteriilor precum specie, alimentație și habitat, permițându-le să își personalizeze experiența în funcție de preferințele lor.

Pentru *angajații* grădinii zoologice, aplicația devine o unealtă de lucru esențială. În plus față de funcționalitățile disponibile pentru vizitatori, aceștia au acces la operațiuni CRUD (inserare, găsire, actualizare, ștergere) pentru gestionarea animalelor din grădină. Astfel, pot actualiza și gestiona informațiile referitoare la fiecare animal în parte, asigurându-se că datele sunt mereu precise și actualizate.

Pentru *administratori* aplicației, aceasta devine un instrument puternic de administrare a întregii platforme. În plus față de facilitățile disponibile pentru vizitatori și angajați, administratorii pot gestiona informațiile legate de utilizatori care necesită autentificare. Aceștia au acces la operațiuni CRUD pentru aceste date, permițându-le să administreze și să monitorizeze utilizatorii și conturile lor în mod eficient. De asemenea, au posibilitatea de a vizualiza lista utilizatorilor care necesită autentificare, facilitând gestionarea și monitorizarea utilizatorilor și accesului lor la aplicație.

Prin intermediul acestei aplicații, grădina zoologică poate oferi o experiență îmbunătățită vizitatorilor săi, asigurându-se că informațiile sunt mereu actualizate și că fiecare categorie de utilizatori beneficiază de funcționalități adaptate nevoilor și rolurilor lor specifice.

INSTRUMENTE UTILIZATE SI JUSTIFICARE LIMBAJULUI ALES



2.

Înainte de a începe implementarea aplicației pentru grădina zoologică, am realizat că este crucial să înțelegem în profunzime contextul problemei și cerințele implicate. Am început prin construirea diagramei cazurilor de utilizare folosind instrumentul online *draw.io*, care ne-a permis să vizualizăm și să clarificăm interacțiunile dintre diferitele funcționalități ale aplicației. Acest pas ne-a ajutat să stabilim clar obiectivele și fluxurile de lucru ale utilizatorilor.

Următorul pas a fost crearea unei diagrame de clase care să respecte arhitectura MVP (Model-View-Presenter), asigurându-ne că structura aplicației va fi coerentă și scalabilă. Această diagramă ne-a servit ca o hartă a entităților și a relațiilor dintre acestea, facilitând dezvoltarea și înțelegerea codului în etapele ulterioare.

Pentru implementarea aplicației, am decis să folosesc limbajul Python în mediul de dezvoltare Visual Studio Code. Alegerea acestui IDE a fost determinată de faptul că îl consider cel mai prietenos și cel mai ușor de utilizat din punctul meu de vedere. În plus, am apreciat bogăția de funcționalități și extensii disponibile, care au contribuit semnificativ la productivitatea și eficiența dezvoltării.

Python a fost alegerea mea pentru codul aplicației datorită mai multor avantaje evidente. Codul în Python este mai lizibil, mai succint și mai ușor de realizat, ceea ce a redus semnificativ timpul și efortul necesar dezvoltării. Un alt beneficiu major al folosirii Python este faptul că nu este necesară o specificare explicită a tipurilor de date, ceea ce a permis o flexibilitate și un flux de lucru mai rapid și mai eficient.

În plus, am optat pentru framework-ul CustomTKinter pentru realizarea interfeței grafice a aplicației desktop. Experiența mea anterioară cu acest framework m-a convins că este soluția cea mai potrivită pentru a obține o interfață estetică și user-friendly. CustomTKinter mi-a oferit un set de instrumente puternice și flexibile pentru crearea unei interfețe grafice atrăgătoare și intuitive, care să îmbunătățească experiența utilizatorilor.

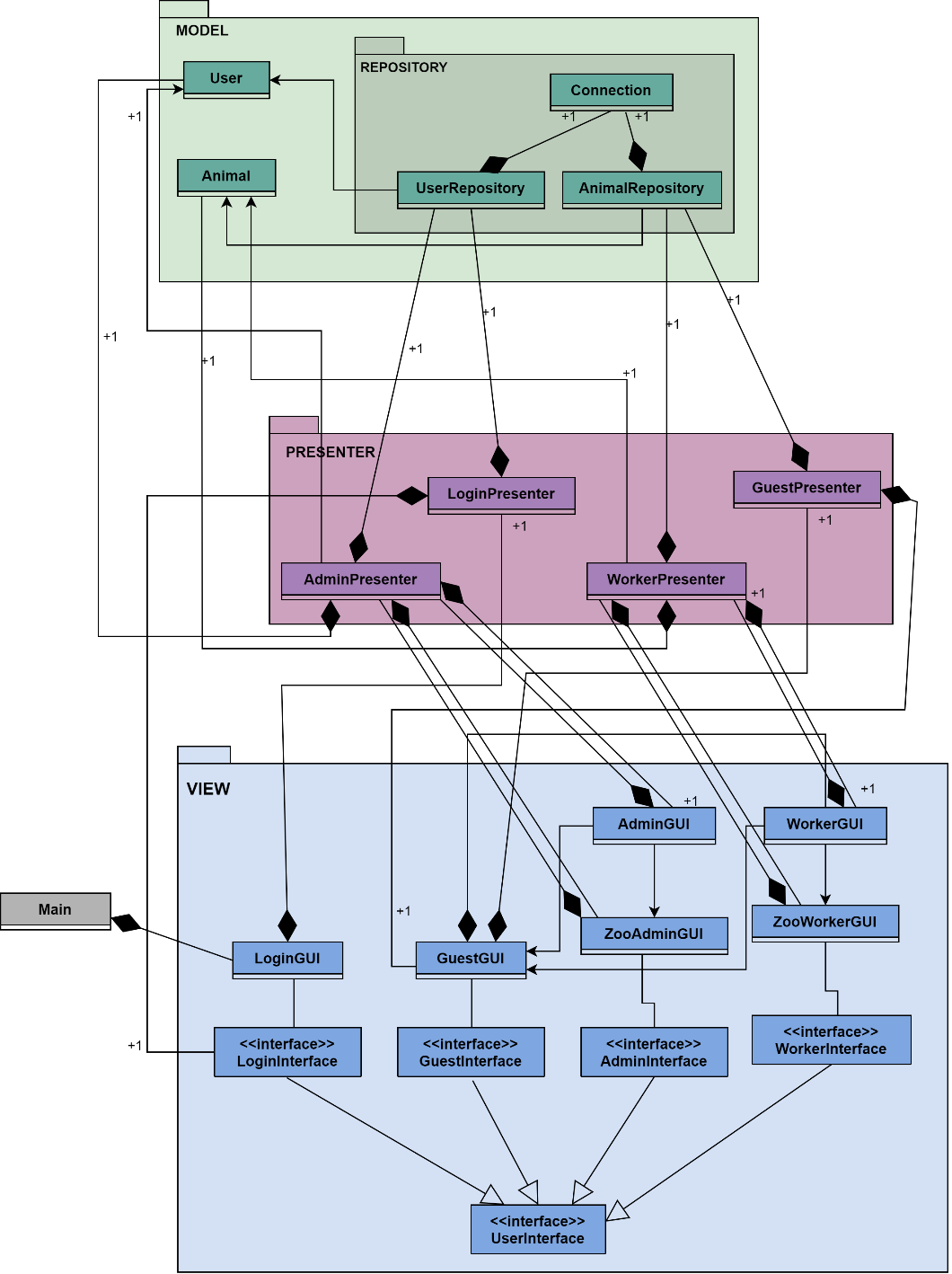
Aceste alegeri mi-au permis nu doar să dezvolt aplicația conform cerințelor, ci și să învăț și să îmi îmbunătățesc abilitățile în utilizarea limbajului Python și în dezvoltarea de aplicații desktop.



3.

DIAGRAME UML

Această primă diagramă ne arată principiul de funcționare a arhitecturii MVP. Arhitectura Model-View-Presenter este un model de proiectare a aplicațiilor software care își propune să separe în mod clar logica de prezentare a datelor (View) de logica de business (Model), cu ajutorul unui intermediar numit Presenter.

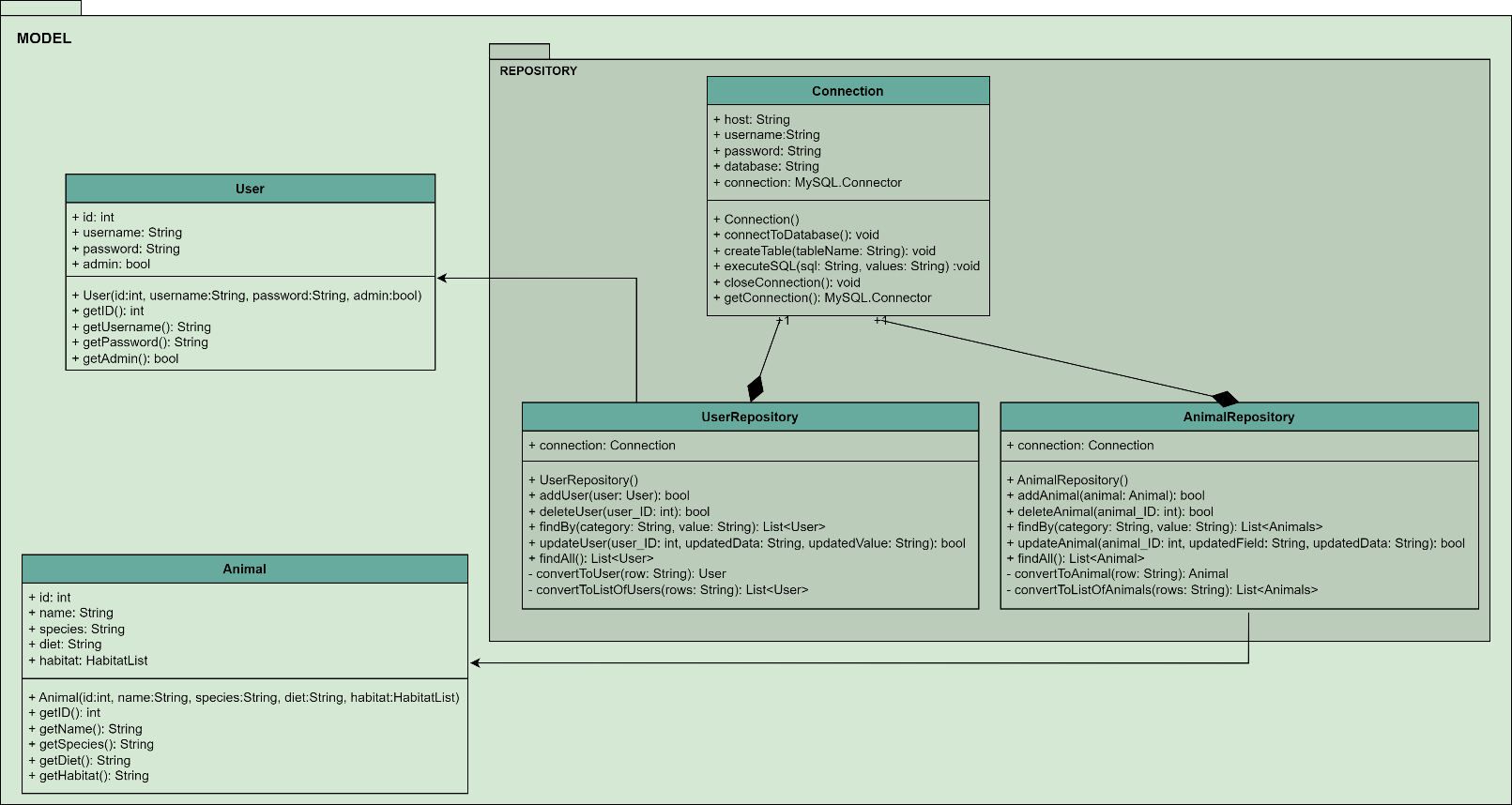


**Figura 1.** Diagrama de clase restrânsă a arhitecturii MVP



În cadrul acestui pattern, componenta Model reprezintă structura și logica de date a aplicației. În contextul specific al pachetului Model, acesta cuprinde clasele User și Animal, care reprezintă entitățile de bază ale aplicației, precum și un alt pachet numit repository.

Pachetul repository conține clase care sunt responsabile pentru gestionarea interacțiunilor cu baza de date. În acest sens, avem clasa Connection, care se ocupă de conectarea la baza de date, execuția instrucțiunilor SQL și închiderea conexiunii într-un mod eficient și sigur. De asemenea, acest pachet include clasele UserRepository și AnimalRepository, care sunt responsabile pentru realizarea efectivă a operațiilor CRUD (Create, Read, Update, Delete) asupra tabelelor de useri și, respectiv, de animale din baza de date.

 Astfel, pachetul Model constituie nucleul aplicației, conținând atât datele și logica lor asociată, cât și mecanismele necesare pentru interacțiunea cu baza de date. Prin împărțirea logică a responsabilităților între componente, MVP facilitează dezvoltarea modulară și scalabilă a aplicației, permitând o gestionare mai eficientă a complexității și o separare clară între straturile de prezentare.

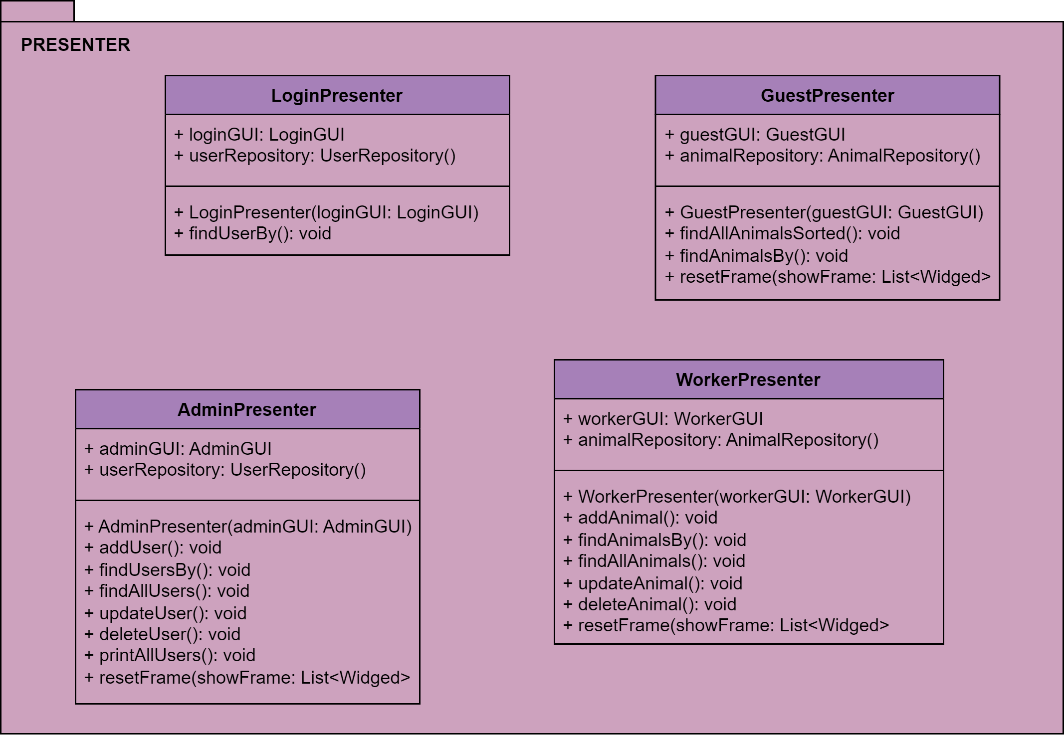
**Figura 2.** Diagrama de clase a pachetului Model



În cadrul arhitecturii Model-View-Presenter (MVP), pachetul Presenter reprezintă componenta care acționează ca intermediar între Model și View, gestionând logica de prezentare și interacțiunea cu interfața grafică a utilizatorului (GUI).

Astfel, pachetul Presenter cuprinde clasele LoginPresenter, WorkerPresenter, GuestPresenter și AdminPresenter, fiecare dintre acestea fiind asociată unui anumit tip de utilizator și acoperind funcționalitățile specifice asociate acestuia. În constructorul lor, aceste clase primesc două obiecte esențiale: o clasă de repository, în funcție de necesitate, și o clasă care realizează conexiunea cu interfața grafică a utilizatorului.

Fiecare clasă Presenter preia datele de la interfața grafică a utilizatorului folosind gettere și le transmite mai departe către repository pentru a efectua operațiile corespunzătoare în baza de date. După ce repository-ul finalizează operațiile, clasele Presenter primesc datele returnate și le transmit înapoi către interfața grafică a utilizatorului prin settere, actualizând astfel starea și afișarea informațiilor în GUI.

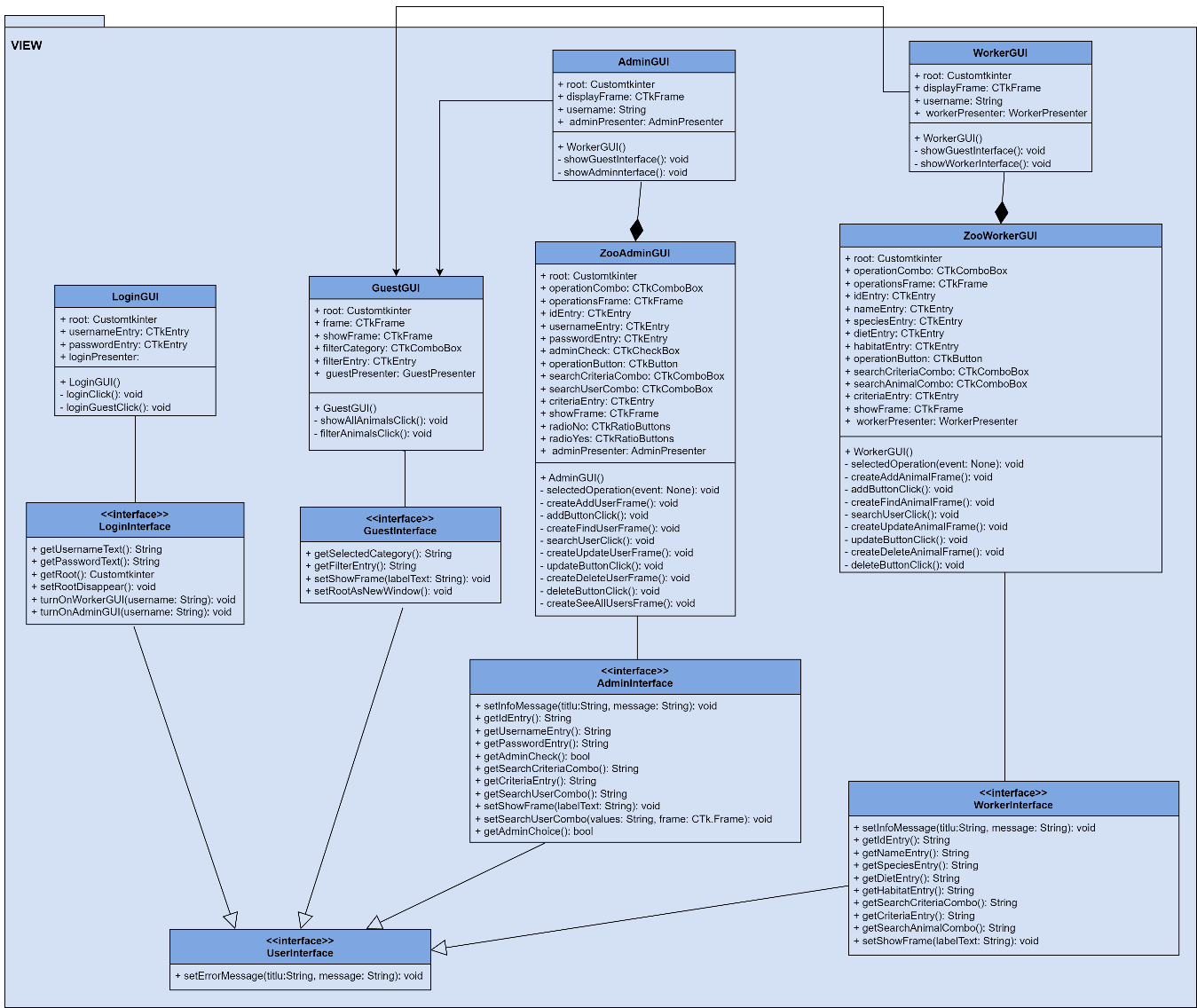
 Această structură modulară și bine definită a pachetului Presenter permite separarea clară a responsabilităților și promovează o gestionare eficientă a logicii de prezentare în cadrul aplicației. Prin intermediul acestor clase, interacțiunea dintre utilizator și sistem este gestionată într-un mod coerent și modular, facilitând astfel dezvoltarea, testarea și întreținerea aplicației

**Figura 3.** Diagrama de clase a pachetului Presenter



Pachetul View reprezintă componenta responsabilă cu prezentarea și gestionarea interfeței grafice a aplicației către utilizator. Acesta cuprinde mai multe clase și interfețe care sunt concepute pentru a oferi o experiență interactivă și impactantă utilizatorilor, utilizând instrumentele și funcționalitățile oferite de CustomTKinter. Interfețele grafice create în cadrul pachetului View folosesc diverse elemente precum label-uri, combo box-uri, câmpuri de introducere text și altele, pentru a afișa informații și a permite interacțiunea utilizatorului cu aplicația într-un mod intuitiv și eficient.

Datele introduse de utilizator prin intermediul interfeței grafice sunt preluate și procesate de către clasele din pachetul View. Aceste date sunt apoi transmise mai departe către clasele Presenter asociate, pentru a fi gestionate și procesate conform logicii de business a aplicației.



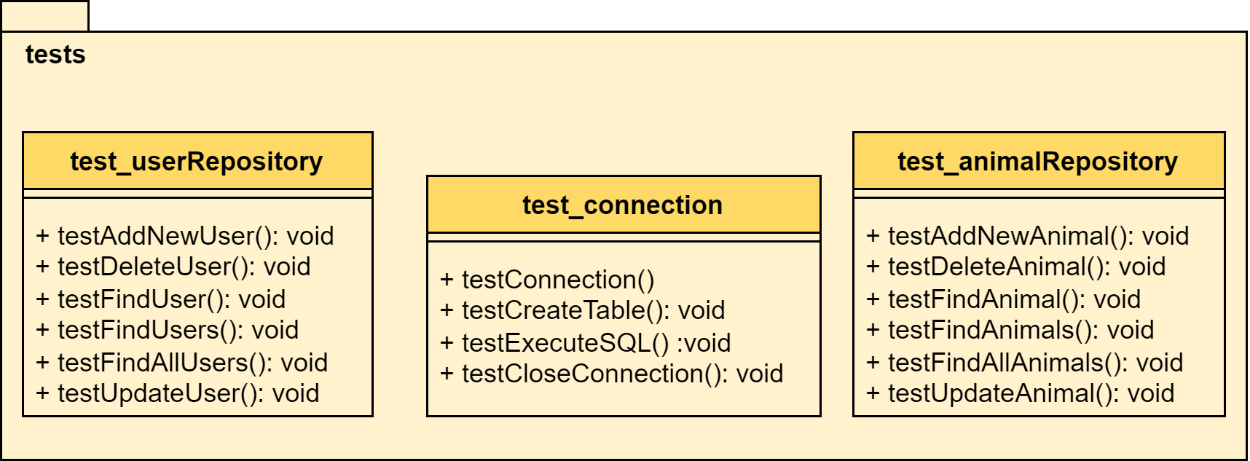
**Figura 4.** Diagrama de clase a pachetului View



Pachetul tests conține trei clase: test\_connection, test\_animalRepository și test\_userRepository. Aceste clase sunt responsabile pentru testarea funcționalității de conectare la baza de date, execuția instrucțiunilor SQL și realizarea operațiilor CRUD în funcție de tipul de tabel accesat. Astfel, aceste teste asigură corectitudinea și fiabilitatea operațiunilor de bază ale aplicației în interacțiunea cu baza de date.

A close-up of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated Pentru realizarea si rularea testelor am folosit framework-ul Pytest.

**Figura 6.** Diagrama enitate-relație

**Figura 5.** Diagrama de clase a pachetului Tests

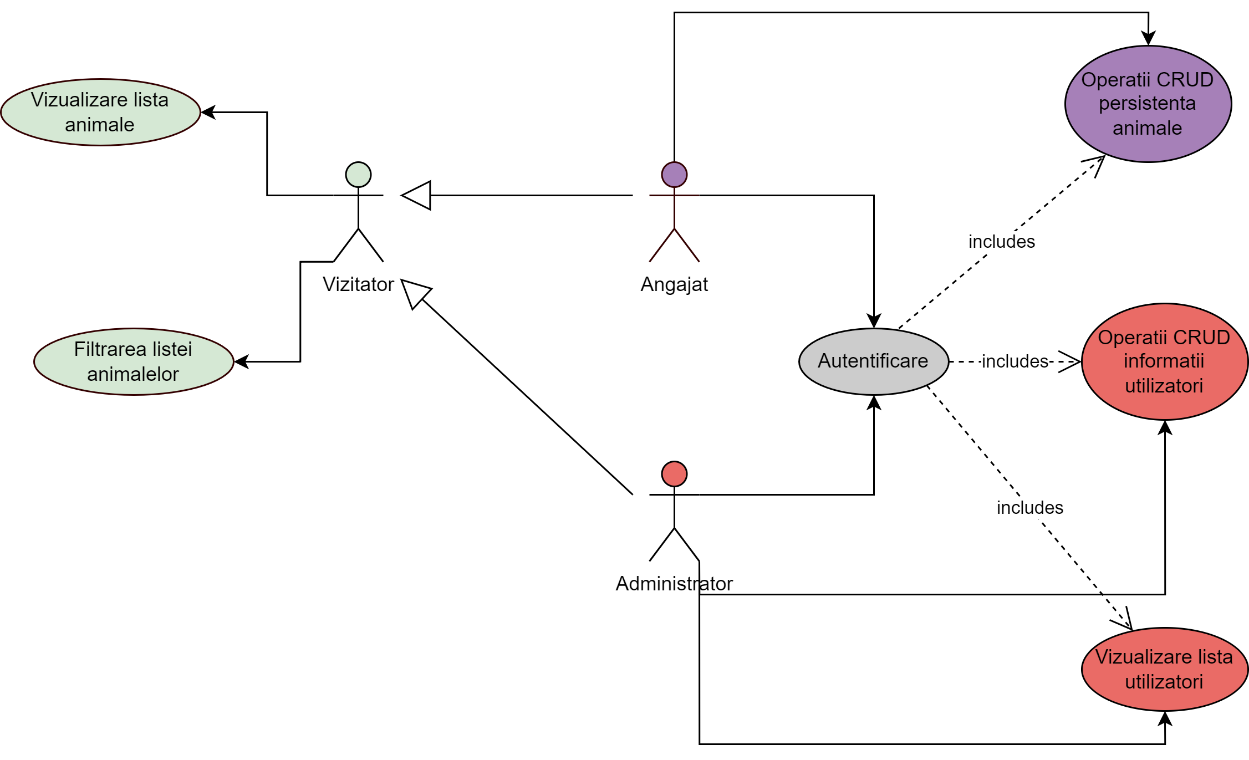


4.

DESCRIEREA APLICAȚIEI

Diagrama prezentată mai jos cuprinde use case scenarios pentru fiecare tip de user și ce acțiuni poate acesta să realize.

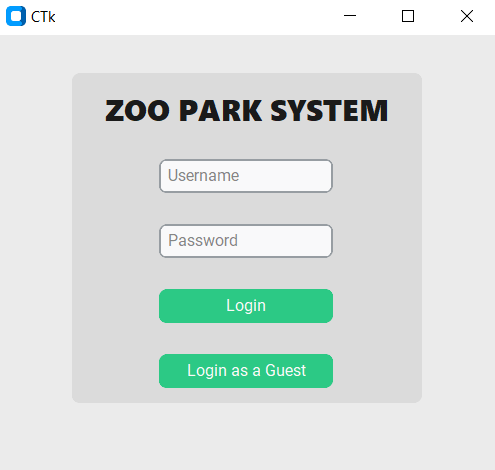
* Un *vizitator* accesează aplicația și vizualizează lista completă a animalelor din grădină, sortate după diferite criterii precum alimentație și specie. Utilizatorul poate aplica filtre pentru a personaliza vizualizarea în funcție de preferințele sale.
* Un *angajat* se autentifică în aplicație și are acces la operațiuni CRUD pentru gestionarea animalelor din grădină. Angajatul poate actualiza informațiile despre animale și poate asigura că datele sunt precise și actualizate în baza de date. Angajatul poate să efectueze și operațiile specifice unui vizitator.
* Un administrator se autentifică în aplicație și are acces la toate funcționalitățile disponibile pentru vizitatori și angajați. În plus, administratorul poate gestiona informațiile legate de utilizatori care necesită autentificare, inclusiv operații CRUD pentru aceste date. De asemenea, administratorul poate vizualiza lista utilizatorilor care necesită autentificare. Administratorul poate să efectueze și operațiile specifice unui vizitator.



**Figura 6.** Diagrama use case



Când aplicația este rulată pentru prima dată, utilizatorii sunt întâmpinați de o fereastră de login simplă. Aici, ei pot introduce informațiile lor de autentificare în două câmpuri de text dacă sunt angajați, urmate de două butoane - unul pentru "*Login*" și celălalt pentru "*Login As A Guest*".

Pentru angajații grădinii zoologice, fie ei simpli angajați sau administratori, autentificarea este cea mai importantă, fără aceasta ei nu își realiza task-urile. După introducerea datelor lor de acces și apăsarea butonului "Login", ei sunt direcționați către o pagină specializată, adaptată rolului lor în cadrul grădinii. Această direcționare se bazează pe informațiile preluate din baza de date a aplicației. Angajații au acces la task-urile specifice atribuite rolului lor - fie că este vorba despre actualizarea și gestionarea informațiilor despre animale(în cazul unui angajat simplu) sau despre administrarea conturilor și a altor aspecte de gestionare(în cazul unui administrator). Este important de menționat că un angajat nu are permisiuni de modificare a datelor despre alți utilizatori, ci doar asupra informațiilor legate de animale și nici invers pentru admnistrator.

În schimb, dacă un vizitator optează să acceseze aplicația fără a se autentifica și apasă butonul "Login As A Guest", o nouă fereastră se deschide pentru a-i oferi experiența dorită. Aici, vizitatorii pot explora și afla informații despre animalele grădinii zoologice, bucurându-se de experiența lor fără a fi nevoie să se autentifice sau să aibă un cont de utilizator.

Pentru un **utilizator de tip vizitator**, interacțiunea cu aplicația este simplificată și intuitivă. După ce s-au conectat, aceștia pot alege din mai multe acțiuni posibile:

* Filtrarea listei de animale: Utilizatorul poate selecta criteriul după care dorește să filtreze lista de animale din grădină. În zona de text, utilizatorul poate specifica valoarea după care să se realizeze filtrarea conform criteriului selectat. După ce au introdus criteriul și valoarea dorită, utilizatorul apasă butonul "Filter" pentru a aplica filtrarea.
* Afișarea tuturor animalelor: Utilizatorul poate apăsa butonul "Show All Animals" pentru a vizualiza lista completă a tuturor animalelor din grădină zoologică.



A screenshot of a computer

Description automatically generated

Lista completă de animale va fi afișată mai jos, oferindu-i utilizatorului o imagine de ansamblu a diversității și bogăției animalelor din grădină. Prin intermediul acestor acțiuni simple și directe, utilizatorii de tip vizitator pot explora și descoperi universul fascinant al animalelor din grădina zoologică, având posibilitatea de a personaliza experiența lor în funcție de interesele și preferințele individuale.

Dacă utilizatorul este angajat al grădinii zoologice(fie angajat simplu fie administrator) în urma apăsării butonului ‘Login’ acesta este direcționat către o nouă fereastră.

Când un utilizator de tip angajat se autentifică și este direcționat către o nouă fereastră, este întâmpinat de un mediu interactiv, unde poate alege între două opțiuni distincte:

* Acțiuni specifice vizitatorului sau angajatului:

Utilizatorul poate opta să efectueze acțiuni care sunt disponibile și pentru vizitatori. Aceste acțiuni includ afișarea listei de animale sau aplicarea unui filtru specific asupra listei.

* Acțiuni specifice postului de angajat la grădina zoologică:

În cazul în care utilizatorul dorește să efectueze acțiuni specifice poziției sale de angajat, este disponibil un meniu din care poate alege ce acțiune să realizeze. Acest meniu include opțiuni precum adăugarea unui animal, găsirea după un anumit criteriu, actualizarea sau ștergerea unui animal. După ce utilizatorul selectează una dintre aceste acțiuni, interfața grafică se adaptează, afișând diferite opțiuni și câmpuri în funcție de acțiunea selectată. De exemplu, pentru adăugarea unui animal, utilizatorul va fi întrebat să completeze informațiile despre noul animal.



A screenshot of a computer

Description automatically generated

Pentru un utilizator de tip administrator, interfața grafică este similară cu cea a angajatului, dar este înzestrată cu acțiuni specifice pentru gestionarea utilizatorilor. Aici, administratorul poate efectua acțiuni precum adăugarea, ștergerea sau actualizarea unui utilizator, gestionând astfel accesul și permisiunile utilizatorilor în aplicație.

Interfața grafică se adaptează în funcție de acțiunile pe care administratorul dorește să le efectueze, oferindu-i toate instrumentele necesare pentru administrarea eficientă a utilizatorilor și a conturilor lor.

5.

POSIBILITĂȚI DE DEZVOLTARE ULTERIOARĂ

Această aplicație a fost realizată în mod minimalist dar aceasta poate fi îmbunătățită și reconstruită și lansată la scară largă. Câteva dintre îmbunătățirile la care m-am gândit eu sunt:

1. *Implementarea unui sistem de raportare și analiză*: Adăugarea unei funcționalități care să permită generarea de rapoarte și analize statistice despre animalele din grădină zoologică.
2. *Integrarea unui sistem de notificare*: Implementarea unui sistem de notificare prin care angajații și administratorii să primească alerte și notificări în timp real despre evenimente importante sau actualizări ale datelor referitoare la animale sau utilizatori.
3. *Implementarea unui sistem de feedback și recenzii*: Adăugarea unei secțiuni în aplicație unde vizitatorii să poată lăsa feedback și recenzii despre experiența lor în grădina zoologică. Aceste feedback-uri pot fi folosite pentru a îmbunătăți serviciile și experiența vizitatorilor.