MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PROIECTARE SOFTWARE

DOCUMENTATIE PROIECT

Firma de colectare deseuri

Filip Iarina-Eden GRUPA 30231

Cuprins

1.Problema de rezolvat

Obiectiv:

Obiectivul acestui proiect este familiarizarea cu șablonul architectural Client/Server, cu șabloanele arhitecturale orientate pe servicii (SOA) și cu șabloanele de proiectare. Pentru persistența informației se va utiliza o bază de date relațională (SQL Server, MySQL, etc.).

Cerințe:

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:

➢ proiectarea dată de diagramele de clase și diagramele de secvență;

➢ 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 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 22

Dezvoltați o aplicație client/server care poate fi utilizată de către o firmă de colectare deșeuri. Aplicația va avea 3 tipuri de utilizatori: angajat, coordonator activitate și administrator.

Utilizatorii de tip angajat pot efectua următoarele operații după autentificare:

❖ Vizualizarea listei cu adresele deșeurilor ce urmează a fi colectate de angajat sortată după adresă;

❖ Vizualizarea traseului optim în funcție de adresa deșeurilor (reprezentare grafică);

❖ Căutarea adresei unui deșeu alocat angajatului după număr și setarea stării acestuia în colectat după colectarea deșeului.

Utilizatorii de tip coordonator activitate pot efectua următoarele operații după autentificare:

❖ Vizualizarea listei tuturor deșeurilor sortată după adresa de colectare;

❖ Filtrarea deșeurilor după starea acestora:

➢ Nealocat unui angajat;

➢ Alocat unui angajat, dar necolectat;

➢ Colectat;

❖ Operații CRUD în ceea ce privește persistența adreselor cu deșeuri;

❖ Alocarea unei adrese cu deșeuri către un un angajat în vederea colectării;

❖ Salvare liste cu informații despre listele cu adresele deșeurilor alocate angajaților în mai multe formate: csv, json, xml, doc;

❖ Vizualizarea unor statistici legate de deșeuri (structură radială, structură inelară, de tip coloană, etc.).

Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

❖ Operații CRUD pentru informațiile legate de utilizatori;

❖ Vizualizarea listei tuturor utilizatorilor;

❖ Filtrarea listei utilizatorilor după tipul utilizatorilor;

❖ Vizualizarea unei statistici cu utilizatorii în funcție de tipul acestora;

❖ 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.

2.2.Diagrama cazurilor de utilizare(Figura 1)

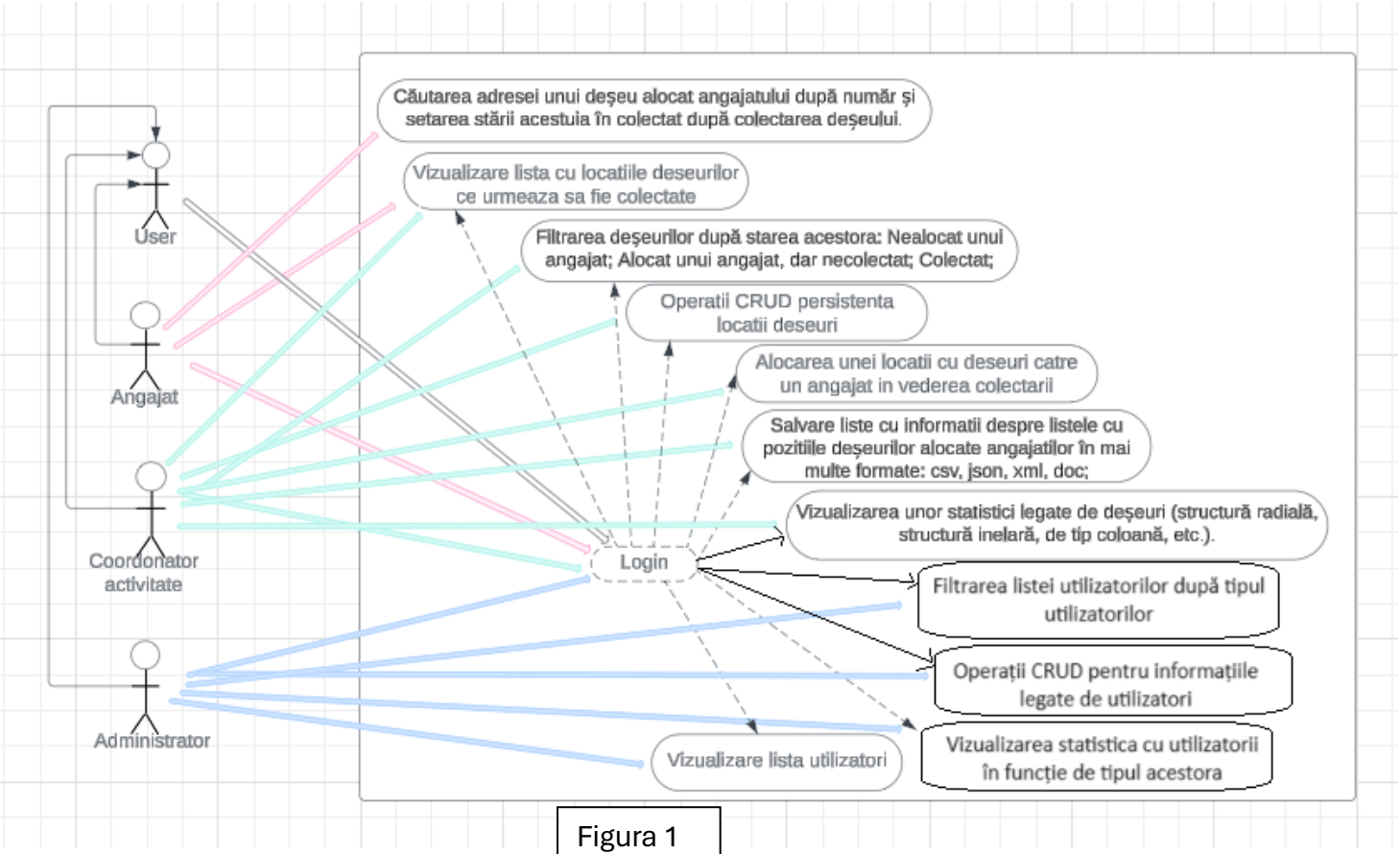

Figura 1

2.3.Diagramele de activitati(Figura 2)

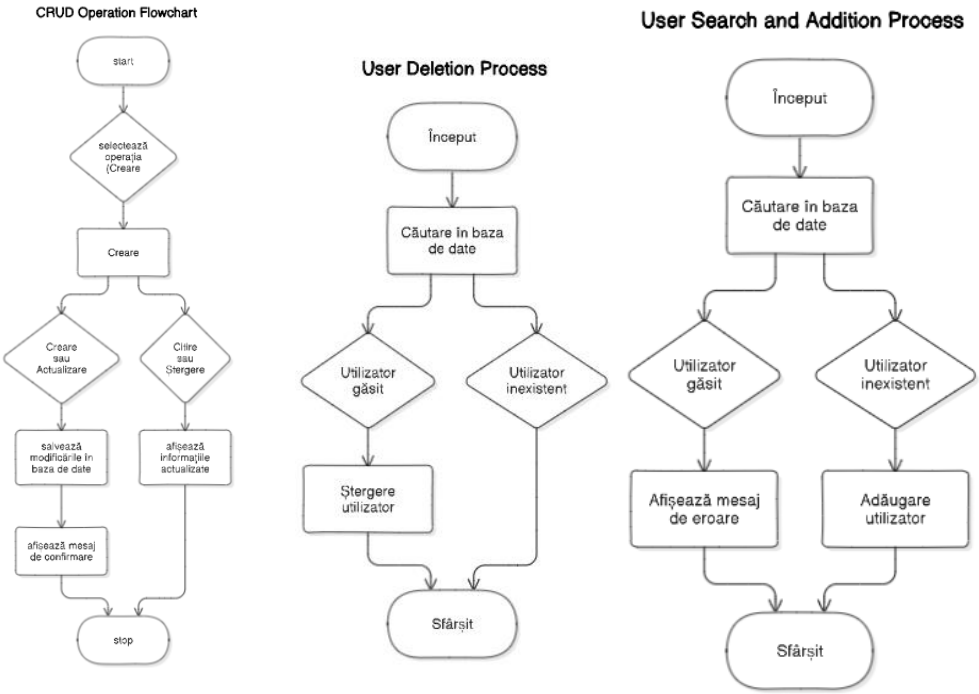

Figura 2

2.4.Diagramele de secventa(Figura 3)

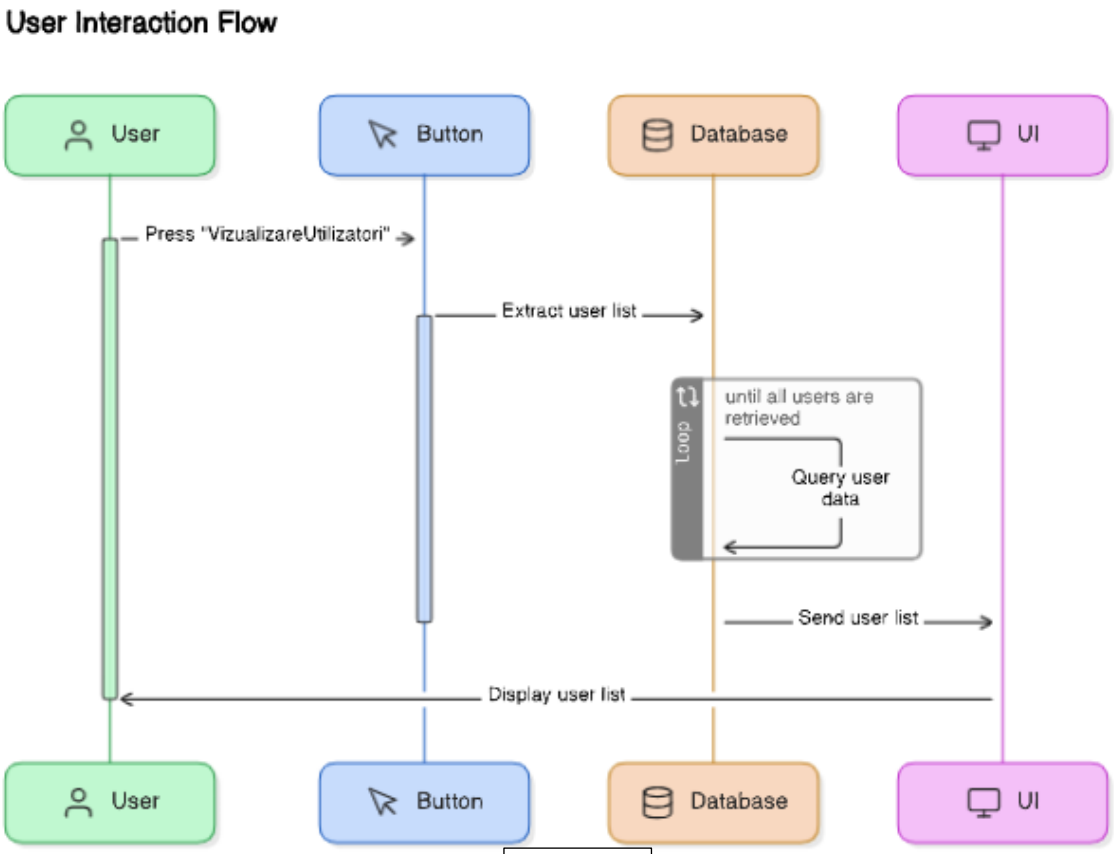

Figura 3

3.Faza de proiectare – Diagrama de clase(Figura 4)

3.1.Structura bazei de date

Tabelele din baza de date:

Această schemă de bază a bazei de date conține următoarele tabele:

1.administrator: Această tabelă stochează informații despre administratorii sistemului. Fiecare administrator are un ID unic (admin_id), nume, email, parolă și număr de telefon.

2.coordonator: Tabelul coordonatorului conține informații despre coordonatorii sistemului. Fiecare coordonator are un ID unic (coordonator_id), nume, email și parolă.

3.angajat: Acest tabel reține detalii despre angajații companiei. Fiecare angajat are un ID unic (angajat_id), nume, email, parolă, departament și ani de experiență.

4.locatieDeseu: Tabelul locației deșeurilor stochează informații despre diferitele locații deșeu disponibile. Fiecare locație deșeu are un ID unic (idLocatie), adresa locației, tipul deșeului și stadiul său (cum ar fi "colectare" sau "adunare").

5.locatieAngajat: Această tabelă realizează o asociere între locațiile deșeurilor și angajați. Fiecare înregistrare conține un ID unic (idLocatieAngajat), ID-ul locației și ID-ul angajatului asociat. Acest tabel este utilizat pentru a atribui angajați la diverse locații deșeu.

În plus, interogările INSERT INTO sunt utilizate pentru a adăuga date de exemplu în tabele.

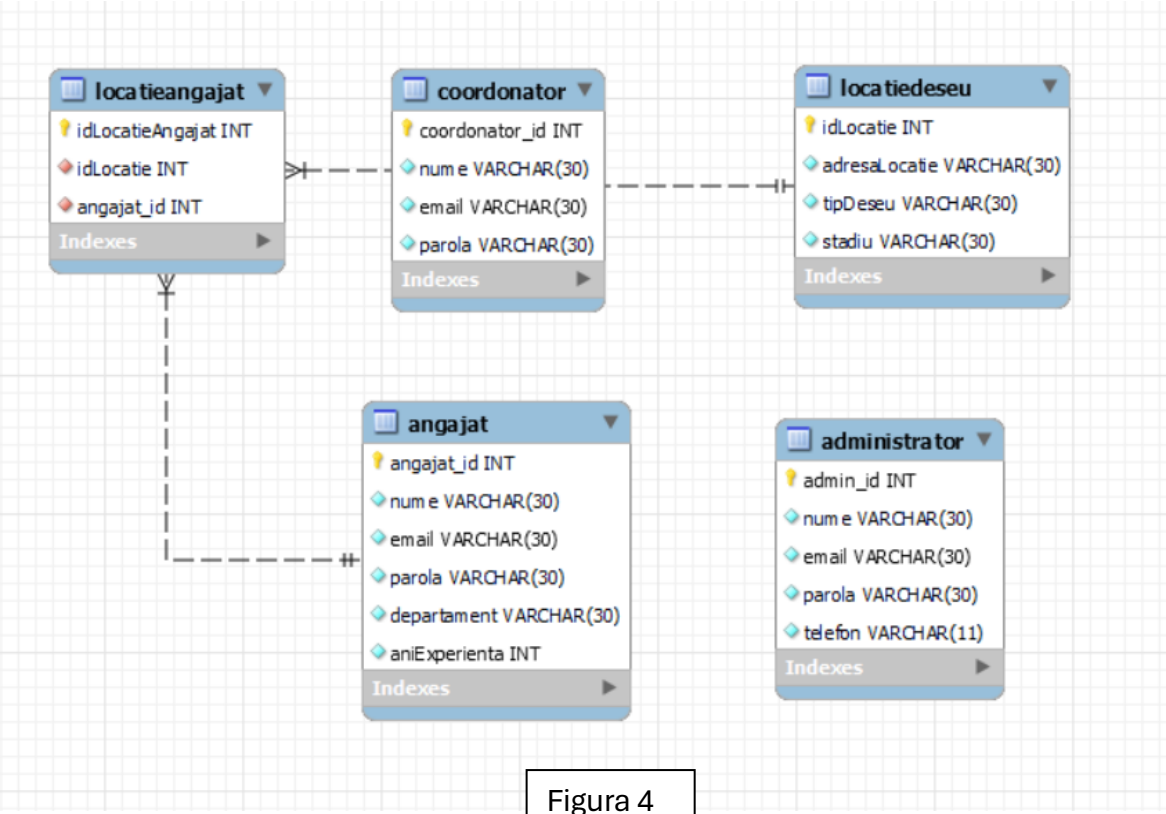

Figura 4

3.2.Diagrama complete de clase(Figura 5)

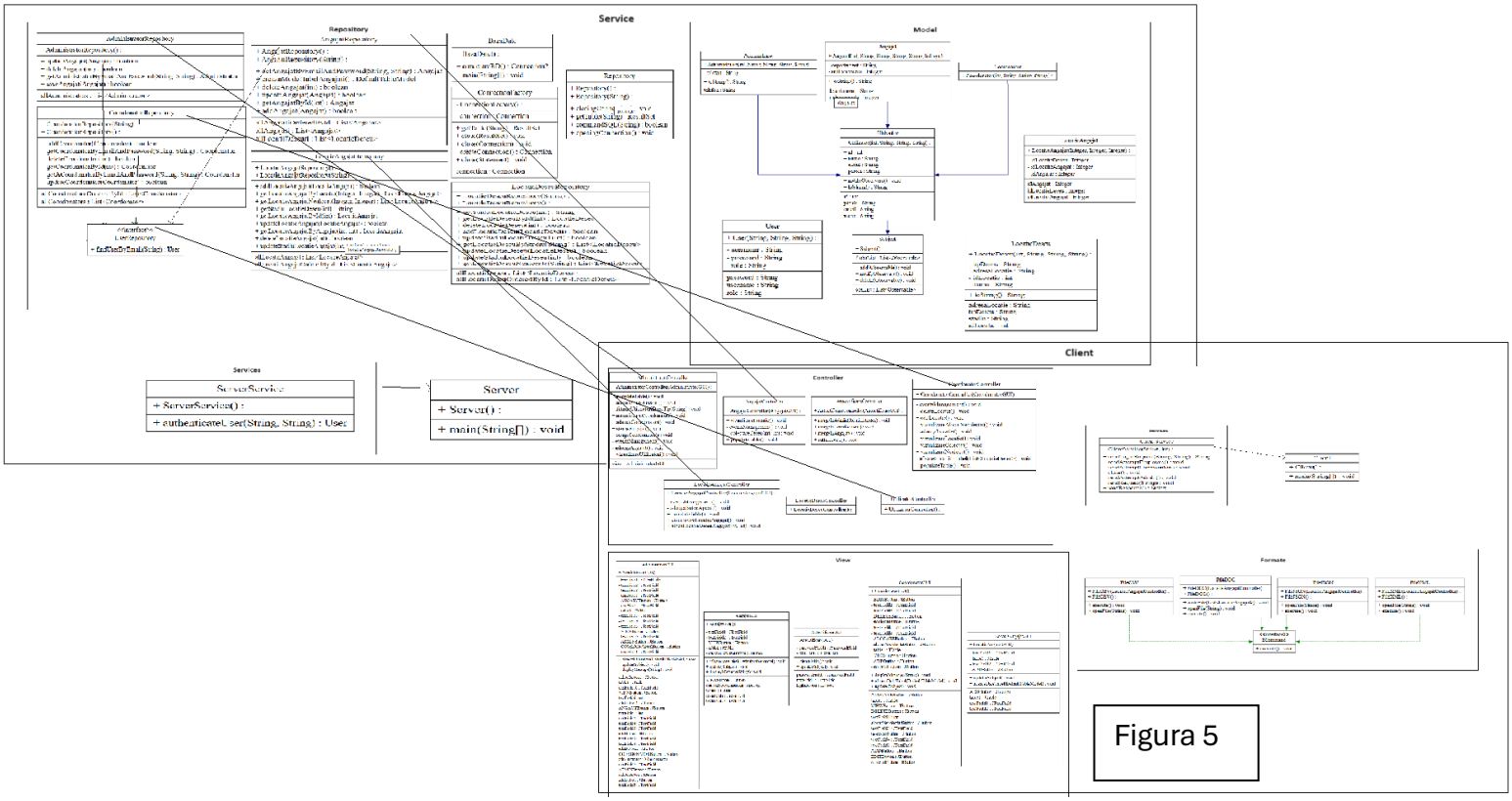

Figura 5

3.3. Diagrama Server(Figura 6):

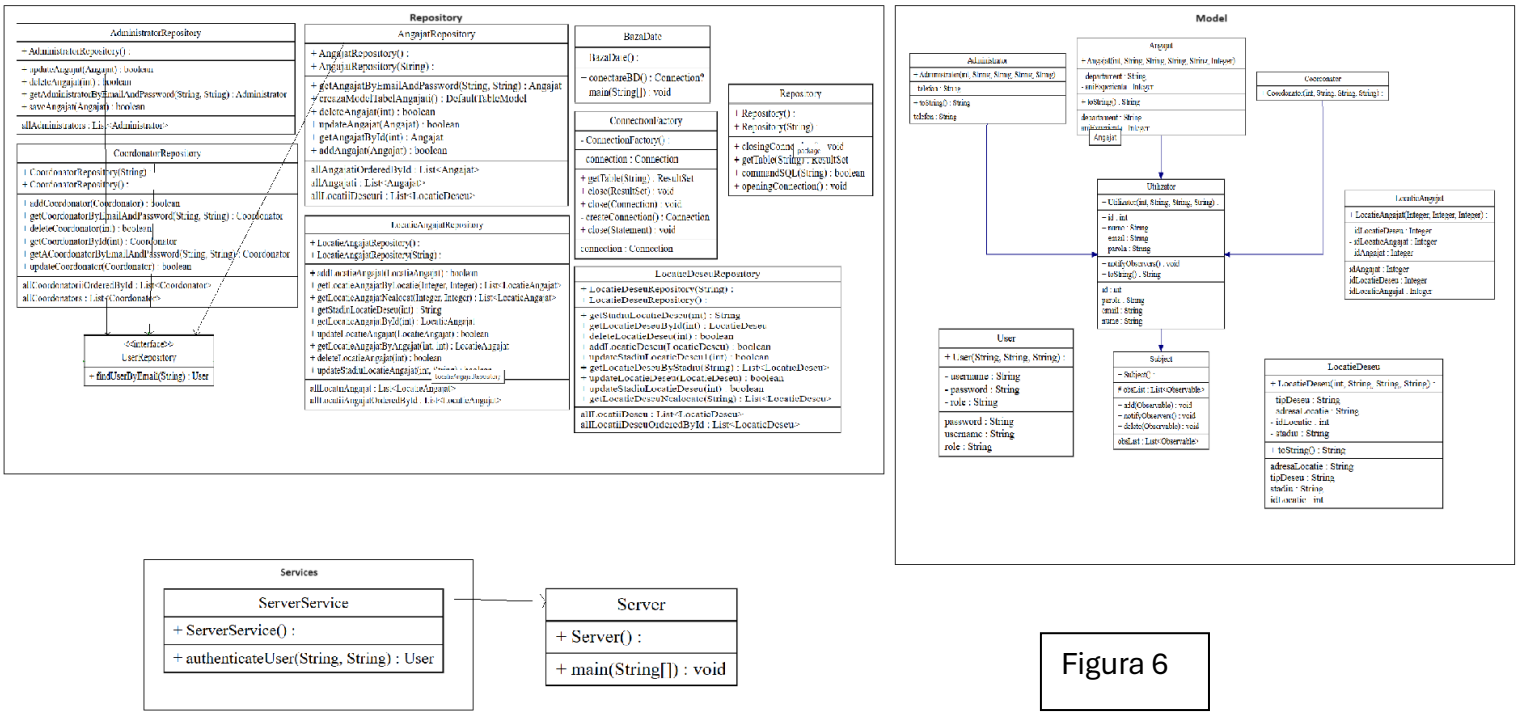

Figura 6

3.3.1.Pachetul Model(Figura 7)

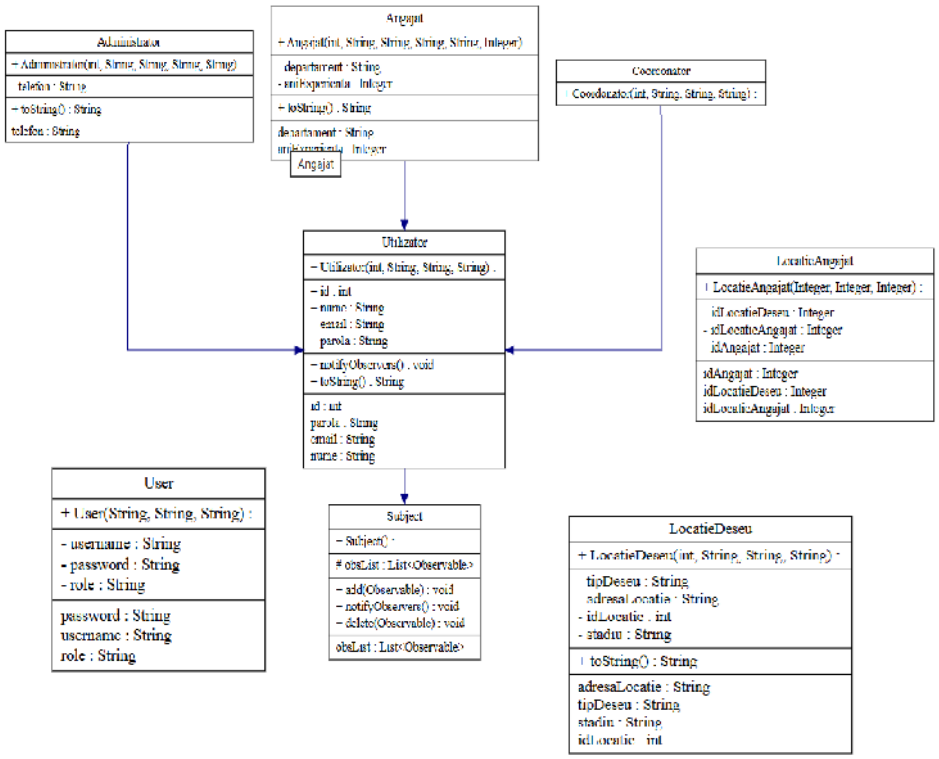

Figura 7

Pachetul Model conține clase care modelează structurile de date și comportamentele asociate acestora în cadrul aplicației. Iată o descriere succintă a claselor din acest pachet:

1.Angajat: Reprezintă un angajat al companiei și conține informații precum nume, email, parolă, departament și ani de experiență.

2.Administrator: Clasa care modelează un administrator al sistemului. Are atribuțiile sale specifice și conține informații similare cu cele ale unui angajat.

3.Coordonator: Clasa care reprezintă un coordonator al sistemului. Asemănător cu administratorul și angajatul, conține detalii personale precum nume, email și parolă.

4.Utilizator: O clasă generală care poate servi drept bază pentru clasele specifice de angajat, administrator și coordonator.

5.LocatieAngajat: Această clasă modelează asocierea între angajați și locațiile deșeurilor. Stochează informații precum ID-ul locației și ID-ul angajatului asociat.

6.LocatieDeseu: Reprezintă o locație specifică deșeu, cum ar fi o adresă și tipul deșeului pe care îl colectează.

7.Observable și Subject: Acestea sunt interfețele utilizate pentru implementarea modelului de proiectare Observator-Observabil, care permite notificarea automată a obiectelor interesate despre modificările survenite în alte obiecte.

8. User: este folosita de Client si imparte utilizatorii in functie de tipul de utilizator care doreste sa se logheze.

Împreună, clasele din pachetul Model definesc structurile de bază și comportamentele necesare pentru gestionarea angajaților, administratorilor, coordonatorilor și locațiilor deșeurilor în cadrul aplicației.

3.3.2.Subpachetul Repository(Figura 8)

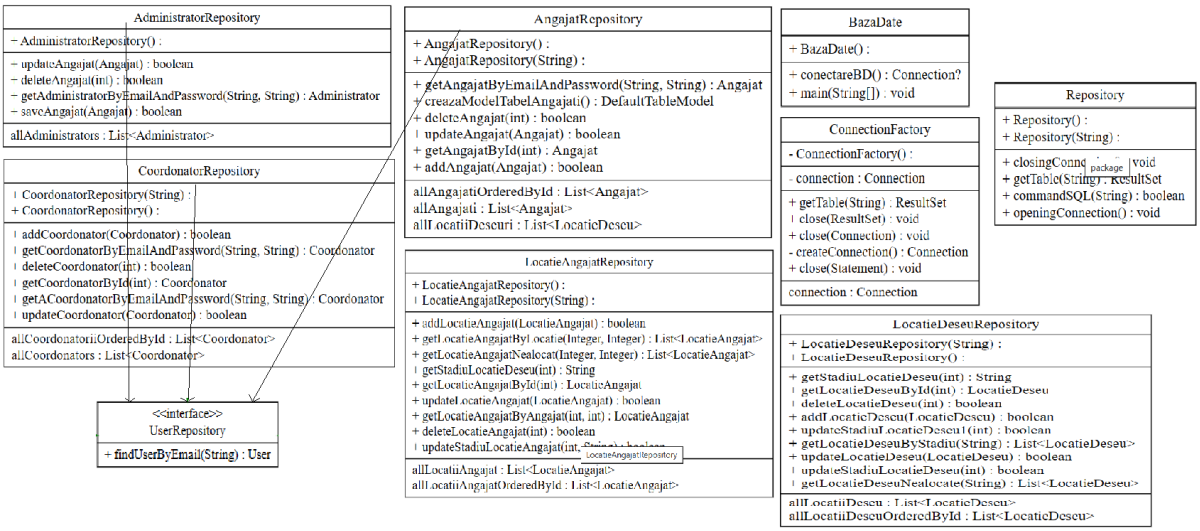

Figura 8

Subpachetul Repository conține clase care se ocupă de interacțiunea cu baza de date și gestionarea datelor. Iată o descriere succintă a claselor din acest subpachet:

1.AngajatRepository: Gestionează operațiile de citire, scriere, actualizare și ștergere pentru entitățile Angajat din baza de date.

2.AdministratorRepository: Similar cu AngajatRepository, dar se concentrează pe entitățile Administrator.

3.CoordonatorRepository: Gestionează operațiile CRUD pentru entitățile Coordonator.

4.LocatieAngajatRepository: Se ocupă de operațiile legate de asocierea între angajați și locațiile deșeurilor. Aici sunt gestionate operațiile specifice pentru această relație.

5.LocatieDeseuRepository: Gestionează operațiile legate de locațiile deșeurilor, precum citirea, scrierea și actualizarea acestora în baza de date.

6.Repository: Este o clasă abstractă care conține metode generice pentru a efectua operațiile CRUD de bază pe orice entitate din baza de date.

7.ConnectionFactory: Clasa responsabilă pentru crearea și gestionarea conexiunii cu baza de date.

8.BazaDate: O clasă care conține informații de configurare pentru conexiunea la baza de date, precum URL-ul bazei de date, numele utilizatorului și parola.

9. UserRepository: este o interfață care definește metoda necesară pentru a găsi un utilizator în baza de date folosind adresa de email. Aceasta oferă un contract pe care alte clase de repository specific (de exemplu, AngajatRepository, CoordonatorRepository, AdministratorRepository) trebuie să-l implementeze.

Împreună, clasele din subpachetul Repository formează un strat de acces la date flexibil și modular, care facilitează interacțiunea aplicației cu baza de date. Aceste clase îmbunătățesc gestionarea datelor și abstractizează detaliile specifice bazei de date de restul aplicației.

3.3.3.Pachetul Services(Figura 9)

Clasa ServerService gestionează operațiunile legate de autentificarea utilizatorilor în sistem. Aceasta interacționează cu diferite repository-uri pentru a valida credențialele utilizatorilor și a returna obiectele utilizator corespunzătoare.

Clasa Server inițializează un server socket care ascultă pe un anumit port și gestionează conexiunile de la clienți. Aceasta procesează cererile de autentificare și alte comenzi de la clienți, comunicând cu ServerService pentru a valida utilizatorii.

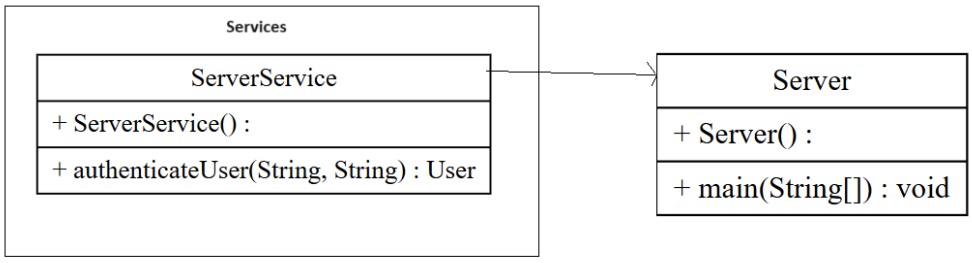

Figura 9

3.4.Client(Figura 10)

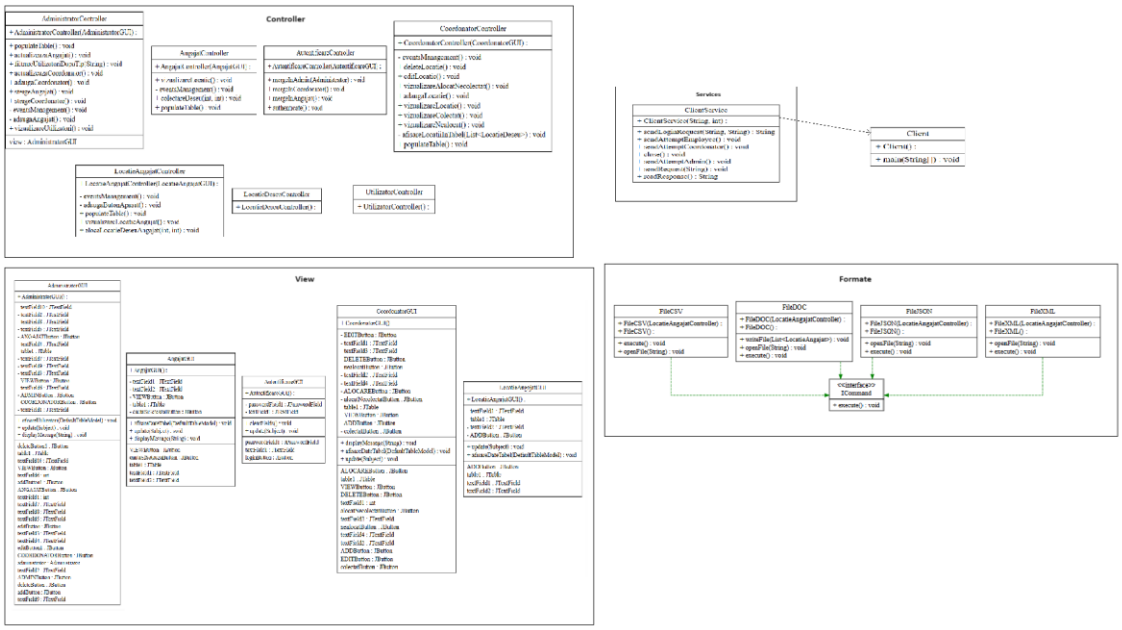

Figura 10

3.4.1.Pachetul Controller(Figura 11)

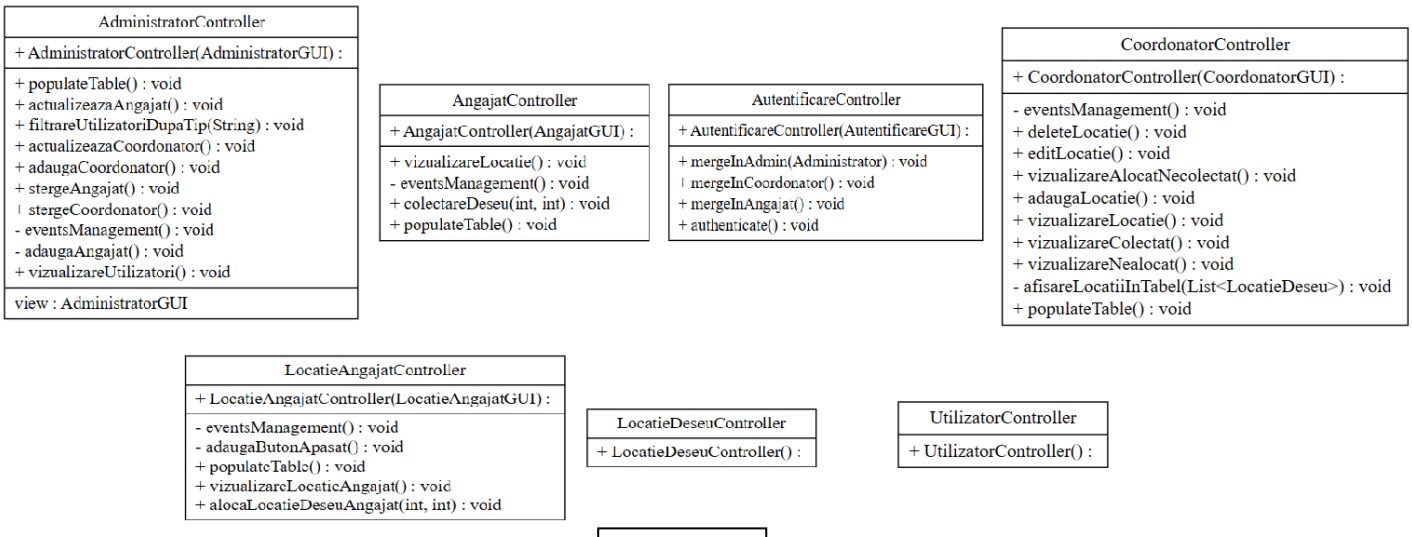

Figura 11

Pachetul Controller conține clasele care se ocupă de logica aplicației și de gestionarea interacțiunilor dintre interfața utilizator și modelul de date. Iată o descriere sumară a claselor din acest pachet:

1.AngajatController: Această clasă gestionează interacțiunile specifice pentru angajați. Ea se ocupă de prelucrarea datelor de intrare ale angajaților, efectuarea operațiilor pe baza acestora și actualizarea interfeței utilizatorului cu rezultatele corespunzătoare.

2.AdministratorController: Similar cu AngajatController, dar destinat administratorilor. Gestionează operațiunile specifice de administrare a sistemului și de gestionare a datelor asociate administratorilor.

3.CoordonatorController: Acesta este controllerul destinat coordonatorilor. Se ocupă de gestionarea operațiilor și funcționalităților specifice atribuite coordonatorilor în cadrul aplicației.

4.UtilizatorController: Această clasă gestionează operațiunile generice care sunt comune tuturor utilizatorilor, cum ar fi autentificarea și gestionarea sesiunilor de utilizator.

5.LocatieAngajatController: Controllerul responsabil pentru gestionarea operațiunilor legate de asocierea locațiilor deșeurilor cu angajați. Se ocupă de procesarea cererilor legate de aceste operațiuni și de actualizarea corespunzătoare a modelului și a interfeței utilizatorului.

6.LocatieDeseuController: Similar cu LocatieAngajatController, dar se ocupă de operațiunile specifice asociate locațiilor deșeurilor. Gestionarea adăugării, actualizării și ștergerii locațiilor deșeurilor și actualizarea corespunzătoare a modelului și a interfeței utilizatorului.

Aceste clase din pachetul Controller reprezintă componentele esențiale ale logicii aplicației și asigură funcționarea corespunzătoare a acesteia, gestionând fluxul de date și interacțiunile utilizatorului în mod eficient și fiabil.

3.4.2.Pachetul Formate(Figura 12)

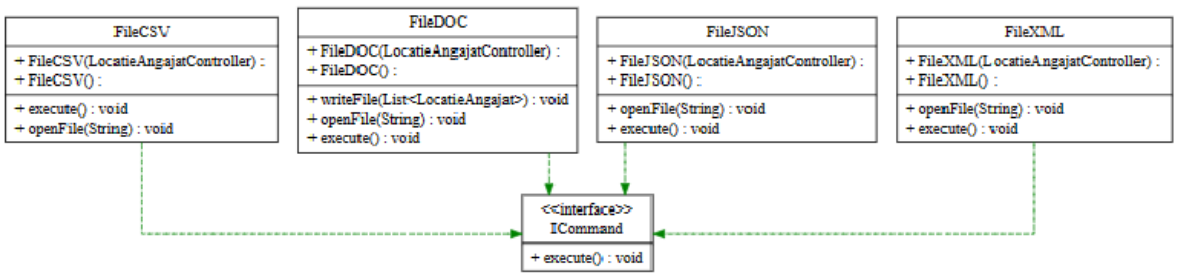

Figura 12

Pachetul Formate conține clasele responsabile pentru manipularea și gestionarea diferitelor formate de fișiere în care datele pot fi salvate sau încărcate. Iată o descriere succintă a claselor din acest pachet:

1.FileJSON: Această clasă se ocupă de manipularea fișierelor în format JSON. Ea oferă metode pentru serializarea și deserializarea datelor în/din format JSON, permițând astfel salvarea și încărcarea acestora din fișiere JSON.

2.FileXML: Similar cu FileJSON, dar pentru fișierele XML. Această clasă oferă funcționalitate pentru a salva și încărca datele în/din fișiere XML, utilizând formatul XML pentru a structura și organiza datele.

3.FileDOC: Această clasă gestionează manipularea fișierelor în format DOC (Microsoft Word). Ea permite crearea, modificarea și salvarea documentelor în format DOC, oferind astfel posibilitatea de a crea documente text complexe sau formate pentru rapoarte și alte scopuri.

4.FileCSV: Această clasă se ocupă de manipularea fișierelor în format CSV (Comma-Separated Values). Fișierele CSV sunt utilizate frecvent pentru stocarea și schimbul de date tabulare. Clasa oferă metode pentru salvarea și încărcarea datelor din/din fișiere CSV și manipularea acestora sub formă de tabele cu valori separate prin virgulă.

5.ICommand: Această este o interfață care definește metoda execute() pentru a executa o comandă specifică. Este utilizată pentru a implementa diferite comenzi în aplicație și pentru a separa logica de execuție a comenzilor de interfața utilizator.

Aceste clase din pachetul Formate oferă instrumentele necesare pentru gestionarea datelor în diferite formate de fișiere, permițând astfel salvarea și încărcarea datelor în funcție de preferințele și cerințele utilizatorului sau ale aplicației.

3.4.3.Pachetul View(Figura 13)

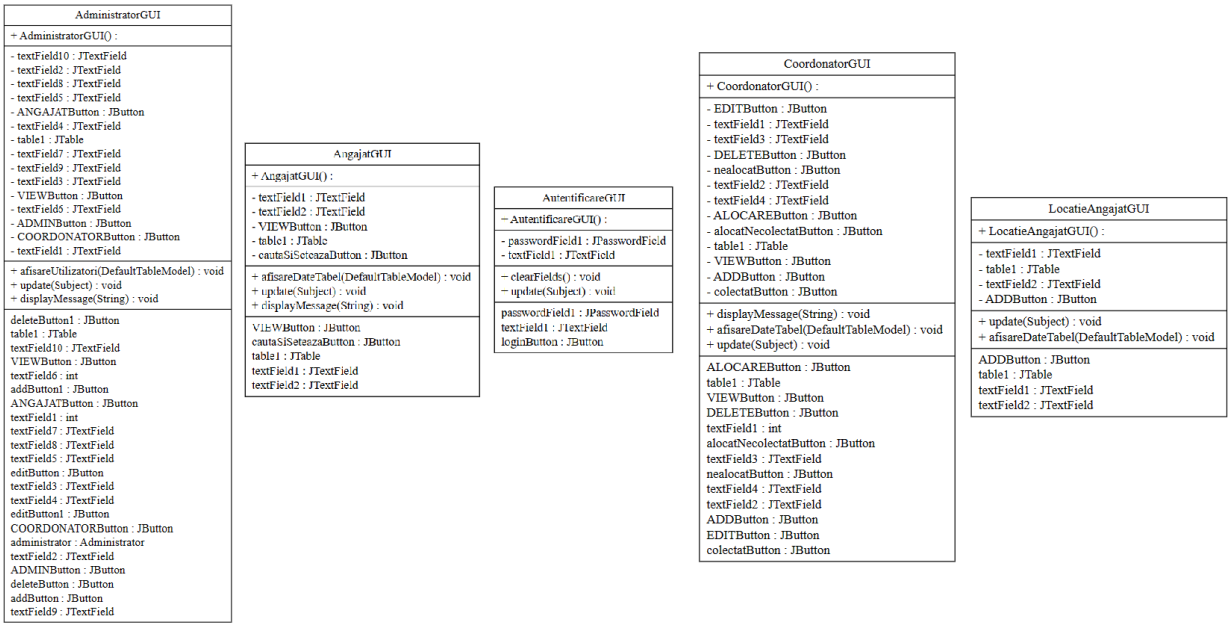

Figura 13

Pachetul View conține clasele responsabile cu interfața grafică a aplicației. Iată o descriere sumară a claselor din acest pachet:

1.AngajatGUI: Acesta este GUI-ul destinat angajaților. Furnizează o interfață utilizator intuitivă pentru interacțiunea cu sistemul din perspectiva angajatului.

2.AdministratorGUI: Similar cu AngajatGUI, dar destinat administratorilor. Oferă funcționalități specifice pentru administrarea sistemului.

3.CoordonatorGUI: Acesta este GUI-ul atribuit coordonatorilor. Permite coordonatorilor să gestioneze și să monitorizeze activitățile specifice.

4.AutentificareGUI: Interfața de autentificare a utilizatorilor în aplicație. Aici utilizatorii introduc datele lor de autentificare pentru a accesa funcționalitățile aplicației.

5.LocatieAngajatGUI: Interfața grafică pentru gestionarea și vizualizarea locațiilor deșeurilor asociate angajaților. Permite adăugarea, modificarea și ștergerea acestor asocieri.

Aceste clase din pachetul View furnizează o interfață prietenoasă utilizatorului pentru a interacționa cu aplicația, facilitând gestionarea datelor și efectuarea operațiilor specifice într-un mod eficient și intuitiv.

4.Faza de implementare – Aplicatie

4.1.Instrumente utilizate

În cadrul dezvoltării acestei aplicații, au fost utilizate diverse instrumente și tehnologii pentru a facilita procesul de implementare, testare și gestionare a proiectului. Mai jos sunt descrise principalele instrumente utilizate:

1.Java Development Kit (JDK)

- Versiune: JDK 8 sau mai recentă.
- Rol: Set de unelte pentru dezvoltarea și rularea aplicațiilor Java, inclusiv compilatorul javac, motorul de rulare java și alte utilitare esențiale pentru dezvoltarea aplicațiilor.

2.IntelliJ IDEA

- Versiune: Community Edition.
- Rol: IDE (Integrated Development Environment) folosit pentru scrierea, depanarea și testarea codului Java. Oferă suport avansat pentru proiectele Java, facilitând gestionarea dependențelor, automatizarea build-urilor și integrarea cu sistemele de control al versiunilor.

3.MySQL

- Versiune: MySQL Server 5.7 sau mai recent.
- Rol: Sistem de gestionare a bazelor de date relaționale utilizat pentru stocarea informațiilor despre utilizatori și alte date relevante ale aplicației. Interacțiunea cu baza de date se face prin intermediul JDBC (Java Database Connectivity).

4.Maven

- Versiune: Maven 3.6.3 sau mai recent.
- Rol: Instrument de management al proiectelor și de automatizare a build-urilor utilizat pentru gestionarea dependențelor și a procesului de build. Fişierul pom.xml definește toate dependențele externe necesare proiectului.

5.Swing (Java Swing)

- Rol: Bibliotecă GUI (Graphical User Interface) utilizată pentru dezvoltarea interfeței grafice a aplicației. Swing oferă componente grafice precum butoane, ferestre, câmpuri de text, necesare pentru realizarea interacțiunii cu utilizatorul.

6.JDBC (Java Database Connectivity)

- Rol: API Java pentru conectarea și executarea interogărilor asupra bazelor de date relaționale. Utilizat pentru interacțiunea aplicației Java cu baza de date MySQL, facilitând operațiunile de CRUD (Create, Read, Update, Delete).

Aceste instrumente au fost esențiale în dezvoltarea, testarea și implementarea aplicației, asigurând un flux de lucru eficient și o calitate ridicată a codului produs.

7.Arhitectura Aplicației

Arhitectura Server-Client

Pentru implementarea aplicației, s-a folosit o arhitectură server-client. Această arhitectură implică faptul că există două componente distincte ale sistemului:

- Serverul: O aplicație care rulează în mod continuu și oferă servicii și resurse pentru clienți. Serverul gestionează cererile primite de la clienți și răspunde în consecință.
- Clientul: O aplicație interactivă utilizată de utilizatori pentru a accesa serviciile și resursele oferite de server. Clientul trimite cereri către server și primește răspunsuri.

Prin adoptarea acestei arhitecturi, s-au obținut beneficii precum scalabilitatea sistemului, gestionarea centralizată a datelor și securitatea sporită.

4.2.Aplicatia

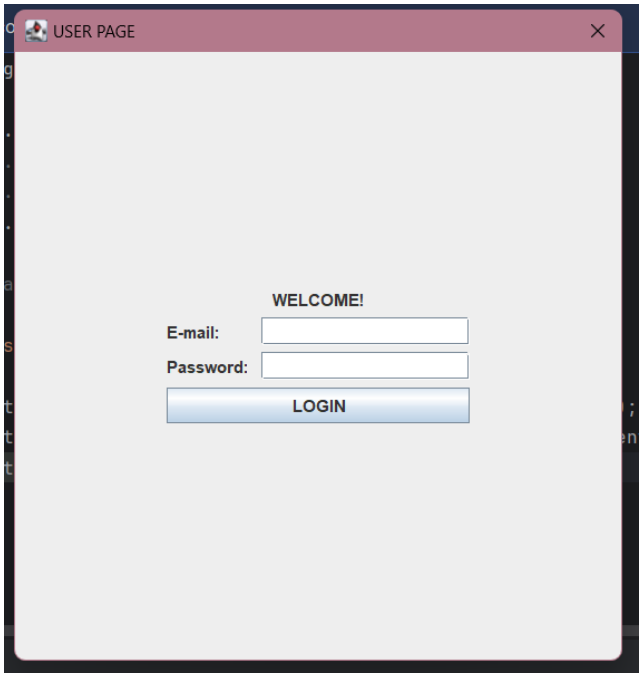

Figura 14

Aceasta este pagina care apare la inceput, cand pornesti aplicatie si ai posibilitatea sa te loghezi ca administrator, coordinator sau angajat(Figura 14).

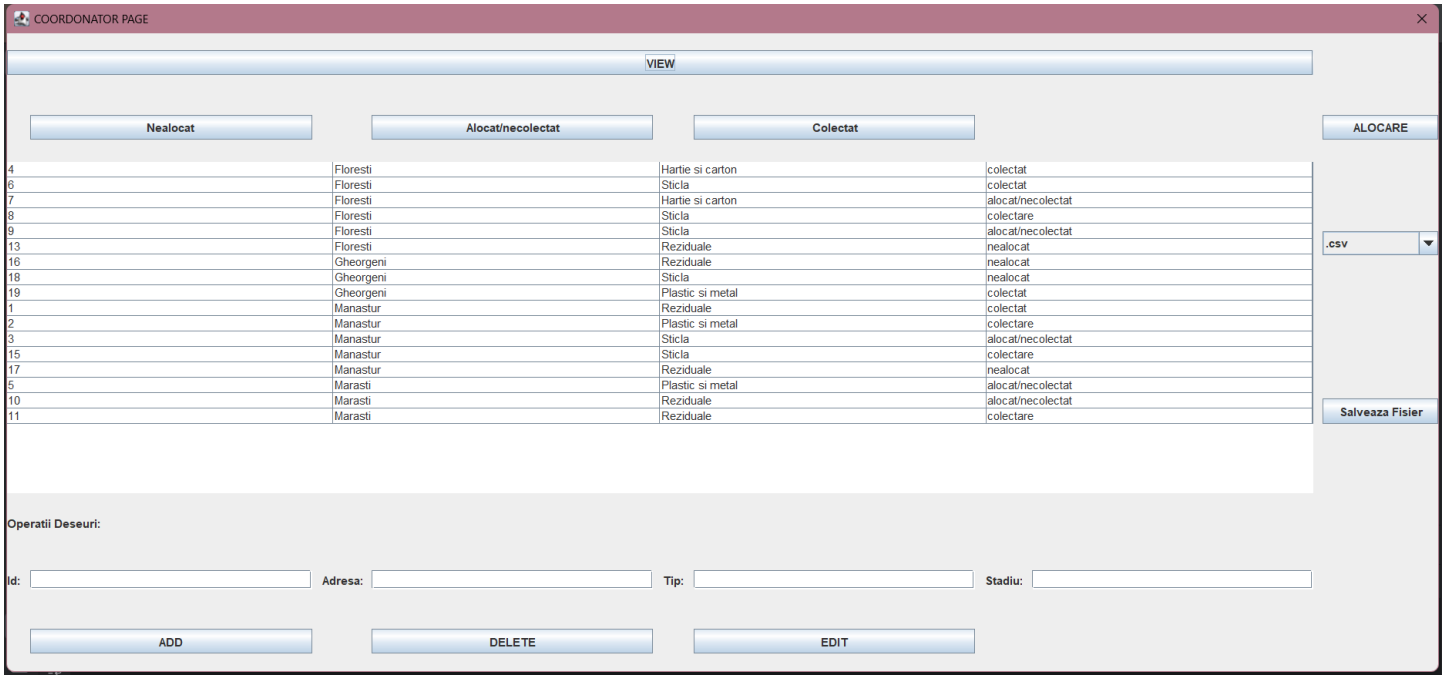

Figura 15

Aceasta este pagina administratorului, unde poate vizualiza tabelul cu utiliztaorii si poate face operatiile CRUD pe ele. Pe langa asta se pot vizualiza filtrat utilizatorii in functie de tipul lor(Figura 15).

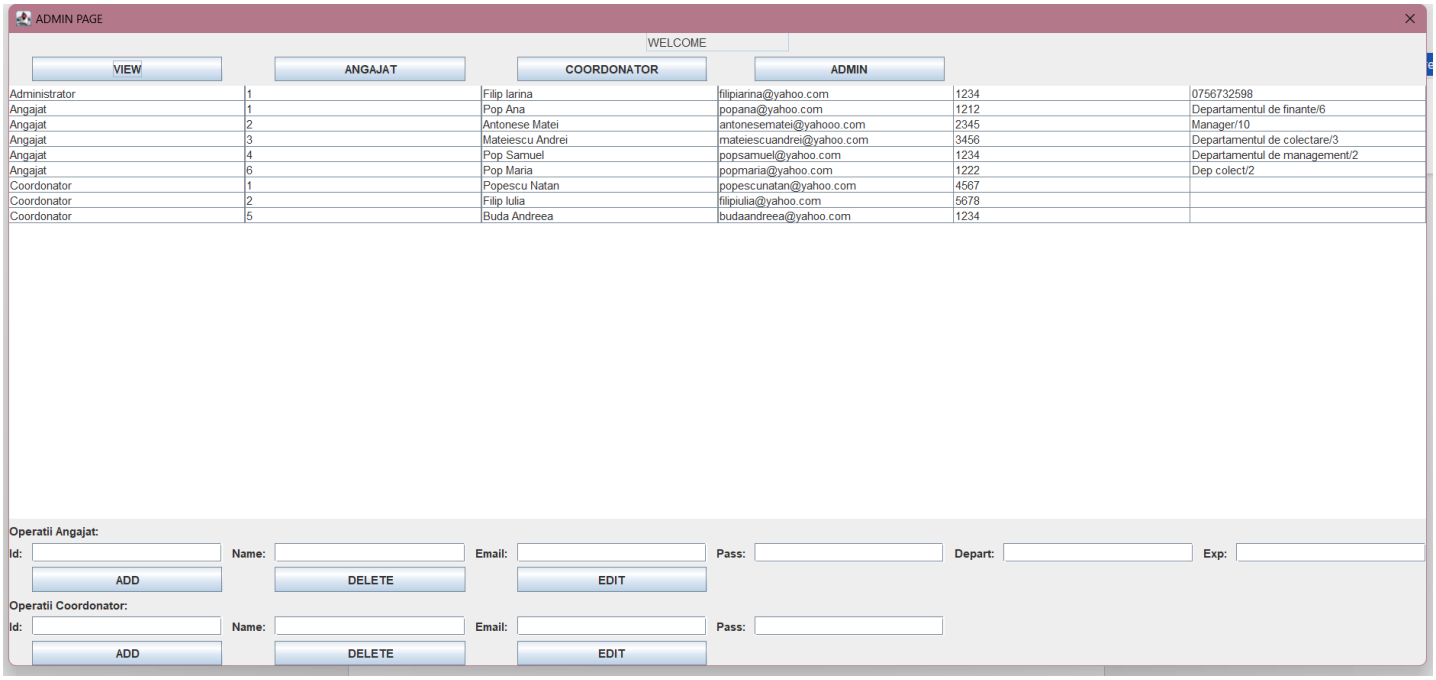

Figura 16

Aceasta este pagina coordonatorului, unde poti vizualiza tabelul cu locatii si poti face operatiile CRUD pe el. De altfel poti filtra si locatiile deseurilor in functie de stadiul lor. Mai poti alege sa generezi fisiere de mai multe tipuri: csv, json, doc si xml. Pe langa toate acestea se mai poate deschide o pagina care sa faca o alocare a unei locatii la un angajat.(Figura 16)

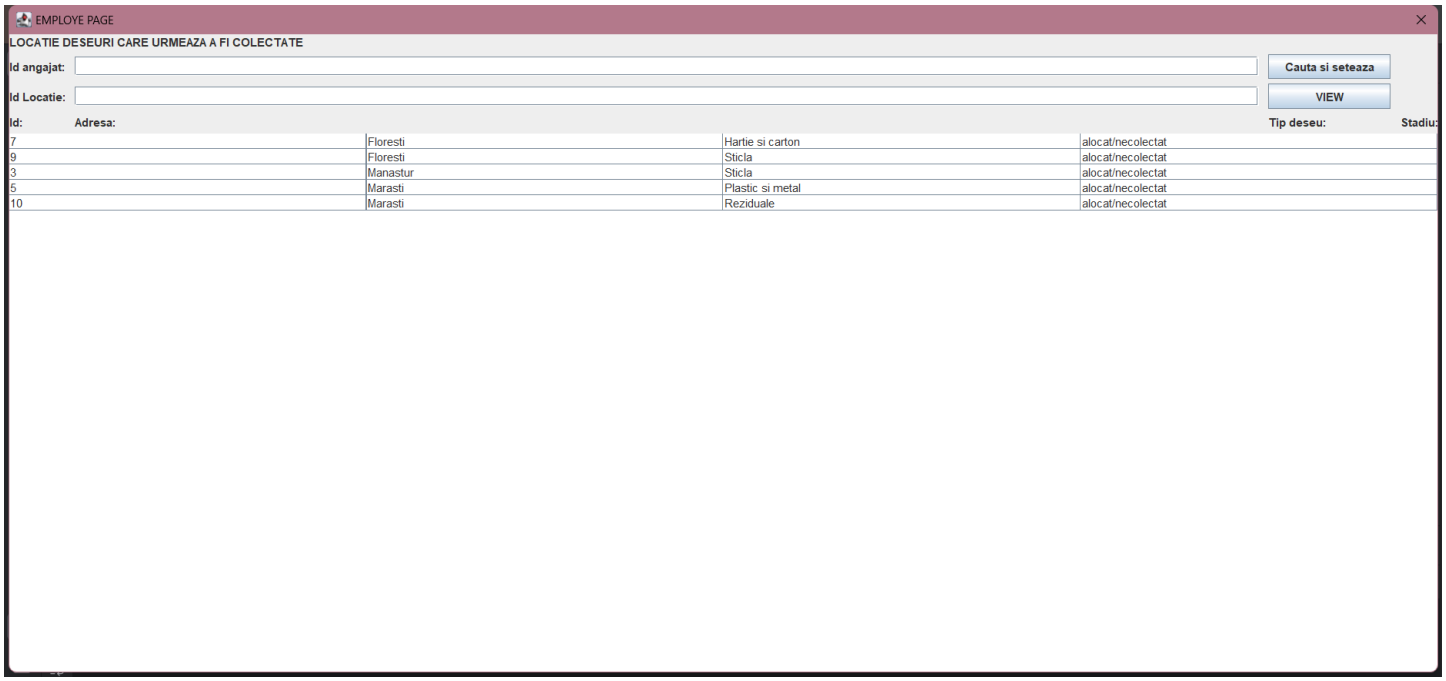

Figura 17

Aceasta este pagina unde se poate face alocarea unel locatii catre un angajat in vederea colectarii(Figura 17).

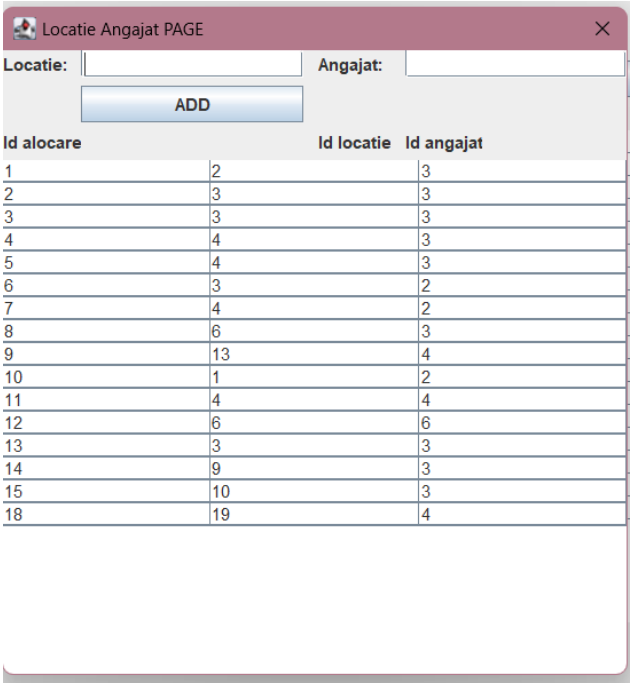

Figura 18

Aceasta este pagina angajatului in care poate vizualiza locatiile care sunt allocate/necolectate si sa caute o locatie atribuita unui angajat si in momentul in care este cautata se seteaza starea la colectat(Figura 18).

Statistici:

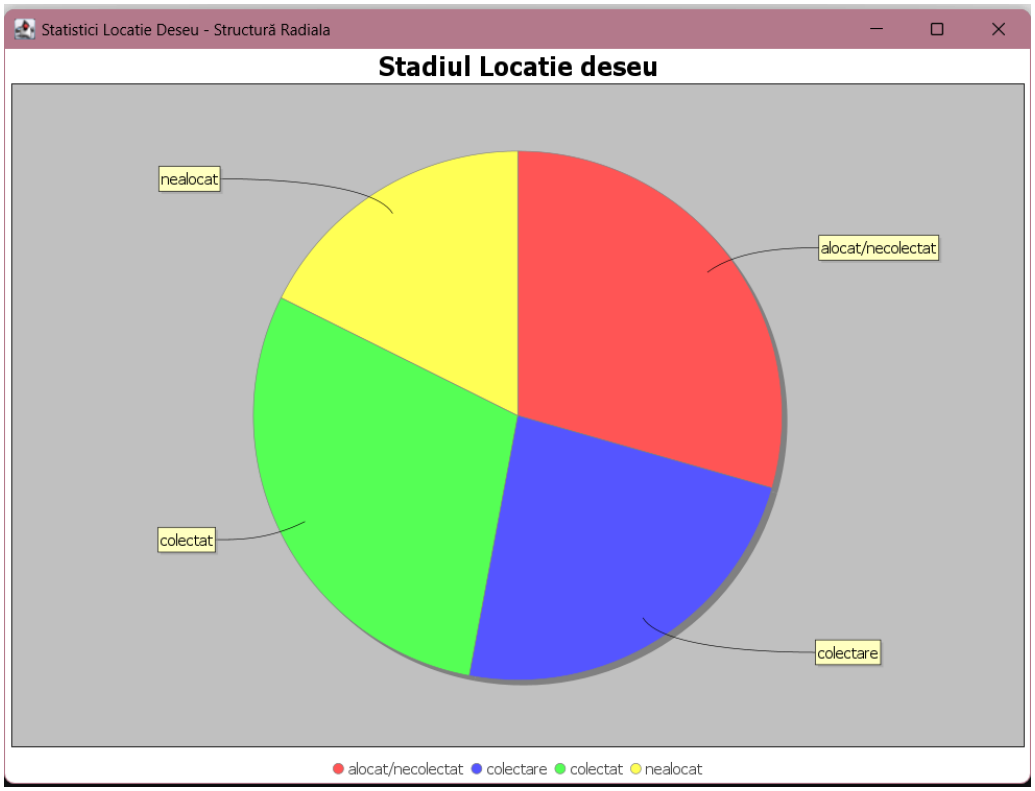

Figura 19

Aici putem observa o statistica de structura radiala pentru cee ace priveste stadiul locatiilor deseurilor.(Figura 19)

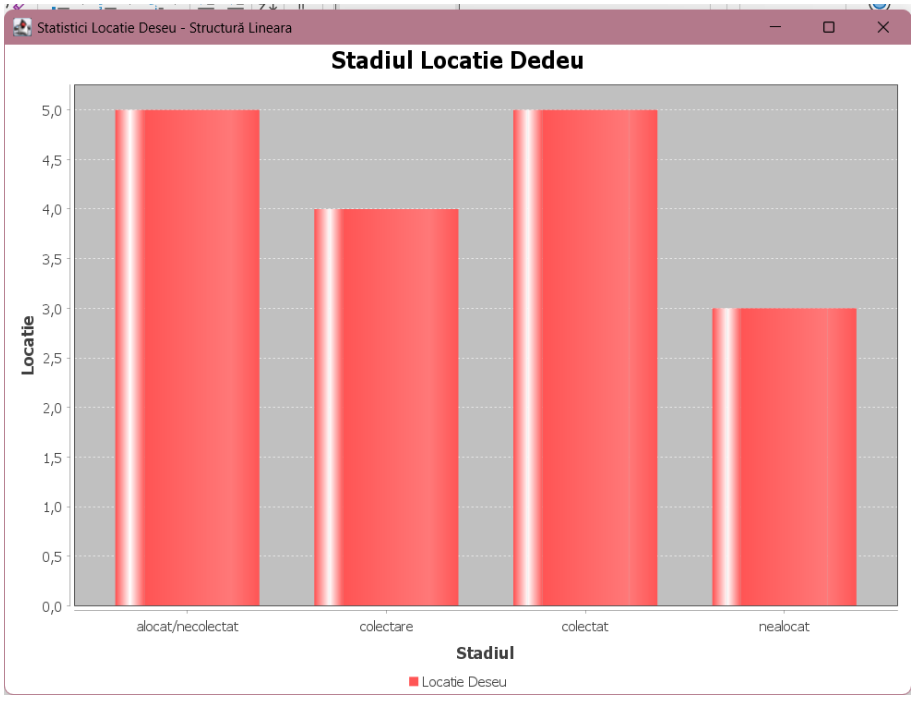

Figura 20

Aici putem observa o statistica de structura lineara pentru cee ace priveste stadiul locatiilor deseurilor.(Figura 20)

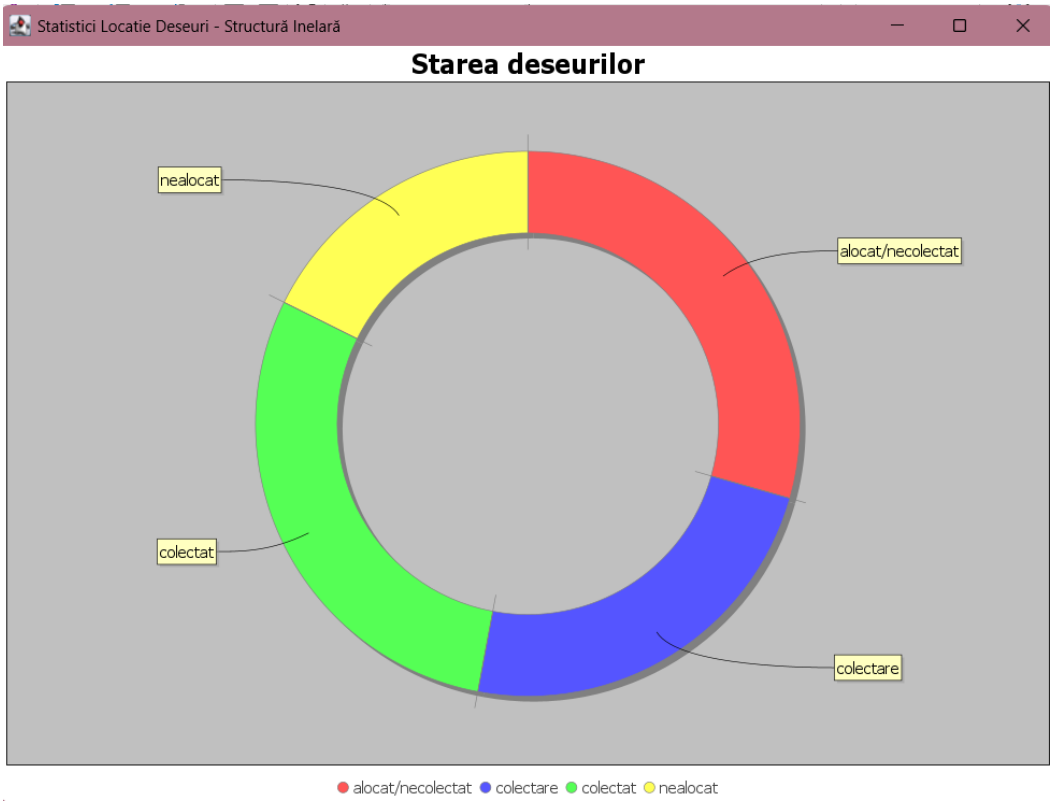

Figura 21

Aici putem observa o statistica de structura inelara pentru cee ace priveste stadiul locatiilor deseurilor.(Figura 21)