



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Proiectare Software

Colectare Deseuri

Profesor îndrumător:

Anca Iordan

Student:

Gavra Anamaria

Grupa: 30236

Cuprins

Enuntul Problemei.....	3
Justificarea limbajului ales.....	4
Justificarea alegerii șablonului de proiectare creațional utilizat.....	5
Etape.....	6
Analiza.....	6
1 Diagrama cazurilor de utilizare.....	6
1 Diagrame de activitate.....	7
Implementare.....	8
1 Diagrama de clase.....	8
Diagrama de clase.....	8
Pachetul Model.....	9
Pachetul View.....	9
Pachetul Controller.....	10
2 Diagrama de relationare a entitatilor.....	10
Diagrama de relationare a entitatilor.....	10
Implementare.....	11
Descrierea Aplicatiei.....	11
Design Pattern.....	12
Singleton.....	12
Observer.....	13

Enuntul Problemei

Descrierea aplicației: Dezvoltați (analiză, proiectare, implementare) o aplicație client/server care poate fi utilizată de către o firmă de colectare deșeuri.

Aplicația client va avea 3 tipuri de utilizatori: angajat, coordonator activitate și administrator.

1. Utilizatorii de tip angajat pot efectua următoarele operații după autentificare:

- Vizualizarea listei cu locațiile deșeurilor ce urmează a fi colectate sortată după locație;
- Vizualizarea traseului optim în funcție de locația deșeurilor.

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

- Toate operațiile permise utilizatorilor de tip angajat;
- Operații CRUD în ceea ce privește persistența locațiilor cu deșeuri;
- Alocarea unei locații cu deșeuri către un angajat în vederea colectării;
- Salvare rapoarte/liste cu informații despre listele cu pozițiile deșeurilor alocate angajaților în mai multe formate: csv, json, xml;



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- Vizualizarea unor statistici legate de deșeuri: procente după stare utilizând grafice (structură radială, structură inelară, de tip coloană, etc.).

3. Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

- Operații CRUD pentru informațiile legate de utilizatori;
- Vizualizarea listei tuturor utilizatorilor și filtrarea acestora după tipul utilizatorilor;
- Notificarea fiecărui utilizator prin cel puțin 2 variante (email, SMS, WhatsApp, Skype, etc.) la orice modificare a informațiilor de autentificare aferente aceluși utilizator.

Interfața grafică a aplicației client va fi disponibilă în cel puțin 3 limbi de circulație internațională (implicit limba română).

Justificarea limbajului ales

Am ales sa realizez implementarea acestui proiect in limbajul „Java”, in primul rand, deoarece a fost unul dintre primele limbaje de nivel inalt pe care le-am studiat. Alte motive pentru care am ales acest limbaj sunt urmatoarele :

- Este un limbaj foarte folosit si cunoscut de toata lumea.
- Este usor de inteles
- Codul este organizat
- Se poate muta usor de pe un sistemul unui computer pe al altuia
- Functiile au denumiri sugestive si sunt usor de utilizat.

Justificarea modului de realizare a persistenței

Pentru persistența am decis să utilizez bazele de date întrucât, din punctul meu de vedere sunt ușor de folosit și oferă o modalitate mai simplă de a accesa datele.

Sistemul de management al bazelor de date este în esență o colecție de date interconectate și un set de programe software care accesează, procesează și manipulează datele. Permite accesul, regăsirea și utilizarea acestor date, luând în considerare măsurile de securitate adecvate. Sistemul de management al bazei de date este cu adevărat util pentru o mai bună integrare și securitate a datelor.

Utilizarea bazelor de date aduce numeroase avantaje printre care se numără:

Transfer mai bun de date: gestionarea bazelor de date creează un loc în care utilizatorii beneficiază de mai multe date și mai bine gestionate.

Securitate mai bună a datelor: Este utilizat pe scară largă în lumea corporativă, unde companiile investesc bani, timp și efort în cantități mari pentru a se asigura că datele sunt sigure și utilizate corespunzător.

O mai bună integrare a datelor: Datorită sistemului de management al bazei de date, avem acces la o formă de date bine gestionată și sincronizată, astfel încât gestionarea datelor este foarte ușoară și oferă o

vedere integrată a modului în care funcționează o anumită organizație și ajută, de asemenea, să țineti evidența modului în care un segment al companiei afectează un alt segment.

Incoerență minimă a datelor: Incoerența datelor apare între fișiere atunci când versiuni diferite ale acelorași date apar în locuri diferite. Dacă o bază de date este proiectată corespunzător, inconsecvența datelor poate fi redusă foarte mult.

Acces mai rapid la date: Sistemul de management al bazei de date ajută la producerea de răspunsuri rapide la interogările bazei de date, făcând astfel accesul la date mai rapid și mai precis.

Simplu: Sistemul de management al bazelor de date (DBMS) oferă o vedere logică simplă și clară a datelor. Multe operațiuni precum inserarea, ștergerea sau crearea de fișiere sau date sunt ușor de implementat.

Reducerea redundanței datelor: Când lucrați cu o bază de date structurată, DBMS oferă caracteristica de a preveni introducerea de elemente duplicat în baza de date.

Justificarea alegerii șabloanelor de proiectare utilizate

1.Cretionale:

Am decis sa folosesc sablonul de proiectare creational **Singleton** pentru a ma asigura ca, conexiunea la baza de date este unica la nivel de aplicatie si nu pot crea mai multe instante ale acestei clase.

In ingineria software, modelul singleton este un model de proiectare software care restricționează instanțierea unei clase la o singură instanță. Acest lucru este util atunci când este necesar un singur obiect pentru a coordona acțiunile în sistem.

Modelul Singleton este unul dintre cele mai simple modele de design din Java. Acest tip de model de design face parte din modelul de creație, deoarece acest model oferă una dintre cele mai bune modalități de a crea un obiect.

Acest model implică o singură clasă care este responsabilă să creeze un obiect, asigurându-se în același timp că este creat doar un singur obiect. Această clasă oferă o modalitate de a accesa singurul său obiect care poate fi accesat direct, fără a fi nevoie de instanțierea obiectului clasei.

Pentru implementarea sablonului am creat o clasă SingleObject. Clasa SingleObject are constructorul cu nivelul de acces privat și are o instanță statică a acesteia.

Clasa SingleObject oferă o metodă statică pentru a aduce instanța sa statică în lumea exterioară. SingletonPatternDemo, clasa noastră demo va folosi clasa SingleObject pentru a obține un obiect SingleObject.

In acest proiect am utilizat sablnul Singleton pentru a realiza conexiunea la baza de date.

2. Structurale:

Adapter este un design pattern structural care permite obiectelor care au interfețe și structuri incompatibile să colaboreze. Acest șablon a fost utilizat la generarea fișierelor cu rapoarte în cazul în care sunt selectate toate tipurile de fișier (csv, json și xml). Pentru implementare am generat un singur fișier json, iar fișierele xml și cel csv au fost create prin parsarea fișierului generat.

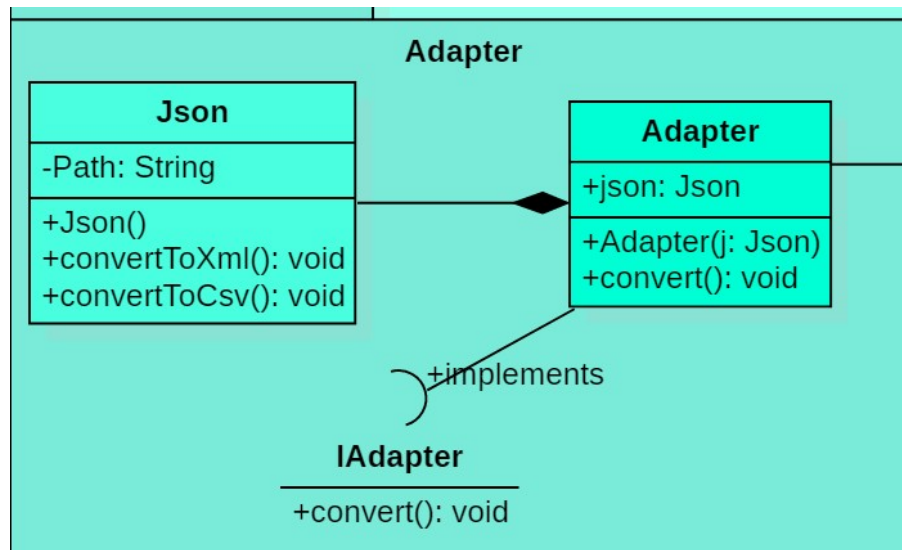


Figure 1: Adapter

Pentru implementare am definit două clase: clasa Json care conține calea către fișierul json generat și două metode de conversie a acestui fișier în celelalte două tipuri dorite; și Adapter, care conține un obiect de tipul prezentat anterior și implementează interfața IAdapter cu metoda pentru conversie.

Sablonul de proiectare structural **“Decorator”** este un sablon care permite atasarea unui nou comportament obiectelor prin “impachetarea” acestuia într-un alt obiect ce conține implementarea acestui comportament. Acesta a fost utilizat în proiect pentru a face posibilă trimiterea de notificări prin diferite mijloace: SMS sau Email.

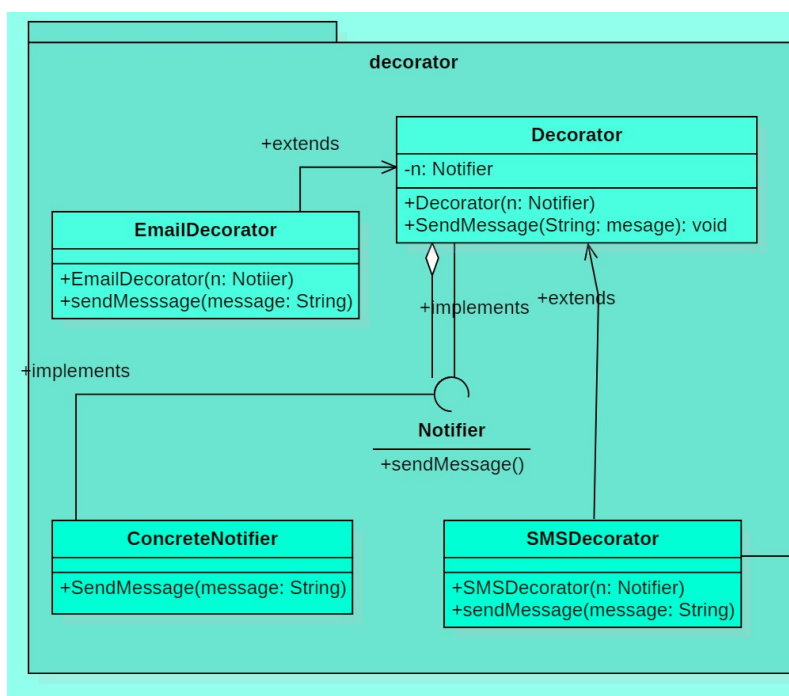


Figure 2: Decorator

Implementarea are la bază interfața “Notifier” care conține metoda de bază, aceea de trimitere a notificării. Această interfață este implementată de un obiect concret, după cum sugerează și numele, **ConcreteNotifier** și de o clasă mai abstractă “Decorator”, care este extinsă de clasele specifice notificărilor trimise : **SMSDecorator** și

EmailDecorator care implementeaza in mod adecvat metoda de baza a arhitecturii.

3. Comportamentale:

Observer este un sablon comportamental care ne permite sa notificam mai multi observatori la modificarea starii unui obiect.

Acesta a fost utilizat in proiect pentru actualizarea listei de locatii din JComboBox-ul din intrfata pentru coordonator, dupa modificarea datelor din lista de locatii.

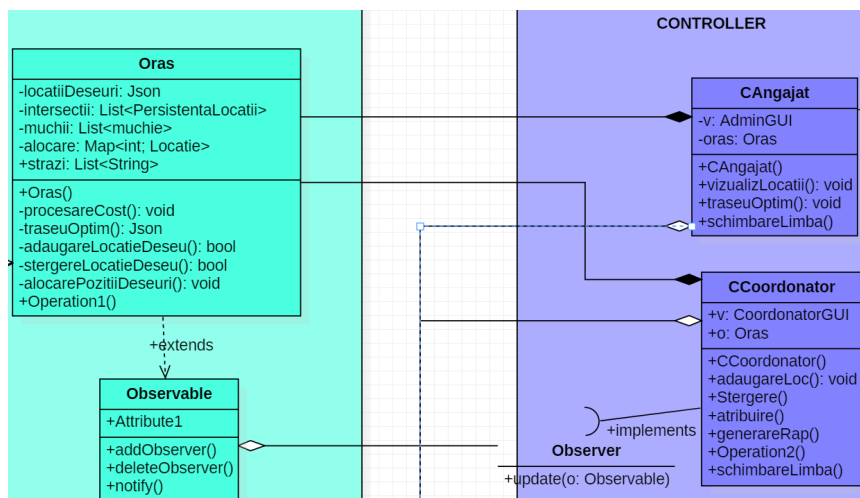


Figure 3: Observer

Pentru implementare, clasa “Oras” care se ocupa de persistenta locatiilor extinde clasa “Observable” care gestioneaza observatorii listelor de locatii, iar Controllerul implementeaza interfata “Observer” ce contine metode pentru raspunsul observatorilor la modificarea obiectelor de tip “Observable”.

State este un design pattern comportamental care permite obiectelor sa isi modifice comportamentul cand starea lor se modifica.

În proiect, acesta este folosit la ștergerea unei Locații de deșeu; astfel, dacă este selectată o locație pentru ștergere, obiectul trece în starea de succes și efectuează operația de ștergere; în caz contrar, se trece în starea de eroare și se afișează un mesaj specific.

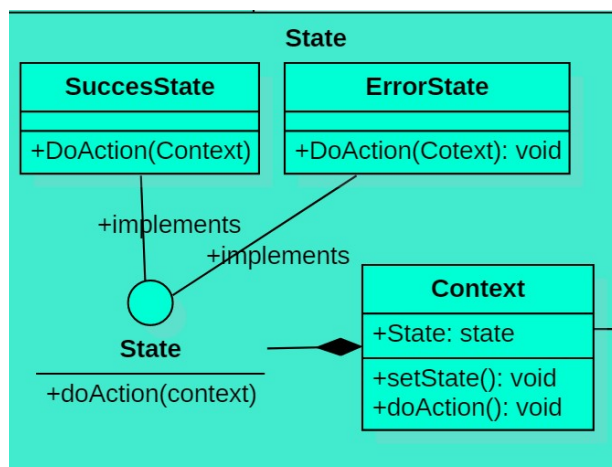


Figure 4: State

Pentru implementare au fost create două clase specifice fiecărei stări în care se poate găsi controllerul coordonatorului în timpul operației de ștergere, implementând interfața *State*, cu metoda ce implementează acțiunea care trebuie realizată în fiecare caz. Pachetul mai conține și clasa “*Context*” care conține o stare, o metodă pentru setarea acesteia în funcție de scenariu, și una care apelează acțiunea din clasa stării curente.

Etape

Analiza

1 Diagrama cazurilor de utilizare

Client:

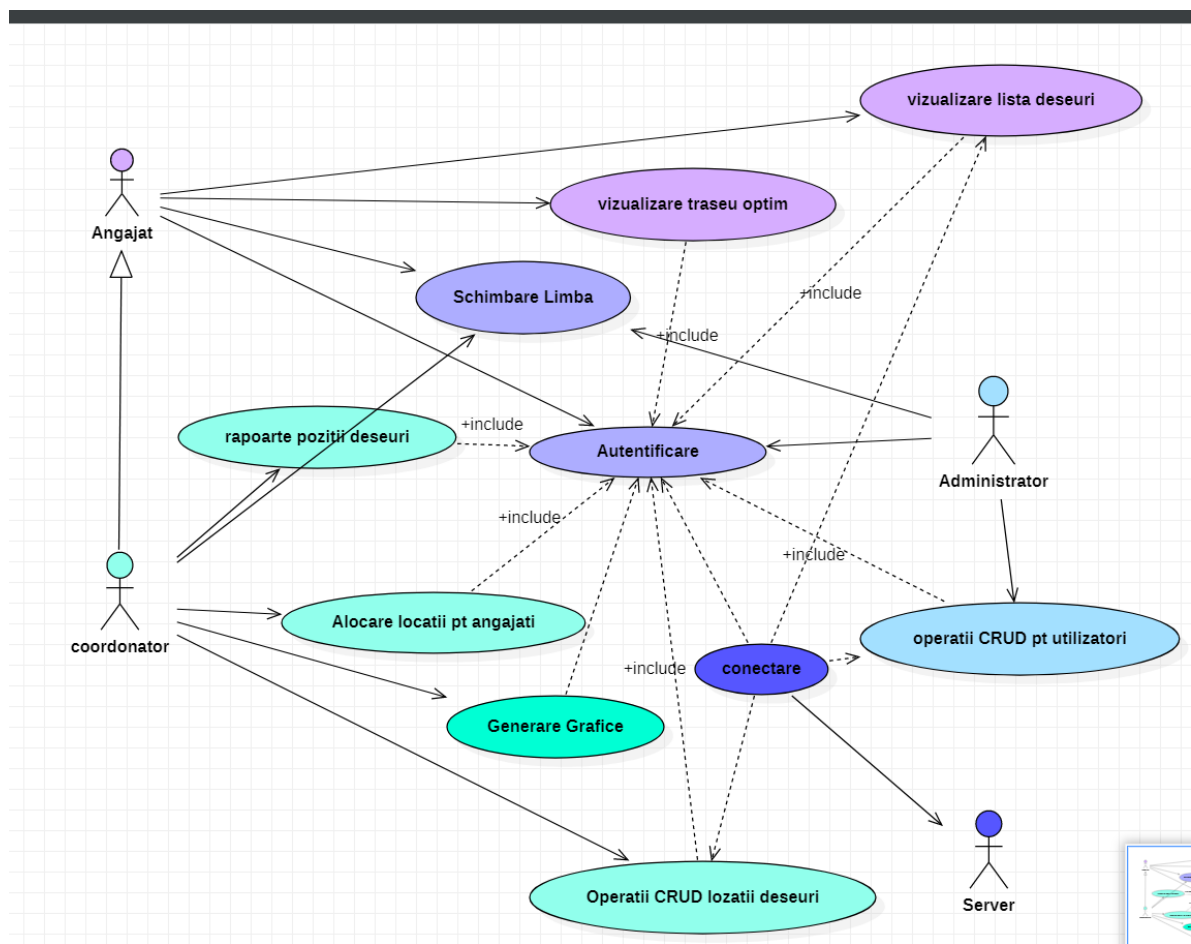


Figure 5: Diagrama Cazuri Utilizare

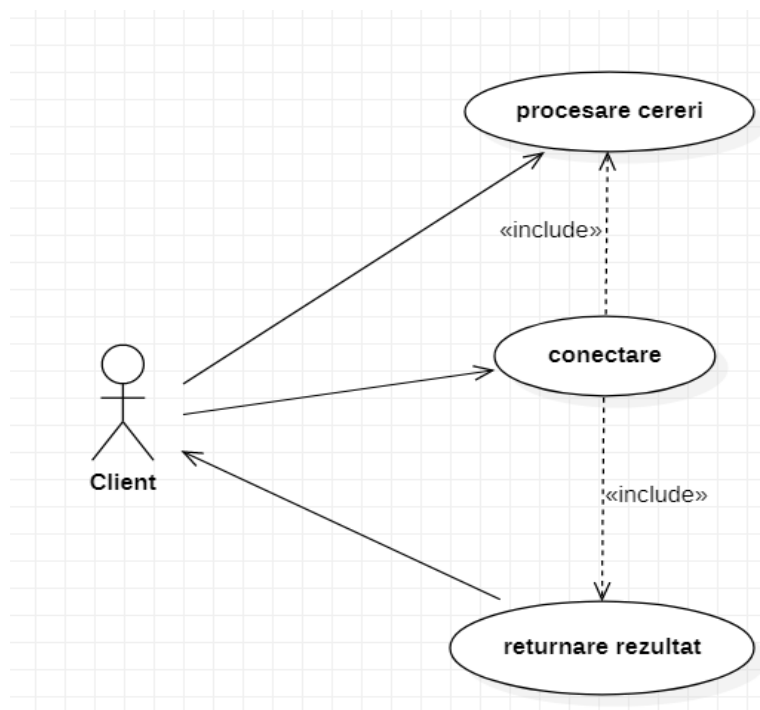


Figure 6: Diagrama Cazuri Utilizare(Server)

În această diagramă sunt prezentate acțiunile pe care le poate face fiecare utilizator prin intermediul interfeței grafice aferente. Aceste funcționalități sunt prezentate și în prima secțiune (“Enunțul Problemei”). Utilizatorii au totuși o funcție comună (cea de login), iar cazurile de utilizare specifice fiecărui utilizator sunt colorate în culorile corespunzătoare.

Aplicația “Server” are un singur tip de utilizator, acesta fiind aplicația client care poate să trimită request-uri, cereri care urmează să fie soluționate și să primească ca răspuns, un mesaj sau un rezultat.

1 Diagrame de activitate

Aceste diagrame contin pasii care sunt parcursi in vederea efectuarii fiecărei operatii din diagrama de cazuri de utilizare.

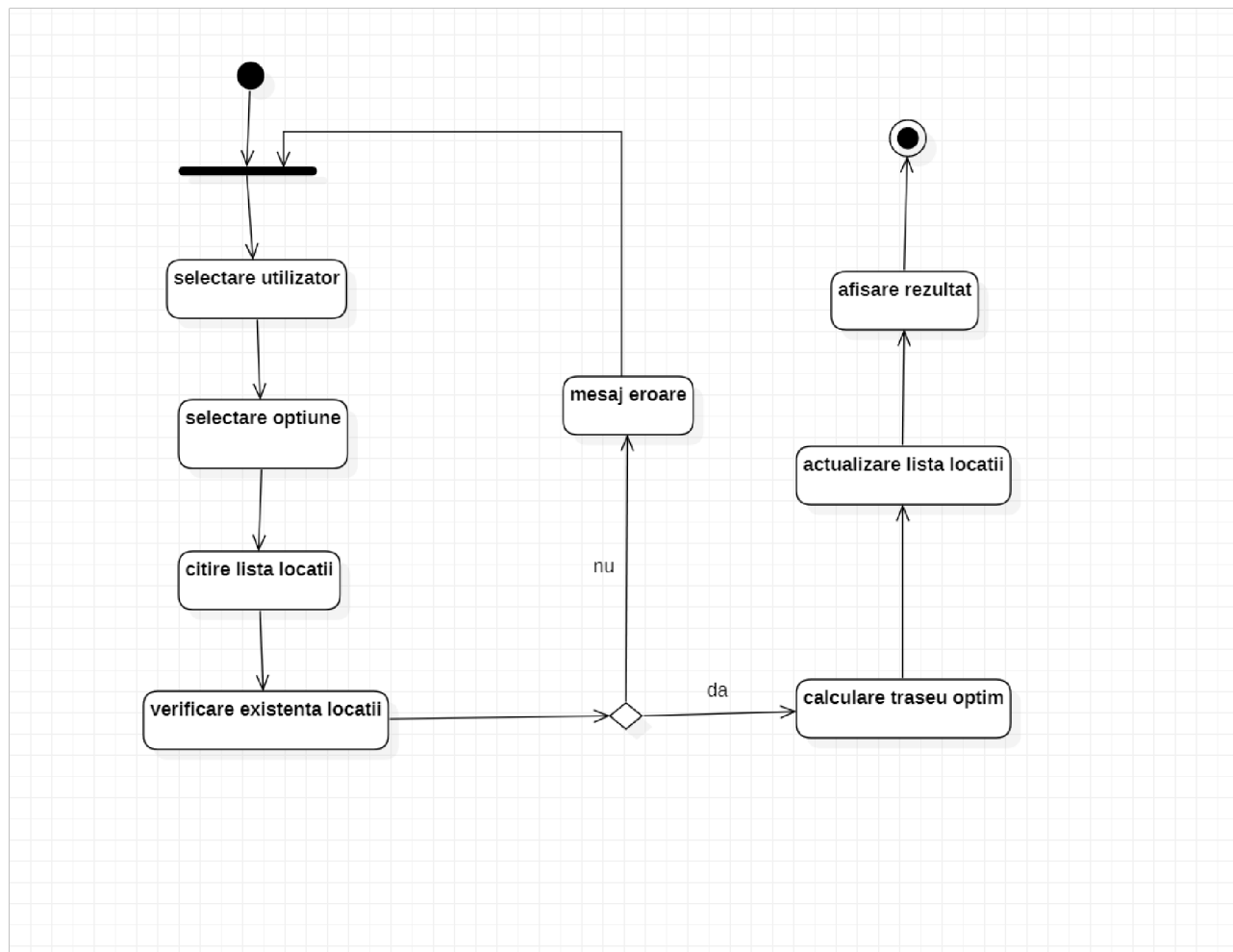


Figure 7: Diagrama Activitate

De exemplu, figura de mai sus reprezinta diagrama specifica operatiei de calculare a traseului optim pentru angajati si coordonatori.



Implementare

1 Diagrama de clase

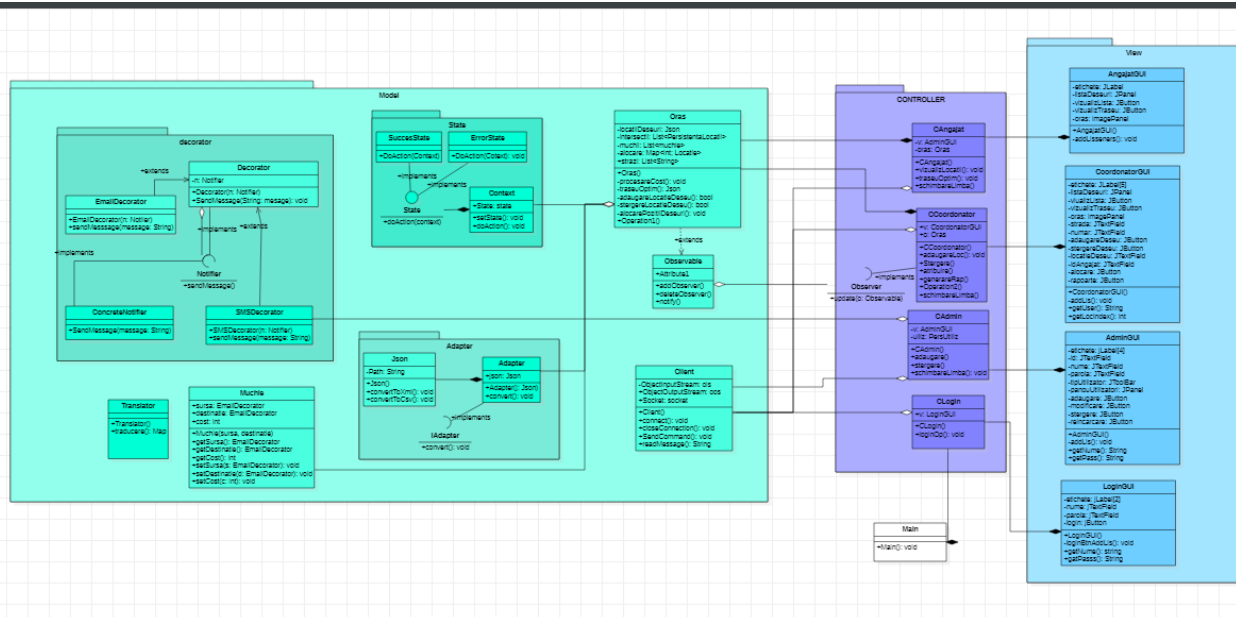


Figure 8: *Diagrama de clase(Client)*

Aceasta prezinta clasele implementate in cadrul proiectului. Clasele sunt organizate in trei pachete principale sugestive pentru arhitectura utilizata:

Pachetul Model

Contine clasele care se ocupa de implementarea functionalizatiilor. Acestea sunt:

- Clasa locatie care memoreaza strada si numarul fiecarei pozitii a deseurilor, dar si numele angajatului caruia ii este atribuita locatia spre colectare.



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- Clasa Translator care contine o singura metoda, fiind folosita pentru a implementa functionalitatile de traducere a textelor din interfata grafica in alte limbi.
- Clasa Utilizator care contine numele de utilizator si parola pentru conectare, dar si tipul utilizatorului conectat.
- Clasa Client care este folosita pentru a realiza comunicatia dintre client si server prin trimiterea comenzilor si receptionarea mesajelor si rezultatelor primite.
- Clasa Muchie care cuprinde doua locatii si distanta dintre acestea
- Clasa Oras care cuprinde strazile orasului, locatiile deseirilor si intersectiile, cu aceste informatii calculand traseul minim
- Clasa Observable care este extinsa de clasa Oras pentru a implementa sablonul "Observer".

Pe langa aceste clase, pachetul Model mai contine alte trei pachete mai mici. Acestea sunt:

Pachetul Decorator – care contine clasele utilizate pentru implementarea sablonului "Decorator";

Pachetul Adapter – ce cuprinde clasele care intra in componenta sablonului "Adapter";

Pachetul State – in componenta caruia se afla clasele ce formeaza sablonul cu acelasi nume.

Clasele din componenta acestor pachete au fost prezentate anterior la sectiunea "Justificarea alegerii şabloanelor de proiectare utilizate".

Pachetul View

Contine interfetele grafice pentru fiecare tip de utilizator si cea pentru logarea utilizatorilor. Fiecare interfata grafica cuprinde metode pentru accesarea sau actualizarea variabilelor si pentru adaugarea ascultatrilor pentru butoane.

Pachetul Controller

Contine cate o clasa Controller asociata fiecărei interfete grafice si o interfata "Observer" care este implementata de clasa corespunzatoare intergetei grafice a utilizatorului de tip "Coordonator".

Fiecare clasa controller este responsabila de conexiunea dintre model si interfata grafica cirespunzatoare, continand cate un obiect de tip view si un model.

Aceste clase mai au responsabilitatea de implementarea functionalitatilor butoanelor pentru variabila de tip view corespunzatoare si atribuirea acestor functii ascultatorilor butoanelor.

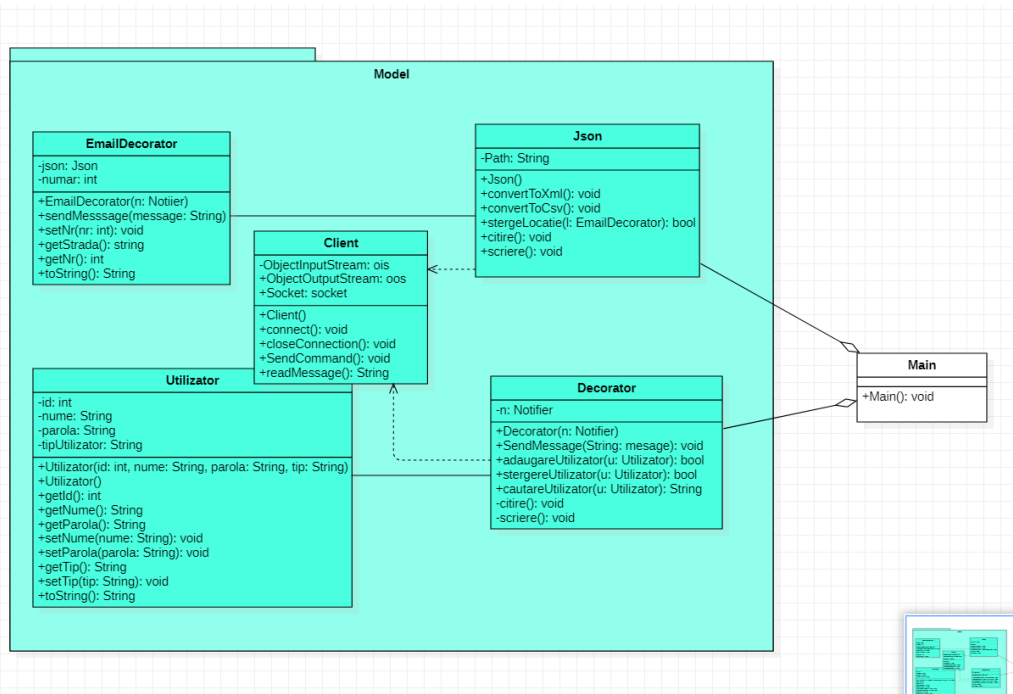


Figure 9: Diagrama de Clase(Server)

Serverul contine clasele care se ocupa de conexiunea la baza de date si de persistenta; astfel, acest contine doar clasa "Main: in care sunt gestionate cererile primite, si pachetul Model, ce contine urmatoarele clase:

- Clasa locatie care memoreaza strada si numarul fiecarei pozitii a deseurilor, dar si numele angajatului caruia ii este atribuita locatia spre colectare.
- Clasa PersistentaLocatii, care, dupa cum indica si numele, se ocupa de persistenta locatiilor, avand functii de citire si scriere in fisier sau de adaugare sau stergere a datelor din lista de locatii.

- Clasa Utilizator care contine numele de utilizator si parola pentru conectare, dar si tipul utilizatorului conectat.
- Clasa PersistentaUtilizatori care contine functiile prin intermediul carora se realizeaza persistenta utilizatorilor.

2 Diagrama de relationare a entitatilor

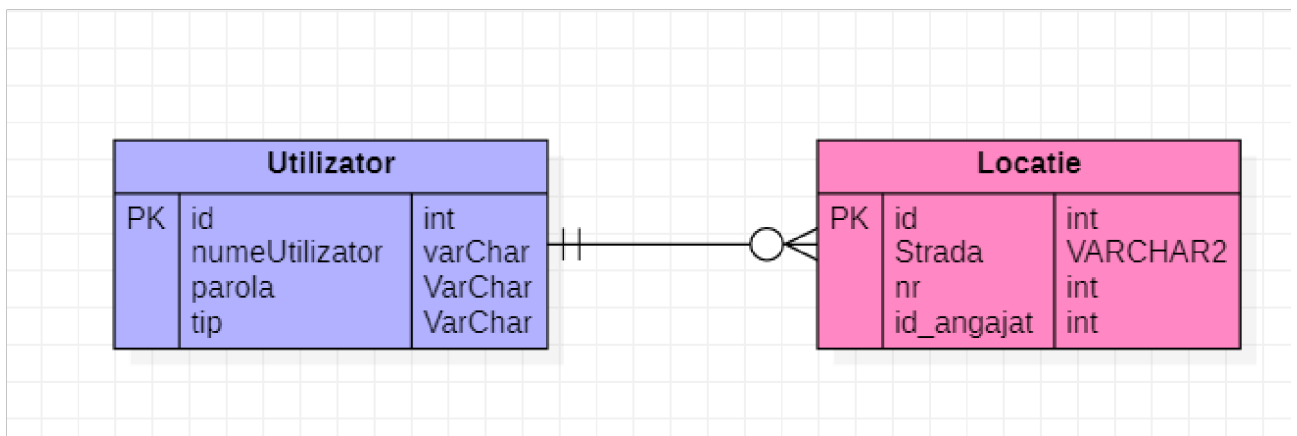


Figure 10: Diagrama de relationare a entitatilor

Aceasta diagrama contine cele doua tabele din baza de date utilizate pentru persistenta datelor si relatia dintre acestea, aceasta fiind o relatie de tip “one to many”. Fiecare tabel contine denumirile coloanelor, cheia primara, dar si tipul datelor fiecărei coloane.

Implementare

Descrierea Aplicatiei



Figure 11: LoginGUI

La pornirea aplicatiei este vizibila interfata grafica pentru autentificarea utilizatorului. Aceasta contine doua textField-uri: unul pentru a introduce numele de utilizator, iar cel de al doilea pentru introducerea parolei. Pe langa acestea, in interfata grafica mai exista un buton care se apasa pentru autentificarea datelor introduse si un checkbox pentru a vizualiza parola. Dupa apasarea butonului, daca datele introduse sunt valide, se deschide o alta interfata grafica in functie de tipul utilizatorului care a introdus credentialele (administrator, angajat sau coordonator).

Interfata pentru angajati cuprinde o lista cu locatiile deseurilor alocate angajatului autentificat in vederea colectarii. Pentru vizualizarea tuturor locatiilor alocate se apasa butonul „Vizualizare locatii”, iar pentru

ordonarea locatiilor pentru a obtine cea mai scurta ruta, se apasa pe butonul „Drum Minim”.

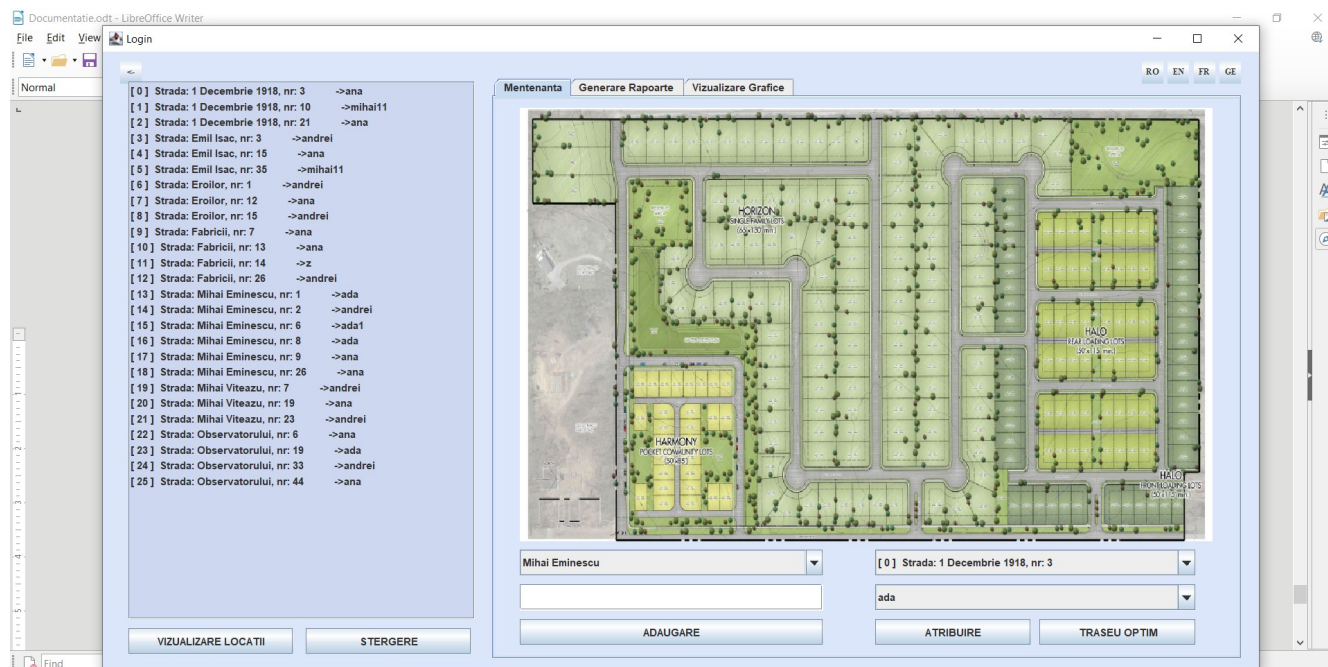


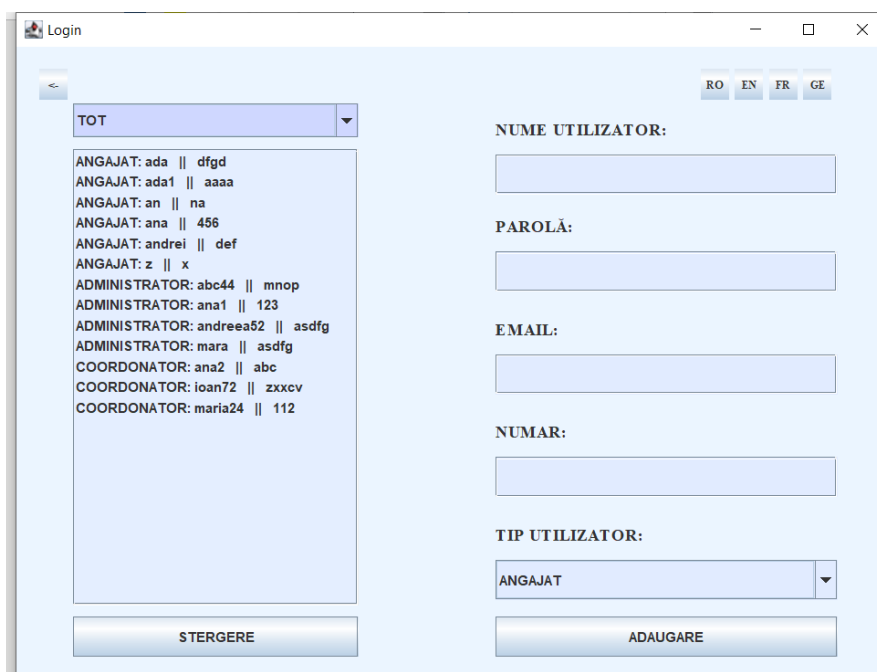
Figure 12: CoordonatorGUI

Utilizatorul de tip coordonator are o interfata prin intermediul careia se pot realiza, pe langa operatiile pentru angajat, operatiile CRUD pentru locatiile deseurilor: pentru adaugare se selecteaza strada si se introduce numarul adresei, iar pentru stergere se selecteaza o locatie din lista si se apasa pe butonul pentru stergere. O alta operatie pe care o poate realiza Coordonatorul este aceea de a atribui o locatie deja existenta unui anumit angajat prin selectarea locatiei si a angajatului si prin apasarea butonului “Atribuire”. Prin apasarea butonului “Traseu optim” se afiseaza ruta de cost minim pentru angajatul selectat in jComboBox-ul de deasupra butonului.

In al doilea tab al meniului din dreapta se pot genera rapoarte prin selectarea formatului dorit (JSON, XML sau CSV), prin introducerea unei cai spre locul in care se doreste salvarea raportului si a numelui fisierului

generat. După ce toate câmpurile au fost completate se apasă butonul “Generare rapoarte”.

În ultimul tab se pot vizualiza diagrame corespunzătoare numărului de locații alocate fiecărui angajat. Se poate selecta una din următoarele diagrame: liniară, circulară sau radială, prin apăsarea butonului corespunzător, rezultatul putând fi vizualizat în panoul de sub butoanele pentru selecție.



Language: RO EN FR GE

TOT

- ANGAJAT: ada || dfgd
- ANGAJAT: ada1 || aaaa
- ANGAJAT: an || na
- ANGAJAT: ana || 456
- ANGAJAT: andrei || def
- ANGAJAT: z || x
- ADMINISTRATOR: abc44 || mnop
- ADMINISTRATOR: ana1 || 123
- ADMINISTRATOR: andreea52 || asdfg
- ADMINISTRATOR: mara || asdfg
- COORDONATOR: ana2 || abc
- COORDONATOR: ioan72 || zxxcv
- COORDONATOR: maria24 || 112

STERGERE

NUME UTILIZATOR:

PAROLĂ:

EMAIL:

NUMAR:

TIP UTILIZATOR:

ANGAJAT

ADAUGARE

Figure 13: AdminGUI

Pentru utilizatorul de tip “Administrator”, sunt implementate funcționalități pentru realizarea operațiilor CRUD pentru utilizatorii din baza de date. Interfața grafică corespunzătoare cuprinde:

- o listă cu utilizatorii existenți ce cuprinde credențialele utilizatorilor și tipul fiecăruia;
- două JTextField-uri unul pentru introducerea numelui utilizatorului pe care dorim să îl adăugăm și unul pentru parola aleasă pentru acesta);



UNIVERSITATEA TEHNICĂ **DIN CLUJ-NAPOCA**

- un JComboBox pentru selectarea tipului de utilizator adaugat;
- un buton pentru adaugarea utilizatorului dupa ce datele au fost introduse corect;
- un buton pentru stergerea unui utilizator selectat din lista descrisa anterior.

Pe langa aceste functionalitati proprii, interfetele mai au si functionalitatea comuna de modificare a limbii in care este prezentata interfata