DOCUMENTATIE

TEMA *3*

NUME STUDENT: Ciobanu Radu-Rares
GRUPA: 30222

Cuprins:

1. Obiectivul temei

1.1 Obiectivul principal

Obiectivul acestei teme este de a invata cum sa facem conexiunile cu o baza de date, si a invata pentru ce este bun JavaDoc-ul, cat si folosirea, metodei de reflectie pentru a avea cod cat mai lizibil.

1.2 Obiectivele secundare

Obiectivele secundare sunt acele obiective pe baza carora intocmim tema propriu zisa (obiectivul principal), iar acestea sunt alcatuite din :

- Crearea unei baze de date care sa corespunda cerintelor;
- Crearea legaturilor dintre tabele si dependentele prezente intre ele;
- Intocmirea unei interfete grafice, care permite utilizatorului sa efectueze cat mai eficient modificari in baza de date;
- Tratarea exceptiilor prezente;
- Scrierea corecta si lizibila a codului, impartit in 6 pachete pentru gestionarea mai usoara a acestuia.

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

2.1 Analiza problemei

Problema principala a acestei teme este determinarea unei metode de a realiza cat mai flexibil o clasa responsabila pentru operatiile de adugare, stergere, editare si de extragere a datelor din tablele, intr-un mod cat mai generic, si usor reutilizabila.

De asemenea trebuie facuta conexiunea dintre o interfata grafica si restul codului responsabil pentru aceste operatii.

2.2 Modelare

2.2.1 Cerinte functionale:

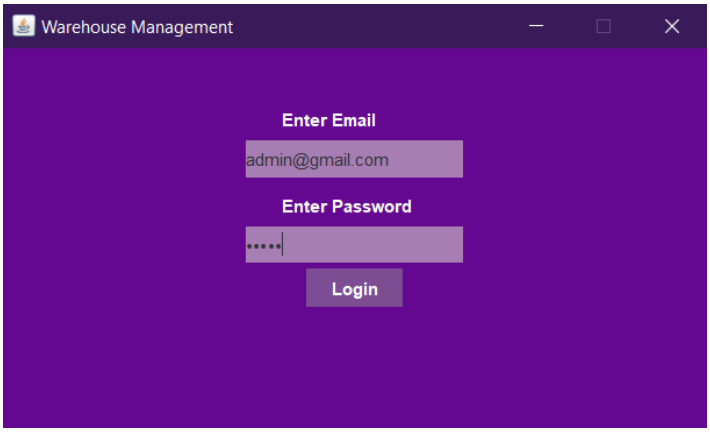
- Utilizatorul trebuie sa aiba permisiuni pentru a efectua operatiile de modificare a tabelelor;
- Programul trebuie sa-i permita utilizatorului sa aleaga operatiile
- Programul trebuie sa fie capabil sa modifice continutul tabelelor la cerinta utilizatorului, si sa le afiseze;
- Programul trebuie sa atentioneze utilizatorul in cazul in care a oferit un input invalid .

2.2.2 Cerinte non-functionale:

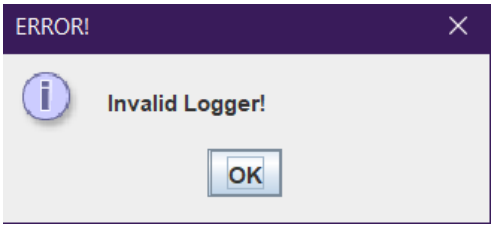
- Programul trebuie sa fie intuitiv si usor de inteles pentru utilizator, si sa ii ofere o mobilitate aparte cand vine vorba de efectuarea modificarilor.

2.3 Scenarii

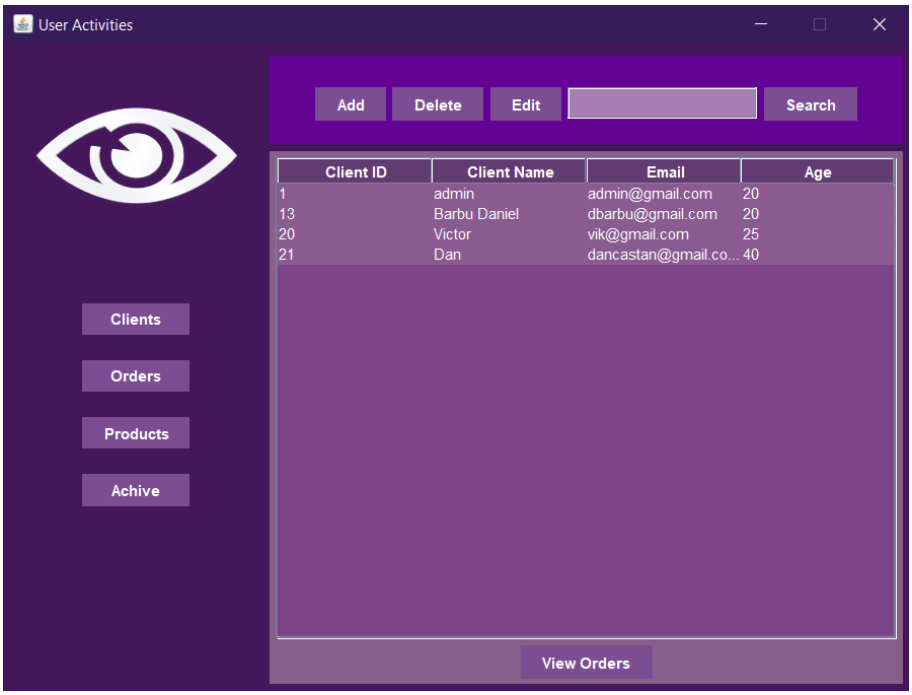
La rularea programului, utilizatorul va fi intampinat de o fereastra care ii cere sa se intre in contul propriu de utilizator. Avand in vedere ca nu se cere ca si clientul sa fie capabil sa intre in cont, am creat un cont fictiv, cu numle de utilizator, emailul din baza de date admin@gmail.com si cu parola, numele din baza de date care in cazul nostru este "admin".



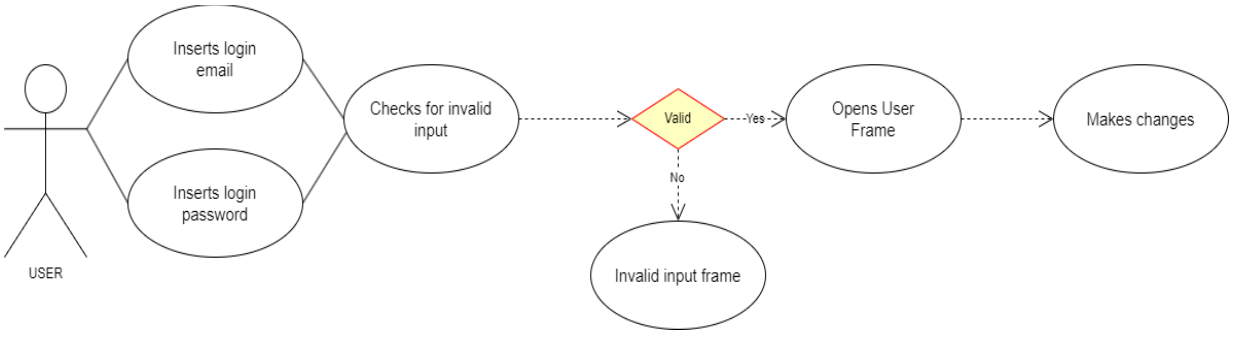
In cazul in care utilizatorul nu are datele respective, va fi intampinat de un mesaj de eroare.



Dupa ce adminul a intrat in cont, fereastra principala va aparea, unde adminul poate vizualiza continutul tabelelor din baza de date, dar le si poate modifica dupa bunul plac.



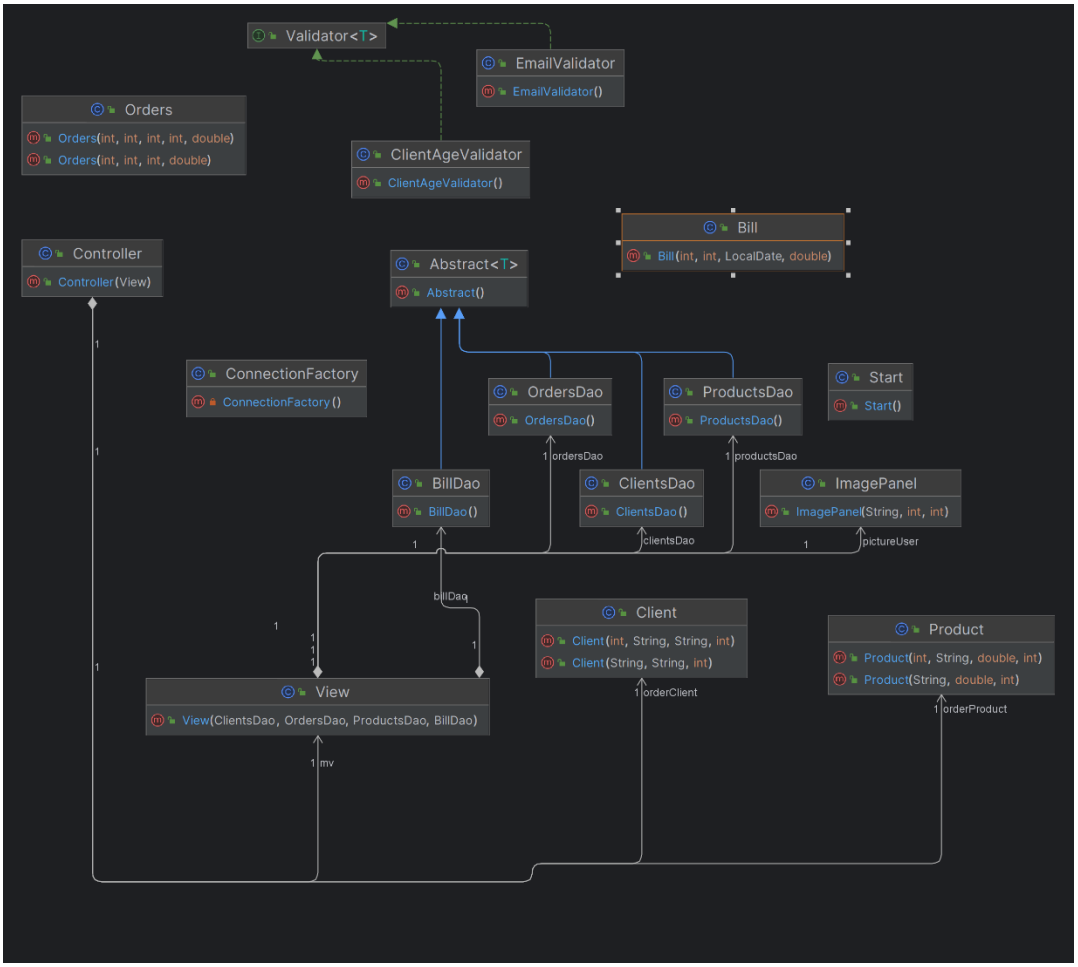
2.4 Cazuri de utilizare



3. Proiectare

Proiectul este impartit in 6 pachete, care corespund unor parti diferite de cod, si anume, Presentation, care contine toate datele despre interfata grafica, plus conexiunea cu restul codului, Model, care contine clasele de obiecte prezente si in baza de date, DAO, care contine clasa abstracta, plus alte clase care exting clasa respectiva, Connection, care face lagaturile cu baza de date, Bll, care are 2 validatori responsabili pentru crearea unui client in modul corect, si pachetul Start, care contine exectuia in sine.

3.1 Diagrama UML:



4. Implementare

Pachetul Bll.Validator:

Pachetul valideaza anumite date prin interfata Validator. Prin aceasta interfata se selecteaza operatia de validare dintre cele 2 operatii, email validator si age validator.

Pachetul DAO:

Pachetul DAO prezinta clasa Abstract, care are metode de tin add, delete, find , edit, care pot fi efectuate pe orice tip de obiect, deoarece are un parametru generic. Aceasta clasa este extinsa de obiectele in sine cu tipul lor de obiect, pentru a avea o referinta doar la operatiile pentru obiectul acela.

Pachetul Model:

In pachetul Model sunt definite clasele Client, Orders, Product si Bill, cu tot cu atributele acestora, si contin in majoritate doar constructori, care au id si care nu au, datorita autoincrementului din SQL.

Pachetul Presentation:

Pachetul acesta, a fost cel mai costisitor cand vine vorba de timp, deoarece am incercat sa fac o interfata cat mai buna pentru program, si a iesit cat de cat bine. Clasa View este responsabila de tot ce vede utilizatorul cand porneste programul. Clasa Controller este responsabila pentru functionarea corecta a programului.

5. Rezultate

Rezultatele se pot vedea in timp real, actualizate, in tabelele prezente in interfata, dar si in baza de date.

6. Concluzii

Programul funtioneaza corect, si executa toate cerintele cerute. O posibila perfectionare a acestuia ar fi adaugarea unui search box functional, permiterea clientilor sa se conecteze pentru a da comenzi, si adaugarea a mai multor produse pe comanda.

7. Bibliografie

1. https://dsrl.eu/courses/pt/
2. https://www.jetbrains.com/help/idea/class-diagram.html
3. https://www.w3schools.com/java/