

**Aplicatie de management pentru comenzi**

**Documentatie**

**Student:**

**Manoilescu Victor**

Cuprins:

1. Obiectivul temei
2. Analiza problemei; Modelare;Scenarii

Cazuri de uitlizare

1. Proiectare
2. Implementare
3. Rezultate
4. Concluzii
5. Bibliografie

***1.Obiectivul temei:***

* Obiectivul principal al temei “Management pentru comenzi” consta in proiectarea si implementarea unei aplicatii ce poate suporta diferite operatii asupra unor comenzi, cat si asupra produselor comandate si a persoanelor ce solicita comanda
* Indeplinirea obiectivului principal necesita indeplinirea unor obiective secundare:

1. Analizarea problemei si identificarea cerintelor:

* Pentru a putea implementa aplicatia trebuie intai sa avem o idee concreta a ce ne dorim de la aceasta. Astfel, intelegerea conceptelor de baza a bazelor de date este necesara
* Dezvoltarea acestui subiect se afla in capitolul 2

1. Identificarea si solutionarea cazurilor de utilizare ale aplicatiei din ceea ce priveste interactiunea utilizatorului cu aceasta:

* Este necesar sa acoperim toate cazurile de utilizare astfel incat aplicatia sa functioneze cum trebuie in orice circumstanta
* Dezvoltarea acestui subiect se afla in capitolul 2

1. Dezvoltarea si proiectarea aplicatiei conform cerintelor din barem respectand paradigmele programarii orientate pe obiecte:

* Se descriu structurile de date si algoritmii utilizati in implementare (capitolul 3 si 4)
* Se prezinta structurarea aplicatiei in pachete, clase si relatiile dintre acestea ( capitolul 3 si 4)
* Se vor utiliza diagrame UML pentru exemplificarea legaturilor dintre componente

1. Evaluarea rezultatelor obtinute:

* Se vor analiza continuturile tabelelor.
* Dezvoltarea acestui subiect se afla in capitolul 5

***2.Analiza problemei; Modelare; Scenarii; Cazuri de utilizare:***

Cerinta problemei este: “Proiectati si implementati o aplicatie de management a comenzilor, datele clientilor, produselor si comenzilor fiind socate intr-o baza de date relationala”.

Pentru rezolvarea acestei problemei intai ne trebuie o suita de cunostinte legate de functionalitatea bazelor de date:

O baza de date reprezintă o modalitate de stocare a unor informații și date pe un suport extern (un dispozitiv de stocare), cu posibilitatea extinderii ușoare și a regăsirii rapide a acestora.

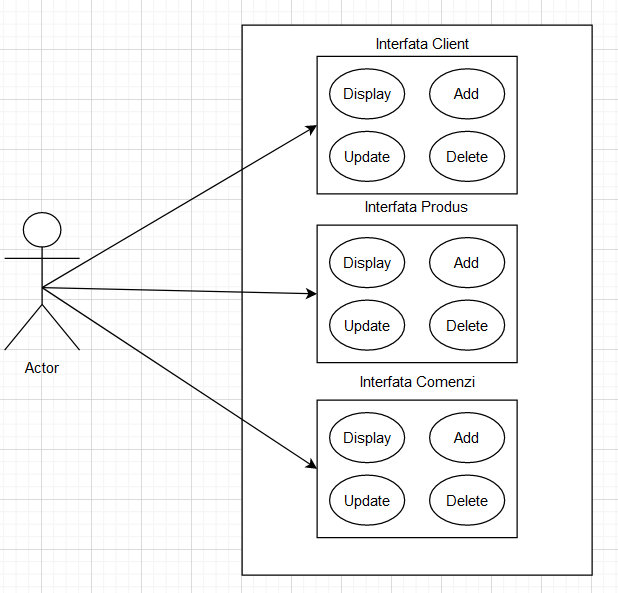
Bazele de date relaționale au elementele organizate ca un set te tabele, cu rânduri și coloane. Tehnologia bazelor de date relaționale oferă cel mai eficient și flexibil mod de a accesa informațiile structurate.

Pentru a manipula datele aflate in bazele de date relationale este necesara utilizarea limbajului de programare SQL(Structured Query Language), ce are ca principale functionalitati inserarea datelor, interogații, actualizare și ștergere, modificarea și crearea schemelor, precum și controlul accesului la date.

***Cazuri de utilizare:***

* Cazurile de uitlizare vor fi evidentiate prin daigrama use-case,

unde Actorul este reprezentat de orice utilizator al aplicatiei



Rularea aplicatiei de management a comenzilor:

Use-case: Selectarea tipului de date dorit, introducerea parametrilor necesari si selectarea operatiei applicate asupra bazei de date.

Primary actor: Utilizatorul

Main success scenario:

Utilizatorul introduce datele corespunzatoare parametrilor necesari operatiei dorite.

Interpretarea datelor utilzatorului.

Aplicarea operatiei dorite asupra tabelei selectate

Alternative sequences:

Introducerea de paramentri incorecti pentru operatia dorita, caz in care va aparea un mesaj de eroare.

***3.Proiectare***

Acest capitol va prezenta diagramele UML pentru pachete si pentru clase.

Diagrama UML pentru package-uri:

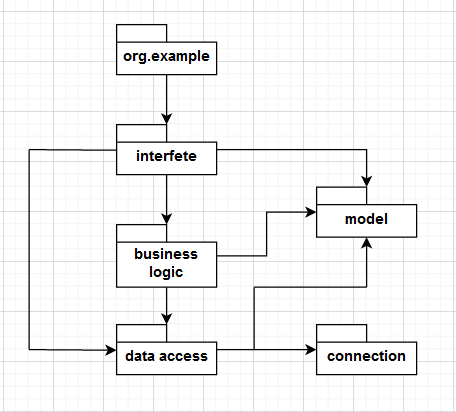


Diagrama UML pentru clase:

A picture containing text, diagram, line, parallel

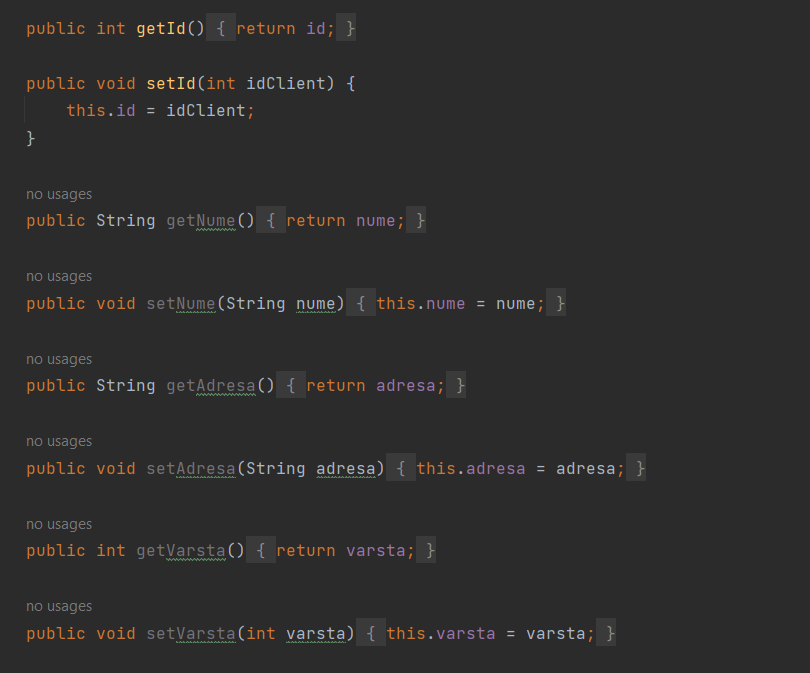
Description automatically generated

***4.Implementare***

Aplicatia contine 20 de clase ce intra in 4 categorii principale:

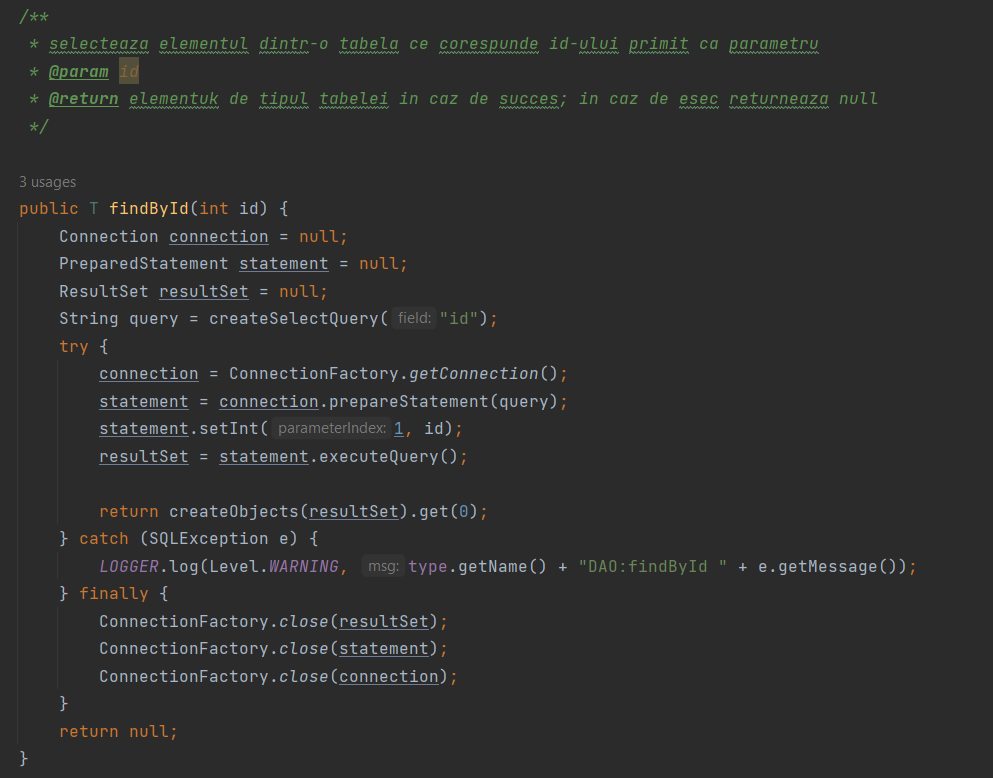
1. Clasele de tip Model:

* Atributele reprezinta coloanele tabelelor reprezentate de fiecare clasa
* Metode:
* *Getters and setters:* Folositi pentru accesarea datelor



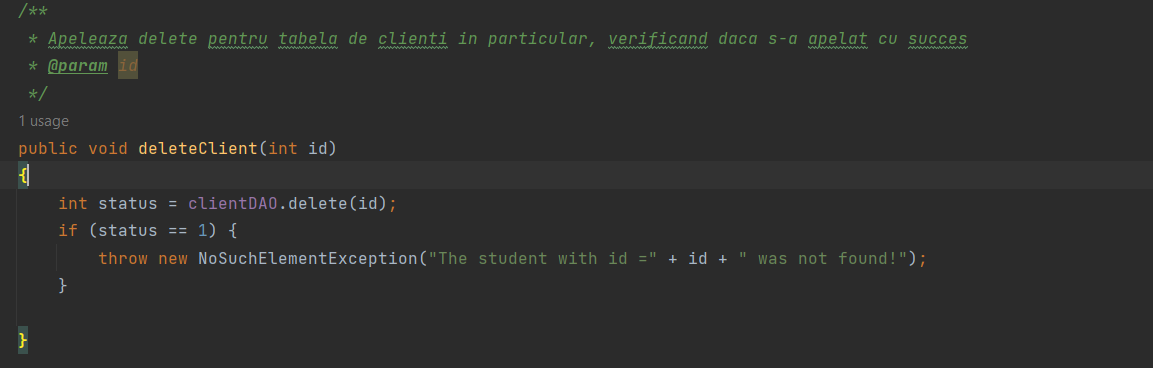
1. Clasele de tip Data Access:

* Doar AbstractDAO contine atribute si Metode; Celelalte clase de tip Data access sunt utilizate pentru a particulariza utilizarea clasei generale AbstractDAO
* Are 2 atribute: logger, folosit pentru a face legatura cu baza de date si type, folosit pentru a distinge clasa
* Metode:
* *Metodele de querry:* Formeaza stringul ce reprezinta querry-ul apelat
* *FindAll:* Gaseste toate elementele dintr-o tabela si le pune intr-o lista de acel tip
* *FindById:* Gaseste elementul dintr-o tabela ce are id-ul primit ca parametru
* *Delete:* Sterge elementul dintr-o tabela ce are id-ul primit ca parametru
* *Insert:* Insereaza elementul primit ca parametru intr-o tabela
* *Update:* Updateaza elementul cu id-ul primit ca parametru 1 cu datele elementului primit ca parametru 2



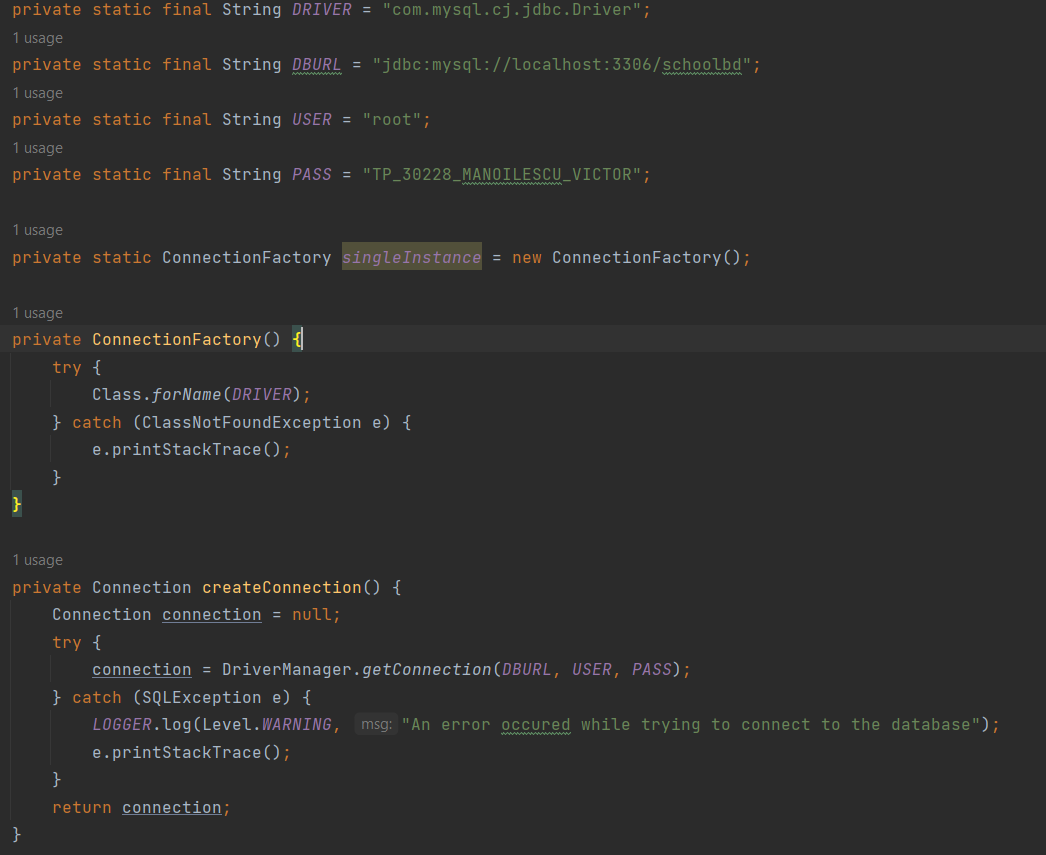
1. Clasele de tip Business Logic:

* Apeleaza functiile de tip data logic si verifica daca acestea s-au realizat cu succes



1. Clasa Connection:

* Creaza conexiunea dintre aplicatie si baza de date



***5.Rezultate***

Date intrare pentru inserarea unui client:

* Id: 1
* Nume: Marcel
* Adresa: str. Memo
* Varsta: 22

Rezultat:



***6.Concluzie***

Concluzionand, aceasta tema a avut un rol important in solidificarea cunostintelor mele atat de java, cat si de SQL, recapituland materia Baze de Date de semestrul trecut.

Aplicatia inca ar avea loc de imbunatatiri, putand adauga operatii de:

* Select in functie de alte criterii
* Ordonarea datelor in functie de diferite ctriterii

***7.Bibliografie***

-https://www.tutorialspoint.com/jdbc/jdbc-update-records.htm

-https://www.geeksforgeeks.org/how-to-update-multiple-columns-in-single-update-statement-in-sql/