Documentatie Tema 3

Order Management

George Adam

Grupa 30225

Profesor Laborator: Dorin Moldovan

[1. Obiectivul temei 3](#_Toc37616378)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc37616379)

[3. Proiectare 4](#_Toc37616380)

[4. Implementare 8](#_Toc37616381)

[5. Rezultate 16](#_Toc37616382)

[6. Concluzii 16](#_Toc37616383)

[7. Bibliografie 17](#_Toc37616384)

Obiectivul temei

Scopul principal al acestei teme este de a implementa o aplicatie de gestiune a comenzilor. Astfel, aceasta preia comenzi dintr-un fisier text, evident se poate implementa si o interfata grafica, si le executa, totodata pentru unele dintre ele generand tabele de raport sub forma de fisiere PDF. Acestea sunt generate la fiecare pas cand se intalneste o instructiune adecvata, deci se poate urmari evolutia executiei comenzilor.

|  |  |  |
| --- | --- | --- |
| Obiectiv secundar | Descriere | Capitol |
| Viteza | Executiile comenzilor in baza de date trebuie sa fie rapida, astfel se adopta relaizarea unei conexiuni globale si care se face o singura data prin intermediul unei clase de tip Singleton. | 3 |
| Verificare pe pasi | Pentru a se putea urmari executia unor comenzi, se genereaza rapoarte pentru a le verifica | 3 |
| Corectitudinea | Prin fisierele PDF generate de aplicatie, se pot verifica toate comenzile executate de aplicatie, citite din fisierul de intrare. | 3 |

Analiza problemei, modelare, scenarii, cazuri de utilizare

* Cerinte functionale.
* Implementarea conexiunii cu baza de date.
* Generarea unui model de baza de date pentru a stoca clienti, comenzi si produse.
* Introducerea clientilor in baza de date.
* Introducerea produselor in baza de date sau actualizarea stocului.
* Introducerea comenzilor in baza de date.
* Generare de rapoarte pentru clienti, produse si comenzi.
* Salvarea rapoartelor se face in fisiere PDF generate la executia respectivei comenzi.
* Use-cases

|  |  |
| --- | --- |
| Use-case | Solutie |
| Conexiunea la baza de date | Conexiunea se realizeaza o singura data prin intermediul unei clase de tip Singleton. S-a ales aceasta implementare deoarece generarea conexiunii de un numar mare de ori duce la performante scazute. |
| Structura bazei de date | Baza de date este impartita in trei tabele: client, orderr, product. Acestea stocheaza informatii care sunt introduse cu ajutorul procedurilor definite special pentru acestea. |
| Alegerea structurilor de date | Se folosesc structuri de date precum liste de string-uri pentru a salva continutul fisierului de intrare si pentru a-l desparti in toate comenzile existente in el. |
| Afisarea in fisier | Rapoartele sunt generate si salvate intr-un fisier PDF salvat in folder-ul aplicatia. Aceste salveaza toate informatiile dintr-o tabela a bazei de date sub un format tabelar. |
| Dezvoltarea algoritmilor | Se folosesc algoritmi de parsing si apelare a procedurilor stocate in baza de date. |
| Impartirea pe clase | Se creeaza clase pentru impartirea functionalitatii in pachete specifice. Se foloseste un pattern Business Logic Architecture pattern. |
| Implementarea solutiei | Se genereaza in faza initiala o diagrama UML pentru a avea structura proiectului, apoi se incepe implementarea propriu-zisa. |
| Testare | Se ruleaza fisiere .txt apoi se verifica corectitudinea prin PDF-urile generate. |

Proiectare

* Decizii de proiectare

Proiectul este impartit in urmatoarele clase: Logic, Parser, PDF, MainFunc, DBase, Data, Client, Order, Product. S-a ales aceasta impartire pentru a separa generarea de fisiere, operatiile pe baza de date si parsarea instructiunilor din fisierul de intrare.

Clasa Client implementeaza toate atributele unui client care sunt preluate din baza de date.

Clasa Order implementeaza toate atributele unei comenzi care sunt preluate din baza de date.

Clasa Product implementeaza toate atributele unei comenzi care sunt preluate din baza de date.

Clasa Dbase are un model de tip Singleton. Aceasta realizeaza conexiunea cu baza de date, care este statica, deci va fi creata doar o singura data pentru a creste performanta.

Clasa Data preia conexiunea create de clasa Dbase si implementeaza metode pentru apel de proceduri si returneaza informatiile transmise de catre baza de date.

Clasa PDF foloseste framework-ul iText pentru a genera fisierele PDF cu informatii. Prima data se genereaza header-ul tabelului din PDF apoi se completeaza cu informatiile. Numele pentru PDF-uri contine numarul acestora care creste pentru fiecare apel al aceleiasi instructiuni.

Clasa Parser desparte fisierul de intrare in instructiuni care trebuie executate de baza de date.

Clasa Logic preia informatiile din clasa parser si le executa, apeland proceduri stocate din baza de date.

* Structuri de date

Structurile de date folosite sunt:

* List: pentru stocarea informatiilor din fisierul de intrare.
* ArrayList: stocarea fisierului de intrarea pentru a putea fi despartit.
* String: folosit la cititrea din fisier si lucru cu comenzile din fisierul text.
* Integer/Double: pentru a salva preturile si stocurile produselor din baza de date.
* Proiectare clase

Clasele sunt:

* Client: structura de client care salveaza informatiile despre acesta.
* Order: structura de comanda care salveaza informatiile despre aceasta.
* Product: stuctura de produs care salveaza informatiile despre acesta.
* MainFunc: utilizat la apelarea functiilor pentru desfasurarea aplicatiei.
* DBase: genereaza conexiunea la baza de date o singura data.
* Data: implementeaza metode pentru preluarea rezultatelor din baza de date sau de apel a procedurilor.
* Logic: implementeaza functii pentru a apela procedurile din baza de date si se foloseste de clasa PDF pentru generarea rapoartelor sub forma PDF.
* Parser: salveaza tot continutul fisierului de intrare, iar apoi il imparte in string-ul aferent apelului de procedura specifica instructiunii din fisier .
* Algoritmi
* Parsing:

Se citeste tot fisierul de intrare si se desparte in instructiunile care trebuie apelate pentru baza de date.

* Apeluri:

Se apeleaza proceduri sau statement de select pentru baza de date.

* Conexiunea la baza de date:

Se realizeaza conexiunea o singura data prin intermediul unei clase de tip Singleton.

* Verificarea operatiilor:

Dupa un numar de adaugari in baza de date se genereaza fisiere PDF cu informatiile din baza de date.

* Scrierea in fisier:

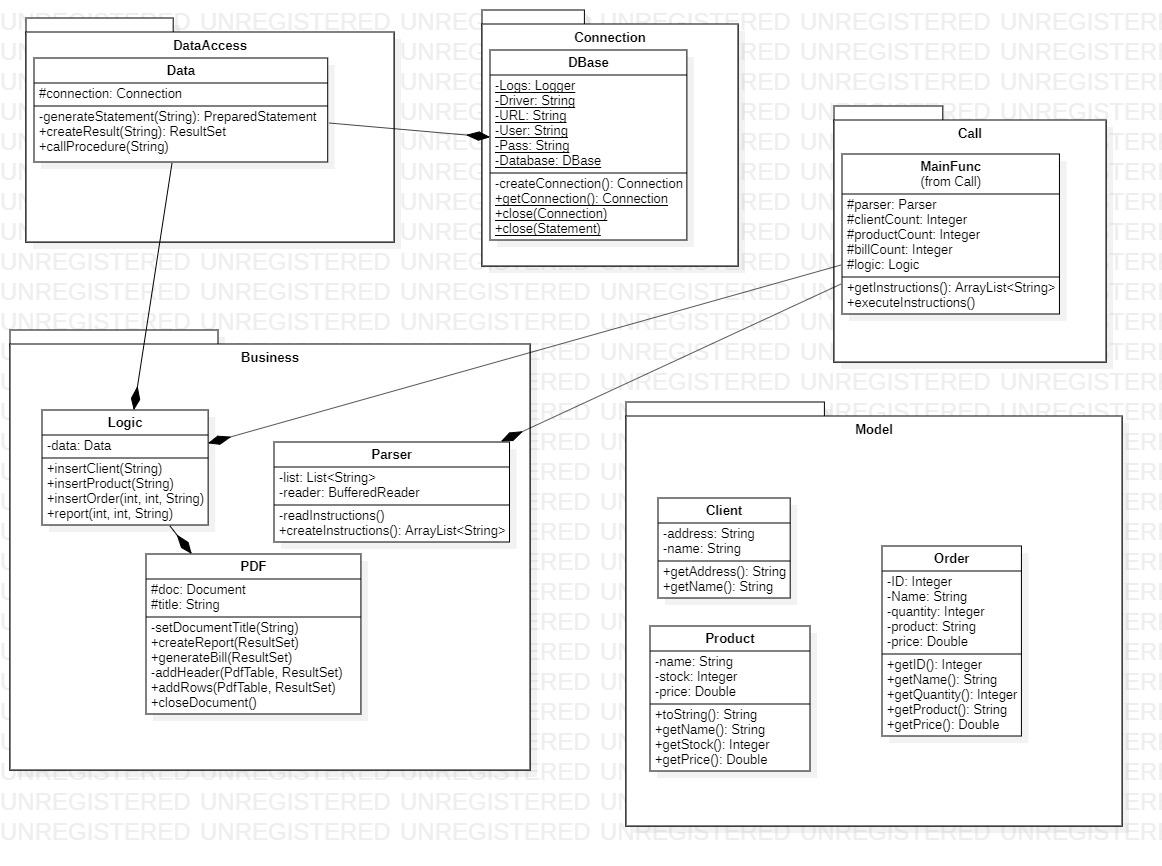
Se genereaza titlu fisierului PDF automat, cu cu un counter pentru a nu se suprascrie. Apoi se scriu informatiile returnate de apelul procedurii bazei de date sub forma de tabel.

* Fisierul de iesire

Fisierele de iesire vor fi create cu numele a ce tabela reprezinta si numarul lor.

Acesta are o structura de forma:

* Titlu:
* Header Tabel:
* Informatii tabel din baza de date.
* Diagrama UML



Clasa Data are relatii de agregare fata de clasele Dbase si Logic.

Clasa Call are relatii de agregare

Clasa Logic are o relatie de agregare cu clasa PDF.

* Packages

Pachetele prezente sunt DataAccess, Connection, Call, Business si Model.

In Connection se intializeaza conexiunea la baza de date o singura data. Atributele din clasa sunt folosite pentru a forma instructiunea necesara pentru conectare.

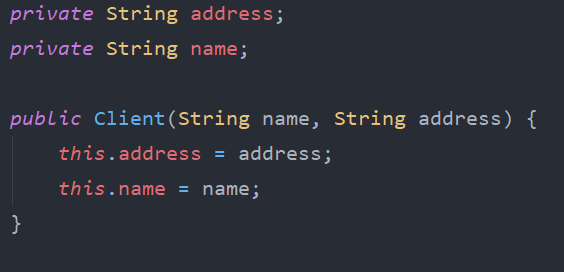
In Call este clasa MainFunc unde sunt apelate functiile pentru rularea aplicatiei.

In Business sunt clasele Logic, Parser, PDF care implementeaza scoaterea instructiunilor din fisierul de intrare. Apoi acestea sunt apelate cu ajutorul functiilor din clasa Logic. Generarea rapoartelor PDF este facuta cu ajutorul functiilor din clasa PDF care utilizeaza framework-ul iText.

S-a ales acest mode de impartire pe pachete si clase pentru a fi separate operatiile pe fisiere si cele pe baza de date. Totodata, aceasta ajuta la vizibilitate si respecta pattern-ul Businesss Layer Architecture.

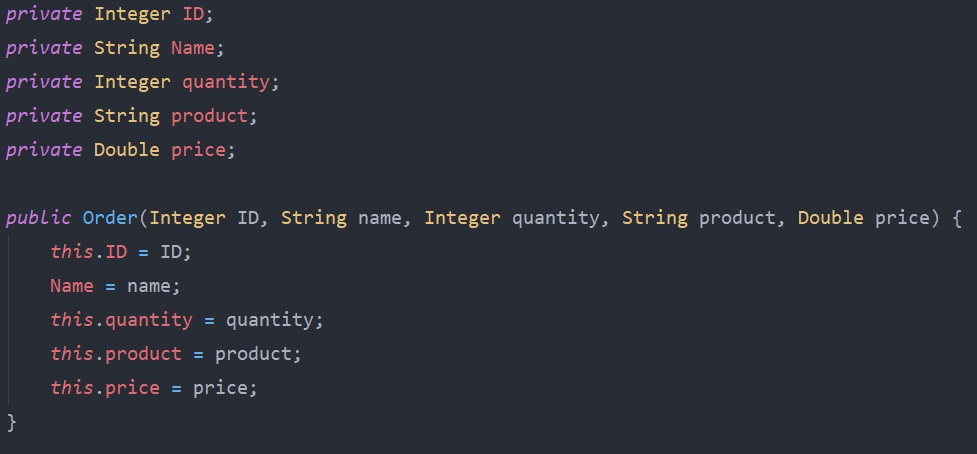
Implementare

* Client



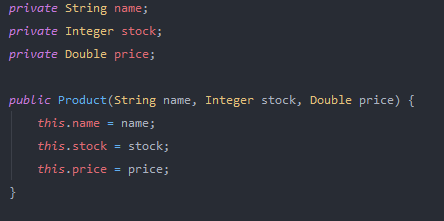
Clasa Client a fost implemenetata pentru a putea retine informatiile despre clienti extrase din baza de date. Acestia sunt definiti de nume si adresa, atat in implementare cat si in baza de date.

* Order



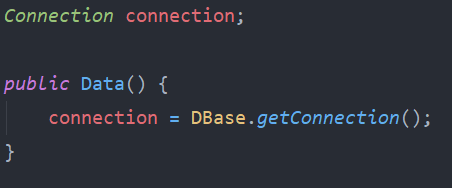
Clasa Order a fost implementata pentru a putea retin informatiile despre comenzi extrase din baza de date si care sunt caracterizate de ID, numele persoanei care o face, cantitatea, produsul specific si pretul acestuia.

* Product

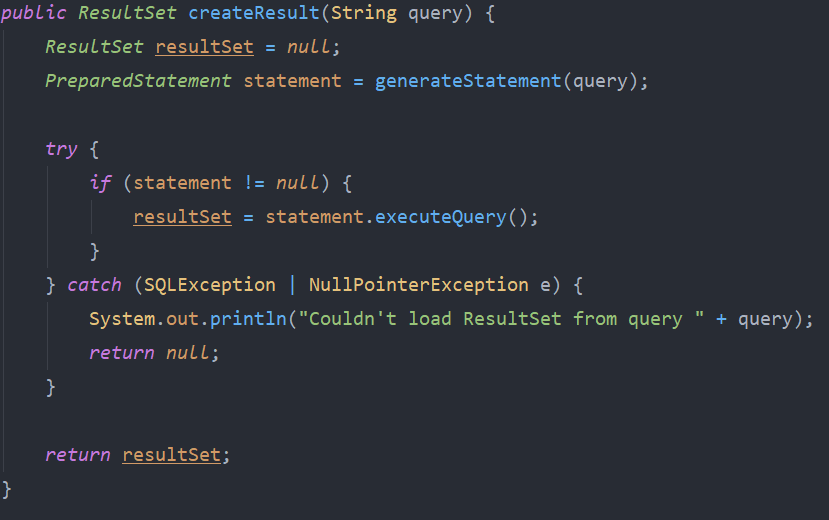


Clasa Product a fost implementata pentru a fi salvate datele despre produse extrase din baza de date. Sunt caracterizate de nume, stocul disponibil si pretul acestora per bucata.

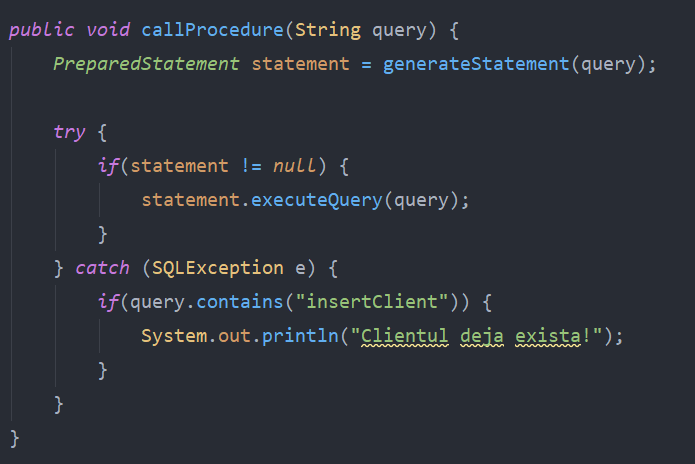
* Data



Clasa Data are un obiect de tip Connection. Aceasta implementeaza functii pentru apeluri de query-uri SQL si returnarea rezultatelor aferente lor.



Executa query-ul trimis ca parametru si returneaza rezultatul acestuia.

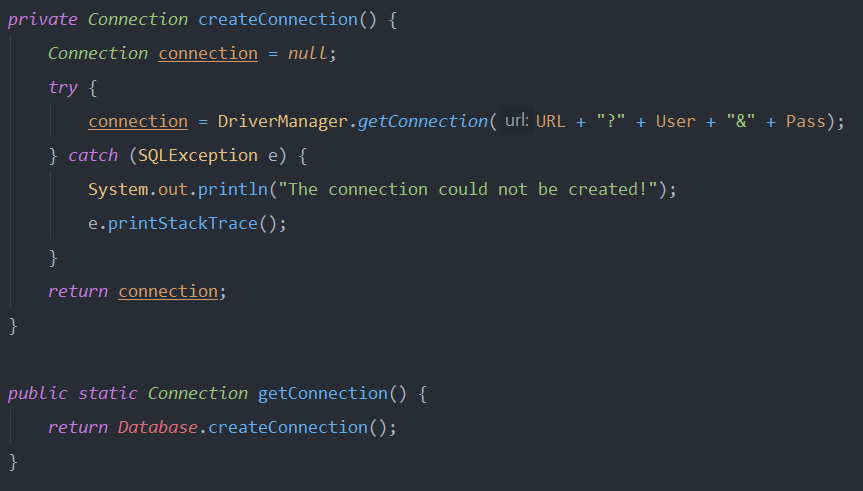


Metoda callProcedure estefolosita pentru a apela o procedura primita ca parametru in query. Aceasta este folosita doar pentru apeluri care nu returneaza valori.

* DBase



Clasa DBase realizeaza conexiunea la baza de date. Aceasta clasa adopta un model Singleton, pentru realizarea conexiunii o singura data, astfel scade timpul de executie pentru ca se reduce numarul de conexiuni create la aceeasi baza de date.



Metoda createConnection creeaza si returneaza obiectul Connection care este folosit pentru a interactiona cu baza de date.

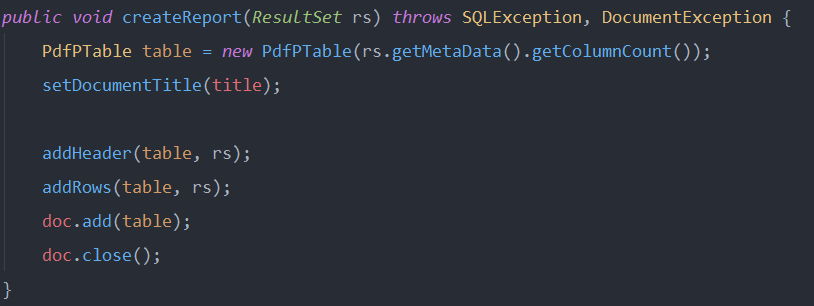
Metoda getConnection returneaza acest obiect, el fiind valabil la compile-time, in clasa Data pentru a fi instantiat si creat doar o singura data.

* PDF



Constructorul clasei PDF decide, pe baza argumentelor primite, ce titlu va primi PDF-ul generat.

Aceasta clasa foloseste framework-ul iText pentru a genera astfel de fisiere.



Creeaza documentul PDF apoi apeleaza functiile addHeader si addRows pentru completarea acestuia.



Creeaza primul rand din tabel, definind un stil diferit de scriere fata de restul si se completeaza cu metadatele rezultatului primit ca parametru.

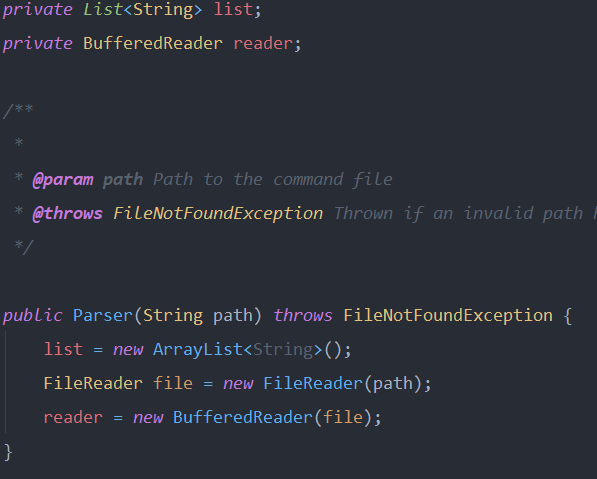


Se comporta la fel ca si adHeader doar ca acesta adauga informatiile din rezultatul primit ca parametru si care corespunde cu informatiile din baza de date la acel moment.



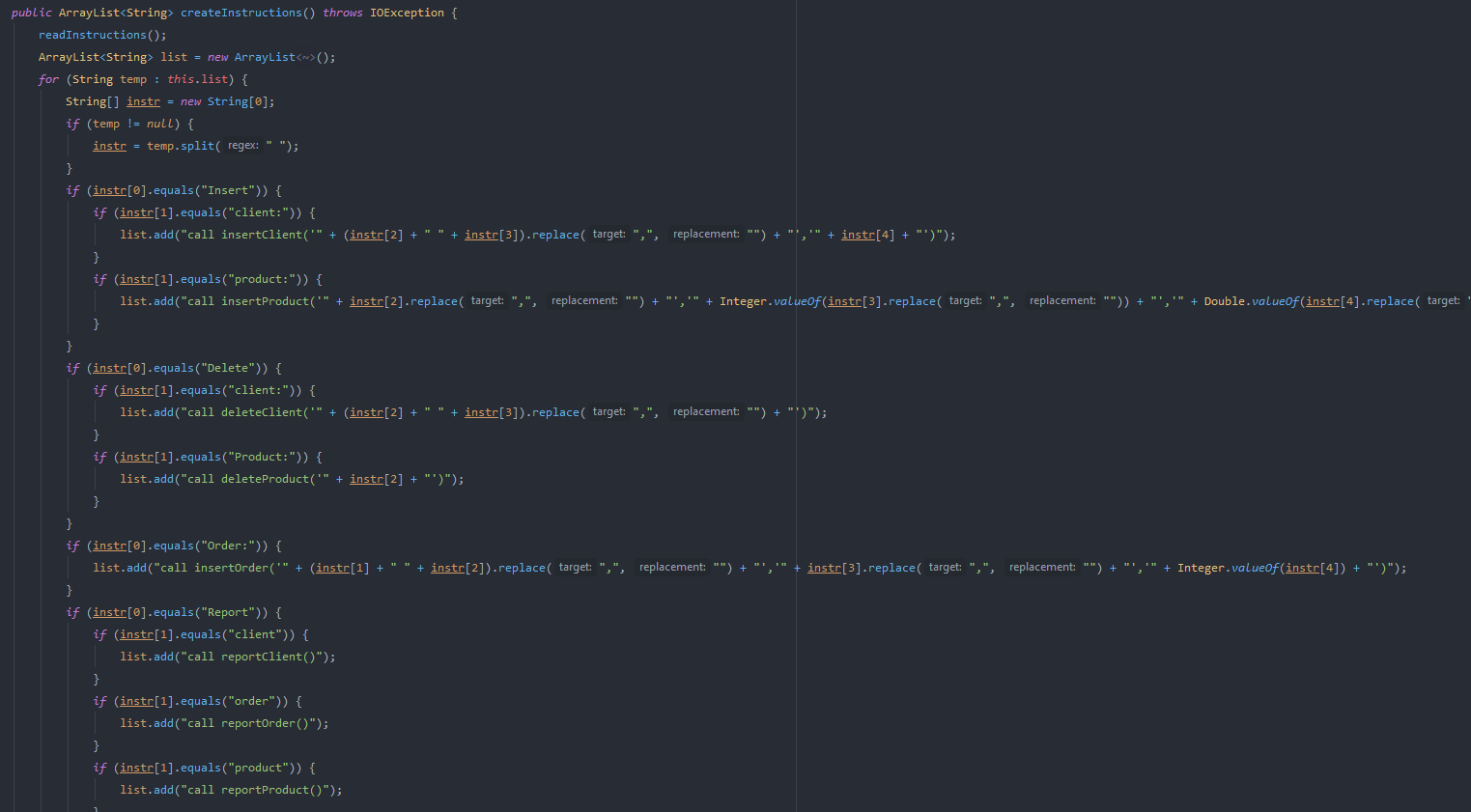
Metoda genereaza un PDF cu informatiile despre comanda efectuata de respectiva instructiune.

* Parser



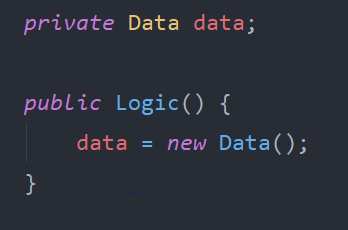
Clasa primeste path-ul catre fisierul de intrare care contine instructiunile de executat de catre aplicatie.

Apoi citeste fiecare linie din acel fisier si le salveaza intr-o lista de string-uri care urmeaza sa fie despartite si transformate in query-uri SQL petntru a fi apelate.

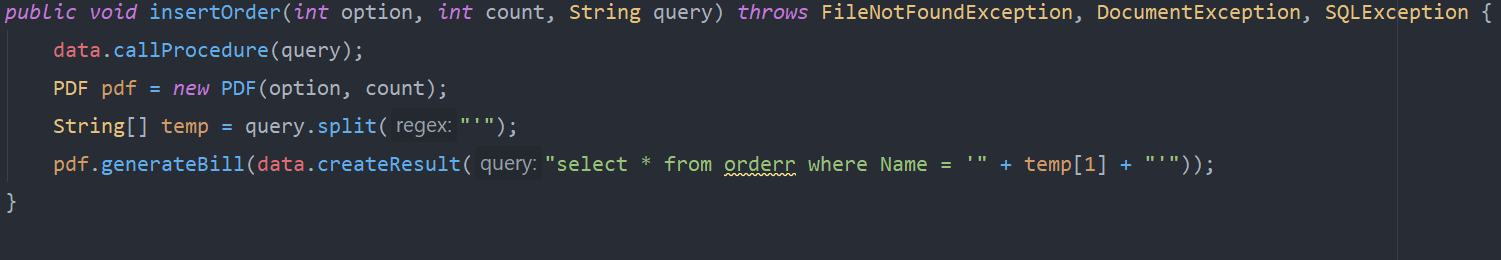


Clasa parcurge toate liniile citite din fisier. Pe fiecare il desparte cu delimitatorul spatiu. Apoi fiecare string rezultat din despartire este verificate pentru a se decide care procedura trebuie apelata.

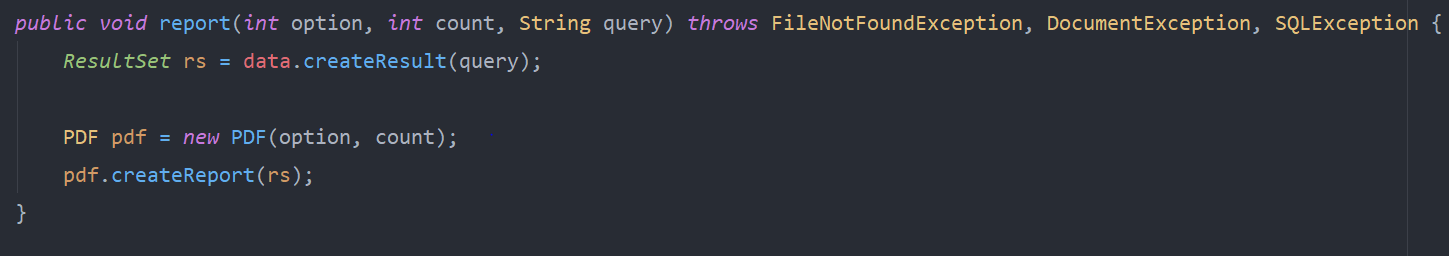
* Logic



Clasa Logic instantiaza un nou obiect de tip Data pentru a putea apela proceduri stocate pentru baza de date.



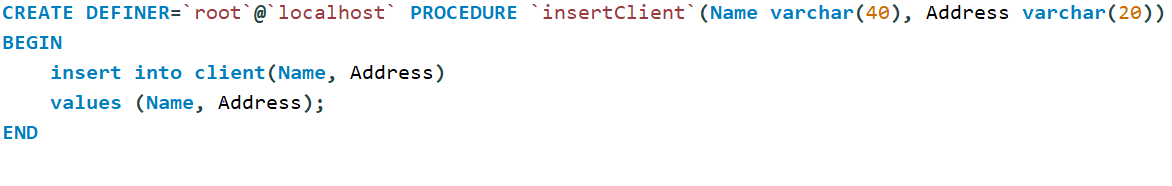
La apelul de procedura pentru inserarea unei noi comenzi, se va genera un fisier PDF care va contine factura pentru respectiva comanda, adica detalii despre aceasta.



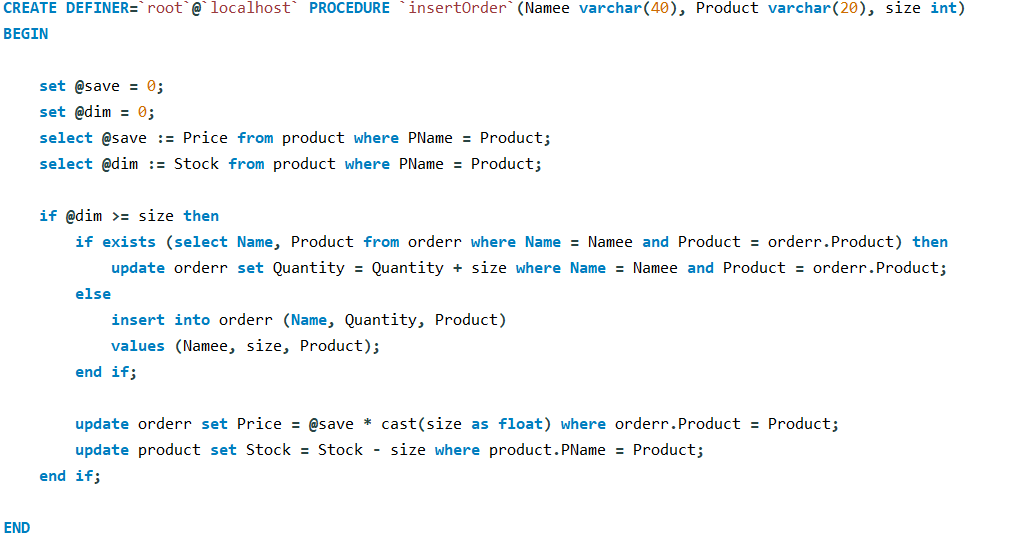
La apelul de procedura pentru a genera un raport, se va genera un fisier PDF care va contine informatii despre respectiva tabela din baza de date.

* Proceduri stocate SQL

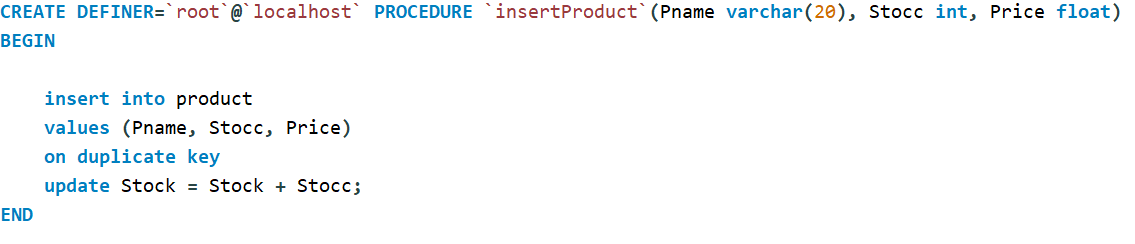
Procedurile sunt: deleteClient, deleteProduct, insertClient, insertOrder, insertProduct, reportClient, reportOrder, reportProduct. Cele care au o implementare care nu este straight-forward sunt insertClient, insertOrder si insertProduct.



Insereaza in table client un nou client, cu numele si adresa date ca parametru.



Aceasta procedura primeste ca parametri numele, produsul si cantitatea care se va comanda. Prima data va salva pretul per bucata al produsului si cantitatea disponibila. Daca cantitatea disponibila va fi destul de mare pentru a se efectua comanda, atunci se trece la pasul urmator. Daca comanda deja exista, atunci se va actualiza cantitatea de produse comandate si pretul final al comenzii. Daca nu exista, atunci se va crea o noua comanda si se va calcula pretul final al acesteia. Dupa realizarea uneia dintre aceste optiuni, in tabela product se va actualiza stocul ramas al respectivului produs.

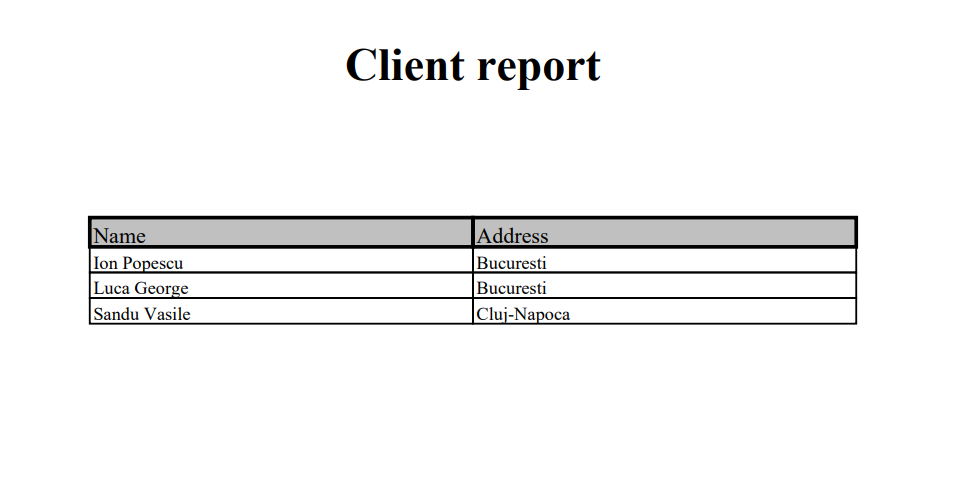


Aceasta procedura primeste ca argumente numele produsului, stocul pe care acesta il are si pretul sau per bucata. Daca produsul nu exista in stoc, atunci va fi adaugat. Daca acesta exista, atunci I se va actualiza doar stocul, adica se aduna parametrul Stocc la stocul existent.

Rezultate

Pentru comenzile de report client, report order, report product, se genereaza cate un fisier PDF denumit sugestiv care contine informatiile despre fiecare tabela in parte.

Se ruleaza aplicatie, iar aceste pdf-uri sunt generate automat daca exsita comenzile de mai sus. Mai jos este un exemplu de fisier generat la comanda report client.



Concluzii

Ideea cea mai importanta care a rezultat din realizarea acestei teme este, cu siguranta, importanta folosirii unui pattern care sa corespunda cu necesitatile de implementare ale proiectului. Astfel, s-a folosit un Business Logic Architecture pattern pentru a diferentia lucrul cu baza de date, cel cu fisiere si cel cu gestiunea informatiilor din baza de date.

Este prezenta si nevoia de realizare a unui cod “curat”. Necesitatea de a realiza un cod atat de inteligibil incat nu are nevoie de a fi comentat este imperativa, deoarece reintoarcerea la el si/ sau modificarea acestuia devine mult mai usoara. Totodata, impartirea codului in clase si pachete specifice unei anumite probleme permite reutilizarea acestuia si evita scrierea unor bucati largi de cod care vor fi folosite doar o singura data.

Elementul care necesita dezvoltare ulterioara este extinderea bazei de date, astfel putand fi adaugate functionalitati ulterioare aplicatiei. Totodata, un must-have pentru acest tip de aplicatii este o interfata grafica, pentru a renunta la folosirea fisierelor text pentru a prelua date de intrare.

Bibliografie

Database Connectivity: <https://www.javatpoint.com/steps-to-connect-to-the-database-in-java>

PDF Generation: <https://www.tutorialspoint.com/itext/index.htm>

File I/O: <https://stackabuse.com/reading-and-writing-files-in-java/>