Tema numărul 2
Aplicație de gestiune a unui depozit

Nume: Șerban
Prenume: Sebastian – Mihai
Grupa: 30223
Specializare: Calculatoare și Tehnologia Informației

Content

1. Obiectivul temei

Principalul obiectiv al temei este acela de a implementa o aplicație de gestiune a unui depozit de marfă pentru of firmă oarecare de curierat. Această aplicație trebuie să fie capabilă să realizeze operații precum: introducere și eliminare din baza de date a anumitor produse sau clienți, plasarea de comenzi și verificarea stocului curent.

Pentru a putea îndeplini acest obiectiv a fost necesară împărțirea aplicației în mai multe obiective precum:

- Abstractizarea și transpunerea clineților, a mărfurilor și a comenzilor în paradigma programării orientate pe obiecte.
- Crearea de clase ce lucrează cu principiul reflexiei pentru a elimina scrierea de cod redundant sau copierea acestuia ineficinet.
- Implementarea operației de adăugare client.
- Implementarea operației de ștergere client.
- Implementarea operației de editareclient.
- Implementarea operației de listare clienți.
- Implementarea operației de adăugare produs.
- Implementarea operației de ștergere produs.
- Implementarea operației de editare produs.
- Implementarea operației de listare poroduse.
- Depuneri de comandă.

2. Analiza problemei. Modelarea. Scenarii și cazuri de utilizare.

Pentru a reuși să duc implementarea la bun sfârșit, a fost nevoie ca în fază incipientă să studiez și să imaginez sisteme de gestiune a depozitelor pentru site-uri specifice precum Emag, Media Galaxy, Amazon, etc. Am observat că majoritatea aveau ceva în comun. Interfața grafică aerisită și o sumedenie de modiltăți prin care clientului să îi fie ușurată munca de a depune o comandă și a căuta produsul potrivit.

Deoarece, deși am încercat să simplific procesul cât mai mult, tot a fost nevoie de introducerea exactă de la tastatură a unor anumite câmpuri, am fost nevoit să declar o sumedenie de validatori care îmi asigură corectitudinea datelor ce intră, iar apoi ies din baza de date. De asemenea, pentru a facilita depunerea de comezni, am conceput un sistem ce folosește un număr infim de date de legătură pentru un ecosistem atât de mare.

Așadar am ajuns la concluzia că este absolut necesară utilizarea unei baze de date scrisă în limbaj mySQL pentru a putea reține toate datele sub o formă de tabel. Am început să proiectez o bază de date relațională care să îmi satisfacă nevoile pentru acest proiect.

În urma unei analize mai amănunțite, am observat că există 3 use case-uri principale și anume: Use case-ul 1 – este vorba despre un patron sau un administrator care are nevoie să updateze datele drespre produs și clienți. Ei bine, pentru acest lucru a fost nevoie de împățirea pe 3 mari categorii ale operațiilor: operațiile care operează cu date despre clienți, operațiile care operează cu date despre produse și operațiile care operează cu plasarea de comenzi. Absolut fiecare pas este făcut în strânsă legătură cu o bază de date pe care am conectat-o la aplicație.

Cel de-al doilea use case este din perspectiva unui client ce dorește să depună o comandă sau să își editeze datele personale. Pentru acest lucru am realizat operațiile create order și edit client. Aceste operații înglobează în două interfețe sugestive un amalgam de pași pe care clientul ar fi trebuit să îl facă.

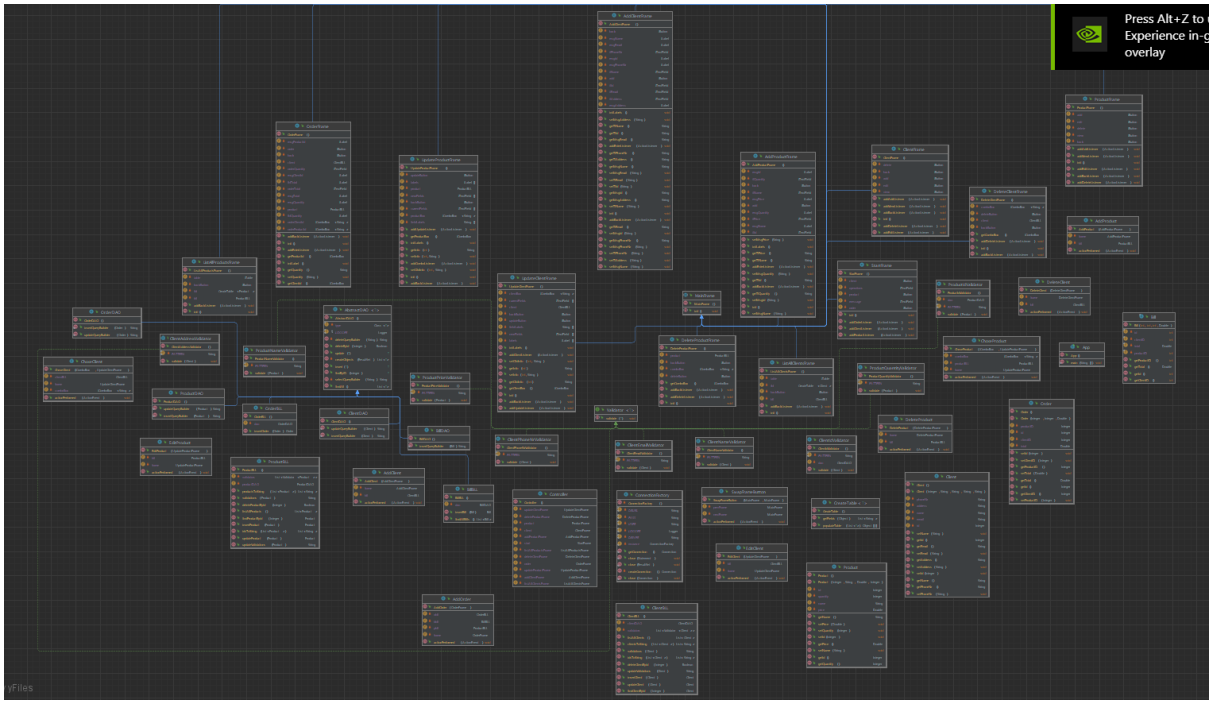
Cel de-al treila use case este cel de ementenanță pe care îl face un angajat de specialitate. Pentru acest use case am adăugat posibilitatea de modificare a datelor despre produs, listare a clienților și a produselor și capacitatea de a putea șterge un produs sau un client. De asemenea ei ar putea avea acces și la acea clasă Bill care este puțin mai specială.

Deoarece numărul de operații pe care trebuie să îl efectuez era foarte mare, a fost nevoie și să împart proiectul în mai multe layer-e, fiecare având un anumit grad de acces la datele din baza de date și nu numai. Astfel am ajuns să am o arhitectură cu 50 de clase și 7 package-uri. Cele 7 layere sunt: Presentantion, Model, Driver, Connection, Data Acess, Business Logic, Buttons.

- Layer-ul Presentation are slab acces la elementele din baza de date și conține toate clasele ce descriu elemente de interfață grafică.
- Layer-ul Model nu are deloc acces la baza de date, el doar stochează un model de date ce replică atât tabelele cât și înregistrările din baza de date.

- Layer-ul Driver se ocupă de punerea laolaltă a tuturor componentelor pentru a face aplicația să funcționeze.
- Layer-ul Connection se află în directă legătură cu baza de date, el are scopul de a furniza conexiuni la baza de date pentru orice calsă necesită așa ceva.
- Layer-ul Data Acess asigură execuția tuturor operațiilor CRUD și are acces direct la Layer-ul Connection.
- Layer-ul Business Logic are scopul de a lega între ele Data Acess și Model pentru a putea să stocăm temporar datele din tabele, fără a fi nevoie să realizăm o mulțime de interogări SQL.
- Layer-ul Buttons se ocupă de toate ascultătoarele de acțiuni și descriu funcționalitatea fiecărui buton în cel mai mic detaliu.

În urma acestui mod de lucru am ajuns la o diagramă UML a claselor care este mult prea mare pentru a putea fi exemplificată în această documentație, dar o să pun o imagine de ansamblu ca să ne putem face o idee despre legăturile dintre clase și relațiile dintre ele.



3. Implementarea

Implementarea este foarte bine descrisă în fișierele javadoc pe care le-am atașat și generat proiectului. Ce pot spune este că cea mai importantă calsa este de departe AbstractDAO deoarece m-am folosit de tehnica reflexiei pentru a construi query-urile dar și clasa CreateTable este aproximativ egal de importantă deoarece mă ajută să creez tabele fără a fi nevoie de a construi un pattern unic pentru fiecare tabel în parte.

Consider că package-ul de butoane este din nou un punct forte deoarece operează atât cu operații CRUD, operează și cu elemente din Layer-ul BLL, din layer-ul model, DAO și GUI. Aș putea spune că acest pachet este cel mai complex și mai bine inclus în proiect.

4. Concluzii

În concluzie, aș putea spune că acest proiect a fost un proiect extrem de interesant și totodată extrem de intrigant deoarece am folosit foarte multe tehnici noi pe care nu le cunoșteam până acum.

Cel mai important asset pe care l-am câștigat în urma acestui proiect este clar dobândirea metodei reflexiei și faptul că acum reușesc cu ușurință să gândesc lucrurile mai general și să standardizez codul fără a avea cod duplicat sau redundant.

De asemenea, am observat un progres pe partea de interfețe grafice, mai ales la partea de alinieri și de așezări în pagină, lucru ce până acum nu era neapărat un punct forte ci dimpotrivă, reprezentau un punct slab al proiectelor mele. Consider că și în acest sens am făcut progrese deoarece am ajuns să lucrez cu containere de componente ceva mai sofisticate decât un simplu Jlabel sau un JtextField.

Ca și îmbunătățiri, consider că încă aș fi putut face și mai modular operațiile CRUD și construcția de query-uri.

De asemenea, clasa Bill aș fi putut să o implementez mai sofisticat și să mă asigur că într-adevăr poate să păstreze datele din Log.

5. Bibliografie

https://chat.openai.com

How to Use JavaDoc to Document Your Classes - dummies

How To Create an Immutable Class in Java | DigitalOcean

How to Use Tables (The Java™ Tutorials > Creating a GUI With Swing > Using Swing Components) (oracle.com)

Delete Contents From Table Using JDBC - GeeksforGeeks

A Java MySQL DELETE example | alvinalexander.com

java - How to update JComboBox content from ArrayList? - Stack Overflow