DOCUMENTATIE

TEMA *3*

NUME STUDENT: …Truta Andrei…
GRUPA: ……30221……

CUPRINS

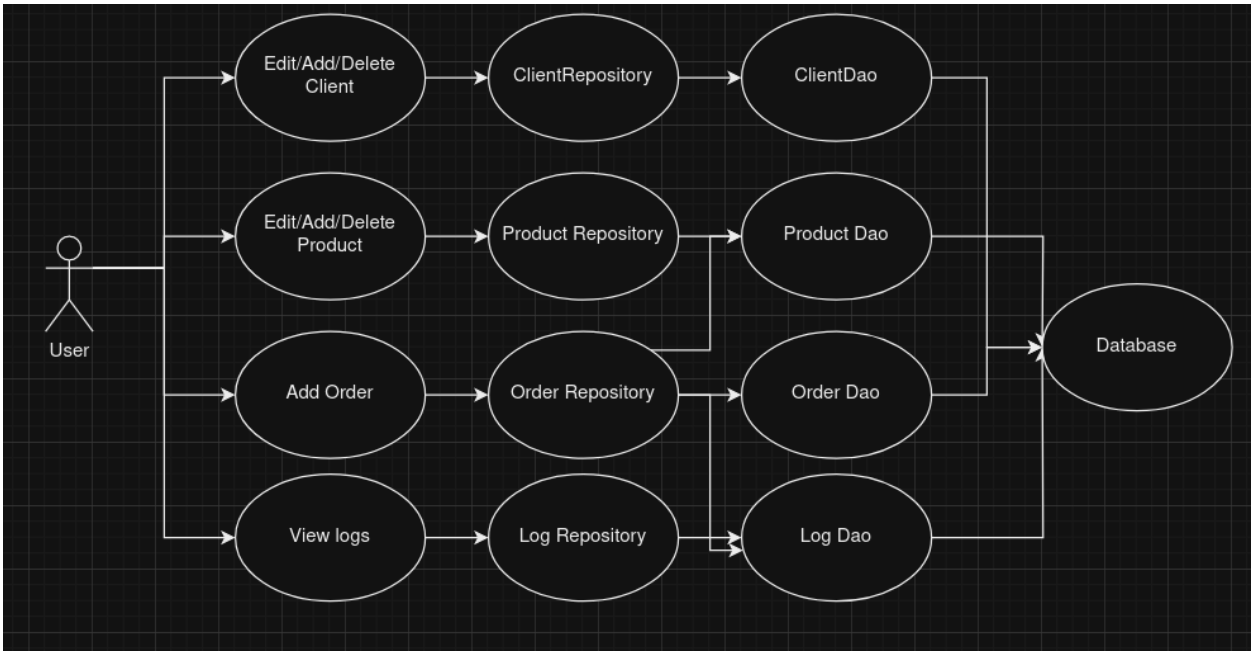
1. Obiectivul temei

Obiectivul principal al temei a fost de a realiza o aplicație ce se folosește de o bază de date pentru stocarea datelor ce necesită persistență. Pașii urmați pentru a dezvolta aplicația au fost următorii:

1. *Gândirea arhitecturii*
2. *Definirea structurii proiectului*
3. *Implementarea layer-ului pentru acces la date*
4. *Implementarea aplicației (business + model + presentation layers)*

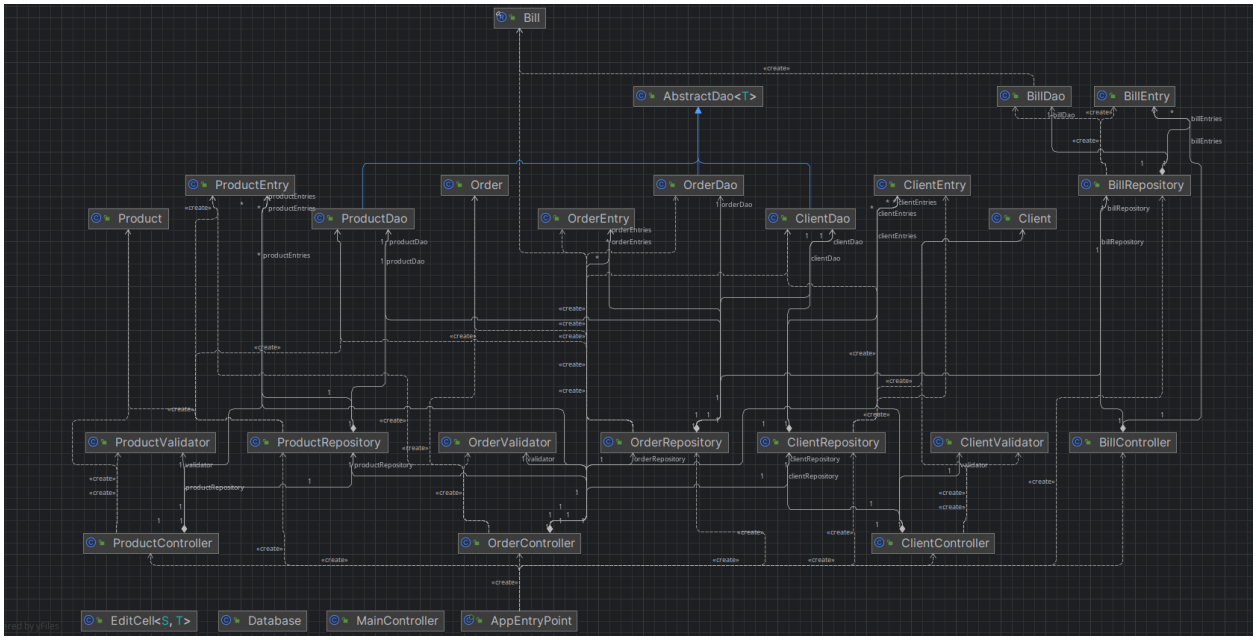
2. Analiza problemei, modelare, scenarii, cazuri de utilizare

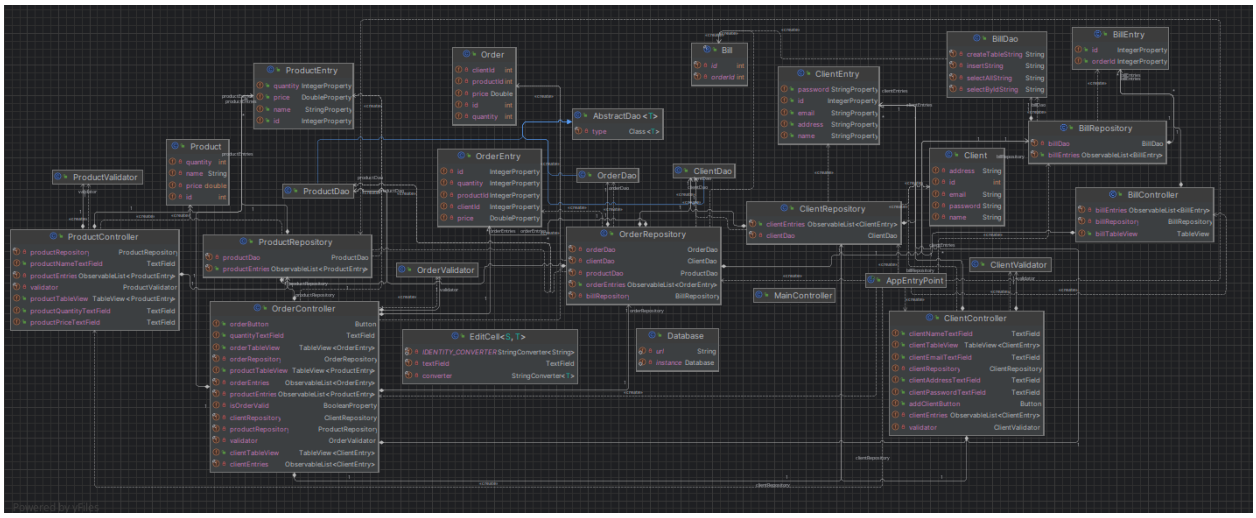
Cerințele funcționale ale aplicației sunt: permiterea utilizatorului să vizualizeze datele din tabelele de clienți, produse, comenzi și log-uri, precum și adăugarea și șteregerea în tabelele de clienți și produse, și adăugarea de comenzi. De asemenea, există validatoare care nu permit continuarea introducerii în baza de date a unor valori fără sens (de exemplu, preț negativ).
Ca și cerințe nefuncționale, avem modalitatea de modificare a tabelelor și de stergere din acestea. Pentru modificare, tabelul este implementat printr-o metoda care-i permite modificarea celulelor, iar pentru ștergere, un clic dreapta va deschide un meniu pentru înregistrarea respectivă cu un buton de ștergere.



3. Proiectare

Mai jos este prezentată structura proiectului pe clase și relațiile dintre acestea.





4. Implementare

Interfața a fost realizată prin intermediul unui TabView, cu 4 tab-uri pentru clienți, produse, comenzi și log-uri.
Tabelele sunt interactive și pentru adăugare regăsim o bară orizontală sub tabele pentru completarea datelor.
Clasele importante și descrierile lor:

- *BillController – controller pentru pagina de log-uri, gestionează generarea tabelului și instanțe de repository pentru popularea acestuia*
- *ClientController – controller pentru pagina de clienți, gestionează generarea tabelului, adăugarea în acesta (preluarea input-ului de la utilizator) și instanțe de repository pentru popularea acestuia*
- *OrderController – controller pentru pagina de comenzi, gestionează 3 tabele pentru generarea și vizualizarea comenzilor, instanțe de repository*
- *ProductController – controller pentru pagina de produse, gestionează tabelul pentru produse, preluarea input-ului de la utilizator și instanțe de repository pentru populare*
- *ClientValidator – clasă pentru validarea unui client; verifică dacă toate proprietățile unei instanțe sunt valide luând în considerare un set de reguli*
- *OrderValidator - clasă pentru validarea unei comenzi; verifică dacă toate proprietățile unei instanțe sunt valide luând în considerare un set de reguli*
- *ProductValidator - clasă pentru validarea unui produs; verifică dacă toate proprietățile unei instanțe sunt valide luând în considerare un set de reguli*
- *AbstractDao – clasă abstractă generică pentru operații CRUD (insert, update, delete, get) generice aplicabile mai multor Dao-uri*
- *BillDao – clasă pentru accesul la tabela de log-uri, operații de get și insert*
- *ClientDao - clasă pentru accesul la tabela de clienți, operații de get, insert, update, delete*
- *OrderDao - clasă pentru accesul la tabela de log-uri, operații de get și insert*
- *ProductDao - clasă pentru accesul la tabela de clienți, operații de get, insert, update, delete*
- *BillRepository – clasă pentru accesul la baza de date cu scopul de a lucra cu obiecte ce țin de log-uri, în cazul de față doar log-uri; de asemenea se ocupă de menținerea tabelelor de log-uri actualizate în toată aplicația*
- *ClientRepository - clasă pentru accesul la baza de date cu scopul de a lucra cu obiecte ce țin de clienți, în cazul de față doar clienți; de asemenea se ocupă de menținerea tabelelor de clienți actualizate în toată aplicația*
- *OrderRepository - clasă pentru accesul la baza de date cu scopul de a lucra cu obiecte ce țin de comenzi, în cazul de față accesează atât tabela de comenzi cât și cea de produse și*

log-uri; de asemenea se ocupă de menținerea tabelelor de comenzi actualizate în toată aplicația

- *ProductRepository - clasă pentru accesul la baza de date cu scopul de a lucra cu obiecte ce țin de produse, în cazul de față doar produse; de asemenea se ocupă de menținerea tabelelor de produse actualizate în toată aplicația*
- *Database – clasă cu metode statice pentru gestiunea conexiunii la baza de date*
- *BillEntry – clasă care modelează un Bill dar cu proprietăți observabile pentru tabel*
- *ClientEntry - clasă care modelează un Client dar cu proprietăți observabile pentru tabel*
- *OrderEntry - clasă care modelează un Order dar cu proprietăți observabile pentru tabel*
- *ProductEntry - clasă care modelează un Product dar cu proprietăți observabile pentru tabel*
- *EditCell – clasă care extinde un TableCell normal în scopul de a-i permite sa fie modificat în timp real prin dublu-click*
- *Bill – clasă record pentru log-uri*
- *Client – clasă model pentru clienți*
- *Order - clasă model pentru comenzi*
- *Product - clasă model pentru produse*
- *AppEntryPoint – clasă care se ocupă de bootstrapping – inițializarea JavaFX, createa repository-urilor și injectarea lor în contructori*

5. Rezultate

Rezultatul temei este o aplicație care permite gestiunea unui sistem de comenzi din perspectiva unui administrator/power-user. Baza de date generată (dump) poate fi gasită sub forma de fișier .sql pentru testare.

6. Concluzii

În cadrul acestei teme am învățat cum se poate lega o bază de date de o aplicație grafică, modul de lucru cu baza de date respectivă, menținerea consistenței bazei de date în toată aplicația și și în baza de date respectivă. De asemenea, am aprofundat lucrul cu JavaFX și tabelul oferit de acesta, folosind factory-uri pentru generarea unor tabele mai avansate care permit editare direct în ele și meniuri de tip context menu în urma unor event-uri precum clic dreapta. De asemenea am învățat să lucrez cu observables, precum liste, care trimit notificări când conținutul lor se schimbă.

7. Bibliografie

1. *pt-layered-architecture - https://gitlab.com/utcn_dsrl/pt-layered-architecture*
2. pt-reflection-example - https://gitlab.com/utcn_dsrl/pt-reflection-example
3. Editable JavaFX Table Cell - https://gist.github.com/james-d/be5bbd6255a4640a5357
4. Java SQLite Tutorial - https://www.sqlitetutorial.net/sqlite-java/
5. Editable Tables in JavaFX - https://dzone.com/articles/editable-tables-in-javafx
6. DAO vs Repository Patterns - https://www.baeldung.com/java-dao-vs-repository