UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Shoes Store

Name:Danci Ionela
Group: 30236

Table of Contents

Project Specification

Aplicatia are doua tipuri de utilizatori: User (comparator) si Admin. Cele doua tipuri de utilizatori au actiuni diferite asupra aplicatiei.

Datele legate despre produse, utilizatori, cos de cumparaturi si favorite sunt pastrate in baza de date.

Functional Requirements

- Identificarea tipului de user
- Cumparatorul poate vizualiza produse pe categorii
- Cumparatorul poate adauga/sterge un produs la favorite
- Cumparatorul poate adauga/sterge un produs in cos
- Cumparatorul poate incrementa cantitatea unui produs in cos
- Cumparatorul poate face o comanda
- Administratorul poate adauga/sterge/modifica produse
- Administratorul poate vedea toate comenzile/utilizatorii

Use Case Model 1

Use Cases Identification

1.
Use case: Adaugare produs
Level: administrator level
Primary actor: Administrator
Main success scenario:
- completare email si parola
- logare cu succes
- Click pe butonul de adaugare a unui produs
- completare formular cu datele cerute
- click pe butonul de adaugare
- primire mesaj de confirmare ca adaugarea s-a facut cu success

Extensions:
- daca email-ul sau parola sunt gresite, accesul in aplicatie nu este permis

2.
Use case: Inregistrare client
Level: user-goal level
Primary actor: Cumparator
Main success scenario:
- introducere email si parola dorita
- primire meaj daca inregistrarea s-a facut cu success

Extensions:
- daca email-ul sau parola au dimensiunea mai mica de 5, accesul in aplicatie nu este permis

3.
Use case: Cautare incaltaminte
Level: User goal

Primary actor: User
Main success scenario: Utilizatorul cauta pentru un tip specific de brand sau o categorie specifica si o sa se afiseze incaltaminte care respecta criterile selectate de acesta

Extensions:
- Utilizatorul poate sorta produsele dupa alt factor, cum ar fi pret sau marime
- Utilizatorul poate selecta un produs anume unde poate vedea toate specificatiile acestuia

UML Use Case Diagrams

Supplementary Specification

Non-functional Requirements

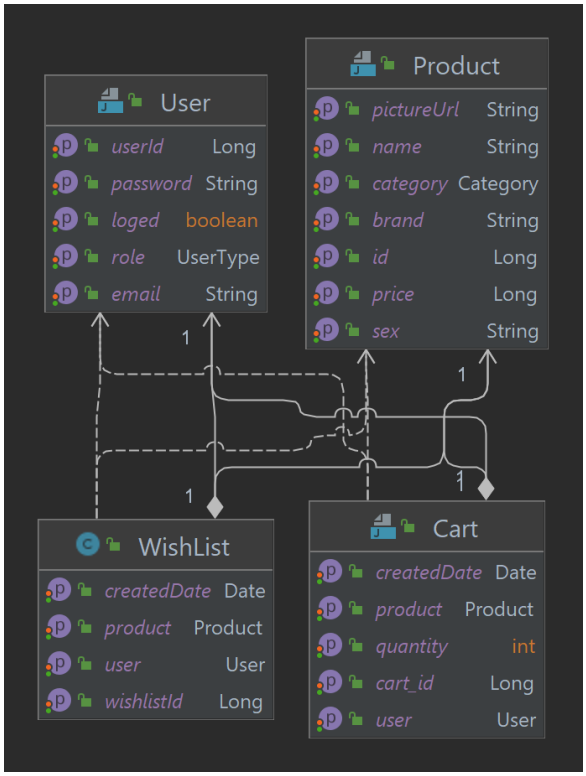
User Experience : Aplicatia este intuitiva, este usor de de navigat in interfata, asigurand o experienta pozitiva pentru user
Usability : Aplicatia este usor de folosit
Privacy : Aplicatia respecta datele private ale utilizatorilor
Performance: Aplicatia raspunde rapid cererilor

Design Constraints

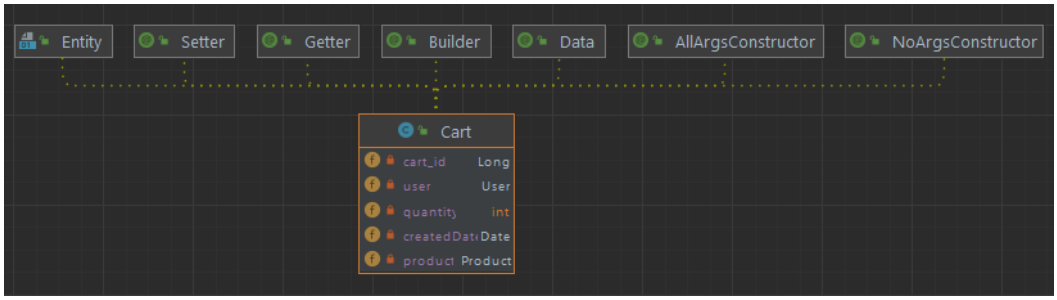
Compatibility: Aplicatia trebuie sa fie compatibila cu toate sistemele de operare
Intuitive Interface: Interfata aplicatiei trebuie sa fie usor de inteles pentru utilizator
Visual Design: Aplicatia trebuie sa aiba un design atragator, in conformitate cu imaginea brandului

Domain Model

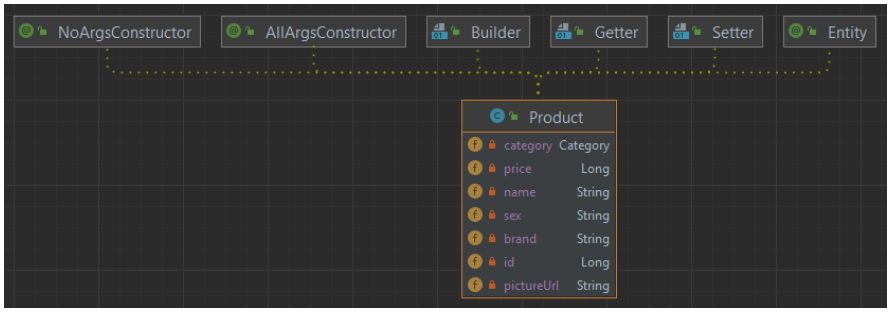
-este locul unde are loc modelarea etajului de business.
In cadrul acestui proiect sunt 4 entitati: User, Product, Cart, WishList



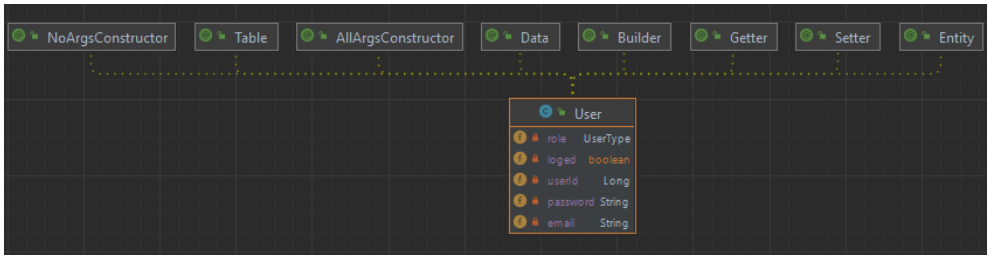
Cart:



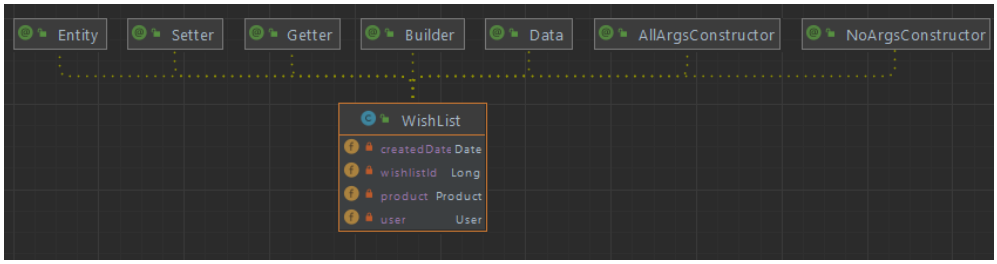
Product:



User:



WishList:



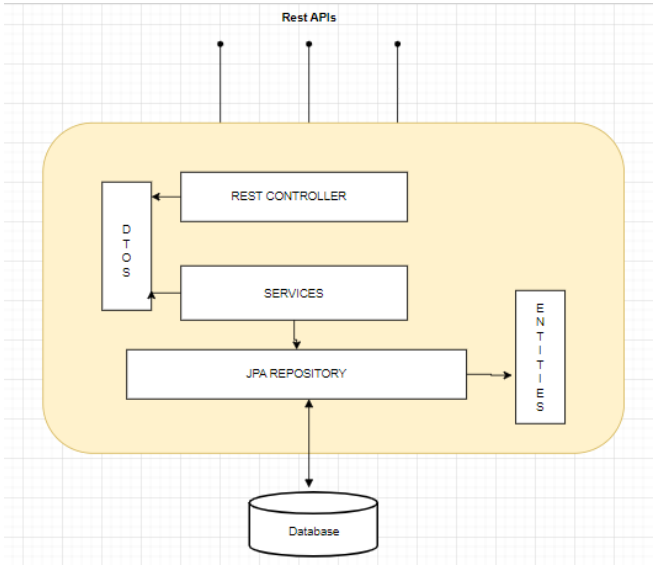
Architectural Design

Conceptual Architecture

In acest proiect am utilizat arhitectura Spring Boot.

Aplicatia contine cele 4 etaje:

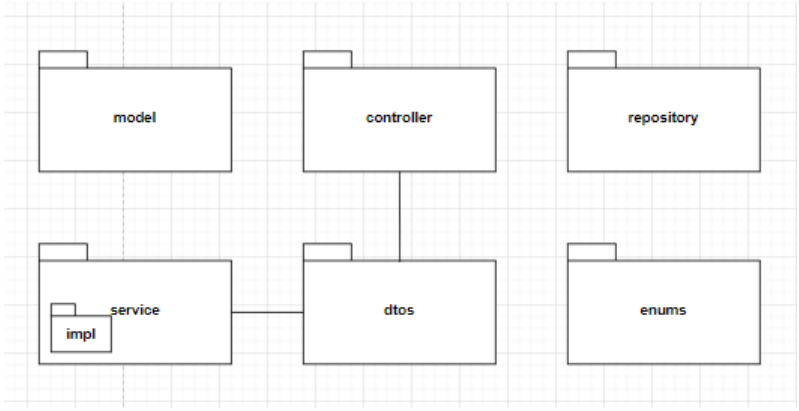
- Presentation Layer: reprezinta partea de front-end, aceasta este realizata utilizand react js.
- Business Layer: consista in toate clasele de service
- Persistence Layer: contine logica de stocare in baza de date
- Database Layer: contine baza de date (MySql)



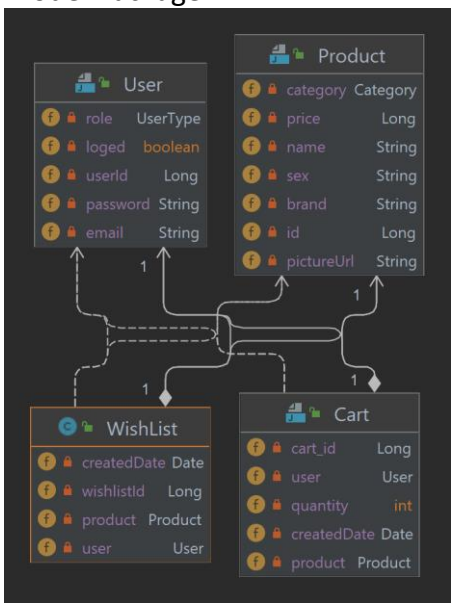
Patterns:

- Singleton: in mod implicit, spring creaza toate bean-urile ca si fiind singleton
- Proxy pattern: am folosit anotatia @Transactional la metodele de delete din CartRepository si din WishListRepostory. Aceasta adnotare indica ca spring sa execute atomic aceste metode. Fara proxy. Spring nu ar putea controla accesul la bean-ul nostru CartRepository, respective WishlistRepository si nu ar putea asigura coerenta tranzactionala a acestuia

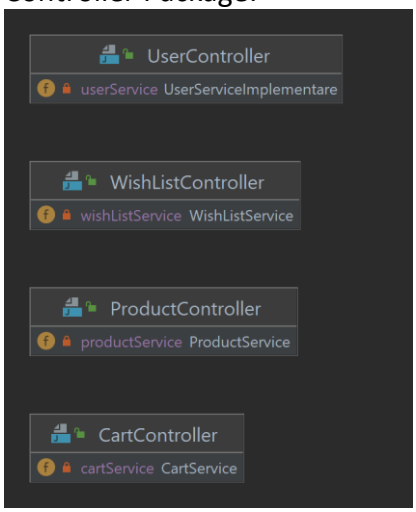
Package Design



Model Package:

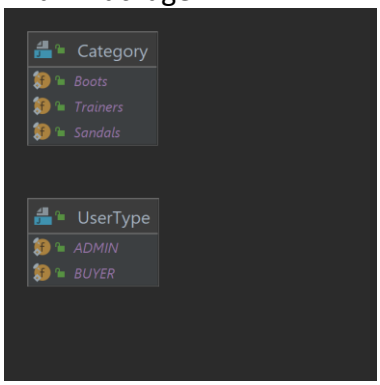


Controller Package:

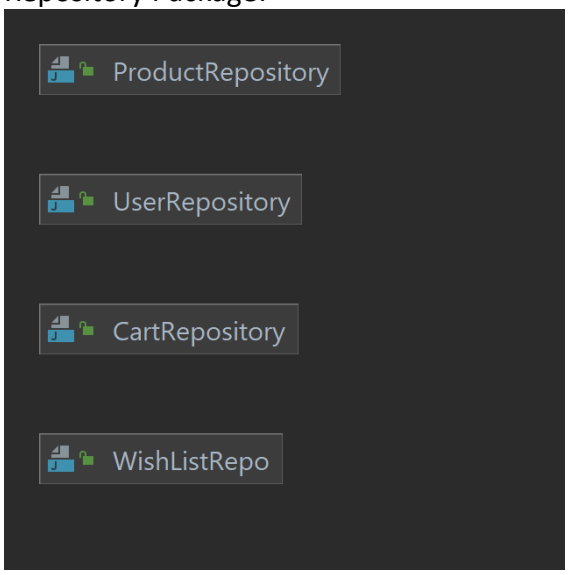


Dto Package:

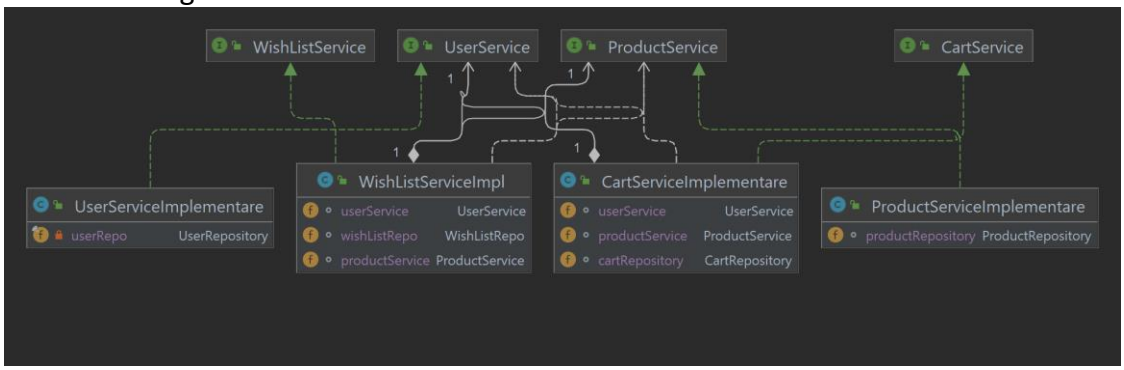
Enum Package:



Repository Package:

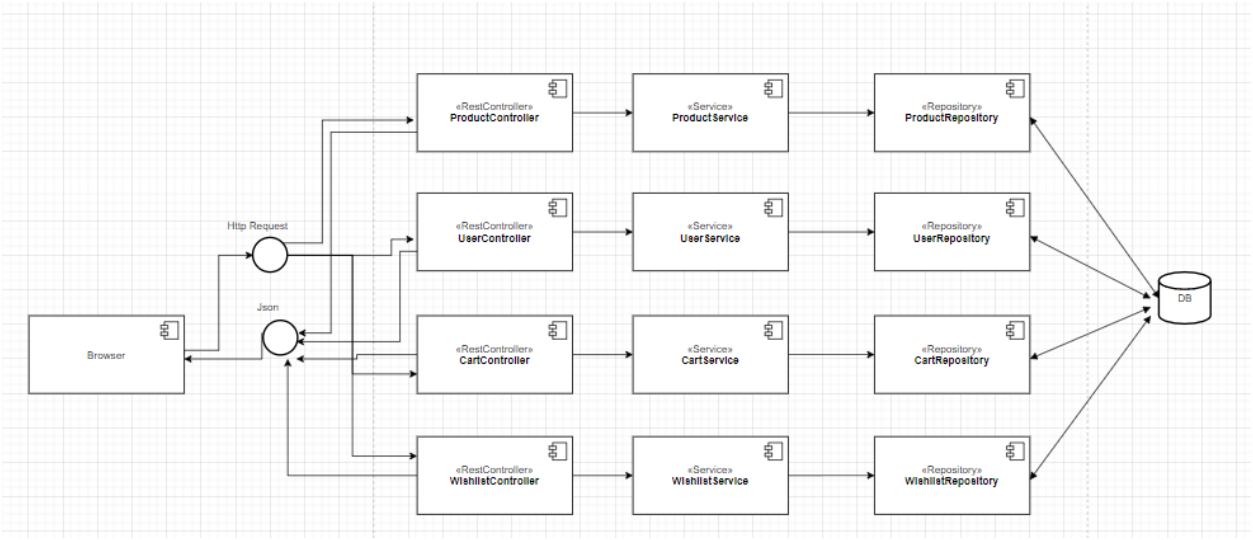


Service Package:

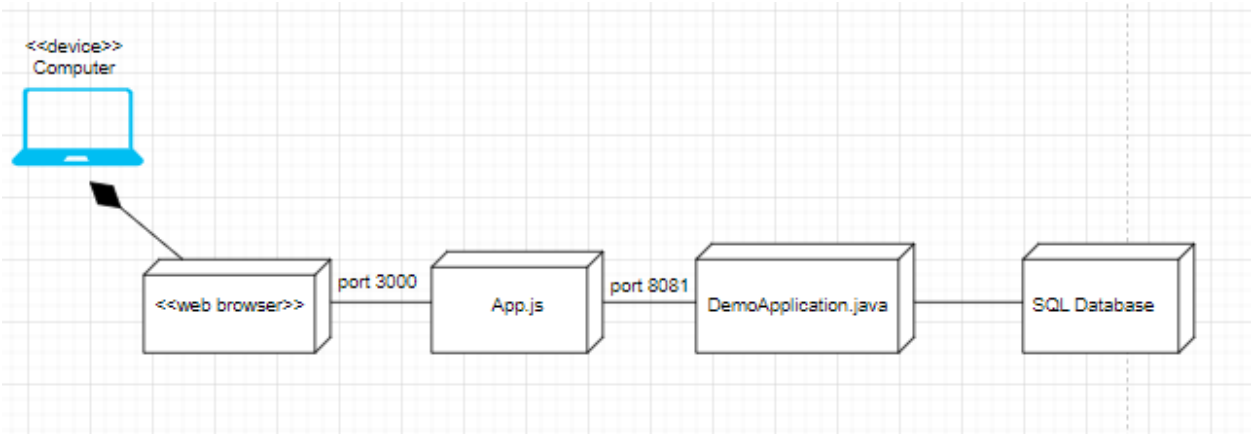


Component and Deployment Diagram

Component diagram

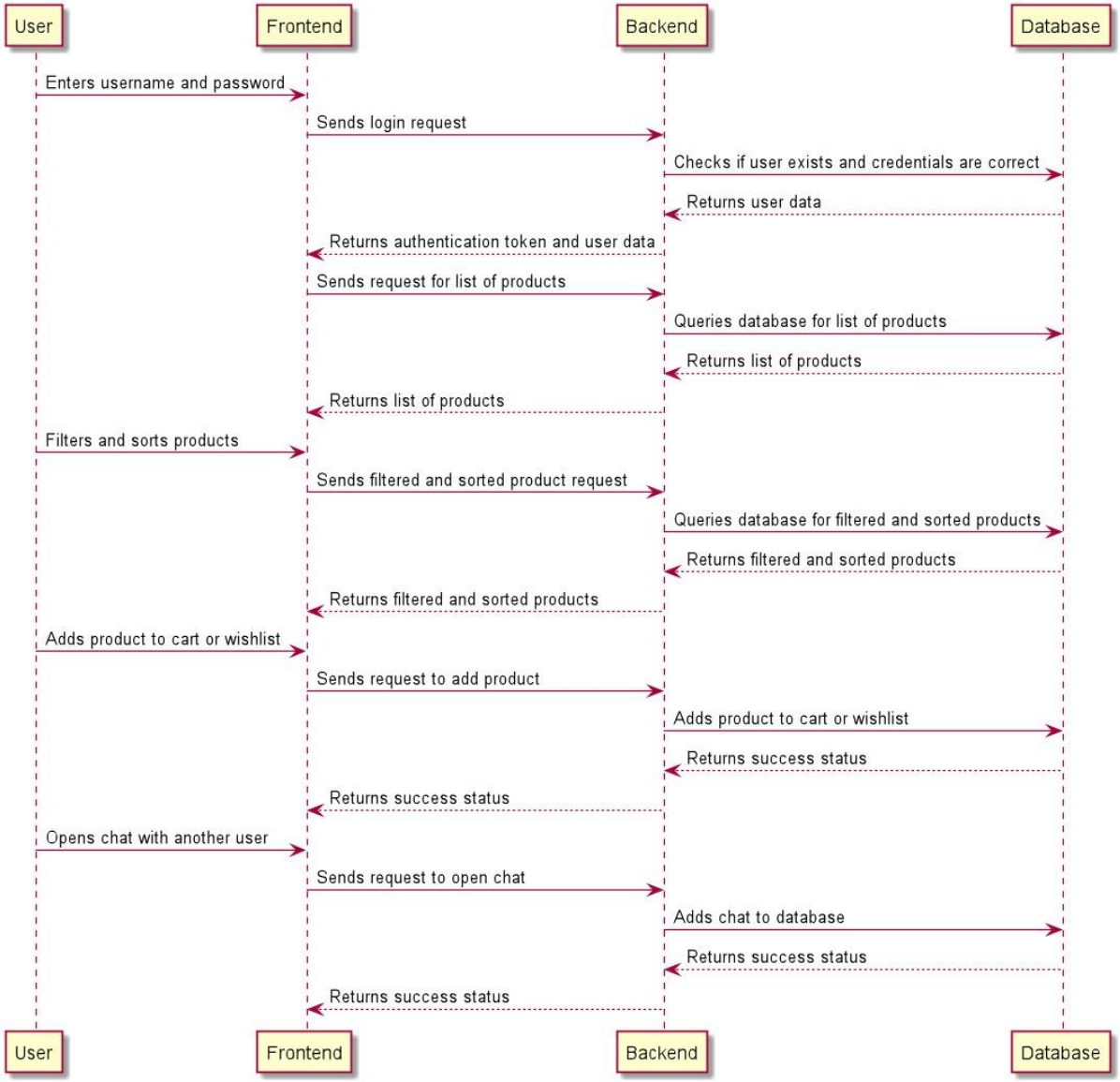


Deployment diagram:

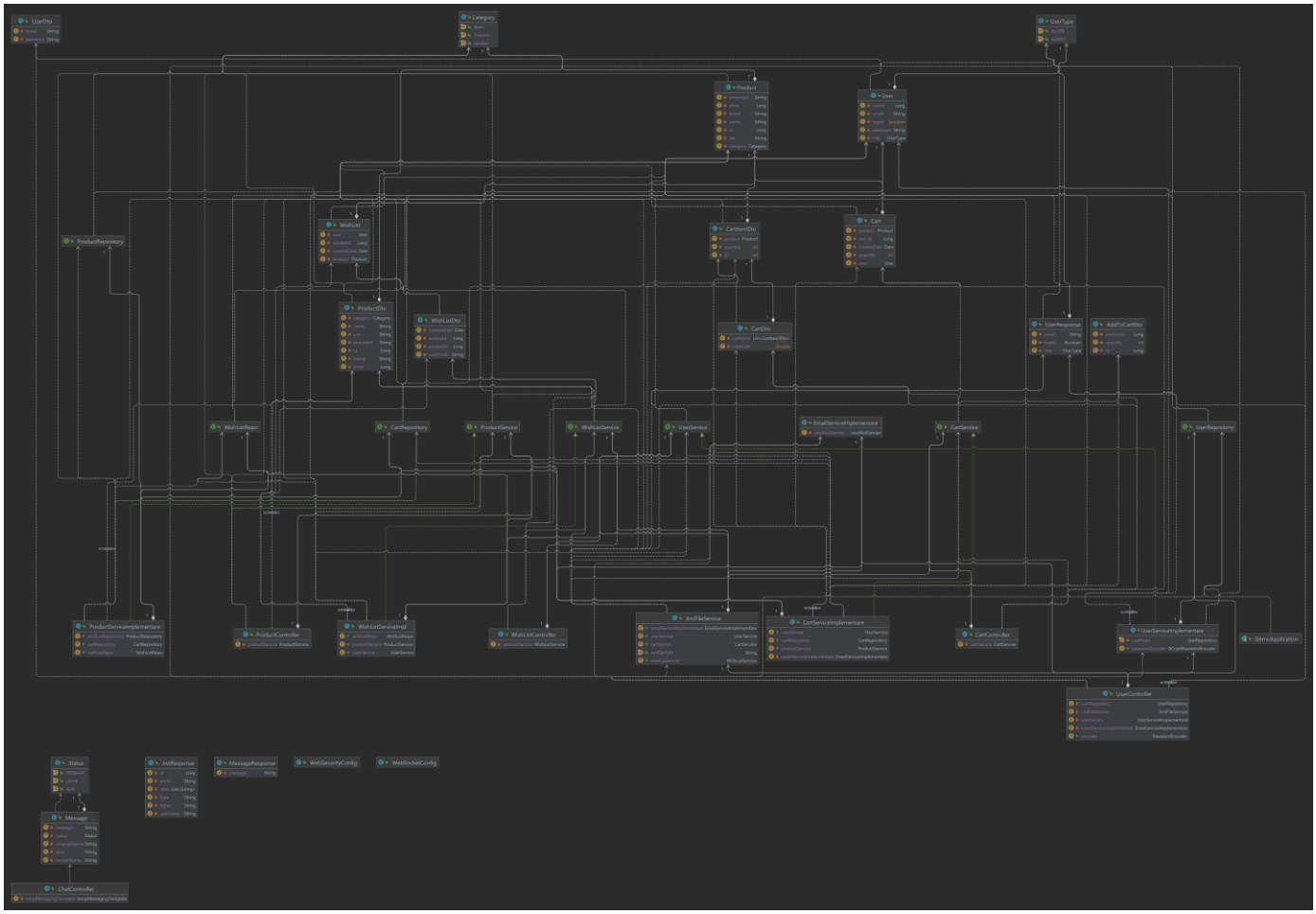


Design Model

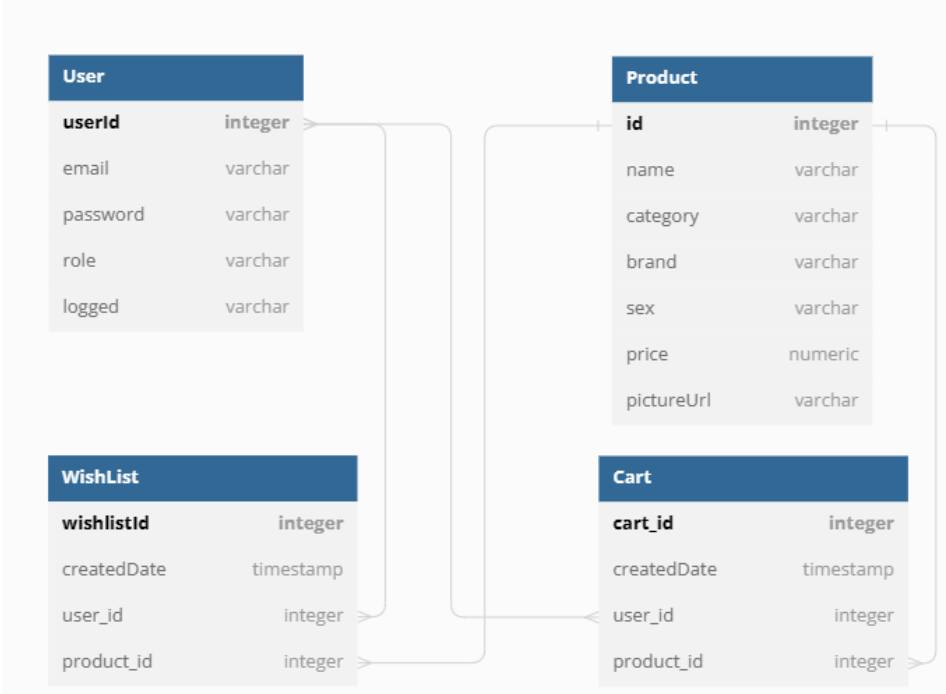
Dynamic Behavior



Class Diagram



Data Model



System Testing

In cadrul acestui proiect am testat cu ajutorul JUnit functionalitatite implementate in pachetul Service. Pentru a testa nivelul de Service, nu avem nevoie sa utilizam baza de date pentru testare. Ideal, ar f sa testam nivelul de Service fara sa facem legatura la baza de date. Pentru a acest lucru, putem utiliza suportul oferit de “Mockito” din cadrul Spring Boot. De asemenea, am utilizat metode din libraria AssertJ precum assertThat() pentru a afirma conditii.

Am implementat teste pentru metodele din toate clasele din pachetul de service, fiind foarte importanta buna functionare a acestora. De asemenea, am testata tat cazurile in care ne asteptam ca metoda sa functioneze conform planului, dar si cazul in care metoda arunca exceptii pentru date invalide.

Cateva exemple de metode din clasele de teste sunt: findByIdTest() , findByIdTestWhenIdDoesntExists(), whenGivenId_shouldDeleteProduct_ifFound(), should_throw_exception_when_product_doesnt_exist().

Future Improvements

Cateva dintre functionalitatiile care ar imbunatati acest proiect sunt:

- Implementarea unui sistem de evaluare a produselor pentru a conferi o experienta cat mai placuta pentru utilizatori
- Implementarea unui sistem de comentarii care sa permita utilizatorilor sa prezinta experienta cu aceasta aplicatie
- Implementarea unui sistem de notificare, pentru a tine utilizatorii informati asupra modificarilor ce pot aparea asupra produselor vizionate de acestia
- Implementare unui formular pentru completarea datelor de livrare si crearea unui tabel pentru pastrarea acestora in baza de date, precum si o noua pagina pentru vizionarea comenzilor facute si statusul acestora. Pentru aceasta noua functionalitate, pentru pagina administratorului ar putea fi imbunatatita prin adaugarea posibilatii de a viziona comenziile si a le putea schimba statusul
- Imbunatatirea perfomantei prin minimizarea interogarilor bazei de date
- Implementarea unor masuri de Securitate aditionale, cum ar fi: autentificarea cu 2 factori

Conclusion

In concluzie, acaeasta aplicatie confera un mod eficient si prietenos pentru utilizatori de a gestiona produsele si utilizatorii. Cu abilitatea de a filtra produsele, a le adauga la favorite sau in cosul de cumparaturi, precum si de a comunica cu alti utilizatori sau cu administratorul site-ului, aceasta aplicatie ofera o solutie completa pentru cumparaturi online. Cu toate acestea, exista intotdeauna loc de imbunatatiri si intentionez sa continui sa imbunatatesc caracteristicele aplicatiei si masurile de securitate pentru a conferi o experinta cat mai buna pentru utilizator.

Bibliography

- https://learnetto.com/blog/react-form-validation
- https://www.baeldung.com/spring-boot
- https://www.javatpoint.com/reactjs-tutorial
- https://www.baeldung.com/spring-boot-testing
- https://www.baeldung.com/websockets-spring
- https://www.baeldung.com/java-email