



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Shoes Store

Name: Danci Ionela
Group: 30236

Table of Contents

Project Specification	3
Functional Requirements	3
Use Case Model	4
Use Cases Identification	4
UML Use Case Diagrams	4
Supplementary Specification	4
Non-functional Requirements.....	4
Design Constraints	4
<i>Deliverable 2</i>	<i>5</i>
Domain Model.....	5
Architectural Design.....	6
Conceptual Architecture	6
Package Design.....	7
Component and Deployment Diagram	10
<i>Deliverable 3</i>	<i>11</i>
Design Model.....	11
Dynamic Behavior	11
Class Diagram	12
Data Model.....	12
<i>System Testing</i>	<i>12</i>
<i>Future Improvements</i>	<i>13</i>
<i>Conclusion.....</i>	<i>13</i>
<i>Bibliography.....</i>	<i>13</i>

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 mesaj 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

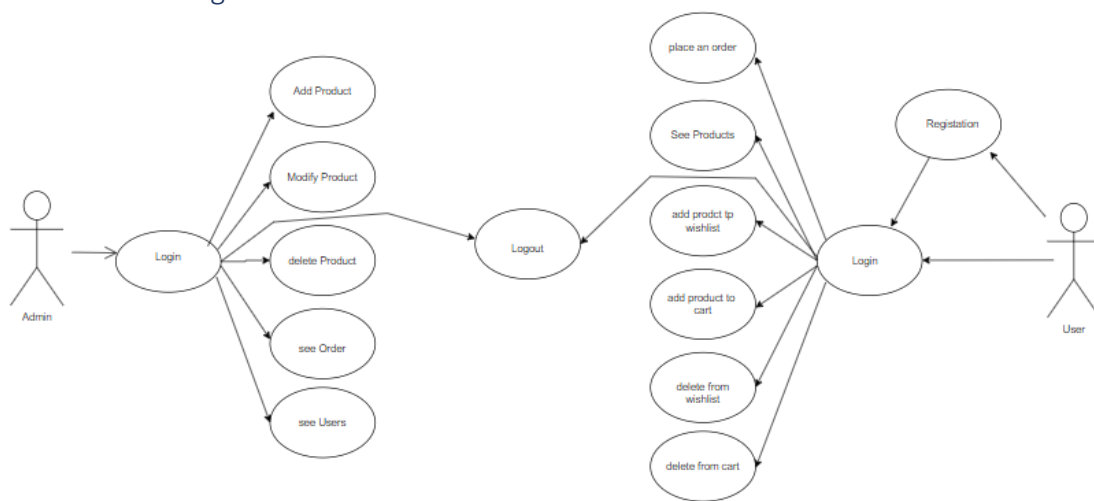
Primary actor: User

Main success scenario: Utilizatorul cauta pentru un tip specific de brand sau o categorie specifica si o sa se afiseze incaltaminte care respecta criteriile 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

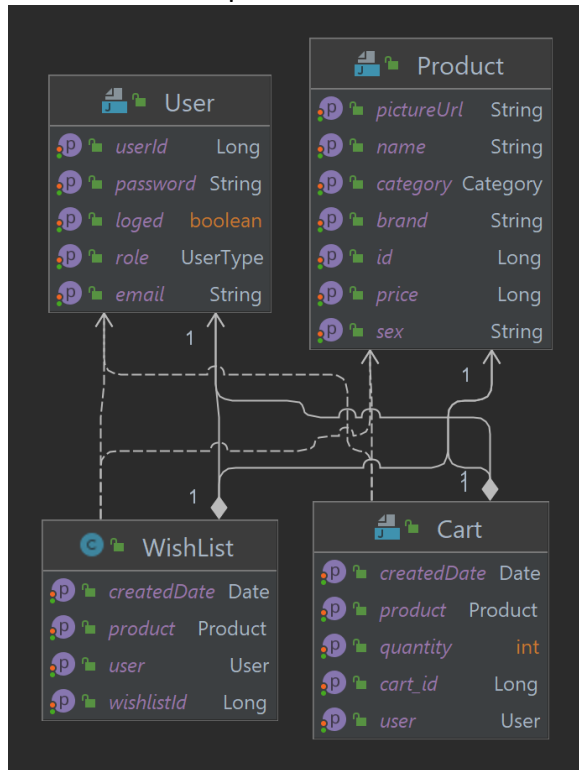
Visual Design: Aplicatia trebuie sa aiba un design atragator, in conformitate cu imaginea brandului

Deliverable 2

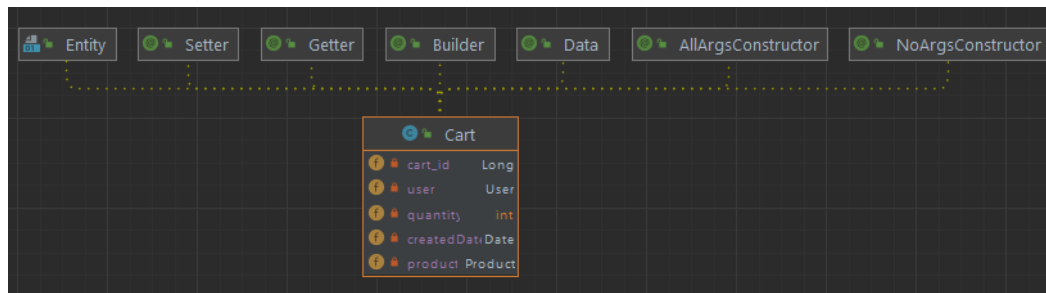
Domain Model

-este locul unde are loc modelarea etajului de business.

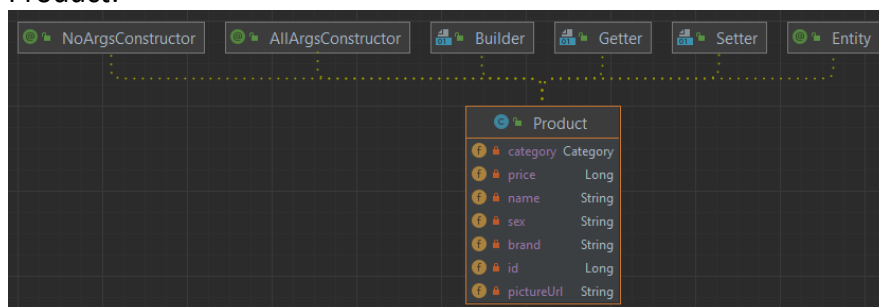
In cadrul acestui proiect sunt 4 entitati: User, Product, Cart, WishList



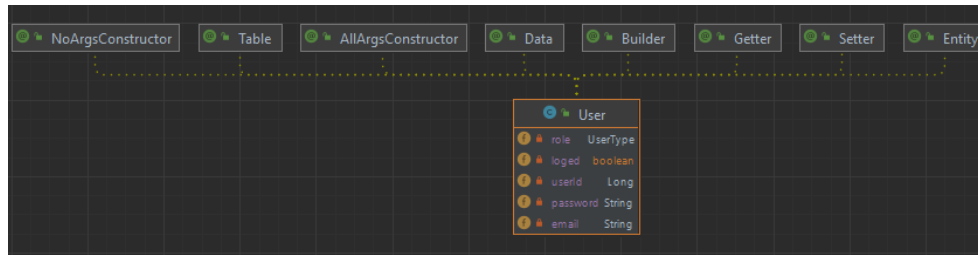
Cart:



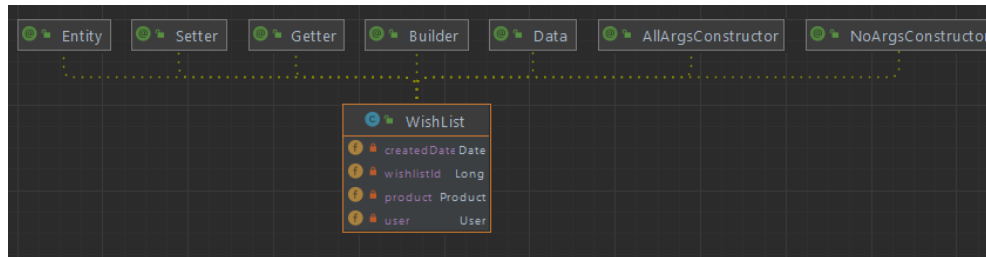
Product:



User:



WishList:



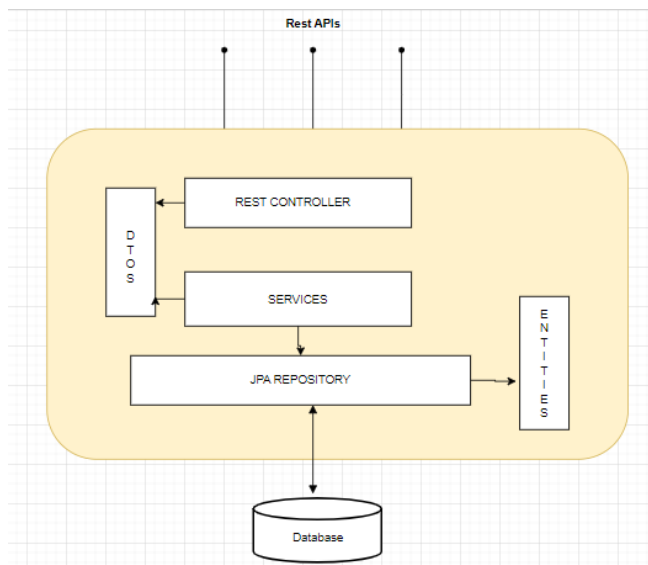
Architectural Design

Conceptual Architecture

In acest proiect am utilizat arhitectura Spring Boot.

Aplicatia contine cele 4 etaje:

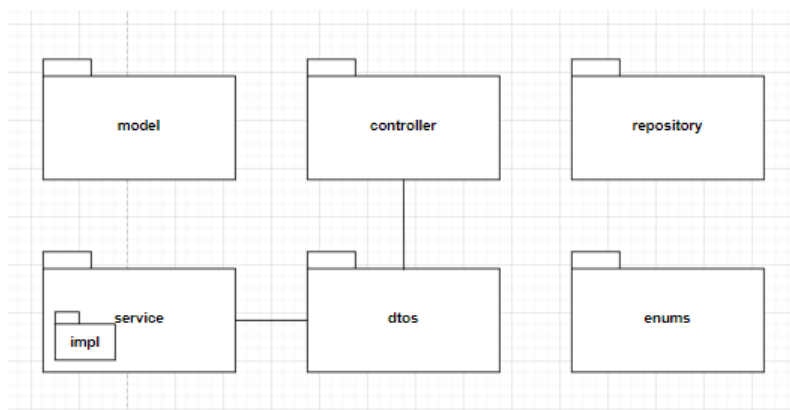
- Presentation Layer: reprezinta partea de front-end, aceasta este realizata utilizand react js.
- Business Layer: consta in toate clasele de service
- Persistence Layer: contine logica de stocare in baza de date
- Database Layer: contine baza de date (MySQL)



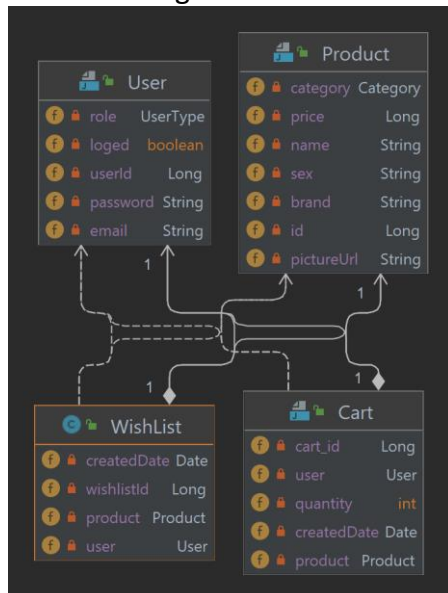
Patterns:

- 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 `WishListRepository`. 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

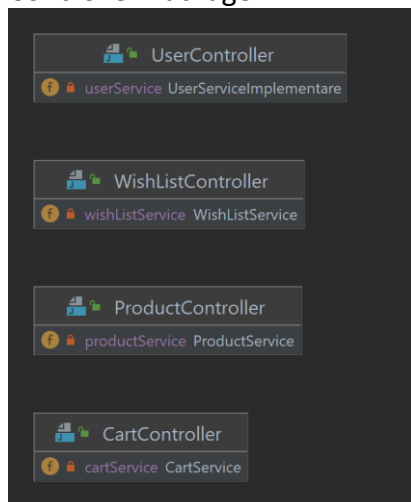
Package Design



Model Package:



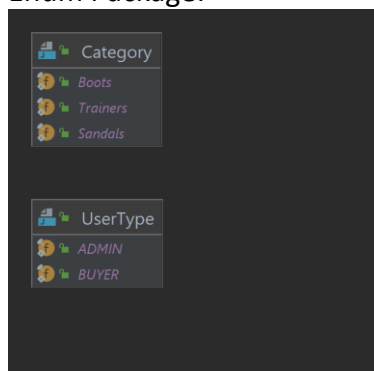
Controller Package:



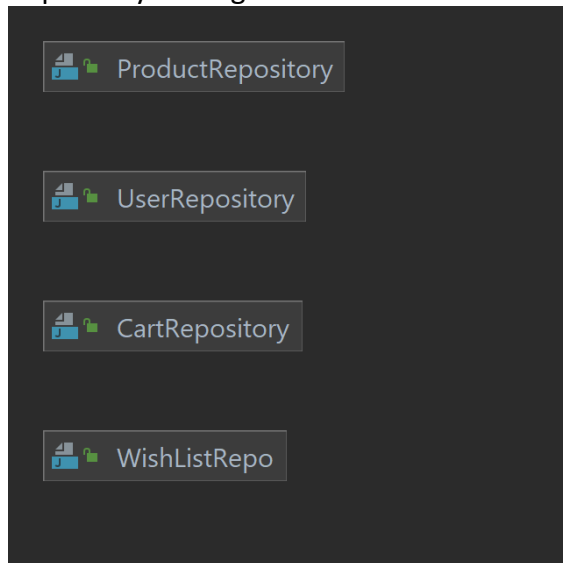
Dto Package:



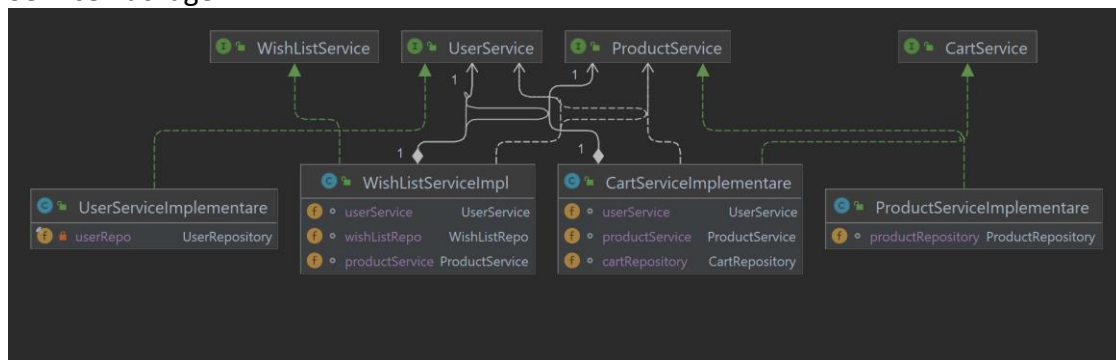
Enum Package:



Repository Package:

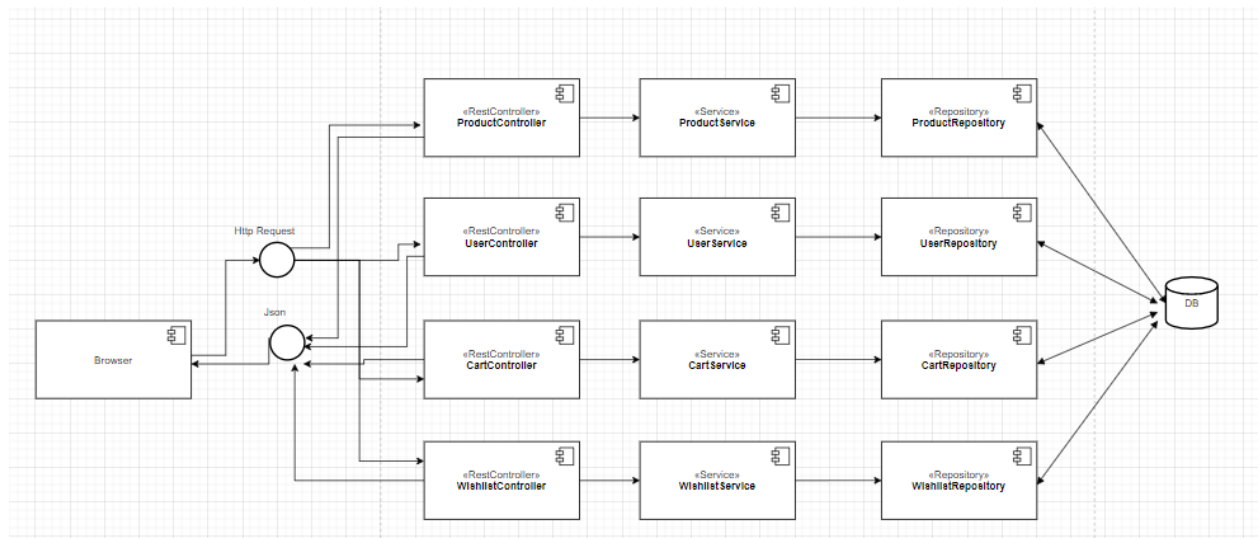


Service Package:

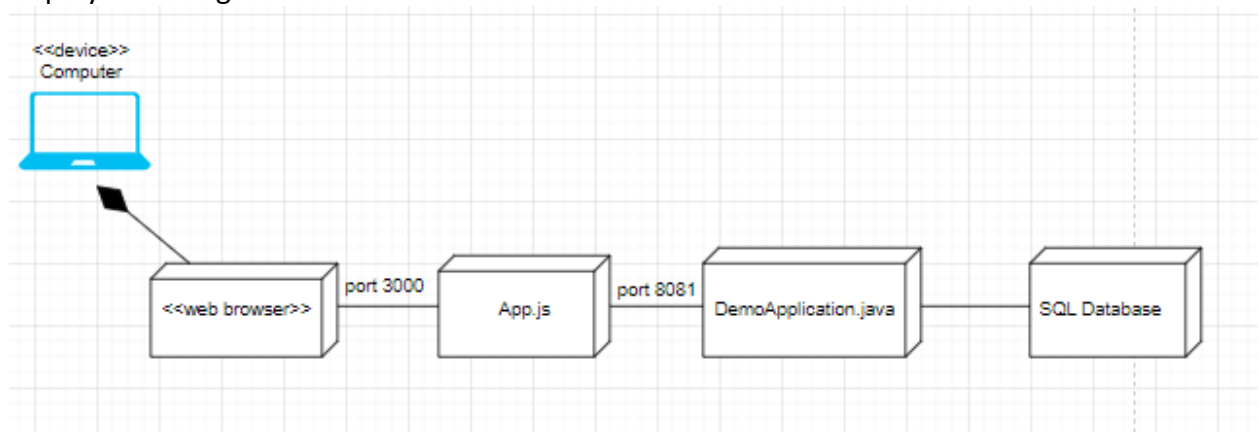


Component and Deployment Diagram

Component diagram



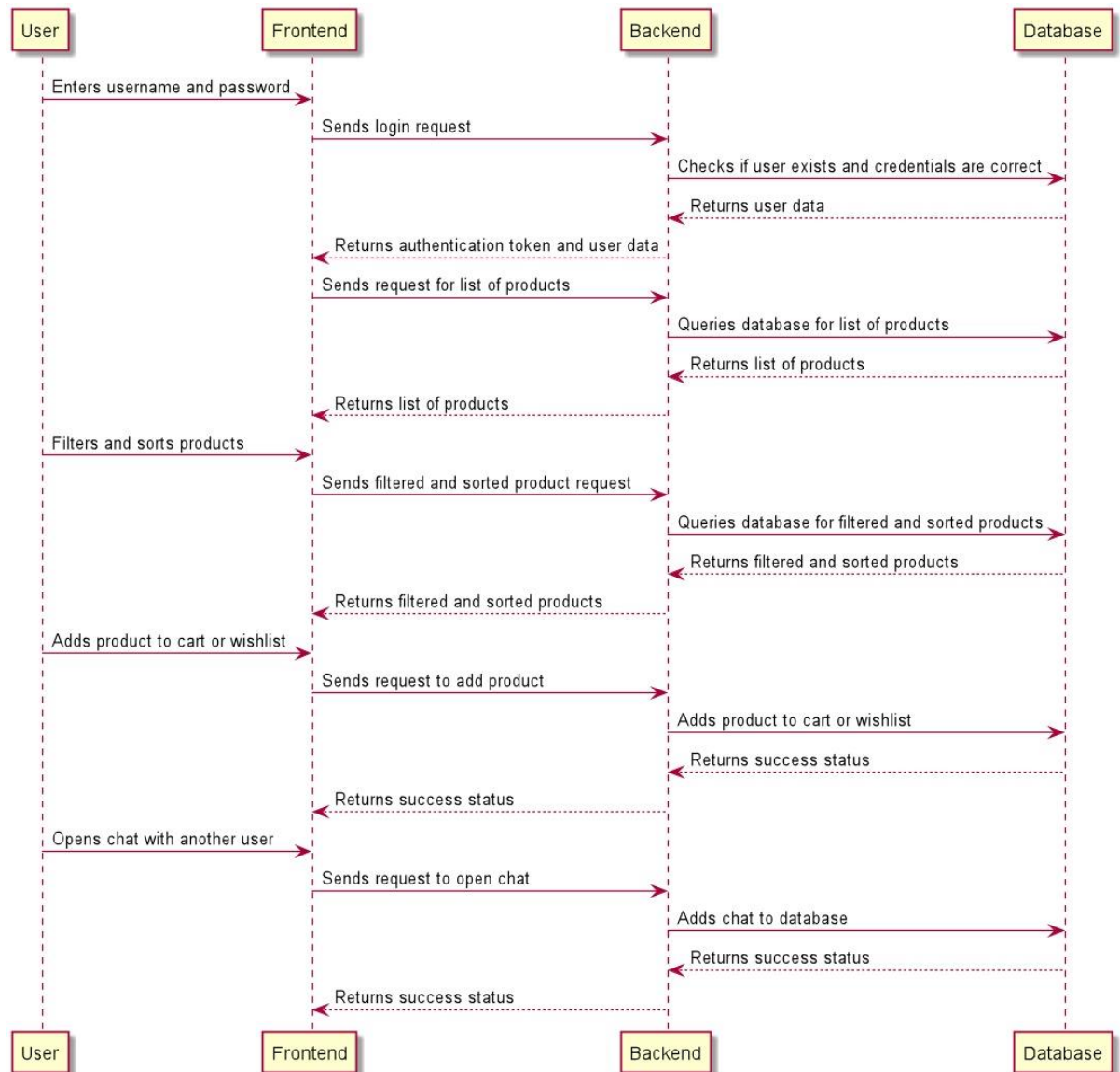
Deployment diagram:



Deliverable 3

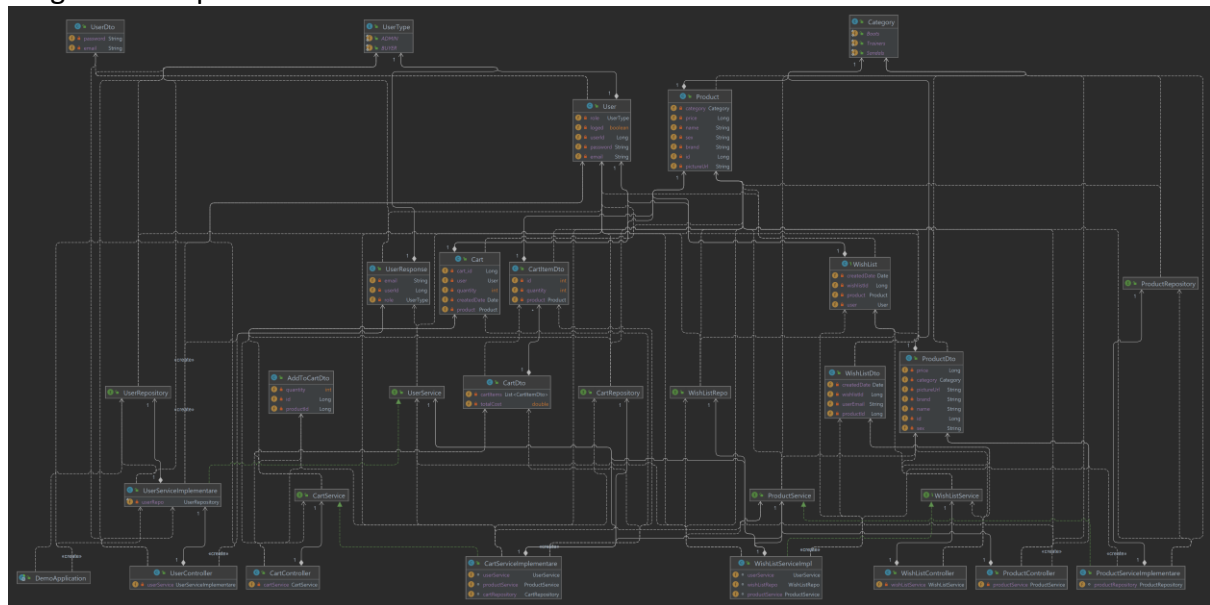
Design Model

Dynamic Behavior

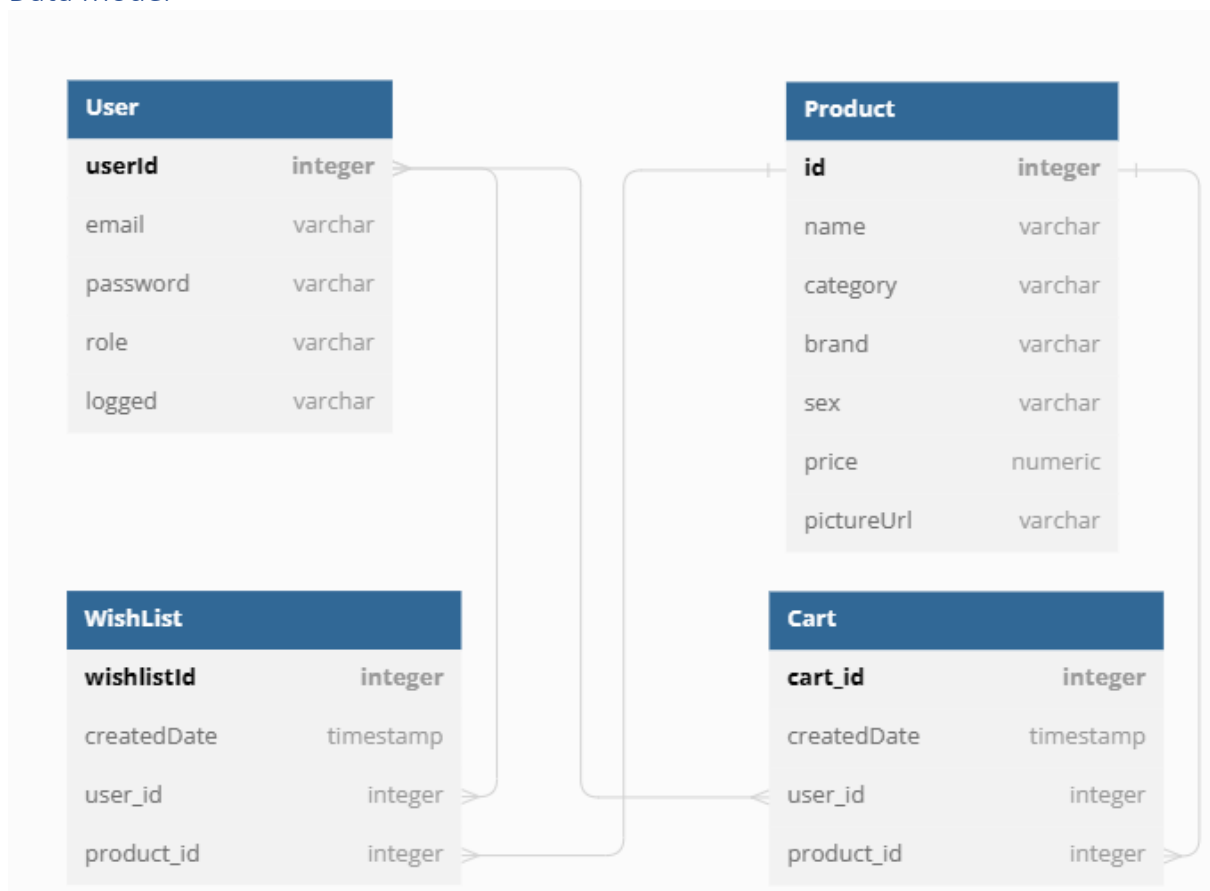


Class Diagram

Diagrama completa:



Data Model



System Testing

[Describe the testing methods and some test cases.]

Future Improvements

[Present some features that apply to the application scope.]

Conclusion

Bibliography