UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

# Flower Shop

Name: Ureche Simona Elena
Group: 4

# Table of Contents



# Deliverable 1

## Project Specification

Flower Shop este o aplicație web destinată gestionării unui magazine online de flori. Aceasta permite clienților să comande buchete de flori și să se aboneze magazinului, administratorilor să gestioneze produsele și utilizatorii, iar livratorilor să preia și să livreze comenzile.

## Functional Requirements

- **User Management**

| Funcționalitate | Descriere |
|---|---|
| Creare utilizator | Utilizatorii noi pot fi înregistrați cu nume, email, parolă și rol (Client, Administrator, Livrator). |
| Logare utilizator | Utilizatorii pot accesa contul lor cu email și parolă. |
| Vizualizare profil | Utilizatorii pot vedea și edita detaliile contului lor. |
| Roluri și permisiuni | Fiecare utilizator are acces doar la funcționalitățile specifice rolului său. |

- **Product Management**

| Funcționalitate | Descriere |
|---|---|
| Adăugare produs | Administratorul poate adăuga produse noi (nume, preț, descriere, stoc). |
| Editare produs | Administratorul poate modifica detaliile unui produs. |
| Ștergere produs | Administratorul poate șterge un produs din catalog. |
| Vizualizare catalog | Clienții pot vedea lista de produse disponibile. |

- **Order Management**

| Funcționalitate | Descriere |
|---|---|
| Plasare comandă | Clienții pot adăuga produse în coș și finaliza comanda. |
| Atribuire comandă | Administratorul poate atribui o comandă unui livrator. |
| Urmărire comandă | Clienții pot vedea statusul comenzii (Pending, In Delivery, Completed). |
| Actualizare comandă | Livratorul poate schimba statusul unei comenzi livrate. |

- **Cart Management**

| Funcționalitate | Descriere |
|---|---|
| Adăugare produs în coș | Clienții pot adăuga produse în coșul lor. |
| Modificare cantitate | Clienții pot modifica cantitatea unui produs din coș. |
| Eliminare produs | Clienții pot elimina produse din coș. |
| Golire coș | Clienții pot șterge toate produsele din coș. |
| Vizualizare coș | Clienții pot vedea produsele adăugate înainte de finalizarea comenzii. |

- **Product Categorization**

| Funcționalitate | Descriere |
|---|---|
| Creare categorie | Administratorii pot crea categorii noi (ex: Buchete, Vouchere, Lalele). |
| Editare categorie | Administratorii pot modifica numele unei categorii. |
| Ștergere categorie | Administratorii pot șterge categorii care nu mai sunt relevante. |
| Atribuire produs | Administratorii pot adăuga produse într-o anumită categorie. |
| Filtrare produse după categorie | Clienții pot naviga produsele pe baza categoriilor. |

- **Delivery Management**

| Funcționalitate | Descriere |
|---|---|
| Atribuire livrare | Administratorul poate atribui o comandă unui livrator. |
| Vizualizare livrări | Livratorii pot vedea comenzile atribuite lor. |

| | |
|---|---|
| Actualizare status livrare | Livratorii pot marca o comandă ca „În curs de livrare” sau „Livrată”. |
| Tracking livrare | Clienții pot vedea statusul livrării comenzilor lor. |

- **Payment Management**

| Funcționalitate | Descriere |
|---|---|
| Selectare metodă de plată | Clienții pot alege între plata online sau ramburs. |
| Procesare plată online | Sistemul gestionează plățile prin card sau alte metode digitale. |
| Confirmare plată | Clienții primesc o confirmare după ce plata a fost procesată. |

- **Reviews & Ratings**

| Funcționalitate | Descriere |
|---|---|
| Adăugare recenzie | Clienții pot scrie recenzii pentru produsele cumpărate. |
| Editare recenzie | Clienții își pot modifica recenziile existente. |
| Ștergere recenzie | Clienții își pot șterge recenziile. |
| Vizualizare recenzii | Toți utilizatorii pot vedea recenziile altor clienți. |

- **Vouchers & Discounts**

| Funcționalitate | Descriere |
|---|---|
| Creare voucher | Administratorii pot crea vouchere cu reduceri. |
| Aplicare voucher | Clienții pot introduce un cod de reducere în coș. |
| Verificare valabilitate | Sistemul verifică dacă voucherul este valid (expirat sau deja folosit). |

| | |
|---|---|
| Calculare reducere | Se aplică discount-ul asupra prețului total. |

## Use Case Model

### Use Cases Identification:

**Use-Case: Login**

- **Level**: Acțiune directă a utilizatorului
- **Primary Actor**: Client / Administrator / Livrator
- **Main success scenario**:
    1. Utilizatorul accesează pagina de login.
    2. Introduce email-ul și parola.
    3. Apasă pe butonul „Login”.
    4. Sistemul verifică datele și autentifică utilizatorul.
    5. Utilizatorul este redirecționat către dashboard-ul său.
- **Extensions**:
    - (4a) Dacă email-ul sau parola sunt greșite → Se afișează un mesaj de eroare.
    - (4b) Dacă utilizatorul și-a uitat parola → Poate solicita resetarea acesteia.

**Use-Case: Creare cont nou**

- **Level**: Acțiune directă a utilizatorului
- **Primary Actor:** Client
- **Main success scenario**:
    1. Utilizatorul accesează pagina de înregistrare.
    2. Introduce numele, email-ul, parola și alte detalii.
    3. Apasă pe „Creează cont”.
    4. Sistemul validează datele și salvează utilizatorul.
    5. Utilizatorul primește un email de confirmare și contul este activat.
- **Extensions:**
    - (4a) Dacă email-ul este deja folosit → Se afișează un mesaj de eroare.
    - (4b) Dacă parola nu respectă regulile → Utilizatorul primește o notificare să o schimbe.

**Use-Case: Adăugare produs nou**

- **Level**: Acțiune directă a administratorului
- **Primary Actor**: Administrator
- **Main success scenario**:
    1. Administratorul accesează secțiunea „Produse”.
    2. Apasă pe butonul „Adaugă produs nou”.
    3. Introduce numele, prețul, descrierea, categoria și stocul produsului.
    4. Apasă pe „Salvează produs”.
    5. Sistemul validează și adaugă produsul în catalog.
- **Extensions:**
    - (4a) Dacă un câmp obligatoriu lipsește → Se afișează un mesaj de eroare.

- (4b) Dacă prețul introdus nu este valid → Se solicită corectarea acestuia.

**Use-Case: Adăugare produs în coș**

- **Level**: Acțiune directă a utilizatorului
- **Primary Actor**: Client
- **Main success scenario**:
  1. Clientul navighează prin catalogul de produse.
  2. Selectează un produs și apasă „Adaugă în coș”.
  3. Sistemul verifică stocul produsului.
  4. Produsul este adăugat în coș cu cantitatea specificată.
  5. Clientul poate continua cumpărăturile sau finaliza comanda.
- **Extensions:**
  - (3a) Dacă produsul nu mai este în stoc → Se afișează un mesaj de eroare.
  - (3b) Dacă clientul nu este logat → Este redirecționat către pagina de autentificare.

**Use-Case: Plasare comandă**

- **Level**: Acțiune directă a utilizatorului
- **Primary Actor**: Client
- **Main success scenario**:
  1. Clientul adaugă produse în coș.
  2. Accesează pagina de finalizare comandă.
  3. Selectează metoda de plată și adresa de livrare.
  4. Apasă pe „Plasează comanda”.
  5. Sistemul verifică detaliile și confirmă comanda.
- **Extensions:**
  - (4a) Dacă stocul unui produs s-a epuizat → Clientul primește o notificare să îl înlocuiască.
  - (4b) Dacă plata online eșuează → Clientul este anunțat și poate încerca din nou.

**Use-Case: Actualizare status livrare**

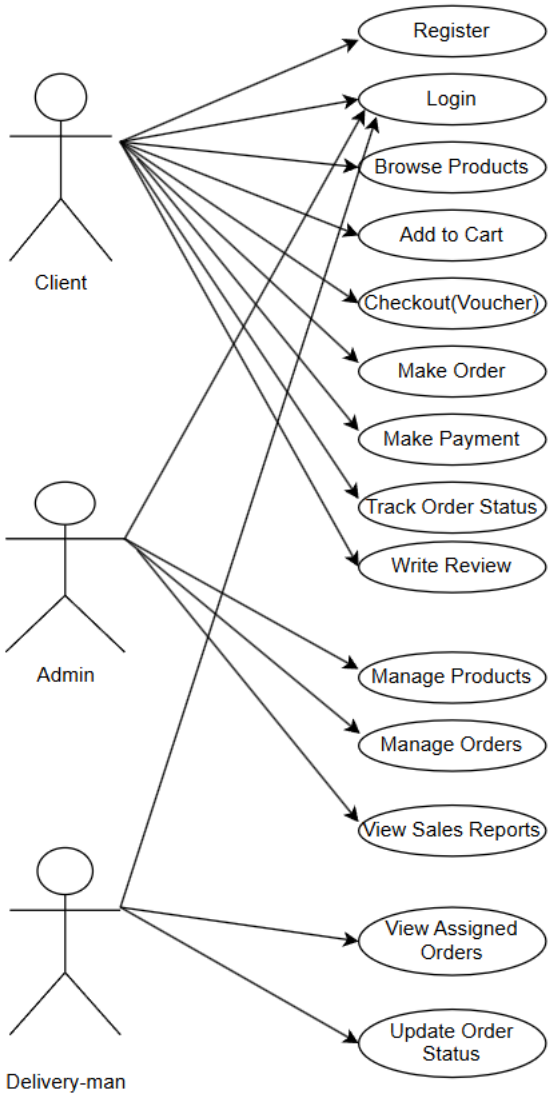
- **Level**: Acțiune directă a livratorului
- **Primary Actor**: Livrator
- **Main success scenario**:
  1. Livratorul accesează lista comenzilor atribuite.
  2. Selectează o comandă și apasă „În curs de livrare”.
  3. După livrare, apasă „Finalizat”.
  4. Statusul comenzii este actualizat în sistem.
  5. Clientul primește o notificare că livrarea a fost finalizată.
- **Extensions:**
  - (3a) Dacă clientul nu este acasă → Livratorul poate marca livrarea ca „Nereușită” și reprograma.

**Use-Case: Procesare plată online**

- **Level:** Acțiune directă a clientului
- **Primary Actor**: Client

- **Main success scenario**:
    1. Clientul finalizează comanda și selectează „Plată cu cardul”.
    2. Introduce datele cardului și confirmă plata.
    3. Sistemul verifică și procesează tranzacția.
    4. Plata este aprobată, iar clientul primește confirmarea.
- **Extensions:**
    - (3a) Dacă fondurile sunt insuficiente → Plata este refuzată și clientul este anunțat.

## UML Use Case Diagrams




## Supplementary Specification

### Non-functional Requirements

| Caracteristică | Descriere | Motivare |
|---|---|---|
| Scalabilitate | Sistemul trebuie să fie capabil să gestioneze un număr mare de utilizatori și comenzi simultan, fără degradarea performanței. | Creșterea numarului mare de utilizatori și comenzi necesită un backend optimizat. |
| Performanța | Aplicația trebuie să răspundă rapid la cereri și să fie optimizată pentru o încărcare mare a datelor (produse, comenzi, utilizatori). | Îmbunătățirea experienței utilizatorului. |
| Securitate | Aplicația trebuie să protejeze datele utilizatorilor, în special informațiile personale (email, adresă, metodă de plată). | Folosim gestionarea rolurilor – fiecare utilizator are acces doar la funcțiile permise de rolul său. |
| Modularitate | Funcționalitățile trebuie să fie dezvoltate independent și întreținute independent. | Codul trebuie să fie modular, bine organizat și ușor de extins. |
| Ușurința în utilizare | Aplicația trebuie să fie intuitivă și ușor de utilizat de către toți utilizatorii. | Interfața trebuie să fie clară, să aibă un design modern și să fie accesibilă de pe orice dispozitiv. |

## Design Constraints

| Tip de constrângere | Descriere | Motivare |
|---|---|---|
| Limbaj de programare | **Backend:** Java 17 + Spring Boot 3.4.3<br>**Frontend:** HTML, Thymeleaf, Bootstrap (React în viitor) | Java oferă performanță ridicată și compatibilitate cu Spring Boot. Thymeleaf permite integrarea ușoară cu backend-ul. |
| Arhitectură Software | **MVC** (Model-View-Controller) | Permite separarea clară a logicii de business, interfației și accesului la date. |
| Bază de date | Inițial H2 salvate în memorie, extensibil la MySQL-Workbench. | H2 este util pentru testare rapidă, MySQL-Workbench oferă scalabilitate în producție. |
| Framework-uri utilizate | Spring Boot, Spring Data JPA, Thymeleaf, Lombok, Bootstrap, MUI | Spring Boot optimizează dezvoltarea backend-ului, Thymeleaf și Bootstrap facilitează interfața utilizatorului. |

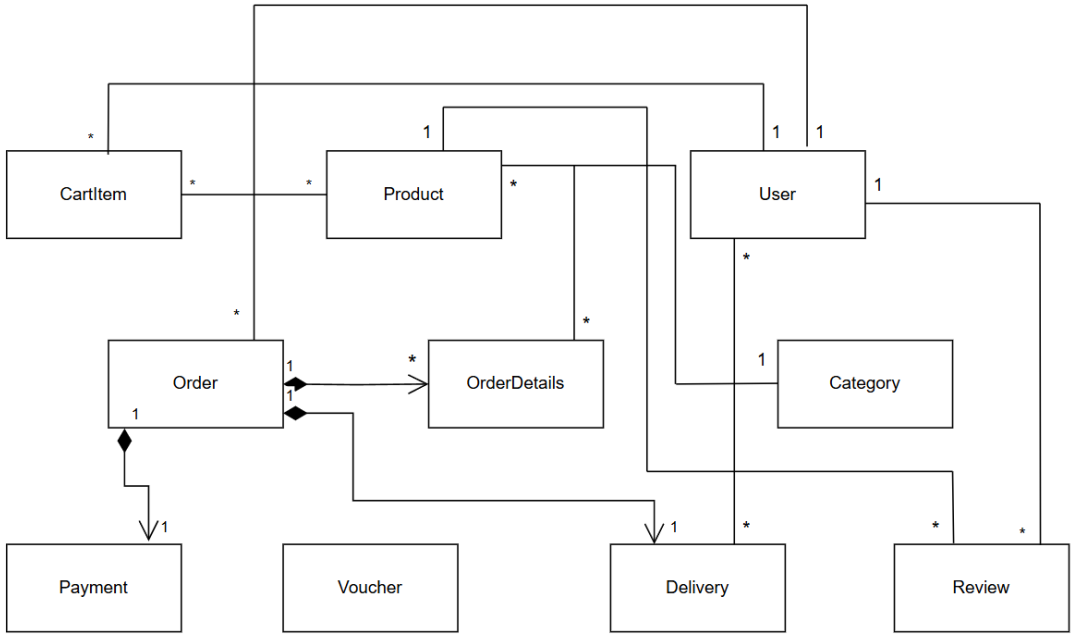
| | | |
|---|---|---|
| Gestionarea dependințelor | **Maven** | Maven permite gestionarea eficientă a pachetelor și a versiunilor librăriilor utilizate |
| Metode de testare | **Unit Tests** cu JUnit & Mockito | Testele unitare vor asigura calitatea codului și prevenirea regresiilor. |
| Interfață utilizator | **Web-first** (desktop & mobile via Bootstrap, MUI) | Design responsiv pentru accesibilitate pe toate dispozitivele. |
| Persistență date | **Repository Pattern cu** Spring Data JPA | Permite gestionarea eficientă a datelor și separarea logicii de acces la baze de date. |

## Glossary

- **User -** O entitate care interacționează cu aplicația. Poate fi de tip CLIENT, ADMIN sau LIVRATOR.
- **Product -** Obiectul de bază vândut în aplicație.
- **Order -** O cerere plasată de un client pentru a achiziționa produse.
- **OrderDetails -** Relația între o comandă și produsele cumpărate, cu cantitate și preț individual.
- **CartItem -** Un produs adăugat temporar în coșul de cumpărături.
- **Voucher -** Un cod care oferă discount pentru o comandă.
- **Payment** - Metoda prin care clientul finalizează o comandă.
- **Delivery -** Procesul prin care o comandă este expediată către client.

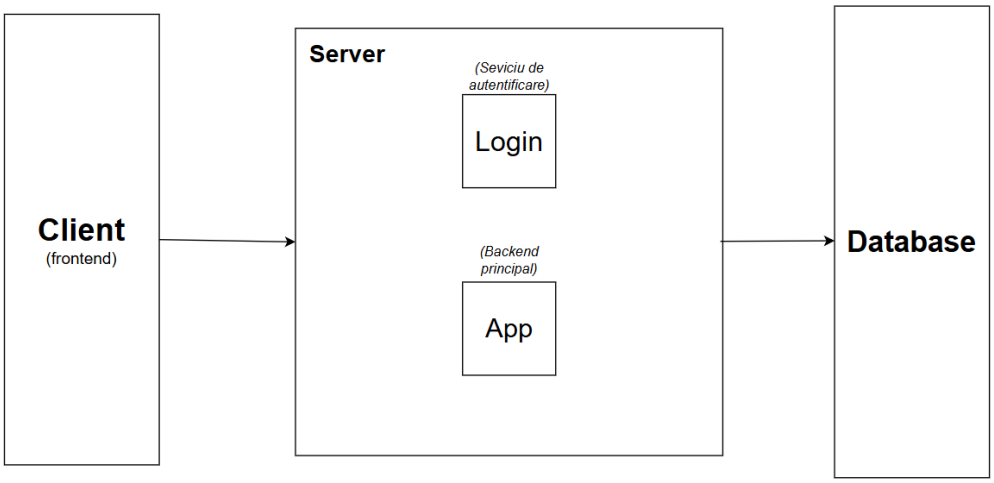
# Deliverable 2

## Domain Model

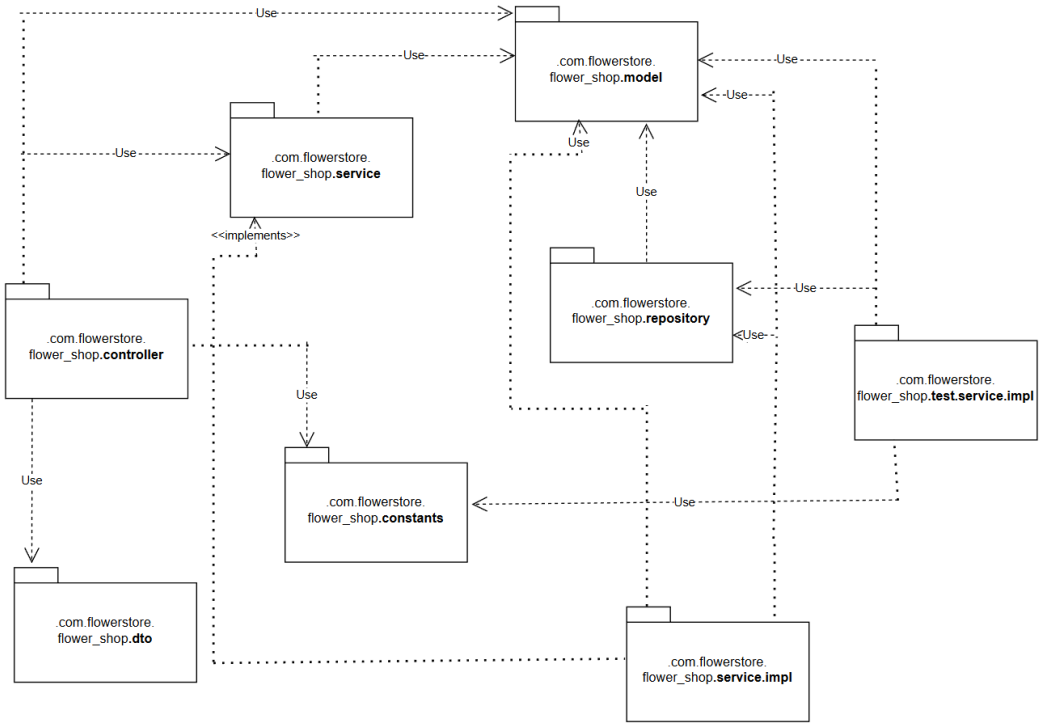



## Architectural Design

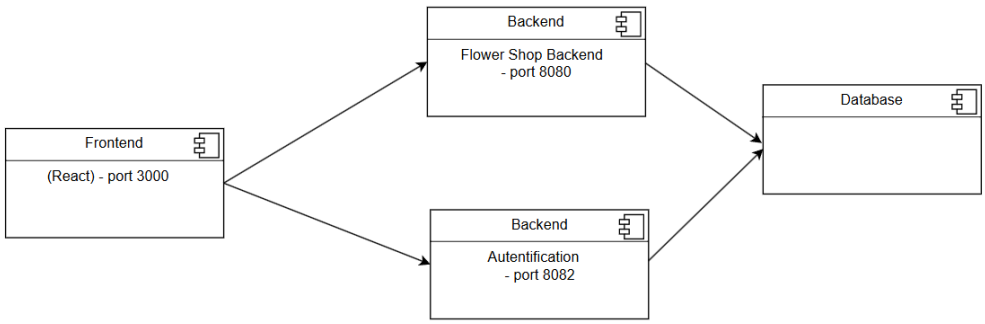
### Conceptual Architecture

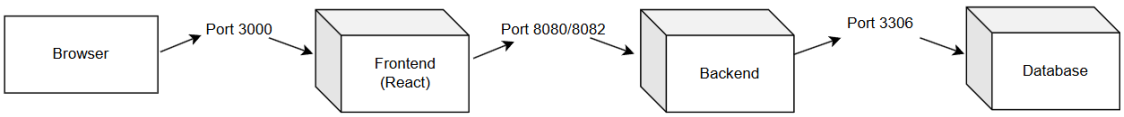


## Package Design




## Component and Deployment Diagram

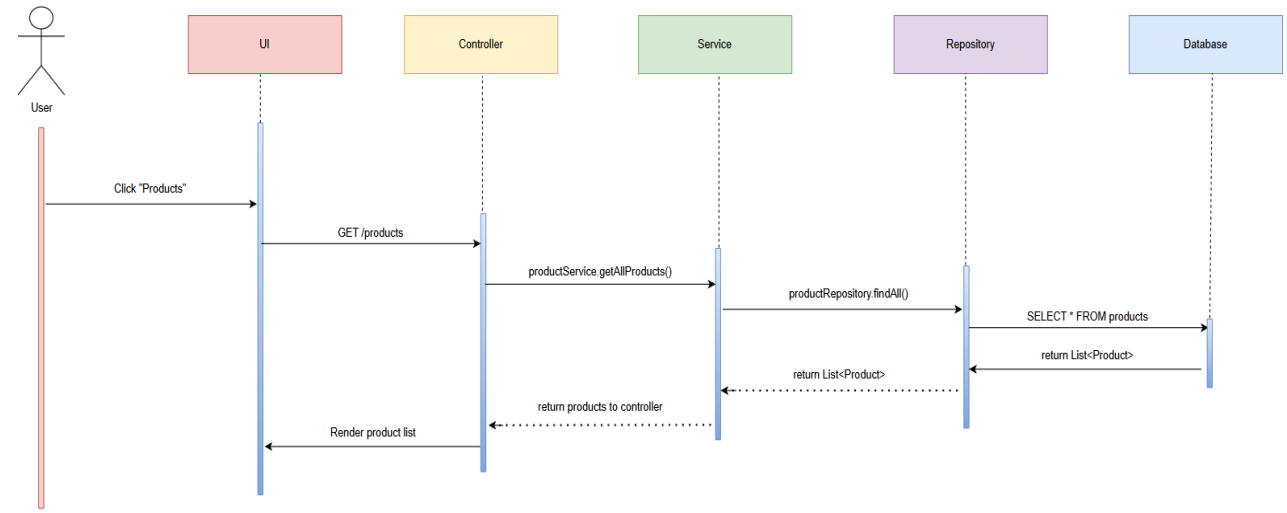


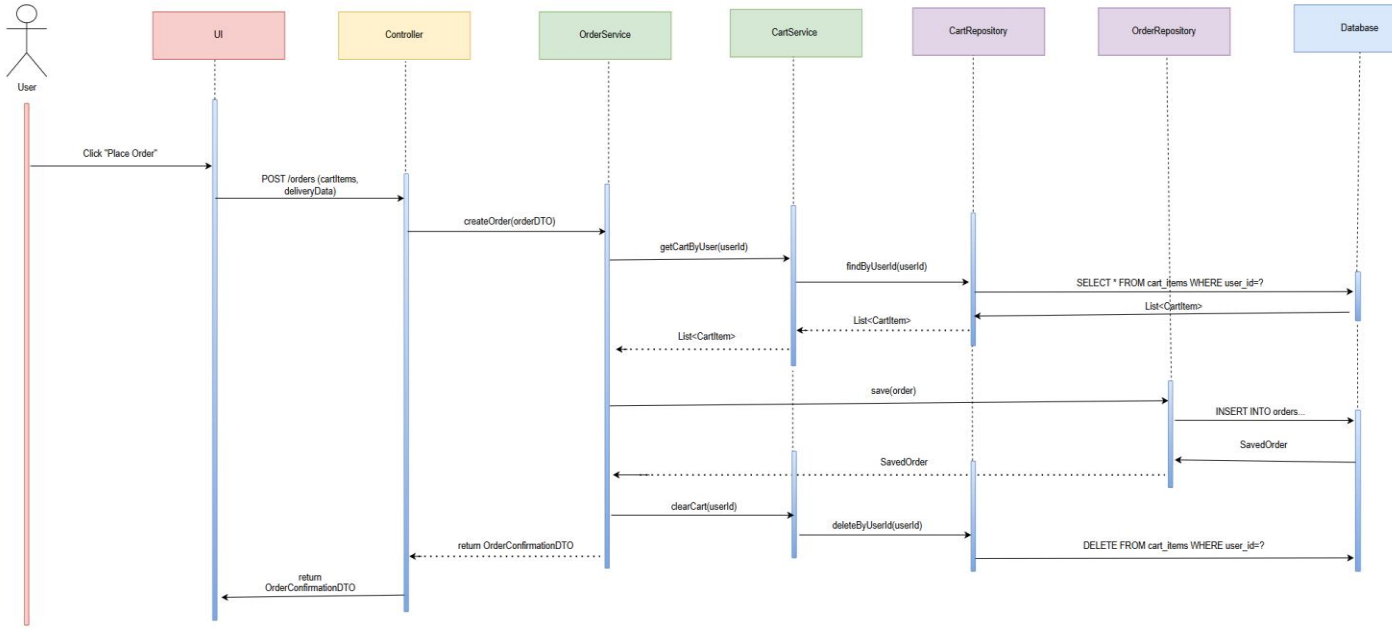



# Deliverable 3

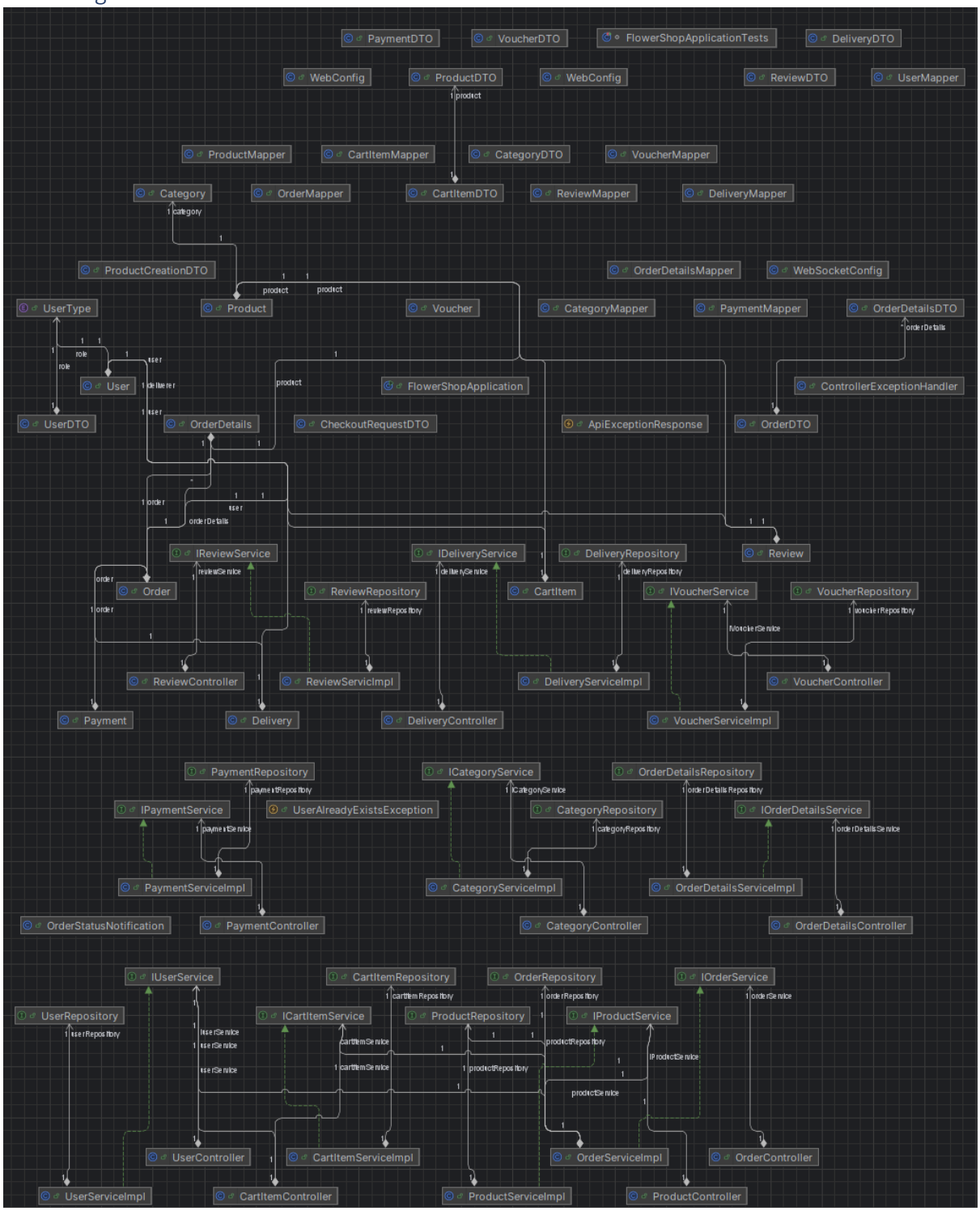
## Design Model

### Dynamic Behavior

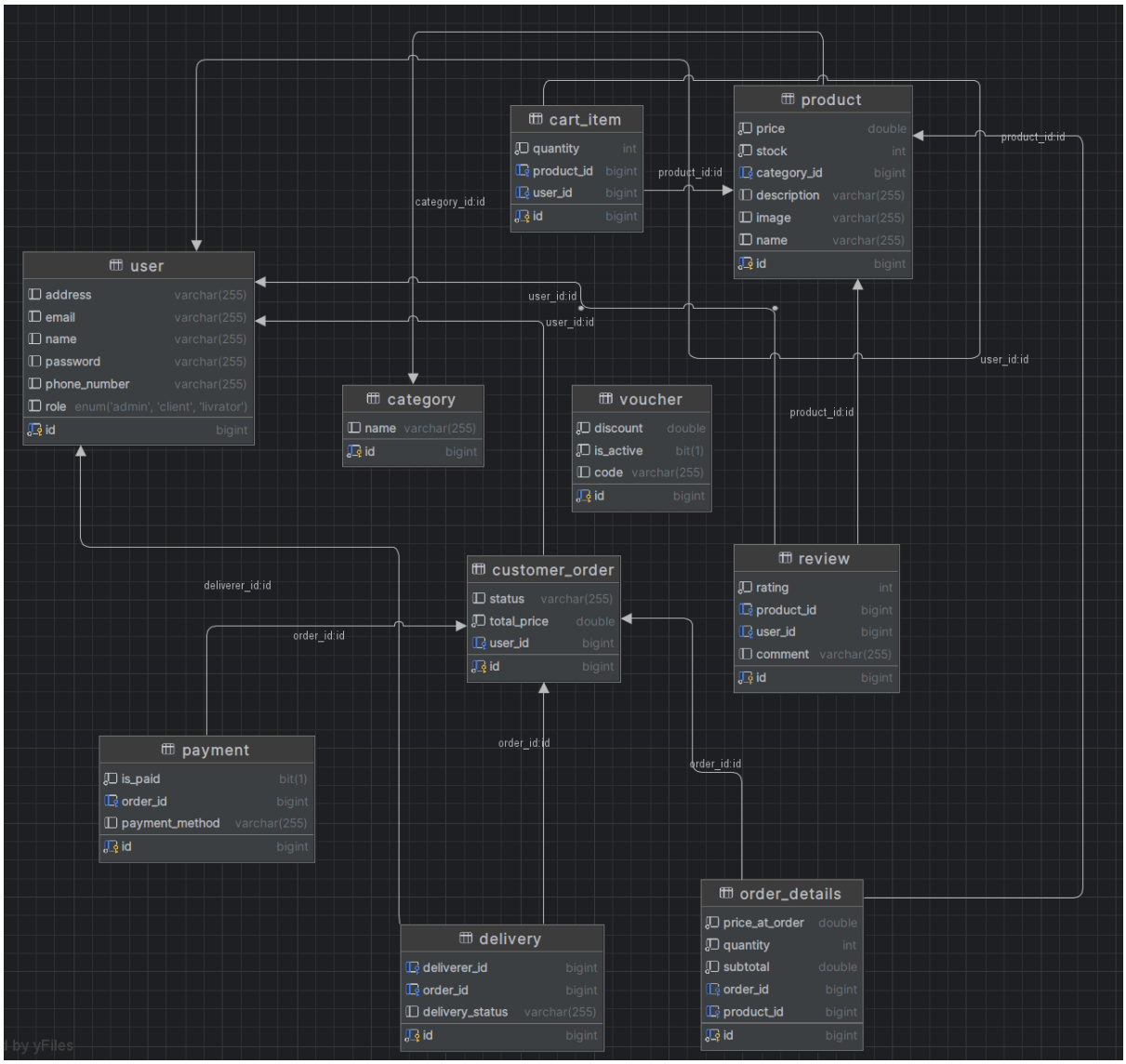


## Class Diagram



## Data Model




## System Testing

- ***Testare unitară (Unit Testing)***

*Am realizat testarea logicii de business din layer-ul de **service**, folosind **JUnit 5** și **Mockito**, mock-uind accesul la baza de date. Testele unitare au fost scrise pentru următoarele componente:*

- ***UserService**: crearea, actualizarea, căutarea și listarea utilizatorilor*
- ***ProductService**: adăugarea, modificarea, ștergerea și listarea produselor*
- ***OrderService**: plasarea, actualizarea, căutarea și ștergerea comenzilor*

*Scopul testelor a fost să verificăm comportamentul corect al metodelor din serviciile noastre, tratările de excepții și apelurile către repository.*

- ***Testare manuală (Functional Testing)***

*Pe lângă testarea automată, aplicația a fost verificată manual în interfața grafică (UI), simulând utilizarea reală. Am testat următoarele:*

- *Înregistrarea și autentificarea utilizatorilor*
- *Adăugarea de produse în coș și plasarea comenzilor*
- *Vizualizarea listelor de produse/comenzi*
- *Funcționalități de administrare (adăugare/ștergere produs, modificare stoc)*

## Future Improvements

1. ***Adăugarea unui sistem de abonament lunar (Subscription)***

- *Clienții pot primi automat un buchet de flori lunar.*
- *Posibilitatea de a alege: tip flori, buget, frecvență (săptămânal, lunar).*
- *Integrare cu plata recurentă*

2. ***Program de fidelizare***

- *Puncte de loialitate la fiecare comandă.*
- *Reduceri sau cadouri pentru clienții frecvenți.*

3. ***Recenzii cu imagini***

- *Clienții pot încărca poze cu florile primite.*
- *Galerie vizuală pentru fiecare produs.*

4. ***Funcționalitate de Chat Live / Asistent virtual***

- *Asistență pentru clienți în timp real.*
- *Chatbot pentru întrebări frecvente sau status comenzi.*

5. ***Autentificare prin Google / Facebook***

- *Login mai rapid și familiar pentru utilizatori.*

## Conclusion

În concluzie, această aplicație de tip **Flower Shop** oferă o soluție completă pentru gestionarea unui magazin online de flori, adresând nevoile atât ale clienților, cât și ale administratorilor. Proiectul a inclus:

- **Integrarea unei baze de date SQL** pentru gestionarea persistentă a datelor.
- **Arhitectură bine structurată** folosind straturi de Controller – Service – Repository.
- Funcționalități de **autentificare, adăugare în coș, checkout, administrare produse și utilizatori**.
- Trimiterea automată de **emailuri de confirmare** după finalizarea comenzii.
- Relații clare între entități (ex: User, Order, Product, CartItem) implementate cu JPA.

Proiectul poate fi extins în viitor prin funcționalități avansate precum abonamente lunare, notificări și sistem de fidelizare, oferind o experiență modernă utilizatorilor.

# Bibliography

**Spring Boot Documentation** – https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

**Lombok Library** – https://projectlombok.org

**OpenAPI / Swagger Documentation** – https://swagger.io/docs/

**Material UI (MUI) React** – https://mui.com