UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

# Anticariat online

*Proiectare Software*

Autor: Marginean Teodor
Grupa: 30236

Facultatea de Automatica
si Calculatoare

28 Martie 2023

# Cuprins

# 1 Project Specification

Proiectul meu are ca scop realizarea unui anticariat online. Publicul tinta sunt cititorii care doresc sa aiba in colectia lor anumite carti din editii mai vechi. Obiectivul principal al acestui proiect este oferirea unui spatiu ideal de comercializare a cartilor pentru orice utilizator, indiferent de nivelul de competente tehnologice detinute de catre acesta. Planul nostru este dezvoltarea unei baze de date care sa cuprinda toate informatiile necesare despre fiecare carte si generarea unei facturi in urma comenzii. Plata se face doar la ramburs. Timpul in care vreau sa ma incadrez este de trei luni de la data inceperii proiectului.

# 2 Functional Requirements

Cerintele functionale ale proiectului sunt:

- adaugarea in baza de date folosindu-se operatii de CRUD;
- stergerea din baza de date folosindu-se operatii de CRUD;
- schimbarea valorilor in baza de date folosinduse operatii de CRUD;
- generarea unei facturi;
- crearea unui nou rand in baza de date la functia de register;
- posibilitatea stergerii unui user doar de catre un admin;
- cautarea in baza de date in functie de numele cartii;
- cautarea in baza de date in functie de numele autorului;

# 3 Use Case Model

Use case-urile care exista in acest proiect sunt:

- Inregitrearea userului;
- Logarea in cont;
- Schimbarea parolei;
- Crearea de anunt;
- Stergerea anuntului;
- Modificarea anuntului;
- Vizionare carti;
- Cautare carte dupa nume;
- Cautare carte dupa autor;
- Stergerea unui utilizator(admin);
- Achizitionarea unuei carti.

## 3.1 Use Case Identification

### 3.1.1 Inregistrearea userului

Level: user goal level

Primary Actor: utiliztorul

Main success scenario: se creaza userul in baza de date si se poate loga

### 3.1.2 Logarea in cont

Level: user goal level

Primary Actor: utiliztorul
Main success scenario: se logheaza in cont si se pot face actiuni in continuare

#### 3.1.3 Schimbarea parolei

Level: user goal level
Primary Actor: utiliztorul
Main success scenario: se modifa parola in baza de date si de deconecteaza usrul

#### 3.1.4 Crearea de anunt

Level: user goal level
Primary Actor: utiliztorul
Main success scenario: Se genereaza un nou anunt vizibil tuturor

#### 3.1.5 Stergerea anuntului

Level: user goal level
Primary Actor: utiliztorul
Main success scenario: se sterge anuntul si nu mai apare in baza de date

#### 3.1.6 Vizionare carti

Level: user goal level
Primary Actor: utiliztorul
Main success scenario: se afiseaza toate cartile

#### 3.1.7 Cautare carte dupa nume

Level: user goal level
Primary Actor: utiliztorul
Main success scenario: se afiseaza toate cartile cu numele respectiv

#### 3.1.8 Cautare carte dupa autor

Level: user goal level
Primary Actor: utiliztorul
Main success scenario: se afiseaza toate cartile de la autorul respectiv

#### 3.1.9 Modificarea anuntului

Level: user goal level
Primary Actor: utiliztorul
Main success scenario: se fac modificarile pe anunt

#### 3.1.10 Stergerea unui utilizator

Level: user goal level

Primary Actor: admin

Main success scenario: utilizatorul este sters din baza de date

#### 3.1.11 Achizitionarea unei carti

Level: user goal level

Primary Actor: utilizator

Main success scenario: se genereaza o factura

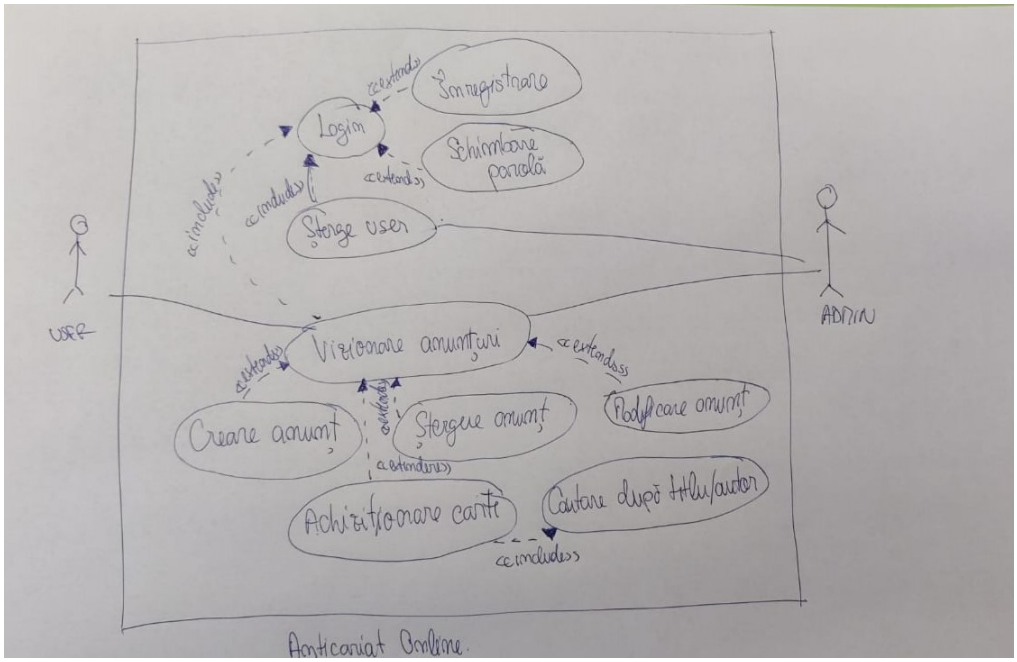



Figura 1: UML Use Case Diagram

# 4 Suplementary Specfication

## 4.1 Non-functional Requirements

Cerintele non-functionale sunt:

- portabilitate - poate functiona pe diferite tipuri de device-uri;
- utilizabilitate - sa fie user-friendly;
- mentenabilitate - sa fie usor de adaptat la un mediu in contiuna schimbare.

## 4.2 Design Constraints

Partea de backend a proiectului o sa fie conceputa cu ajutorul limbajului Java si Spring, parte de frontent o sa fie facuta in React iar Baza de date o sa fie implementata si stocata cu ajutorul aplicatiei MySQL, simplificarea codului va fi realziata cu ajutorul Lombok, maparea intre baza de date si java se face cu JPA. IDE-ul folosit ieste InteliJ. In cadrul proiectului o sa fie folosita o arhitectura stratificata care o sa respecte principiile SOLID. O sa avem cinci mari module: Model, Repo, UI, Controller si Service, care vor avea fiecare multiple clase.

# 5 Domain Model

Diagrama modelează domeniul aplicației și este utilă în înțelegerea modului în care diferitele entități din sistem interacționează între ele. Diagrama de mai jos prezintă entitățile principale ale sistemului și relațiile dintre ele.

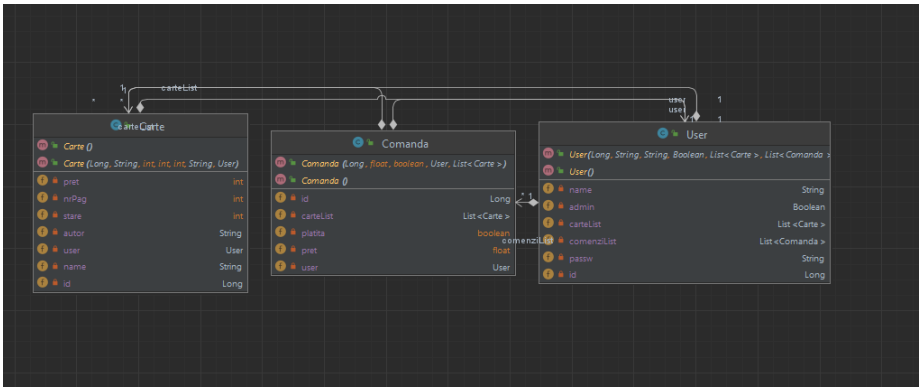



Figura 2: Domain Model

Diagrama cuprinde următoarele entități:

- **Carte** - reprezintă o carte care poate fi vândută sau cumpărată în cadrul anticariatului online. Aceasta are următoarele atribute:
  - *name* - titlul cărții;
  - *autor* - autorul cărții;
  - *nrPag* - numarul de pagini;
  - *stare* - starea cartii;
  - *user* - userul care a acreat anuntul cu cartea;
  - *preț* - prețul cărții;
  - *id* - identificator unic;
- **User** - reprezintă un utilizator al aplicației, care poate fi fie un client, fie un angajat al anticariatului. Aceasta are următoarele atribute:
  - *name* - numele utilizatorului;
  - *passw* - parola utilizatorului;
  - *admin* - verifica daca userul e admin;
  - *carteList* - lista de carti;
  - *comenziList* - lisat de comenzi;
  - *id* - identificator unic;
- **Comanda** - reprezinta o comanda
  - *id client* - id-ul unic al clientului;
  - *pret total* - preturl comenzii;
  - *platita* - starea platii;
  - *carteList* - lista de carti;
  - *user* - userul care da comanda;

# 6 Architectural Design

## 6.1 Conceptual Architecture

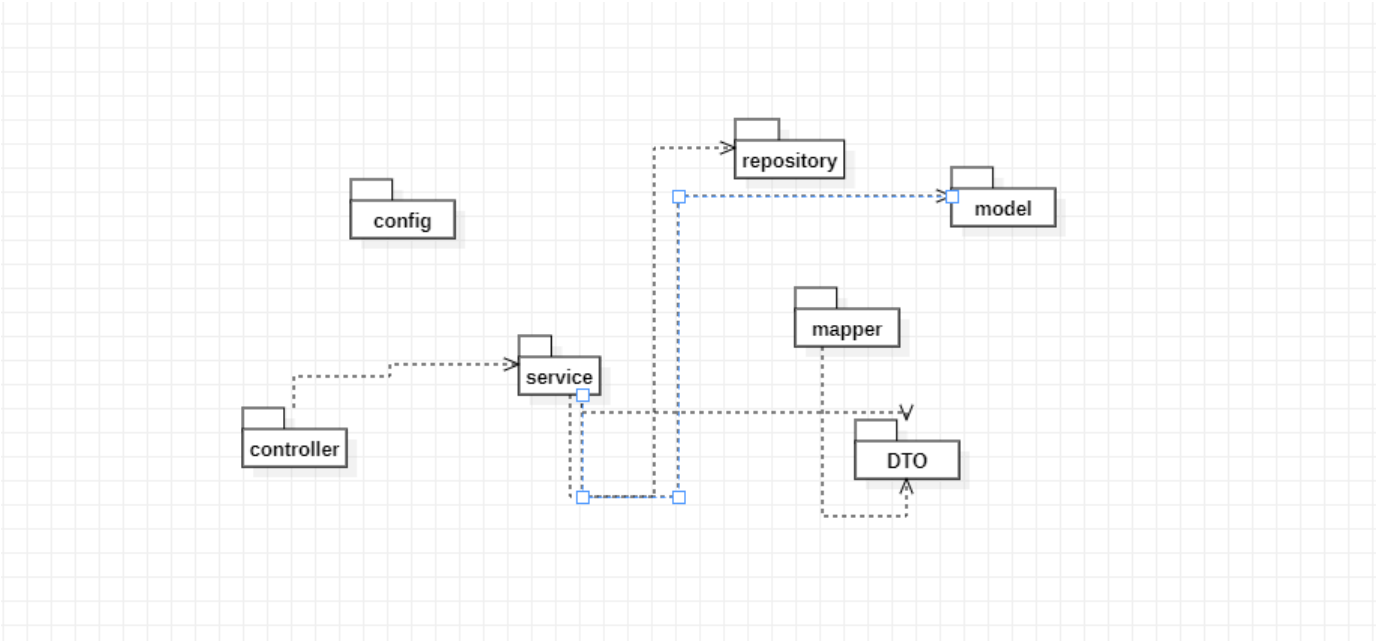
Arhitectura se bazează pe o arhitectură în straturi (layered architecture), care are următoarea structură:

1. Straturi de prezentare (Presentation Layer)
   - Interfața de utilizator (UI) oferă o interfață grafică utilizatorului și se ocupă de afișarea datelor utilizatorului. Această interfață poate fi o aplicație web sau o aplicație desktop.
   - Straturile inferioare de prezentare sunt responsabile pentru afișarea informațiilor utilizatorului și preluarea interacțiunii utilizatorului (ex: input-ul de la tastatură sau mouse). Acestea pot fi reprezentate prin componente precum HTML/CSS sau librării de interfață utilizator (ex: React, Vue.js, etc.).
2. Straturi de afaceri (Business Layer)
   - Acestea reprezintă logica de afaceri și se ocupă de prelucrarea datelor primite din straturile inferioare. Aici sunt definite toate entitățile și operațiile legate de afacere, cum ar fi operațiile de CRUD (create, read, update, delete) și verificarea validității datelor.
   - În acest strat sunt incluse și serviciile (service layer) care expun logica de afaceri prin interfețe API pentru a putea fi utilizate în alte componente ale sistemului. Aceste servicii pot fi implementate, de exemplu, cu ajutorul framework-urilor precum Spring sau Node.js.
3. Straturi de acces la date (Data Access Layer)
   - Acestea sunt responsabile de comunicarea cu baza de date și manipularea datelor. În acest strat sunt incluse framework-uri ORM (Object Relational Mapping) care facilitează operațiunile cu baza de date.
   - În plus, acest strat poate include și cache-ul, pentru a optimiza performanța și timpul de răspuns al aplicației.

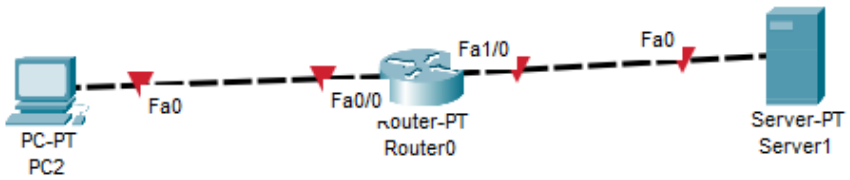
Arhitectura în straturi permite separarea logicii de afaceri de interfața utilizator și de accesul la date, permițând dezvoltarea mai ușoară și extinderea ulterioară a aplicației. În plus, această arhitectură permite dezvoltarea și testarea separată a fiecărui strat, reducând riscul de erori și întârzieri în timpul procesului de dezvoltare.

Este important de menționat că, în funcție de specificul proiectului, arhitectura poate fi adaptată și modificată în consecință.

### 6.2 Package Design




### 6.3 Component and Deployment Diagram




## 7 Design Model

Pentru a realiza un model de design pentru un proiect similar, putem urma următoarele etape:

1. Identificarea cerințelor: Înțelegerea cerințelor proiectului este esențială înainte de a începe să proiectăm. Această etapă implică interacțiunea cu clientul pentru a înțelege cerințele proiectului.

2. Proiectarea arhitecturii: După ce am înțeles cerințele proiectului, putem începe să proiectăm arhitectura sistemului. În această etapă, vom identifica componentele sistemului, interacțiunile dintre ele și fluxul de date prin sistem.
3. Proiectarea interfeței grafice: Proiectarea interfeței grafice implică crearea unui aspect atractiv și intuitiv pentru utilizatori. Este important să se țină cont de modul în care utilizatorii interacționează cu sistemul și să se creeze o interfață care să fie ușor de utilizat și să ofere o experiență pozitivă utilizatorilor.
4. Proiectarea bazei de date: În această etapă vom proiecta schema bazei de date și vom identifica relațiile între diferitele tabele.
5. Implementarea: După finalizarea etapelor anterioare, putem începe să implementăm sistemul. Implementarea poate fi împărțită în diferite etape, în funcție de complexitatea sistemului și timpul disponibil.
6. Testare: După finalizarea implementării, vom efectua diferite teste pentru a ne asigura că sistemul funcționează corect și îndeplinește cerințele proiectului.
7. Implementarea și întreținerea sistemului: După finalizarea testelor, sistemul poate fi lansat în producție. Întreținerea sistemului poate fi necesară pentru a ne asigura că acesta funcționează în continuare și pentru a efectua actualizări și remedieri de erori în viitor.

## 7.1 Class Diagram

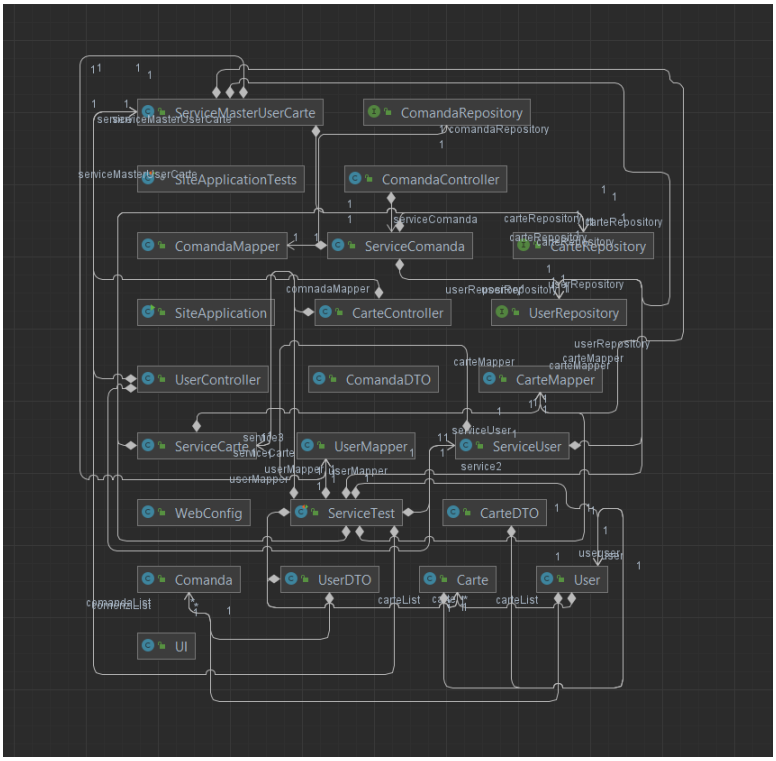



Figura 3: Class Diagram

## 7.2 Data Model

Un data model reprezintă o reprezentare abstractă a datelor și relațiilor dintre acestea, utilizată pentru a proiecta bazele de date și pentru a gestiona informațiile într-un mod eficient.

Există mai multe tipuri de data model, dar cel mai comun este modelul relațional. În acest model, datele sunt organizate în tabele, iar relațiile dintre acestea sunt reprezentate prin intermediul cheilor primare și străine. Ceea ce se si foloseste la priectul de fata.

## 8 System Testing

Metodele de testare pe care le-am folosit pentru proiectul meu:

1. Testare unitară: Am folosit testarea unitară pentru a testa componentele sau modulele individuale ale aplicației. Aceasta m-a ajutat să mă asigur că fiecare modul funcționează conform așteptărilor în timpul procesului de dezvoltare. Exemple de teste unitare: Am testat funcția getMovieByName() pentru a mă asigura că returnează detaliile corecte ale filmului. Am testat funcția updateMovieWatchList() pentru a mă asigura că actualizează corect lista de filme de urmărit pentru un anumit film.
2. Testare de sistem: Am folosit testarea de sistem pentru a testa întregul sistem în ansamblu, pentru a mă asigura că îndeplinește cerințele specificate.

Exemple de teste de sistem: Am testat sistemul de înregistrare și conectare a utilizatorilor prin crearea unui cont de utilizator nou și conectarea cu detaliile contului creat recent. M-am asigurat că utilizatorul a fost redirecționat către pagina corectă după conectarea cu succes. Am testat sistemul de căutare a filmelor prin introducerea unui nume de film și asigurarea că sistemul returnează o listă de filme disponibile care se potrivesc criteriilor de căutare. Am testat sistemul de rezervare prin selectarea unui film și rezervarea unui bilet. M-am asigurat că pagina de confirmare a rezervării afișează detaliile corecte ale rezervării și costul total.

## 9 Future Improvements

Am regasit urmatoarle impbunatatiri:

1. Îmbunătățirea experienței de utilizare: Acest lucru poate fi realizat prin simplificarea și optimizarea fluxurilor de utilizator, reducerea timpului de încărcare și îmbunătățirea interfeței de utilizator. Dezvoltatorii ar putea utiliza diverse tehnici și instrumente pentru a analiza comportamentul utilizatorilor și pentru a obține feedback, astfel încât să poată îmbunătăți în mod constant experiența de utilizare.
2. Îmbunătățirea performanței: În cazul unei aplicații web, performanța este un aspect crucial. Dezvoltatorii ar putea utiliza diverse tehnici, cum ar fi optimizarea bazei de date, utilizarea cache-ului și a tehnologiilor de încărcare progresivă, pentru a îmbunătăți performanța aplicației.
3. Îmbunătățirea securității: Securitatea este un alt aspect important în cazul unei aplicații web. Dezvoltatorii ar trebui să se asigure că aplicația lor este protejată împotriva atacurilor cibernetice, precum și să ia măsuri pentru a proteja datele utilizatorilor.
4. Dezvoltarea unor funcționalități noi și avansate: După ce s-au implementat funcționalitățile de bază, dezvoltatorii ar putea să lucreze la dezvoltarea unor funcționalități noi și avansate, cum ar fi integrarea cu servicii terțe sau implementarea unor funcții avansate de căutare.

## 10 Conclusion

Concluzia generată în urma realizării proiectului unui anticariat online de cărți în Spring Boot, Java și Angular poate fi una pozitivă, în sensul că această combinație de tehnologii poate fi eficientă și poate oferi o experiență plăcută utilizatorilor.

Folosind Spring Boot, se poate dezvolta o aplicație rapidă și scalabilă, care poate gestiona eficient datele din baza de date și poate interacționa cu alte componente ale sistemului. Java este un limbaj de programare robust și versatil, care poate fi utilizat pentru a dezvolta funcționalități complexe și avansate, iar Angular este o tehnologie modernă de frontend, care poate fi utilizată pentru a oferi o interfață de utilizator atractivă și ușor de utilizat.

În general, dezvoltarea unui anticariat online de cărți poate fi un proiect interesant și provocator, care poate necesita abilități de dezvoltare backend și frontend, precum și abilități de gestionare a bazei de date și a altor componente ale sistemului. În final, succesul unui astfel de proiect depinde în mare măsură de capacitatea dezvoltatorilor de a integra toate componentele și de a oferi o experiență de utilizare excelentă.

# 11 Bibliography

`https://www.youtube.com/watch?v=yYlMsNdAXJ4&list=PLGRDMO4rOGcNzi3CpBWsCdQSzbjdWWy-f&index=16`