UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Bookstore

Name: Cociubei Antonia
Group: 30238

Table of Contents

Deliverable 1

Project Specification

Obiectivul acestui proiect este dezvoltarea unui magazin online de cărți, pentru un nou competitor intrat de curând pe piață. Acesta dorește își extindă vânzările de cărți discounted dincolo de magazinul fizic, astfel clienții săi să poată comanda carțile preferate printr-o aplicație user-friendly, fără a se deplasa, bucurandu-se de opțiuni eficinete de filtrare și căutare, de plată sigură și prețuri avantajoase.

Functional Requirements

Site-ul are două tipuri de utilizatori: client și admin.

Adminul dispune de credentiale de loagre în contul său ce îi conferă drepturi depline asupra întregii pagini web putând să adauge, să modifice și să șteargă produsele din baza de date a magazinului.

Clientul își poate crea cont și se poate loga ulterior în cont utlizilând numele si parola. Acesta poate căuta și filtra cărțile după diferite criterii, precum autor, editură, gen literar sau interval de preț. Poate adauga produse în coș, goli coșul de cumpărături, trmite comenzi și vizualiza istoricul comenzilor sale.

Use Case Model 1

Use Cases Identification

Use-Case: Register

Level: Client-Goal

Primary Actor: Client

Main success scenario:Clientul își introduce datele și este creat cu success contul personal.

Extensions: Emaiul este deja asociat unui cont sau formatul acestuia ori al parolei nu este valid.

Use-Case: Log-in

Level: User-Goal

Primary Actor: Admin/Client

Main success scenario: User-ul își introduce datele in fieldurile associate, iar daca logarea s-a realizat cu success, se va deschide pagina corespunzătoare tipului de user.

Extensions: În cazul în care parola este incorectă aceasta poate fi schimbată.

Use-Case: Căutare produs

Level: Client Goal

Primary Actor: Client

Main success scenario: Găsește produsul pe care îl caută, iar acesta este disponibil.

Extensions: Solicită să fie notificat dacă produsul revine în stoc.

Use-Case: Adăugare produs în coș

Level: Client-Goal

Primary Actor: Client

Main success scenario: A adăugat produsele dorite în coș și merge spre pagina de finalizare a comenzii.

Extensions: Dacă stocul este mai mic decât numărul de exemplare introdus, va fi avetrizat și nu va putea continua până nu este introdusă o cantitate ce se află în stoc.

Use-Case: Golește coș

Level: Client-Goal

Primary Actor: Client

Main success scenario: Coșul este golit cu success.

Extensions: -

Use-Case: Plasare comandă

Level: Client goal

Primary Actor: Client

Main success scenario: Daca nu este autentificat, se va autentifica. Trimite cu success comanda.

Extensions: În cazul unui cont invalid va fi atenționat și se va cere autentificarea.

Use-Case: Vizualizare istoric comenzi

Level: Client-Goal

Primary Actor: Client

Main success scenario: Se afiseaza lista ce cuprinde comenzile anterioare și detaliile sumare despre fiecare comandă.

Extensions: -

Use-Case: Gestionare produse

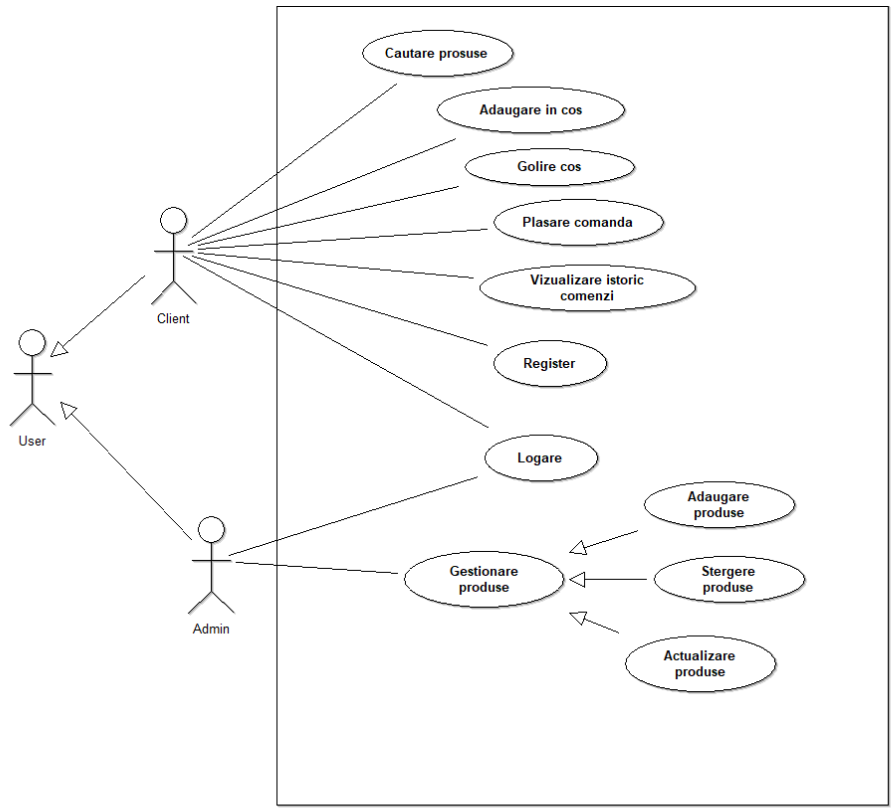
Level: Admin-Goal

Primary Actor: Admin

Main success scenario: Adminul adaugă/șterge/modifică un produs.

Extensions: Sunt omise anumite câmpuri obligatorii. S-a șters produsul greșit.

UML Use Case Diagrams



Supplementary Specification

Non-functional Requirements

Availability: Pagina web va fi disponibila 24 de ore din 24, 7 zile din 7, cu perioade de mentenanță minime, pentru a se asigura disponibilitatea magazinului oricând doresc clienții.

Usability: Interafața siteului web va fi intuitivă astfel încat experiența utilizăriii să fie una plăcută și accesibilă și utilizatorului nespecializat.

Performance: Website-ul va trebui să se încarce rapid și să răspundă prompt la input-ul utilizatorului pentru a asigura o experință de cumpărare cât mai satisfăcătoare.

Scalability: În viitor acest proiect va putea fi scalat prin adăugarea unor noi funcționalități.

Design Constraints

Aplicația este dezvoltată folosind mediul integrat de dezvoltare Intellij și implementată în limbajul de programare Java împreună cu frameworkul Spring, care este integrat cu o bază de date MySQL. Datele sunt preluate și testate folosind Postman. Toate clasele au fost structurate în pachete pentru o mai bună mentenanță și o ușoară scalare a proiectului. Interfața cu utilizatorul va fi creată folosind React.

Deliverable 2

Domain Model

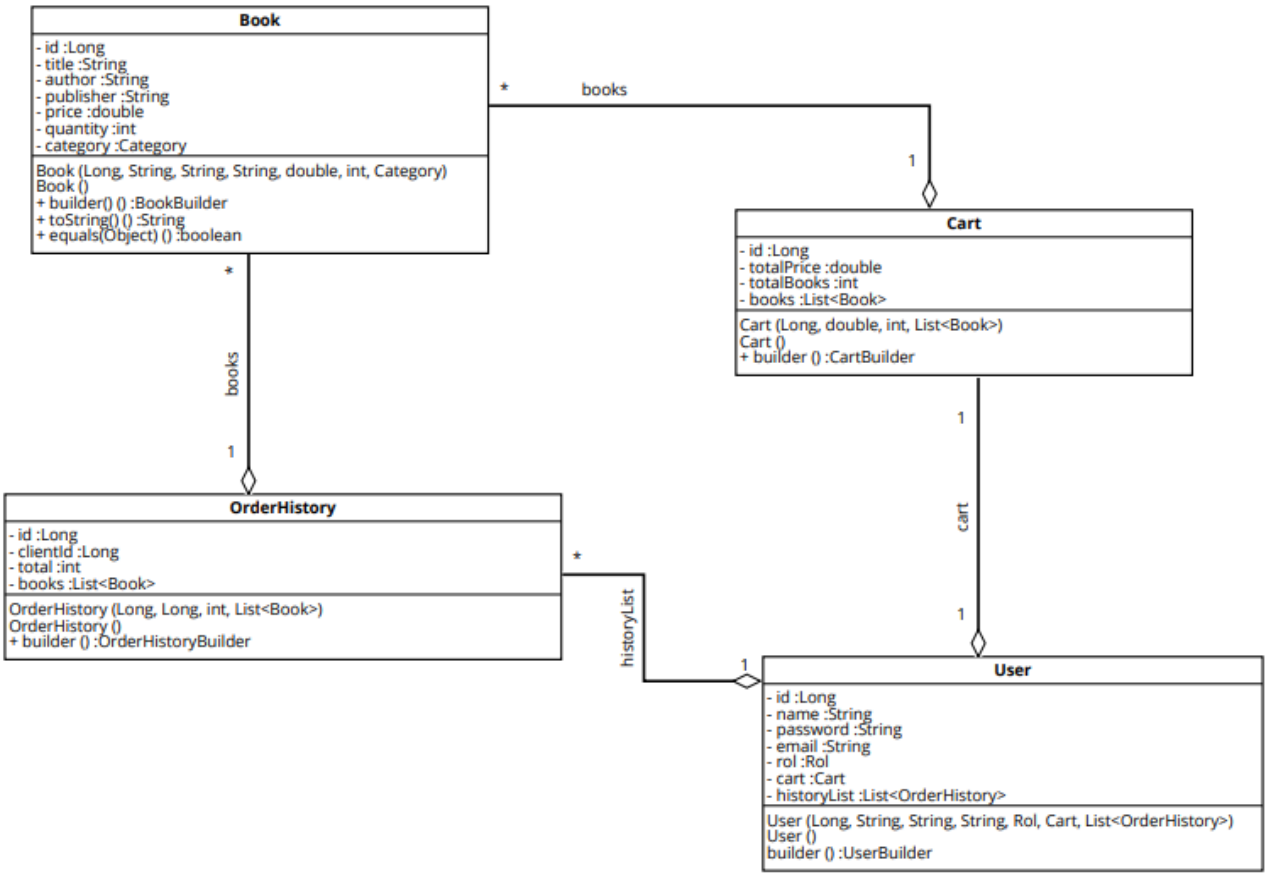
Un model de domeniu este un model conceptual care încapsulează cunoștințele și interacțiunile unui anumit domeniu de problemă. Este utilizat în ingineria software pentru a înțelege mai bine și a reprezenta domeniul de afaceri pe care software-ul intenționează să-l modeleze.

Modelul de domeniu este adesea prezentat ca un diagramă vizuală, cum ar fi un diagramă de clase, și poate include diverse elemente, cum ar fi: entități, relații, atribute, operațiuni.

Entitățile sunt principalele obiecte sau concepte din domeniu.

Relațiile descriu modul în care diferite entități se raportează una la alta.

Atributele reprezintă detalii sau caracteristici ale entităților, iar *operațiunile sau comportamentele* sunt acțiunile care pot fi efectuate de entități sau care afectează entitățile.



Architectural Design

Conceptual Architecture

Arhitectura proiectului este bazată pe designul ***Layered***, o abordare care ne permite să descompunem programul în grupuri de subtask-uri, fiecare cu un nivel specific de abstractizare. Fiecare nivel oferă servicii pentru cel următor, creând un flux coerent și logic.

Această structură bazată pe Layered Architecture oferă un grad înalt de organizare, îmbunătățește scalabilitatea și ușurează procesul de mentenanță și dezvoltare a proiectului.

Având în vedere pachetele și clasele menționate, putem face următoarea asociere cu nivelele de arhitectură:

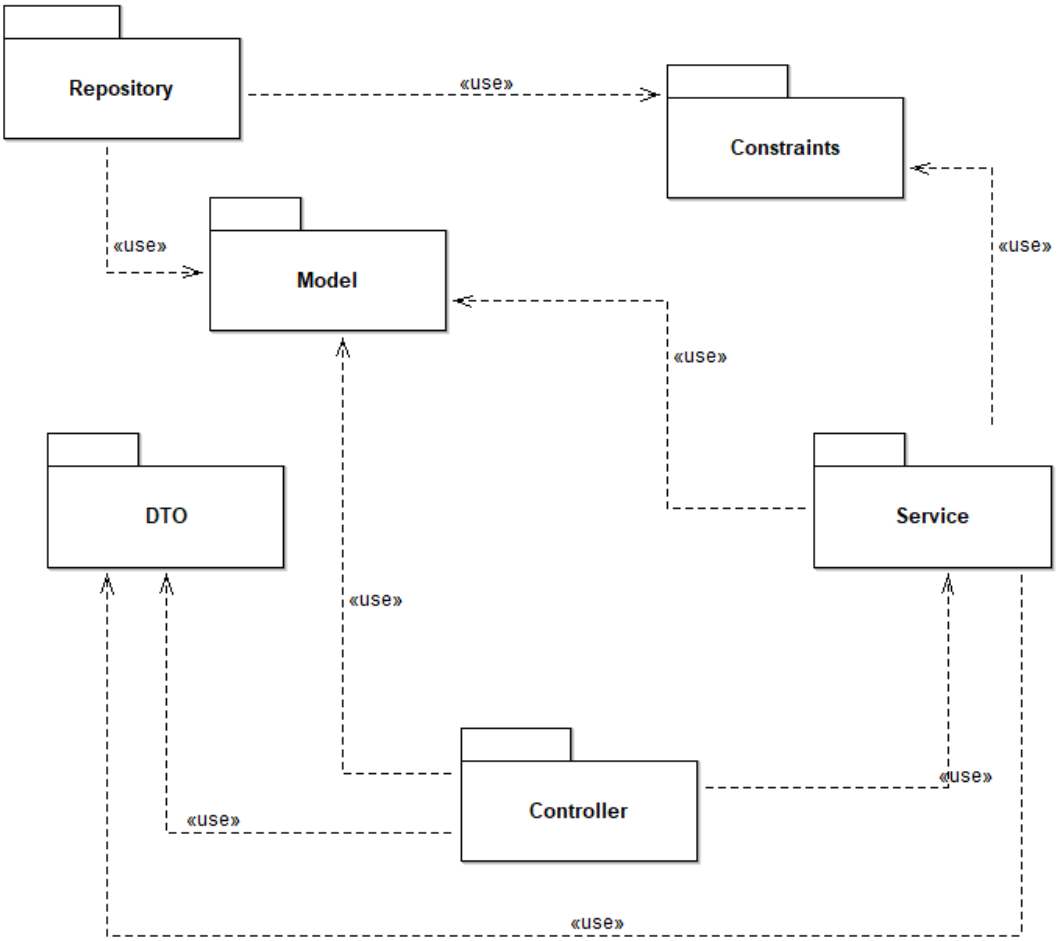
1. Presentation Layer: Acest nivel include pachetul *controllers*. Acestea gestionează interacțiunile cu utilizatorul, procesând solicitările și răspunsurile HTTP.

2. Application Layer/Service Layer: Acest nivel include pachetul *services* și subpachetul său *imp.* Acestea implementează logica de afaceri și servesc ca punte între nivelul de prezentare și cel de date. Se ocupă cu manipularea datelor primite de la nivelul Persistence Layer. Acesta realizează diverse servicii necesare în cadrul aplicației, precum interogări, ștergere, inserare și actualizare.

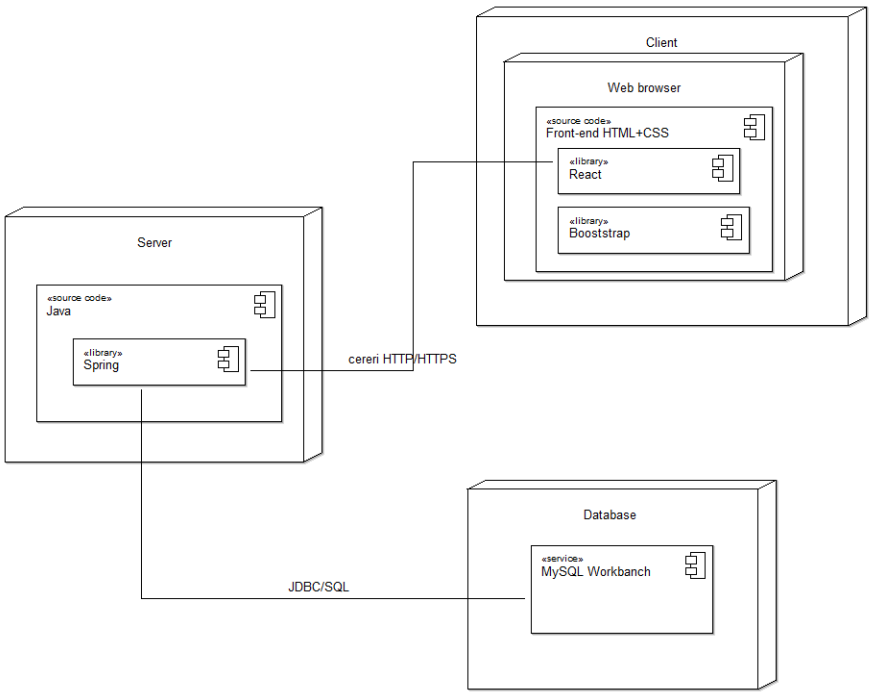
3. Business Logic Layer/Model Layer: Acest nivel include pachetul *model, dto* și *constraints*. Acestea reprezintă structurile de date și logica de afaceri specifică aplicației. DTO (Data Transfer Objects) sunt utilizate pentru a transfera date între sub-sisteme sau între server și client într-un mod eficient și sigur.

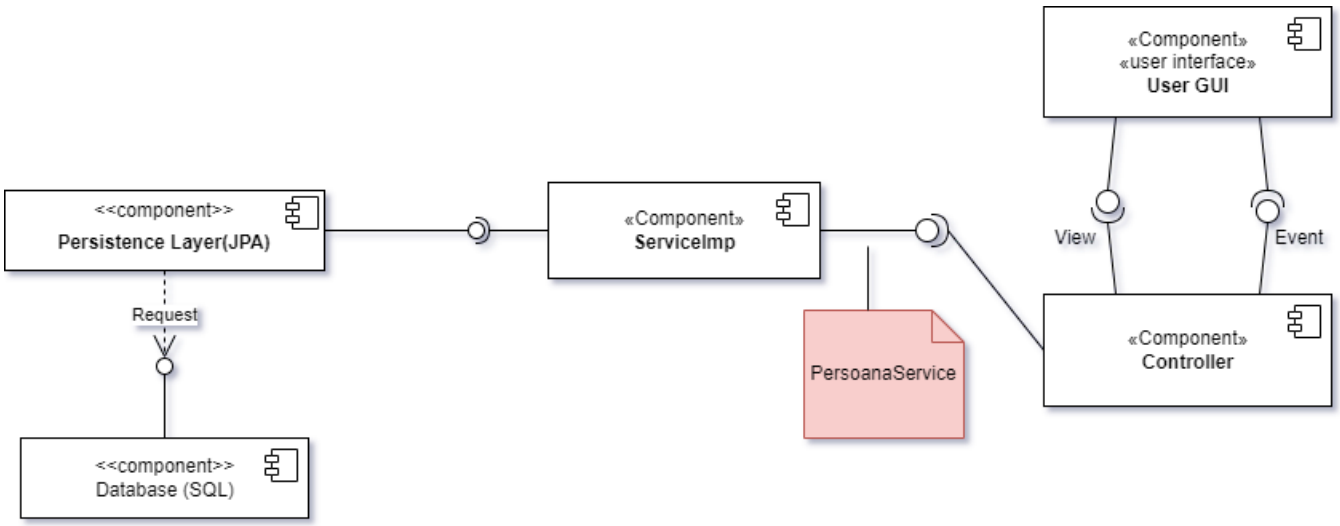
4. Data Access Layer/Persistence Layer: Acest nivel include pachetul *repository.* Acesta este o sub-ramură a Data Access Layer și servește ca punct de acces la baza de date. Acesta interacționează cu baza de date prin intermediul API-ului JpaRepository, care oferă posibilitatea de a efectua operațiuni de tip CRUD.

Package Design



Component and Deployment Diagram





Deliverable 3

Design Model

Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

Class Diagram

[Create the UML class diagram; apply GoF patterns and motivate your choice]

Data Model

[Create the data model for the system.]

System Testing

[Describe the testing methides and some test cases.]

Future Improvements

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

Conclusion

Bibliography