Booking System

~ Website pentru Rezervare de

Unitati de Cazare ~

Name: Flinta Claudiu

Group: 30236

Table of Contents

[**Deliverable 1** 3](#_Toc64843130)

[Project Specification 3](#_Toc64843131)

[Functional Requirements 3](#_Toc64843132)

[Use Case Model 3](#_Toc64843133)

[Use Cases Identification 3](#_Toc64843134)

[UML Use Case Diagrams 3](#_Toc64843135)

[Supplementary Specification 3](#_Toc64843136)

[Non-functional Requirements 3](#_Toc64843137)

[Design Constraints 3](#_Toc64843138)

[Glossary 3](#_Toc64843139)

[Deliverable 2 3](#_Toc64843140)

[Domain Model 3](#_Toc64843141)

[Architectural Design 4](#_Toc64843142)

[Conceptual Architecture 4](#_Toc64843143)

[Package Design 4](#_Toc64843144)

[Component and Deployment Diagram 4](#_Toc64843145)

[Deliverable 3 4](#_Toc64843146)

[Design Model 4](#_Toc64843147)

[Dynamic Behavior 4](#_Toc64843148)

[Class Diagram 4](#_Toc64843149)

[Data Model 4](#_Toc64843150)

[System Testing 4](#_Toc64843151)

[Future Improvements 4](#_Toc64843152)

[Conclusion 4](#_Toc64843153)

[Bibliography 4](#_Toc64843154)

# Deliverable 1

## Project Specification

Proiectul Booking System reprezinta un website destinat zonei de turism, mai exact promoveaza diferite unitati de cazare, iar prin intermediul site-ului se pot realiza rezervari pentru unitatiile de cazare disponibile. Dat fiind faptul ca industria de turism incepe sa se dezvolte din ce in ce mai tare in ultima perioada, apare o cerere in directia digitalizarii acestei zone.

Principalul obiectiv este acela de a dezvolta o aplicatie “user oriented”, in care experienta de utilizare sa fie cat mai intuitiva, simpla si rapida. Un alt obiectiv ar fi acela de a intercoecta si de a reusi sa aducem intru-un singur loc cat mai multe unitati de cazare, atat in regim hotelier, cat si in regim propriu, si de a le expune unei mese extinse de indivizi.

Privind procesul de dezvoltare, aplicatie este create cu ajutorul urmatoarelor tehnologii de dezvoltare software: Java&Spring, React, MySQL.

## Functional Requirements

Registration and Log-In: Utilizatorii au posibilitatea de a se înregistra și de a crea un cont pe platformă prin furnizarea anumitor informațiilor personale.

Funcționalitatea de căutare și rezervare: Aplicația permite utilizatorilor să navigheze și să caute proprietățile disponibile în funcție de preferințele lor, cum ar fi locația, gama de prețuri, tipul de proprietate etc. Utilizatorii ar trebui, de asemenea, să poată rezerva proprietăți direct prin intermediul aplicației.

Gestionarea unitatiilor de cazare: Administratorul poate lista cu ușurință cazarile, inclusiv imagini, detalii despre proprietate și prețuri.

Comunicare: Aplicația va avea un sistem de comunicare care să permita interactiunea intre utilizatori. Va fi un fel de forum, in care utilizatorii vor putea imparatasi experiente, vor putea pune intrebari si vor primi raspunsuri.

User review and rating system: Aplicația permite utilizatorilor să evalueze și să ofere o nota unitatii de cazare, ajutând alți utilizatori să ia decizii informate în timp ce aleg o unitate de cazare.

## Use Case Model 1

### Use Cases Identification

**Use-Case**: Registration

**Level**: primary level

**Primary Actor**: Administrator/Normal User

**Main success scenario**: User-ul introduce datele in fieldurile associate, dupa care contul de salveaza in tabela corespunzatoar utilizatorilor platformei

Extensions: -

**Use-Case**: Log-in

**Level**: primary level

**Primary Actor**: Administrator/Normal User

**Main success scenario**: User-ul introduce datele in fieldurile associate, dupa care se valideaza si in caz de success, se deschide pagina corespunzatoare tipului de user.

**Extensions**: schimbarea parolei in cazul in care a fost uitata

**Use-Case**: Add/Delete/Update Hotels

**Level**: primary level

**Primary Actor**: Administrator

**Main success scenario**: Administratorul poate realiza diferite operatiuni asupra unitatiilor de cazare disponibile pe site. Poate adauga unitati noi, modifica datele celor existente sau le poate sterge definitive.

Extensions: -

**Use-Case**: View Reservations

**Level**: primary level

**Primary Actor**: Normal User

**Main success scenario**: User-ul poate sa vizualizeze rezervarile pe care le are active, dar si pe cele precedente.

Extensions: -

**Use-Case**: Search for hotels by certain criteria

**Level**: primary level

**Primary Actor**: Normal User

**Main success scenario**: User-ul poate filtra unitatiile de cazare in functie de anumite criterii, cum ar fi: data, regiunea, tip-ul camerei etc.

Extensions: -

**Use-Case**: Make a reservation

**Level**: primary level

**Primary Actor**: Normal User

**Main success scenario**: User-ul poate crea o rezervare a unei unitati de cazare pentru o anumita perioada de timp.

Extensions: -

### UML Use Case Diagrams

Diagram

Description automatically generated

## Supplementary Specification

### Non-functional Requirements

Securitate: Aplicația trebuie să ofere un mediu sigur pentru toti utilizatorii, inclusiv pe partea de criptare a datelor (unde este cazul) si pe partea de autentificarea utilizatorilor.

Fiabilitate: Aplicația trebuie să fie extrem de fiabilă, cu timpi minimi de nefuncționare, sa gestioneze eventualele erori și sa fie remediate in timp minimal.

Utilizare: Aplicația trebuie să fie user oriented, simpla, cu instrucțiuni clare și concise pentru oaspeți și gazde.

Performanță: Aplicația trebuie să aibă timpi de răspuns rapizi, timpi rapizi de încărcare a paginii și latență minimă pentru a asigura o experiență buna a utilizatorului.

### Design Constraints

Limbaj de programare : Java&Spring (backend), React (frontend) si MySQL (database).

Unit Testing : JUnit

Integration Testing

IDE : Intellij

# Deliverable 2

## Domain Model

Domain modelul poate fi definit in cazul de fata de procesul prin care utilizatorii se conecteaza la paltforma web si realizeaza rezervari pentru diferite unitati de cazare, in functie de criteriile lor de selectie.

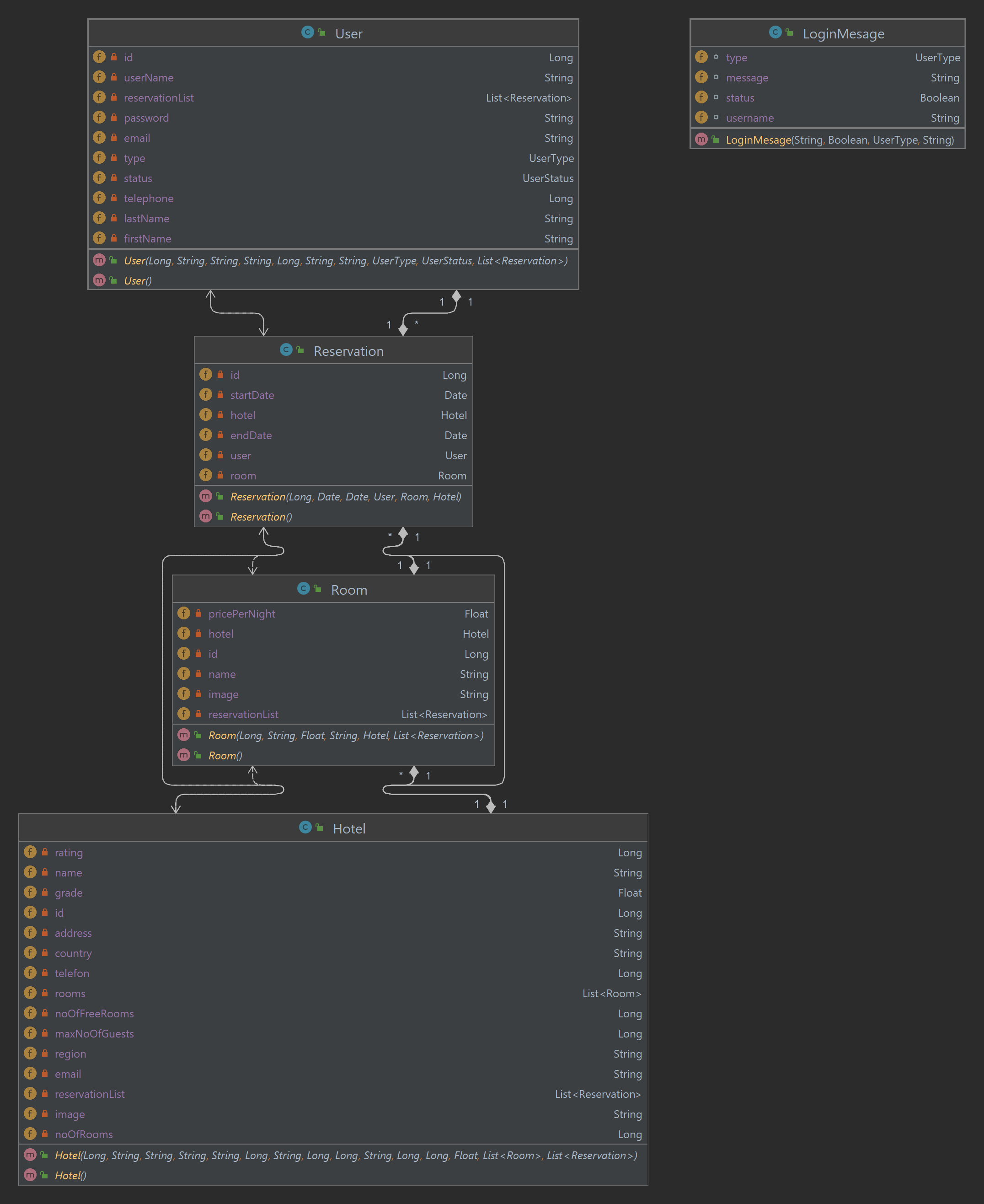
*Bazandu-ne pe modelul definit mai sus, se desprind 4 clase principale : User, Hotel, Room si Reservation.*

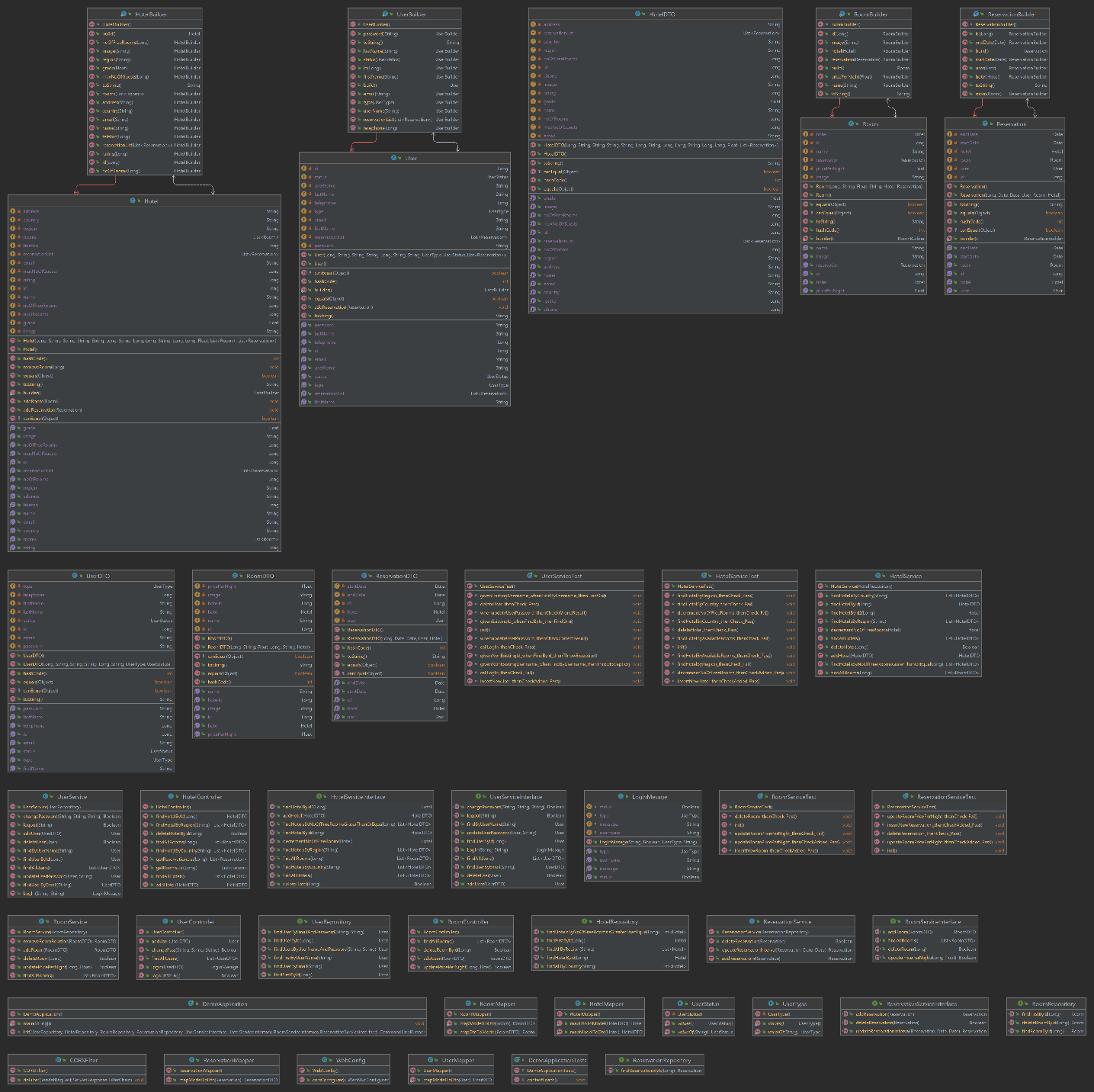
***Clasa User*** *defineste datele utilizatorului. Acesetei clase ii este atasat un Enum in care se specifica tipul de utilizator : Normal User sau Admin User.*

***Clasa Hotel*** *insumeaza datele corespunzatoare fiecarei unitati de cazare ce va fi listata pe site.*

***Clasa Room*** *incapsuleaza datele referitoare la camerele fiecarui hotel. Intre aceasta clasa si clasa Hotel exista o relatie de dependenta.*

***Clasa Reservation*** *tine date referitoare la rezerarile pe care utilizatorii aplicatiei le creaza.*

**

**

Graphical user interface

Description automatically generated

## Architectural Design

### Conceptual Architecture

*Pentru Aplicatia de Rezervari, am folosit modelul architectural Model-View-Controller (MVC), adaptat desigur la cerintele functionale si non-functionale mentionate mai sus.*

*Componenta Model reprezintă datele și logica sistemului. Aceasta include clasele și obiectele care reprezintă diferitele entități din sistem, cum ar fi rezervările, clienții, camerele și hotelurile. Modelul include, de asemenea, metodele și algoritmii care efectuează calculele și operațiile necesare asupra datelor, preluate din partea de JPARepository.*

*Componenta View reprezintă interfața cu utilizatorul a sistemului, care include paginile web si restul elementelor care permit utilizatorului să interacționeze cu sistemul. Componenta View este responsabilă pentru extragerea si trimiterea datelor din model către utilizator, precum și pentru colectarea datelor de intrare și trimiterea acestora către Componenta Controller.*

*Componenta Controller reprezintă intermediarul dintre Model și View și este responsabilă pentru gestionarea datelor introduse de utilizator și actualizarea modelului în consecință. Controller-ul conține logica care determină modul în care ar trebui să fie procesată intrarea utilizatorului și este responsabil pentru actualizarea modelului.*

*Un avantaj al utilizării arhitecturii MVC pentru un site de rezervări este modularitatea si faptul ca este usor de modificat, si in consecinta usor de realizat partea de mentenanta. Componenta Model este responsabilă pentru gestionarea datelor și a logicii care pot fi actualizate sau modificate cu ușurință fără a afecta celelate 2 component: View si Controller.*

### Package Design

*Diagram

Description automatically generated*

### Component and Deployment Diagram

A picture containing text, diagram, parallel, plan

Description automatically generated

# Deliverable 3

## Design Model

### Dynamic Behavior

A picture containing text, screenshot, number, font

Description automatically generated

### Class Diagram

A screenshot of a computer screen

Description automatically generated with low confidence

## Data Model

A picture containing text, screenshot, diagram, font

Description automatically generated

# System Testing

Unit Testing: Această metodă implică testarea componentelor individuale sau a “unit-urilor” de cod în mod izolat. Se asigură că fiecare unitate funcționează corect, in maniera in care a fost proiectata. Partea de unit testing se concentrează in mod special pe validarea comportamentului unităților de cod mici, cum ar fi funcții, metode sau clase.

Cateva cazuri de testare ar fi:

1. Testarea functionalitatii de Login.
2. Testarea functionalitatii de Register
3. Testarea functionalitatii de Adaugare Rezervare
4. Testarea functionalitatii de Adaugare a unui Hotel
5. Testarea functionalitatii de asignarea a unei camere unui anumit hotel

# Future Improvements

1. Payment Gateway Integration

Integrarea unui payment gateway sigur și fiabil pentru a permite utilizatorilor să efectueze plăți online pentru rezervările lor. Odata cu aceasta integrare ar trebui crescut si nivelul de secutitate al site-ului, pentru a proteja datele bancare ale utilizatorilor.

1. Suport Tip Multi-Language

Deoarece site-ul targeteaza un public internațional, trebuie luat in considerare adăugarea de asistență în mai multe limbi și posibilitatea de a afișa prețurile în monede diferite. Acest lucru va face site-ul mai accesibil și mai ușor de utilizat pentru utilizatorii din diferite regiuni.

1. Mobile Integration

Trebuie sa asiguram o receptivitate a site-ului la dispozitive mobile și optimizare pentru diferite dispozitive și dimensiuni de ecran. Designul prietenos cu dispozitivele mobile va permite utilizatorilor să acceseze și să navigheze fără probleme pe site de pe smartphone-uri și tablete, îmbunătățindu-și experiența generală.

# Conclusion

În concluzie, dezvoltarea unui site de rezervări hoteliere este o sarcină complexă care implică diverse aspecte ce trebuie luate in considerare. In ceea ce am incercat sa surprind prin aceasta documentatie, am discutat mai multe aspecte cheie și îmbunătățiri pentru un astfel de proiect. De-a lungul dezvoltarii am intampinat diverse impedimente, legate de ideile pe care am incercat sa le pun in practica, dar am reusit in cele din urma sa adaptez lucrurile in asa maniera incat sa imbrace perfect mediul de dezvoltare.

# Bibliography

1. [React Tutorial (w3schools.com)](https://www.w3schools.com/react/default.asp)
2. [Spring boot tutorial (w3schools.blog)](https://www.w3schools.blog/spring-boot-tutorial)
3. [Getting Started | Securing a Web Application (spring.io)](https://spring.io/guides/gs/securing-web/)
4. [Working with Relationships in Spring Data REST | Baeldung](https://www.baeldung.com/spring-data-rest-relationships)
5. [Create a Slideshow With React - DEV Community](https://dev.to/anxiny/create-a-slideshow-with-react-1pb1)
6. [How to use map() in React applications - Upmostly](https://upmostly.com/tutorials/how-to-use-map-in-react-applications)