# VACATION BOOKING

## *WEB APPLICATION*

Proiect realizat de:

Baluti Laura Loredana
Cotoc Daniel Beniamin

01

# MOTIVATIE

*De ce o aplicatie de Booking?*

## Adresarea unei Nevoi Urgente pe Piață:

- În zilele noastre, totul se întâmplă online, de la cumpărături la rezervări de vacanțe. Aplicația noastră vine cu un răspuns acestei nevoi printr-o platformă simplă, rapidă și la îndemâna tuturor. Fie că vrei să-ți rezervi un hotel, o masă la restaurant sau un bilet la un eveniment, noi suntem aici să te ajutăm.

## Îmbunătățirea Experienței Utilizatorilor

- Ne dorim să facem rezervările cât mai ușoare și mai plăcute pentru utilizatori. Cu o interfață modernă și ușor de folosit, poți să faci rezervări de pe orice dispozitiv, fără bătăi de cap. Astfel, economisești timp și scapi de stresul metodelor tradiționale de rezervare.

02

# BACKEND

*Ce limbaj am folosit?*
*Unde și cum am salvat datele?*

# BACKEND

**Clasele Java:**

**Hotel:** oferă obiectivelor sale metode de get și set

**Room:** defineste camerele unui hotel

**Reservation:** creeaza rezervari pentru useri, definind data vacantei și hotelul ales

**User:** reține datele utilizatorilor aplicației

1

**Baza de date:**

**Tabele:** 4, cate unul pentru fiecare clasa, indexarea datelor fiind automată cu autoincrement

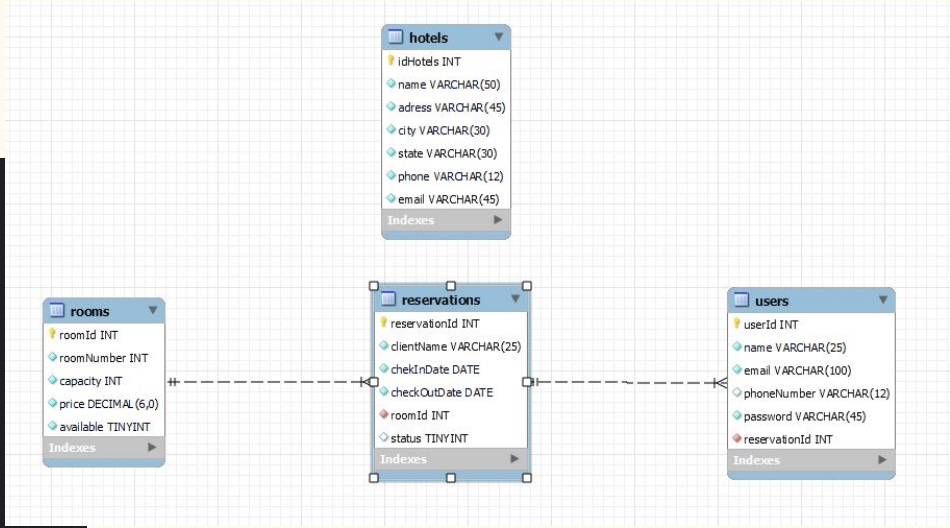
**Spring Boot:** folosit pentru comunicarea datelor către baza de date și pentru preluarea datelor din tabele

(Services și Controllers)

2

# EXEMPLE:

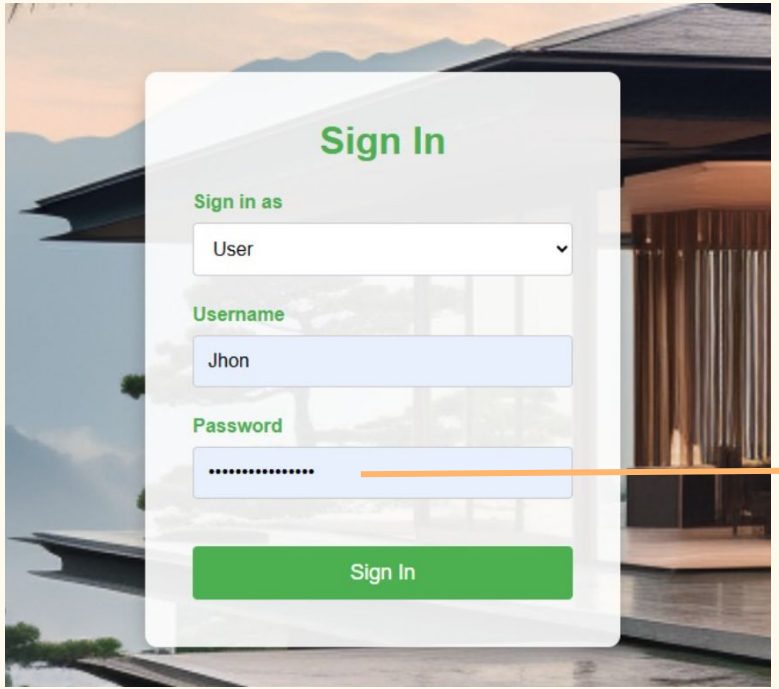
```java
@Component  4 usages  danyel513
public class HotelDBC {
    private final Connection connection;  2 usages
    private static final String DB_URL = "jdbc:mysql://127.0.0.1:3306/bookingapp";  1 usage
    private static final String DB_USER = "root";  1 usage
    private static final String DB_PASSWORD = "letmein";  1 usage

    public HotelDBC() throws HotelException {  danyel513
        Properties properties = new Properties();
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);
        try {
            connection = DriverManager.getConnection(DB_URL, properties);
        } catch (SQLException e) {
            throw new HotelException(e.getMessage());
        }
    }

    public void insert(Hotel hotel) throws HotelException {  danyel513
        try (PreparedStatement preparedStatement = connection.prepareStatement(INSERT_HOTEL_QUERY)) {
            preparedStatement.setString( parameterIndex: 1, hotel.getName());
            preparedStatement.setString( parameterIndex: 2, hotel.getAddress());
            preparedStatement.setString( parameterIndex: 3, hotel.getCity());
            preparedStatement.setString( parameterIndex: 4, hotel.getState());
            preparedStatement.setString( parameterIndex: 5, hotel.getPhone());
            preparedStatement.setString( parameterIndex: 6, hotel.getEmail());
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            throw new HotelException("Database could not be accessed: " + e.getMessage());
        }
    }
}
```




*Ce vedem?*

O clasa care adauga date în tabelul “hotels” folosind framework-ul JDBC.

# PASSWORD ENCRYPTION




```java
public void insert(User user) throws HotelException  danyel513 +1
{
    try (PreparedStatement preparedStatement = connection.prepareStatement(INSERT_USER_QUERY))
    {
        preparedStatement.setString( parameterIndex: 1, user.getName());
        preparedStatement.setString( parameterIndex: 2, user.getEmail());
        preparedStatement.setString( parameterIndex: 3, user.getPhoneNumber());
        // password encryption
        String encryptedPassword = BCrypt.hashpw(user.getPassword(), BCrypt.gensalt());
        preparedStatement.setString( parameterIndex: 4, encryptedPassword);
        preparedStatement.setInt( parameterIndex: 5, user.getReservation().getId());
        preparedStatement.executeUpdate();
    }
    catch (SQLException e)
    {
        throw new HotelException("Database could not be accessed: " + e.getMessage());
    }
}

public boolean checkPassword(String plainPassword, String hashedPassword) {  1 usage  Laura
    return BCrypt.checkpw(plainPassword, hashedPassword);
}
```

Parola nu este salvata ca plain text, ea este trecută printr-un filtru de criptare.

# CUM TESTAM?

**@BeforeAll:**

Vom inițializa obiectul testării.

**@AfterAll:**

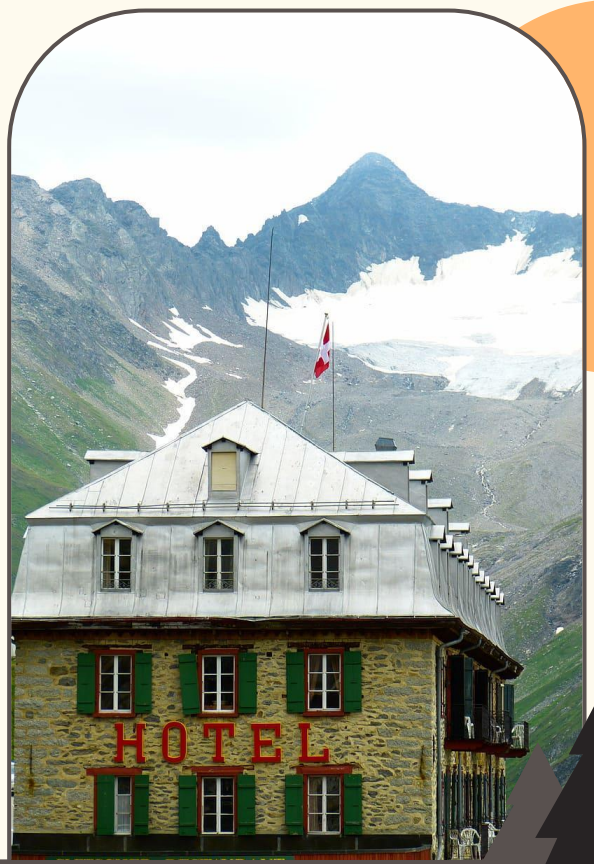
Vom elimina datele inserate in scopul testarii.

HotelDBCTest.java

```java
public class HotelDBCTest {  danyel513

    @BeforeAll  danyel513
    public static void setUp() throws HotelException
    {
        hotelDBC = new HotelDBC();
    }

    @AfterAll  danyel513
    public static void tearDown() throws SQLException
    {
        try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
             Statement statement = connection.createStatement())
        {
            String deleteQuery = "DELETE FROM hotels WHERE name = 'Test Hotel'";
            statement.executeUpdate(deleteQuery);
        }
    }

    @Test  danyel513
    public void testInsert()
    {
        Hotel testHotel = new Hotel( name: "Test Hotel",  address: "Test Address",  city: "Test City",  state: "Test State",  phone: "1234567890",  email: "test@example.com");

        assertDoesNotThrow(() -> hotelDBC.insert(testHotel),  message: "Insert should not throw an exception.");

        try {
            Hotel retrievedHotel = retrieveTestHotel();
            assertNotNull(retrievedHotel,  message: "Hotel should have been inserted into the database.");
            assertEquals(testHotel.getName(), retrievedHotel.getName(),  message: "Inserted hotel name should match.");
            assertEquals(testHotel.getAddress(), retrievedHotel.getAddress(),  message: "Inserted hotel address should match.");
            assertEquals(testHotel.getCity(), retrievedHotel.getCity(),  message: "Inserted hotel city should match.");
            assertEquals(testHotel.getState(), retrievedHotel.getState(),  message: "Inserted hotel state should match.");
            assertEquals(testHotel.getPhone(), retrievedHotel.getPhone(),  message: "Inserted hotel phone should match.");
            assertEquals(testHotel.getEmail(), retrievedHotel.getEmail(),  message: "Inserted hotel email should match.");
        }
        catch(SQLException e)
        {
            System.out.println(e.getMessage());
        }
    }
```

**@Test:**

Vom insera un obiect în tabela cu metoda realizata de noi apoi vom citi ultimul Hotel introdus și vom facem comparația.




03

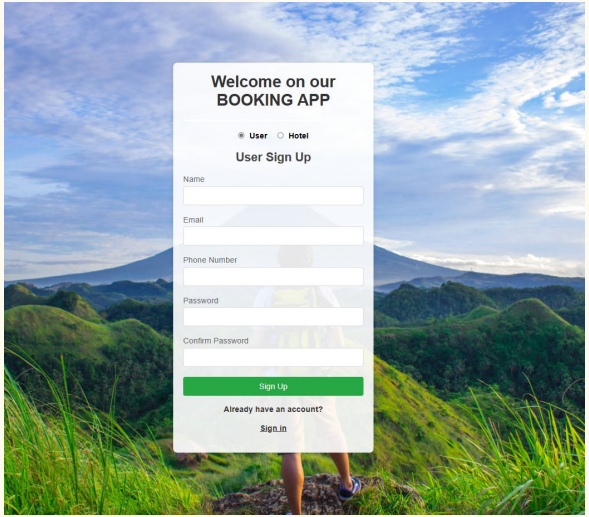
# FRONTEND

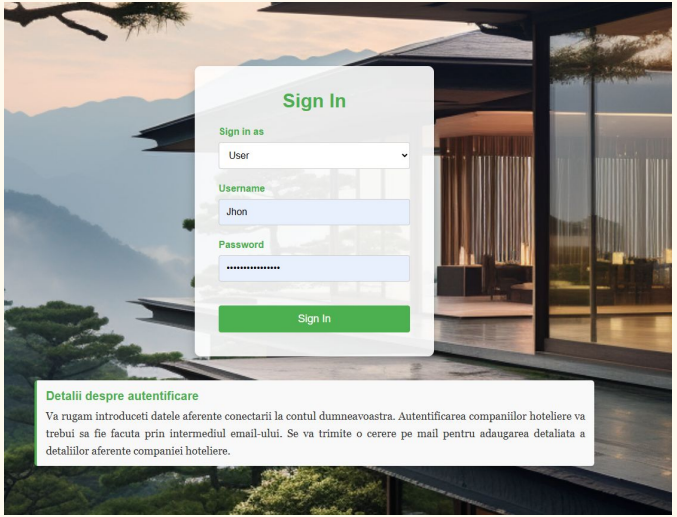
*Ce vede utilizatorul vs*
*ce cunoaște aplicația?*

# PRIMA INTERACȚIUNE CU APLICAȚIA

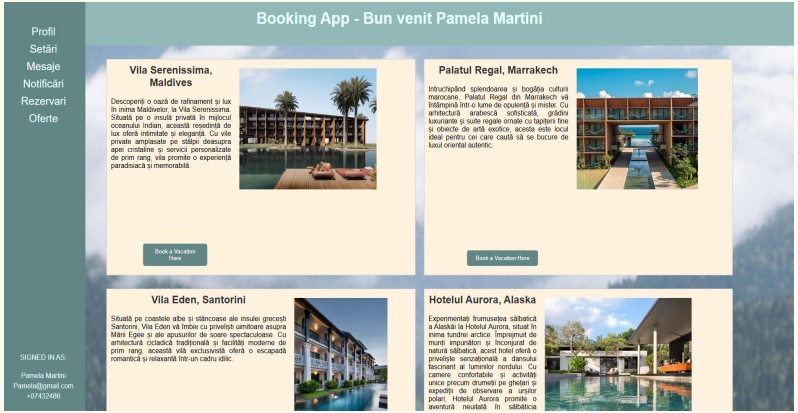
## *Cum interacționează aplicația cu fiecare utilizator?*

La pornirea aplicației utilizatorul va fi intampinat de o pagina de sign-up. Acesta poate alege sa își creeze cont sau sa se autentifice într-un cont deja existent.

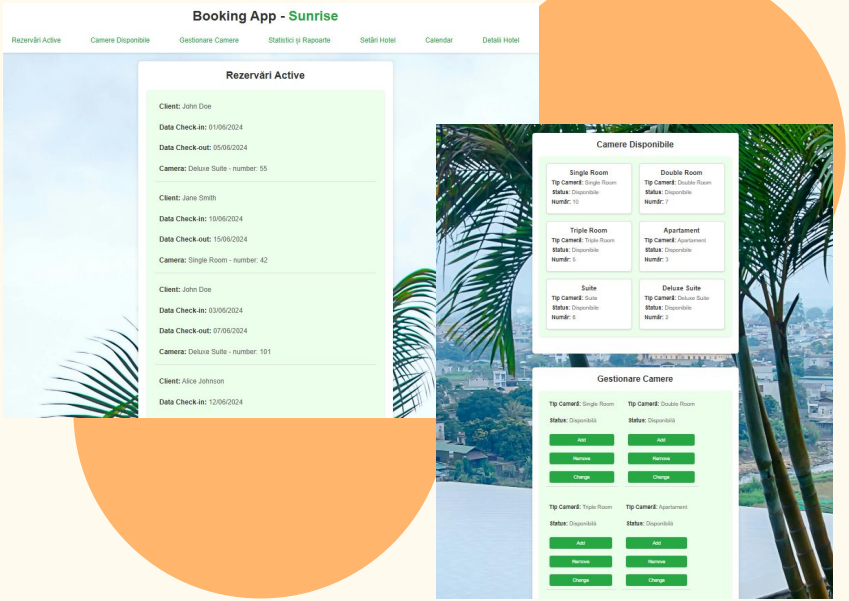


# USER HOMEPAGE




Oferă utilizatorilor recomandări pentru locații de vacanță exclusive, cu descrieri detaliate și opțiunea de a rezerva direct din aplicație.




# HOTEL HOMEPAGE

Permite adăugarea, eliminarea și modificarea camerelor disponibile, gestionând starea acestora pentru a reflecta disponibilitatea curentă.


04

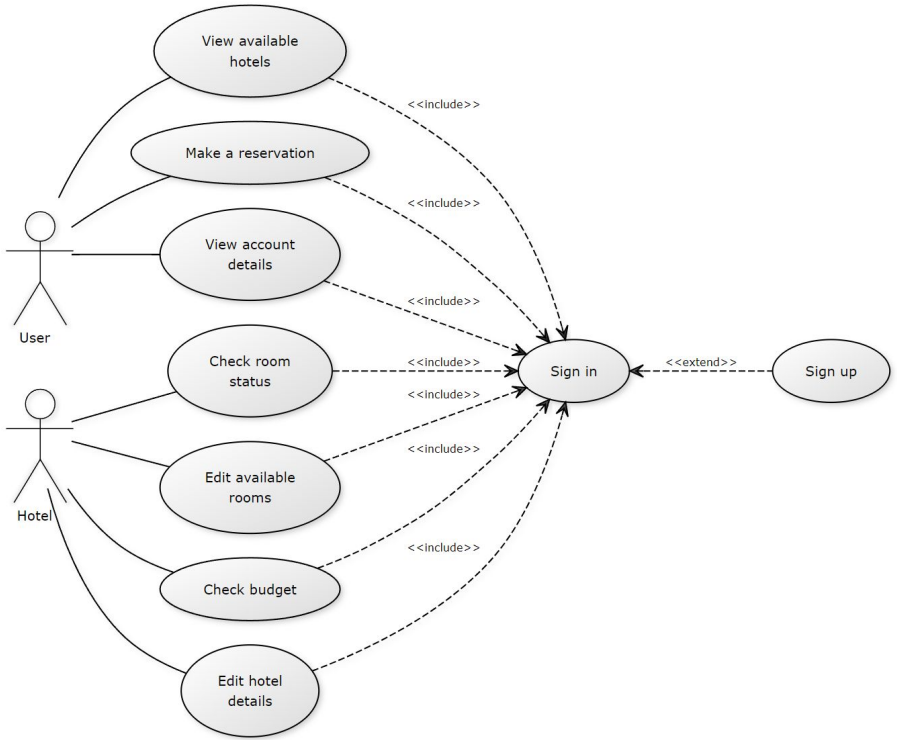
# Diagrama Use Case

*Cum interacționează aplicația cu fiecare utilizator?*

# USER și HOTEL

Atât userii cat și hotelurile vor avea nevoie să se conecteze pentru a putea face oricare din acțiunile disponibile.

În cazul în care User-ul sau Hotelul se află pentru prima dată pe pagina noastră, aceștia au opțiunea de a crea un cont nou.



# STATISTICI

Câte fișiere au fost necesare realizării proiectului?

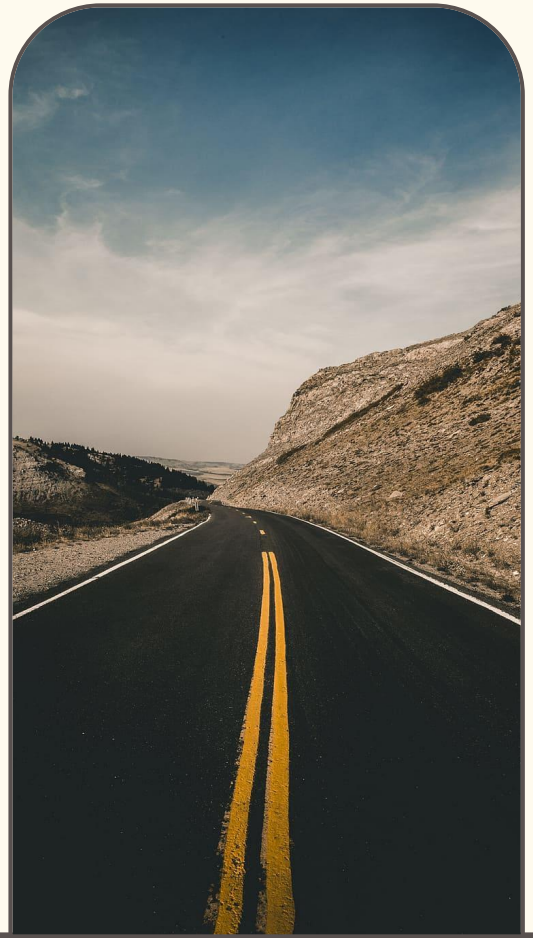
Proiectul consta în **20** fișiere în partea de backend structurate cu ajutorul building tool-ului **MAVEN.**

Acesta are în partea de frontend un total de **5** componente **ANGULAR,** un total de **18** fisiere de cod.

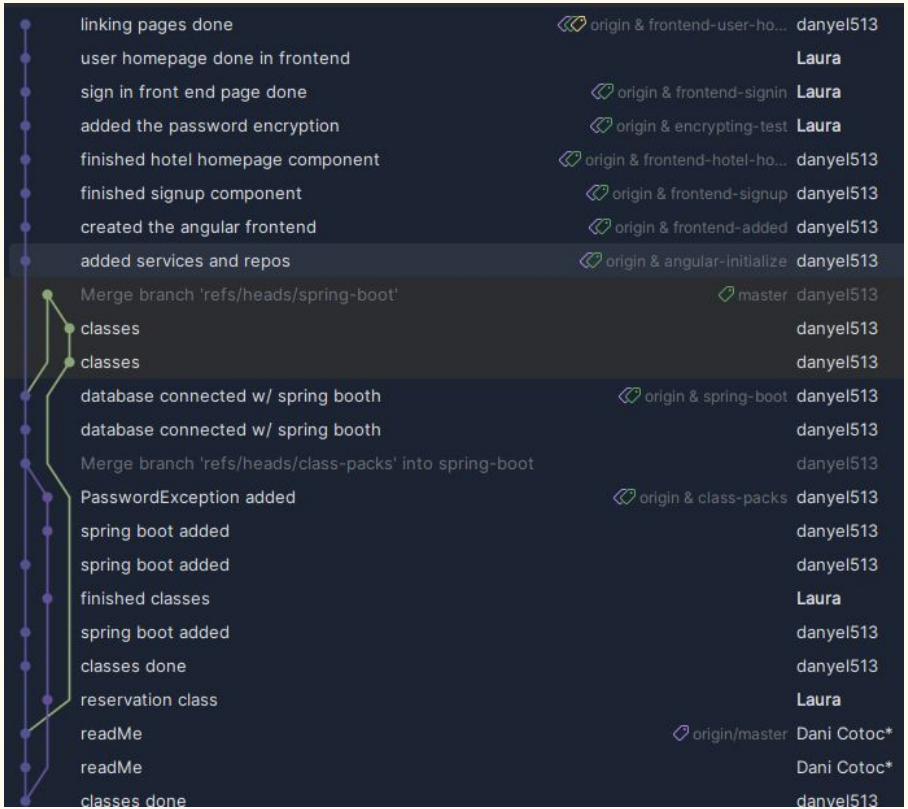
**TOTAL:** cateva mii de linii de cod, ~40 fișiere scrise, timp acumulat: 3 saptamani (4-5h /zi)

## *De ce asa mult?*

Cel mai mult a durat sa invatam cum funcționează unele tehnologii noi, și de ce sunt acestea mai bune decat ce știam sa facem și ni se părea relativ ușor.



# STRUCTURA COMMIT-URILOR




## STATISTICI OBTINUTE

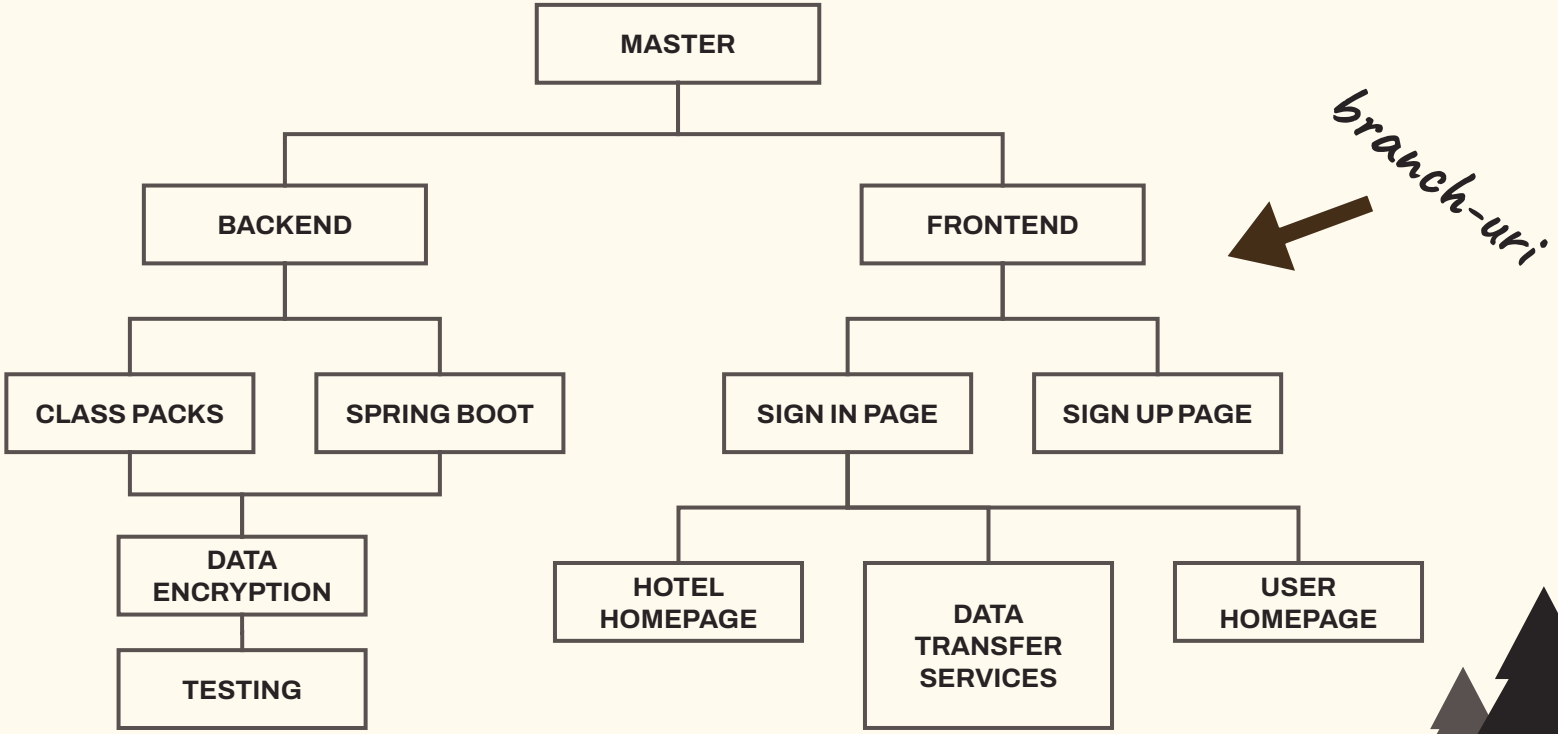
Numărul total de commit-uri:

25

S-a încercat simularea realizării unui proiect real. S-au creat branch-uri pentru diferite features.

Dani: 17
Laura: 8

# STRUCTURA COMMIT-URILOR



# IMBUNATATIRI VIITOARE

## FUNCTIONALITATI NOI

1. Sistem de validare a email-ului,
2. Sistem de schimbare a parolei în cazul uitării acesteia de către utilizatori,
3. Serviciu de mesagerie între utilizatori și hotele.

## FEEDBACK

1. Posibilitate de star rating hoteluri.
2. Feedback personalizat cu plusuri și minusuri.
3. Oferire recomandări personalizate bazate pe istoricul călătoriilor și pe AI

# THANKS

**Va multumim pentru atentia acordata!**