

---

# ***Conception Et Implémentation d'un système de management d'hôtel***

---

Réalisé par : Barakate Imane, Marir Oussama, Lafrougui Zouhair

# Cahier De Charges :

**Projet :** Application de réservation d'hôtel

**Version :** 1.0

**Date :** 2024-01-04

## 1. Introduction

Ce document définit le cahier des charges pour le développement d'une application de réservation d'hôtel. L'application visera à faciliter les processus de réservation pour les clients et la gestion des opérations hôtelières pour les administrateurs.

Objectif du Projet

Développer une application offrant une interface intuitive pour la réservation en ligne et la gestion hôtelière, divisée en deux parties principales :

- Partie Back-office : destinée aux administrateurs de l'hôtel.
- Partie Front-office : destinée aux clients de l'hôtel.

## 2. Spécifications Techniques

- Base de Données : Utilisation de MySQL ou PostgreSQL.
- Accès aux Données : JPA et Hibernate.
- Back-office : Application Desktop avec JavaFX.
- Front-office : Interface Web avec JSP/Servlet.
- Authentification : Sécurisation des accès.
- Gestion de Listes : Recherche, filtrage et tri.
- Communication : Envoi d'emails asynchrones via JavaMail.
- Versioning : Gestion de versions avec GitHub

## 3. Spécifications Fonctionnelles

Partie Back-office :

- Gestion des Employés/Utilisateurs
- Affichage, ajout, modification, suppression.
- Attribution de rôles et permissions.
- Gestion des Clients
- Opérations CRUD sur les données clients.
- Gestion des Chambres/Suites
- Gestion de la disponibilité et des caractéristiques.
- Gestion des Réservations
- Suivi complet des réservations.

#### Partie Front-office

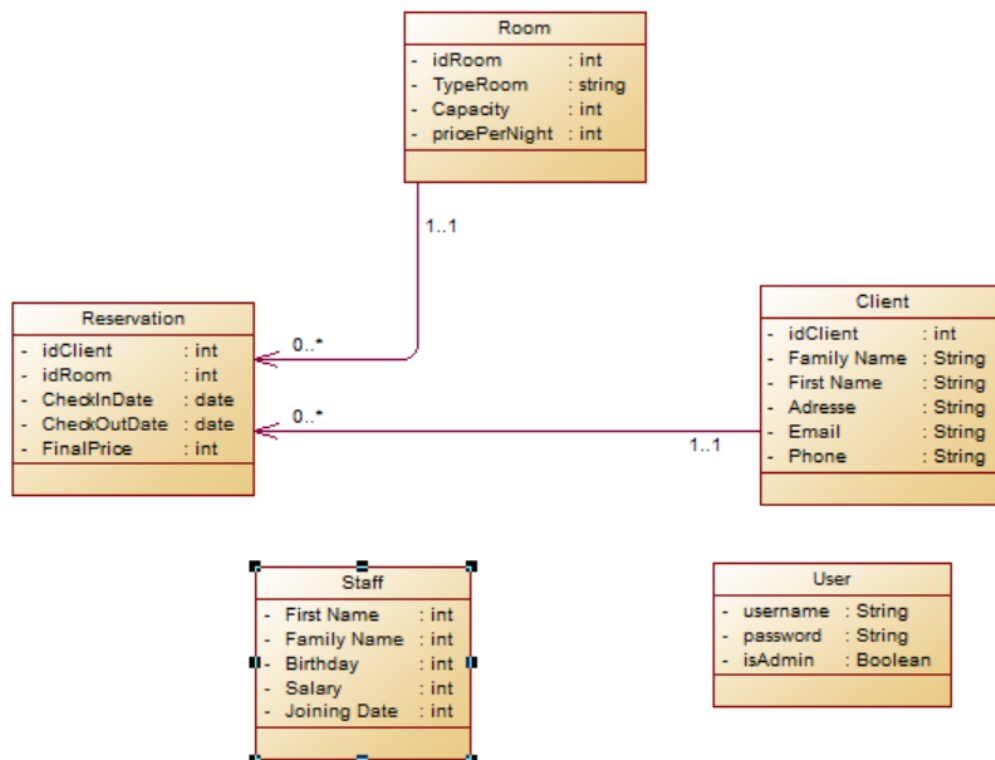
- Inscription des Clients
- Collecte des informations personnelles et de contact.
- Processus de Réservation
- Sélection de dates, type de chambre, informations de paiement.
- Confirmation par Email
- Envoi automatique de confirmations.
- Génération de bon de réservation PDF.

# Conception

## Introduction

La phase de conception pour l'application de réservation d'hôtel est cruciale pour assurer la réussite du projet. Cette section détaille l'architecture du système, les choix technologiques, et les modèles de conception pour une mise en œuvre efficace et sécurisée de l'application.

## Diagramme de Classes



## Architecture du Système

### 1. Architecture Globale :

- **Architecture Intégrée** : Front-office et back-office partagent une base de données commune.

- **Cohérence des Données** : Garantie par la gestion centralisée de la base de données.
2. **Architecture de Données** :
- **Base de données relationnelle unique** : Servant à la fois le front-office et le back-office.
  - **Modèle de Données Unifié** : Assurant l'intégrité et l'accessibilité des données.

## Choix Technologiques

### 1. Base de Données : MySQL

- **Popularité et Communauté** : MySQL est l'une des bases de données les plus utilisées, avec une large communauté et de nombreuses ressources.
- **Performance et Fiabilité** : Excellente performance pour les opérations CRUD (Create, Read, Update, Delete), essentielles pour une application de réservation.
- **Coût-Efficacité** : MySQL est une solution open-source, ce qui réduit les coûts liés aux licences.
- **Compatibilité** : Bonne compatibilité avec divers langages de programmation, en particulier Java.
- **Gestion Facile** : Interface utilisateur intuitive pour la gestion de la base de données.

### 2. Back-End : Java

- **Stabilité et Maturité** : Java est un langage bien établi, offrant une grande stabilité pour les applications d'entreprise.
- **Écosystème Riche** : Large éventail de bibliothèques et de frameworks disponibles.
- **Performance** : Java offre une bonne performance pour les applications à gros volume de données.
- **Portabilité** : Code indépendant de la plateforme, ce qui facilite le déploiement sur différents systèmes.

### 3. ORM : Hibernate

- **Gestion Facilitée des Données** : Simplifie l'interaction avec la base de données en utilisant des objets Java.
- **Portabilité de la Base de Données** : Permet une plus grande flexibilité en cas de changement de la base de données.
- **Performance** : Techniques de mise en cache et de lazy loading pour optimiser les performances.

### 4. Interface Back-Office : JavaFX

- **Richesse de l'Interface Utilisateur** : Permet de créer des interfaces riches et interactives pour le back-office.
- **Compatibilité avec Java** : Intégration naturelle avec le back-end Java.
- **Portabilité** : Applications facilement déployables sur différents systèmes d'exploitation.

### 5. *Front-End : HTML/CSS/JavaScript/Bootstrap*

- **Standards du Web** : Fondation universelle pour le développement web.
- **Flexibilité et Personnalisation** : Permet de créer des interfaces utilisateur personnalisées et réactives.
- **Compatibilité entre Navigateurs** : Prise en charge par tous les navigateurs modernes.

### 6. *Technologie Web : JSP/Servlets*

- **Intégration avec Java** : Complément naturel pour les applications Java côté serveur.
- **Simplicité et Efficacité** : Convient bien pour générer des vues dynamiques.
- **Gestion des Sessions** : Capacités intégrées pour gérer les sessions utilisateurs.

### 7. *Serveur : Apache Tomcat*

Tomcat est un choix populaire pour déployer des applications Java, en particulier celles utilisant JSP et Servlets.

- **Compatibilité avec Java** :
  - **Intégration Naturelle** : Tomcat est optimisé pour les applications Java, notamment celles utilisant JSP et Servlets.
  - **Prise en Charge des Spécifications JEE** : Supporte les spécifications Java Servlet et JSP.
- **Léger et Performant** :
  - **Faible Empreinte Mémoire** : Plus léger que des serveurs d'application complets, idéal pour des applications de taille moyenne.
  - **Démarrage Rapide** : Temps de démarrage rapide, augmentant l'efficacité du développement et des tests.
- **Large Utilisation et Communauté** :
  - **Support Étendu** : Large communauté de développeurs et abondance de documentation.
  - **Fiabilité Éprouvée** : Utilisé par de nombreuses entreprises et projets de taille variée.
- **Facilité de Configuration et de Déploiement** :
  - **Configuration Simple** : Fichiers de configuration clairs et bien documentés.
  - **Déploiement Aisé** : Déploiement direct de fichiers WAR sans nécessiter de configurations complexes.
- **Sécurité** :
  - **Mises à jour Régulières** : Corrections régulières des failles de sécurité.
  - **Personnalisation de la Sécurité** : Possibilité de configurer des politiques de sécurité avancées.

## 8. Gestion de dépendances : Maven

L'utilisation de Maven pour la gestion des dépendances dans le projet d'application de réservation d'hôtel présente plusieurs avantages significatifs :

- **Gestion Centralisée des Dépendances :**
  - **Simplification :** Maven simplifie la gestion des bibliothèques et des frameworks en déclarant les dépendances dans un fichier POM (Project Object Model).
  - **Résolution Automatique des Dépendances :** Télécharge et intègre automatiquement les dépendances nécessaires à partir du dépôt Maven central.
- **Standardisation du Projet :**
  - **Structure de Projet Uniforme :** Encourage une structure de projet standard, rendant les projets Maven faciles à comprendre et à gérer pour les nouveaux développeurs.
  - **Convention sur la Configuration :** Réduit la nécessité d'une configuration détaillée grâce à des conventions par défaut.
- **Intégration avec les IDE et les Outils de CI/CD :**
  - **Compatibilité :** Intégration transparente avec des IDE populaires comme Eclipse, IntelliJ IDEA et des systèmes de CI/CD comme Jenkins.
  - **Automatisation des Builds :** Facilite l'automatisation du processus de build, de test et de déploiement.
- **Cycle de Vie du Build :**
  - **Phases Définies :** Maven définit un cycle de vie de build clair (clean, compile, test, package, install, deploy) qui peut être personnalisé et étendu.
  - **Plugins :** Large éventail de plugins disponibles pour étendre les fonctionnalités de Maven, comme la génération de rapports, l'analyse de code, etc.
- **Gestion de Projets Multiples :**
  - **Projets Multi-Modules :** Gère efficacement les projets multi-modules, ce qui est essentiel pour les grandes applications avec plusieurs sous-projets.
- **Documentation et Communauté :**
  - **Documentation Abondante :** Large documentation disponible pour aider à résoudre les problèmes et à comprendre le fonctionnement de Maven.
  - **Communauté Active :** Communauté active pour le support et le partage des meilleures pratiques.
- **Réutilisabilité et Maintenance :**

- **POM Héritable** : Permet de définir une configuration parente dans un POM qui peut être héritée par des sous-projets, favorisant la réutilisabilité et facilitant la maintenance.

## Conclusion

La conception du projet d'application de réservation d'hôtel est guidée par l'objectif de créer une application performante, sécurisée, et facile à utiliser, tant pour les administrateurs que pour les clients. L'architecture choisie, les technologies et les pratiques de développement reflètent cet objectif, tout en offrant la flexibilité nécessaire pour adapter l'application à l'évolution des besoins et des technologies.

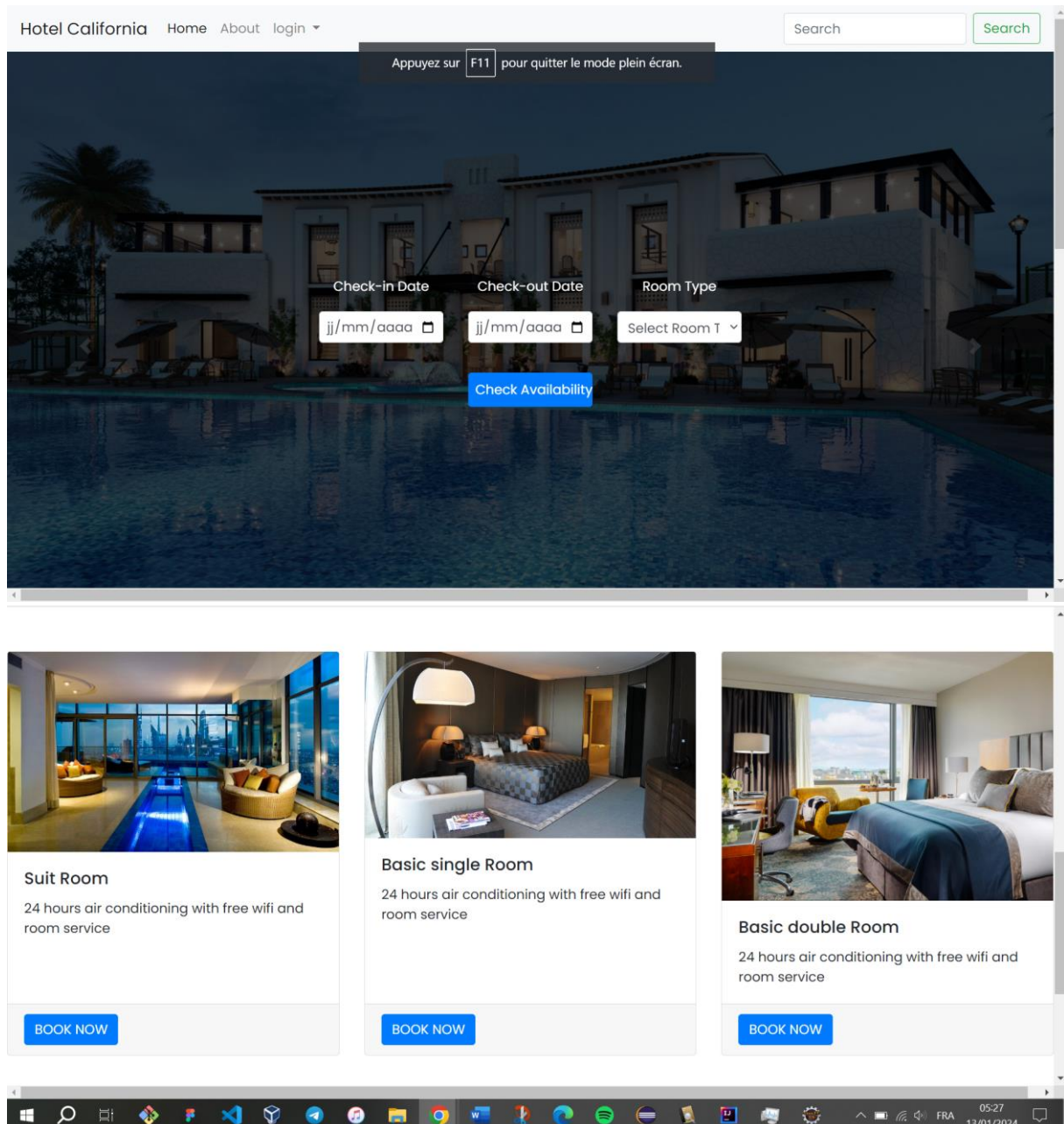


# Fonctionnalités et Mise En Œuvre

## 1. Partie Front Office :

La partie Front Office offre les fonctionnalités suivantes :

### 1. *Page d'accueil Et recherche de disponibilités:*



2. *Page d’affichage des chambres disponibles:*

## Available Room Details

Room ID: 1

Room Type: Single

Price Per Night: 1000 MAD

Breakfast: included

Feel free to proceed with the booking!

Reserve

## Available Room Details

Sorry, no available room matches your search criteria.

### 3. Saisie des informations du client et validation de la réservation

#### Reservation Form

First Name

Imane

Family Name

BARAKATE

Address

adresse imane

Email

ibarakate0@gmail.com

Phone Number

0629991607

Validate Reservation

### 4. Récapitulation de la Réservation avec possibilité de générer le bon de réservation en tant que pdf ou de l'envoyer par email :

## Reservation Recap

Room Number: 1

Check-in Date: 2024-01-12

Check-out Date: 2024-01-20

First Name: Imane

Family Name: Barakate

Address: adresse imane

Email: ibarakate0@gmail.com

Phone Number: 0629991607

Download PDF

Send Email

La conception et implémentation de ces fonctionnalités dans la partie Front Office ce sont faites sous plusieurs étapes décrites comme suit :

## 1. Configuration de l'Environnement de Développement

### Eclipse et Maven :

- **Installation d'Eclipse** : Eclipse IDE pour Java EE Developers est téléchargé et installé, offrant des outils intégrés pour JSP, Servlets, et autres technologies web.
- **Configuration de Maven** : Création d'un projet Maven dans Eclipse. Configuration du fichier **pom.xml** pour inclure les dépendances nécessaires (Servlet API, JSTL, JDBC driver pour MySQL, Hibernate).

```
<!-- https://mvnrepository.com/artifact/jakarta.servlet/jakarta.servlet-api -->
<dependency>
  <groupId>jakarta.servlet</groupId>
  <artifactId>jakarta.servlet-api</artifactId>
  <version>6.1.0-M1</version>
  <scope>provided</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/jakarta.persistence/jakarta.persistence-api -->
<dependency>
  <groupId>jakarta.persistence</groupId>
  <artifactId>jakarta.persistence-api</artifactId>
  <version>3.2.0-M1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/jakarta.servlet.jsp.jstl/jakarta.servlet.jsp.jstl-api -->
<dependency>
  <groupId>jakarta.servlet.jsp.jstl</groupId>
  <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
  <version>3.0.0</version>
</dependency>

  <!-- https://mvnrepository.com/artifact/com.mysql/mysql-connector-j -->
  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>8.0.31</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
```

## 2. Création et Gestion des Pages JSP

### Structure et Développement JSP :

- **Structure de Projet** : Le projet est organisé en respectant la structure standard de projets web Java, avec un répertoire **WEBAPP** pour les fichiers de configuration et de sécurité, un dossier pour les pages JSP, et des dossiers pour les ressources statiques comme CSS et JavaScript.



- 
- **Développement JSP** : Les pages JSP sont créées pour différentes interfaces utilisateur telles que la page de réservation, la gestion des profils, et la visualisation des chambres disponibles. Les JSP utilisent des tags JSTL pour afficher les données dynamiquement et gérer la logique côté serveur de manière élégante.

### 3. Conception des DAO (Data Access Objects)

#### DAOs pour l'Accès aux Données :

- **Création de Classes DAO** : Des classes DAO sont conçues pour chaque entité principale, comme **RoomDAO** et **ReservationDAO**. Ces classes encapsulent toutes les interactions avec la base de données, offrant une couche d'abstraction pour les opérations CRUD.

```

1 package com.jsp.dao;
2
3 import java.util.List;
4
5 public class RoomDao {
6     private SessionFactory sessionFactory;
7
8     // Constructor that takes a Hibernate SessionFactory
9     public RoomDao(SessionFactory sessionFactory) {
10         this.sessionFactory = sessionFactory;
11     }
12
13     // Method to add a new room
14     public void addRoom(Room room) {
15         try (Session session = sessionFactory.openSession()) {
16             Transaction transaction = session.beginTransaction();
17             session.save(room);
18             transaction.commit();
19         }
20     }
21
22     // Method to get all rooms
23     public List<Room> getAllRooms() {
24         try (Session session = sessionFactory.openSession()) {
25             return session.createQuery("FROM Room", Room.class).list();
26         }
27     }
28
29     public Room getFirstAvailableRoom(String checkinDate, String checkoutDate, String
30         try (Session session = sessionFactory.openSession()) {
31             Transaction transaction = session.beginTransaction();

```

- **Configuration des DAOs** : Les DAOs sont configurées pour communiquer avec la base de données MySQL, en utilisant les informations de connexion définies dans Maven ou Eclipse.

#### 4. Intégration de Hibernate pour la Gestion des Entités

##### Mapping et Gestion des Entités :

- **Définition des Entités** : Des classes d'entité comme **Room** et **Reservation** sont définies en Java, avec des annotations Hibernate pour le mapping objet-relationnel.

```

1 package com.jsp.dto;
2
3 import jakarta.persistence.Column;
4
5 @Entity
6 @Table(name = "rooms")
7 public class Room {
8     @Id
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10    @Column(name = "room_id")
11    private int roomId;
12
13    @Column(name = "type")
14    private String type;
15
16    @Column(name = "price_per_night")
17    private int pricePerNight;
18
19    public Room(int roomId, String type, int pricePerNight) {
20        this.roomId = roomId;
21        this.type = type;
22        this.pricePerNight = pricePerNight;
23    }
24
25    // Getters and setters
26
27    public int getRoomId() {
28        return roomId;
29    }
30
31    public void setRoomId(int roomId) {
32        this.roomId = roomId;
33    }
34 }

```

- **Configuration Hibernate** : La configuration Hibernate est établie pour gérer les sessions de base de données, assurant une gestion efficace et performante des transactions et des requêtes.

## 5. Utilisation et Configuration de Servlets

### Gestion des Requêtes et Réponses :

- **Création de Servlets** : Des Servlets spécifiques sont développées pour traiter les requêtes HTTP, interagir avec les DAOs, et retourner des réponses appropriées. Par exemple, `AvailabilityServlet` vérifie les disponibilités selon les spécificités demandées.

```

package Servlets;

import java.io.IOException;

public class AvailabilityServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    private SessionFactory sessionFactory;

    @Override
    public void init() throws ServletException {
        // Initialize Hibernate SessionFactory (You need to set up Hibernate configuration)
        sessionFactory = HibernateUtil.getSessionFactory();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // Retrieve form parameters
        String checkinDate = request.getParameter("checkinDate");
        String checkoutDate = request.getParameter("checkoutDate");
        String roomType = request.getParameter("roomType");

        // Query Hibernate to check for available rooms
        RoomDao roomDao = new RoomDao(sessionFactory);
        Room availableRoom = roomDao.getFirstAvailableRoom(checkinDate, checkoutDate, roomType);

        if (availableRoom != null) {
            // Forward the request to the JSP page that will display the available room
            request.setAttribute("availability", availableRoom);
            RequestDispatcher dispatcher = request.getRequestDispatcher("/availability.jsp");
            dispatcher.forward(request, response);
        }
    }
}

```

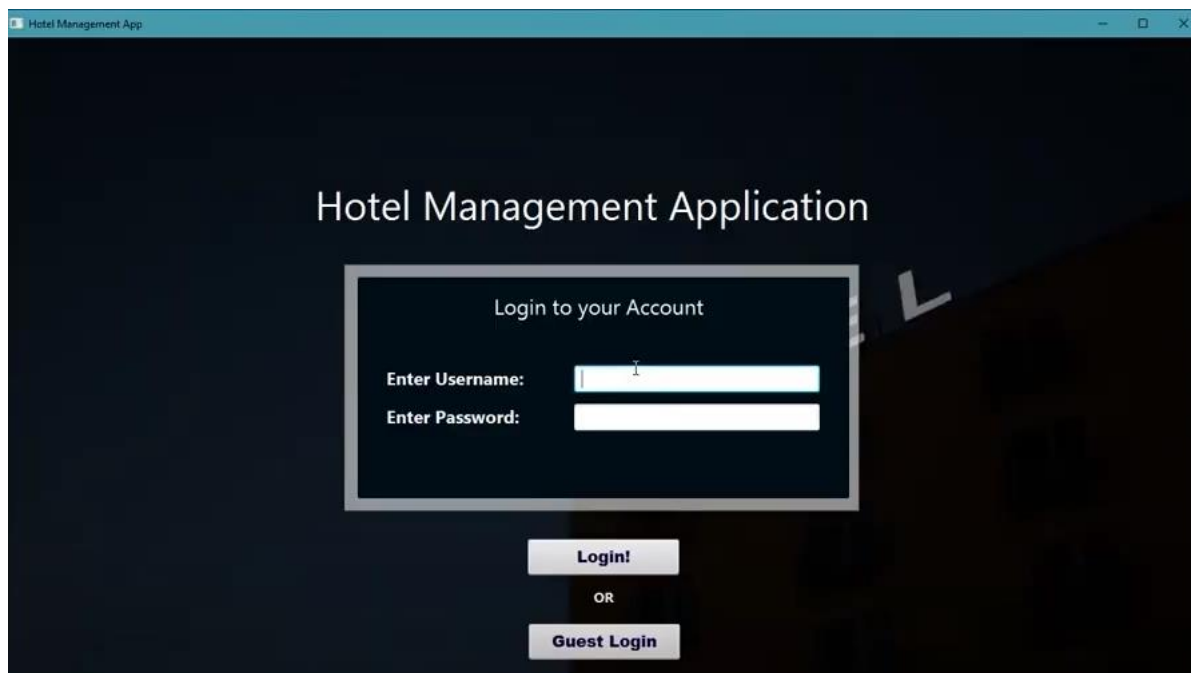
- **Flux de Travail Servlet-JSP** : Les Servlets récupèrent, traitent et transmettent les données aux pages JSP, facilitant un modèle MVC simplifié et une séparation claire des préoccupations.

## 2. Partie Back Office :

La partie Back Office offre les fonctionnalités suivantes :



## 1. Page de Login



The screenshot shows a web browser window titled "Hotel Management App". The main heading is "Hotel Management Application". Below it, a login form is centered. The form has a title "Login to your Account". It contains two input fields: "Enter Username:" and "Enter Password:". Below the input fields are three buttons: "Login!", "OR", and "Guest Login". The background of the page is dark with a faint image of a building.

Hotel Management App

# Hotel Management Application

Login to your Account

Enter Username:

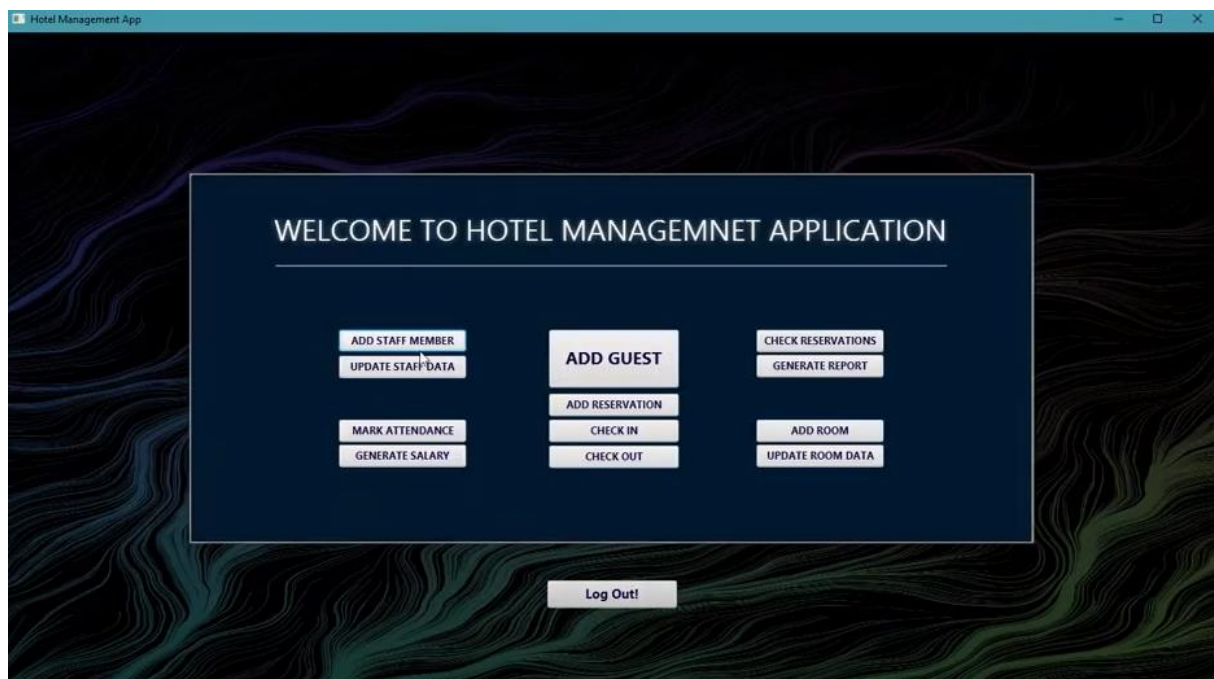
Enter Password:

Login!

OR

Guest Login

## 2. Page d'accueil



The screenshot shows a web browser window titled "Hotel Management App". The main heading is "WELCOME TO HOTEL MANAGEMNET APPLICATION". Below the heading, there are several buttons arranged in a grid. The buttons are: "ADD STAFF MEMBER", "UPDATE STAFF DATA", "ADD GUEST", "CHECK RESERVATIONS", "GENERATE REPORT", "MARK ATTENDANCE", "GENERATE SALARY", "ADD RESERVATION", "CHECK IN", "CHECK OUT", "ADD ROOM", and "UPDATE ROOM DATA". At the bottom center, there is a "Log Out!" button. The background of the page is dark with a wavy, abstract pattern.

Hotel Management App

# WELCOME TO HOTEL MANAGEMNET APPLICATION

ADD STAFF MEMBER  
UPDATE STAFF DATA

ADD GUEST

CHECK RESERVATIONS  
GENERATE REPORT

MARK ATTENDANCE  
GENERATE SALARY

ADD RESERVATION  
CHECK IN  
CHECK OUT

ADD ROOM  
UPDATE ROOM DATA

Log Out!

### 3. Ajout d'employés et modification des données relatives à eux

Hotel Management App

Back

## ADD STAFF MENU

Full Name:	<input type="text"/>	Staff Number:	<input type="text"/>
Contact No.	<input type="text"/>	Address:	<input type="text"/>
Email Address:	<input type="text"/>	City:	<input type="text"/>
CNIC No.	<input type="text"/>	Date Of Birth:	<input type="text"/>
Designation	<input type="text"/>	Date of Joining:	<input type="text"/>

Save Data!

Hotel Management App

Back

## UPDATE STAFF MENU

Staff Number:

Full Name:	<input type="text"/>	Address:	<input type="text"/>
Contact No.	<input type="text"/>	City:	<input type="text"/>
Email Address:	<input type="text"/>	Date Of Birth:	<input type="text"/>
CNIC No.	<input type="text"/>	Date of Joining:	<input type="text"/>
Designation	<input type="text"/>		

Update Data!

#### 4. Ajout des clients

The screenshot shows a web application window titled "Hotel Management App". The main heading is "ADD NEW GUEST MENU". In the top left corner, there is a "Back" button. The form contains the following fields:

Full Name:	<input type="text"/>	Guest Number:	<input type="text"/>
Contact No.	<input type="text"/>	Address:	<input type="text"/>
Email Address:	<input type="text"/>	City:	<input type="text"/>
CNIC No.	<input type="text"/>	Date Of Birth:	<input type="text"/>

At the bottom center of the form, there is a "Save Data!" button.

#### 5. Ajout des réservations

The screenshot shows a web application window titled "Hotel Management App". The main heading is "GUEST MENU". The form contains the following fields:

Guest Number:	<input type="text"/>
Room No:	<input type="text"/>
Booking Date:	<input type="text"/>
Reservation No.	<input type="text"/>

At the bottom center of the form, there is a "Reserve a Room" button.

## 6. Ajout de chambres et modification des informations relatives à elles

The image displays two screenshots of a web application titled "Hotel Management App".

The top screenshot shows the "ADD NEW ROOM MENU" screen. It features a "Back" button in the top left corner. The main heading is "ADD NEW ROOM MENU". Below this, there is a form with the following fields and options:

- Room No.:
- Floor No.:
- No. of Beds:
- Is Reserved: ☐ Yes ☐ No
- Room Type:
- Air Conditioning: ☐ Yes ☐ No
- MiniBar: ☐ Yes ☐ No
- Mini Kitchen: ☐ Yes ☐ No

A "Save Data!" button is located at the bottom center of the form.

The bottom screenshot shows the "UPDATE ROOM MENU" screen. It also features a "Back" button in the top left corner. The main heading is "UPDATE ROOM MENU". Below this, there is a form with the following fields and options:

- Room No.:
- Floor No.:
- No. of Beds:
- Mini Kitchen: ☐ Yes ☐ No
- Room Type:
- Air Conditioning: ☐ Yes ☐ No
- MiniBar: ☐ Yes ☐ No

A "Fetch Data!" button is located to the right of the Room No. field. An "Update Data" button is located at the bottom right of the form.

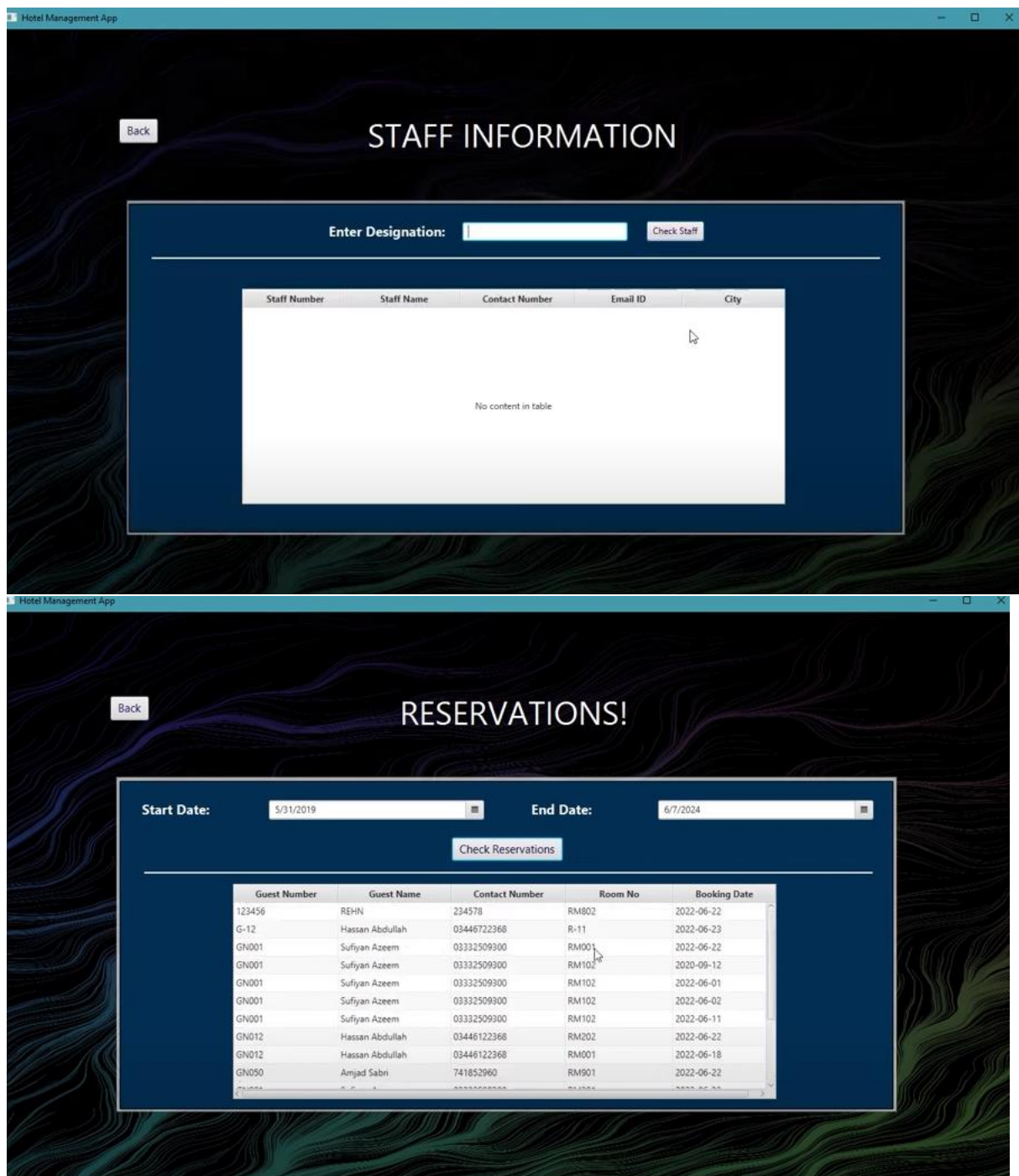
## 7. *CheckIn et CheckOut des clients*

The image displays two screenshots of a 'Hotel Management App' interface.

The top screenshot shows the 'CHECK IN MENU' screen. It features a 'Back' button in the top left corner. The main area contains two input fields: 'Reservation ID' with the text 'RN' entered, and 'Arrival Date:'. Below these fields is a 'Confirm Check In' button.

The bottom screenshot shows the 'WELCOME TO HOTEL MANAGEMNET APPLICATION' screen. It has a dark background with a wavy pattern. The main area contains a grid of buttons: 'ADD STAFF MEMBER', 'UPDATE STAFF DATA', 'MARK ATTENDANCE', 'GENERATE SALARY', 'CHECK RESERVATIONS', 'GENERATE REPORT', 'ADD ROOM', and 'UPDATE ROOM DATA'. A 'CHECK OUT' button is located at the bottom center. A 'Log Out!' button is at the bottom right. An 'Input' dialog box is open in the center, prompting the user to 'Enter Reservation No.' with an 'OK' and 'Cancel' button.

## 8. Recherche des réservations et clients et leur gestion :



La conception et implémentation de ces fonctionnalités dans la partie Front Office ce sont faites sous plusieurs étapes décrites comme suit :

### Configuration de l'Environnement de Développement

#### 1. NetBeans et JavaFX :

- **Installation de NetBeans** : Téléchargement et installation de NetBeans IDE, qui supporte nativement JavaFX.



- **Configuration JavaFX** : Intégration du kit de développement JavaFX dans NetBeans pour le développement d'interfaces utilisateur graphiques.

### **Création d'Interfaces Utilisateur avec JavaFX**

#### **1. Conception d'UI JavaFX :**

- **Structure de Projet** : Création d'une structure de projet claire avec des packages séparés pour les interfaces utilisateur, la logique métier, et les accès à la base de données.
- **Développement de Vues** : Utilisation de Scene Builder pour créer des vues JavaFX, telles que les tableaux de bord de gestion, les formulaires de réservation et les interfaces de gestion des utilisateurs.

#### **2. Développement des Contrôleurs JavaFX :**

- **Création de Contrôleurs** : Développement de contrôleurs Java pour gérer les interactions de l'utilisateur avec les éléments de l'interface utilisateur.

```
public class LoginController implements Initializable {

    //=====
    public static Stage window;
    public static JFXDecorator decorator;
    public static User user;
    //=====

    @FXML
    private JFXButton ButtonLogin;
    @FXML
    private AnchorPane AnchorPane;
    @FXML
    private JFXTextField UserNameField;
    @FXML
    private JFXPasswordField PasswordField;
    @FXML
    private ImageView key_pic_Login_Btn;
    //=====

    @Override
    public void initialize(URL url, ResourceBundle rb) {

    }
}
```

- 
- **Logique Métier** : Intégration de la logique métier dans les contrôleurs pour traiter les données de l'interface utilisateur.

```

try {
    if (User.isUserValid(user)) {
        goToHomePage(event);
    } else {
        try {
            System.out.println("Executing popup window");
            new switchScreen().popUp(event, "/login/ErrorPopUp.fxml", 370, 250, "", "/img/Error01.png");
        } catch (Exception ex) {
            System.out.println("error on popUp window");
            System.out.println(ex);
        }
    }
} catch (Exception e) {
    System.out.println("isUserValid Exception");
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Error");
    alert.setHeaderText("Connection Error");
    alert.setContentText("Error while trying to connect to database ..");
    ((Stage) alert.getDialogPane().getScene().getWindow()).getIcons().add(new Image("/img/Error01.png"));
    alert.show();
    return;
}

```

•

## ***Intégration de la Base de Données et Hibernate***

### **1. Configuration de Hibernate :**

- **Configuration de Session Hibernate** : Mise en place de la session Hibernate pour gérer la connexion à la base de données et la persistance des données.

### **2. DAOs pour le Back-Office :**

- **Création de DAOs** : Développement de classes DAO pour interagir avec la base de données, utilisant Hibernate pour le mappage objet-relationnel.

## ***Création de Modèles et de Services***

### **1. Modélisation des Données :**

- **Définition des Modèles** : Création de classes modèles représentant les entités de l'application, comme les réservations, les chambres, etc.

### **2. Développement de Services :**

- **Services Métier** : Création de services pour encapsuler la logique métier, communiquant entre les DAOs et les contrôleurs JavaFX.