Project Specification Document

- Requirements
  - Background
  - Requirements
    - Must Have
    - Should Have
    - Could Have
    - Won't Have
  - Method
    - Architettura
    - Componenti del sistema
    - Design delle Componenti
    - Schemi di database
    - Algoritmi
- Method
  - Architettura
  - Componenti
    - HOTELIERCustomerClient
    - HOTELIERServer
      - Design delle Componenti
      - Schemi di Database
      - Algoritmi
      - Implementazione
- Implementation
  - Passi per l'Implementazione
    - 1. Preparazione dell'Ambiente di Sviluppo:
    - 2. Definizione delle Strutture Dati:
    - 3. Implementazione del Server:
    - 4. Implementazione del Client:
    - 5. Configurazione delle Connessioni di Rete:
    - 6. Persistenza dei Dati:
    - 7. Test e Debugging:
  - Struttura del codice
    - Classe User
    - Classe Hotel
    - Classe HOTELIERServer
    - Classe HOTELIERCustomerClient

        - Configurazione delle connessioni
            - 1. TCP Connection:
            - 2. UDP Multicast:
        - Persistenza dei Dati
    - Mileston
        - Fase 1: Preparazione dell'Ambiente di Sviluppo
        - Fase 2: Definizione delle Strutture Dati
        - Fase 3: Implementazione del Server
        - Fase 4: Implementazione del Client
        - Fase 5: Configurazione delle Connessioni di Rete
        - Fase 6: Persistenza dei Dati
        - Fase 7: Test e Debugging
        - Fase 8: Documentazione e Consegna
    - Gathering Results
        - Valutazione dei Requisiti
            - 1. Test di Funzionalità:
            - 2. Test di Usabilità:
            - 3. Test di Performance:
            - 4. Test di Robustezza:
        - Analisi delle Prestazioni Post-Produzione
            - 1. Disponibilità del Sistema:
            - 2. Utilizzo delle Risorse:
            - 3. Feedback degli Utenti:
            - 4. Manutenzione e Aggiornamenti:

Requirements

Background

Il progetto HOTELIER è nato come progetto di fine corso per l'anno accademico 2023/24 del corso "Modulo Laboratorio III". L'idea di base è quella di implementare un servizio che si ispira a TripAdvisor, un popolare sito di recensioni per hotel, ristoranti ed altre attività legate ai viaggi. HOTELIER mira a fornire una versione semplificata di alcune delle principali funzionalità di TripAdvisor, concentrandosi esclusivamente sugli hotel situati nelle città capoluogo delle 20 regioni italiane.

TripAdvisor permette agli utenti di registrarsi e recensire varie strutture, consultare recensioni esistenti e visualizzare classifiche basate sulle recensioni degli utenti. Le recensioni includono punteggi sintetici per diverse categorie come posizione, pulizia, servizi e rapporto qualità/prezzo. Inoltre, TripAdvisor utilizza un algoritmo di ranking che considera la qualità, quantità e attualità delle recensioni.

HOTELIER si propone di implementare funzionalità analoghe in maniera semplificata, offrendo un servizio di recensione e consultazione specifico per gli hotel, senza supporto per immagini e recensioni testuali, ma mantenendo il sistema di punteggi e ranking.

Requirements

Must Have

- Gli utenti devono poter registrarsi al servizio fornendo uno username e una password.
- Gli utenti devono poter effettuare il login per accedere al servizio.
- Gli utenti devono poter effettuare il logout dal servizio.
- Gli utenti devono poter cercare un hotel specifico per nome e città.
- Gli utenti devono poter cercare tutti gli hotel di una specifica città, ordinati secondo il ranking.
- Gli utenti devono poter inserire una recensione per un hotel, fornendo un punteggio globale e punteggi specifici per posizione, pulizia, servizi e rapporto qualità/prezzo.
- Il sistema deve calcolare il ranking locale degli hotel in base a qualità, quantità e attualità delle recensioni.
- Gli utenti devono poter visualizzare il proprio distintivo di esperienza.
- Il server deve memorizzare le informazioni relative a tutti gli utenti e a tutti gli hotel.
- Il server deve aggiornare periodicamente il ranking degli hotel di ogni città.

Should Have

- Gli utenti devono poter ricevere notifiche quando un ranking locale cambia, registrandosi al servizio di notifica.
- Il server deve inviare notifiche agli utenti registrati ogni volta che cambia il primo classificato in un ranking locale.

Could Have

- Il client potrebbe fornire un'interfaccia grafica opzionale oltre alla CLI (Command Line Interface).
- Il sistema potrebbe supportare la registrazione di interessi per ricevere notifiche su aggiornamenti di specifici ranking locali.

Won't Have

- Gli utenti non potranno caricare foto degli hotel.
- Gli utenti non potranno inserire recensioni testuali per gli hotel.
- Non sarà possibile registrare hotel al di fuori delle città capoluogo delle 20 regioni italiane.

Method

Architettura

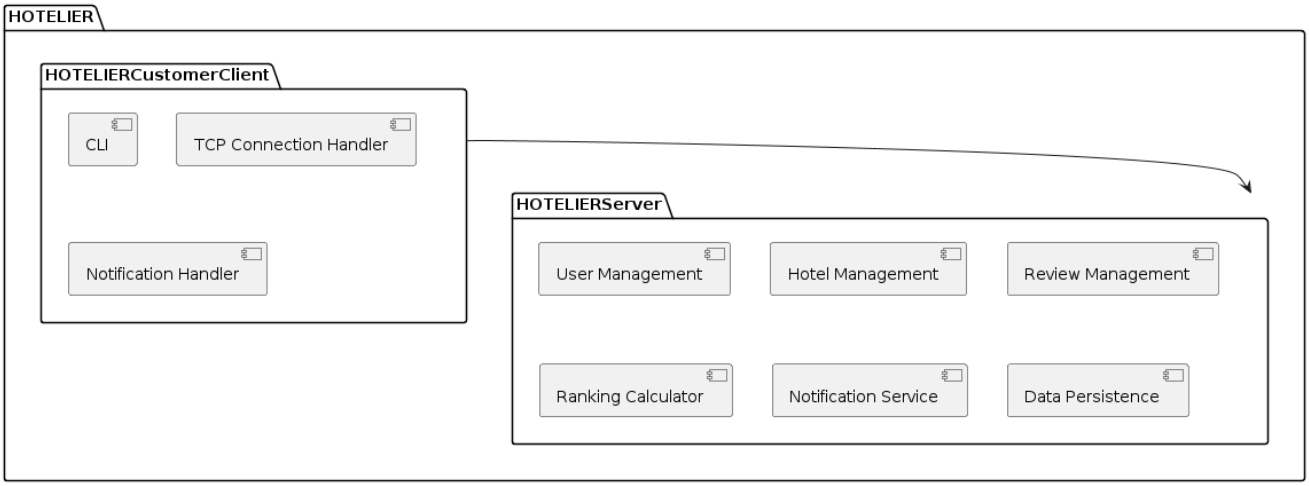
L'architettura del sistema HOTELIER seguirà il modello client-server. Saranno presenti due componenti principali:

1. **HOTELIERCustomerClient**: un client che gestisce l'interazione con l'utente tramite una Command Line Interface (CLI).
2. **HOTELIERServer**: un server che gestisce le operazioni richieste dai client e mantiene i dati degli utenti e degli hotel.

Componenti del sistema

1. **HOTELIERCustomerClient**
   - **CLI**: Interfaccia a linea di comando per l'interazione con l'utente.
   - **TCP Connection Handler**: Gestisce le connessioni TCP con il server per le operazioni di registrazione, login, logout, ricerca e inserimento recensioni.
   - **Notification Handler**: Gestisce le notifiche inviate dal server in caso di aggiornamento dei ranking locali.
2. **HOTELIERServer**
   - **User Management**: Gestisce la registrazione, login e logout degli utenti.
   - **Hotel Management**: Gestisce le informazioni sugli hotel e le recensioni.
   - **Review Management**: Riceve e memorizza le recensioni degli utenti, aggiornando il ranking degli hotel.
   - **Ranking Calculator**: Calcola periodicamente il ranking degli hotel basandosi su qualità, quantità e attualità delle recensioni.
   - **Notification Service**: Invia notifiche agli utenti registrati quando i ranking locali cambiano.
   - **Data Persistence**: Memorizza periodicamente le informazioni degli utenti e degli hotel in formato JSON.

Design delle Componenti



Schemi di database

```
{
  "users": [
    {
      "username": "john_doe",
      "password": "hashed_password",
      "reviews": 5,
      "badge": "Contributore"
    }
  ],
  "hotels": [
    {
      "name": "Hotel Roma",
      "city": "Rome",
      "globalScore": 4.5,
      "scores": {
        "position": 4,
        "cleanliness": 5,
        "service": 4,
        "price": 4
      },
      "reviews": 10,
      "rank": 1
    }
  ]
}
```

Algoritmi

1. **Calcolo del Ranking**: L'algoritmo di ranking considera la qualità (media dei punteggi globali), la quantità (numero di recensioni) e l'attualità (peso maggiore alle recensioni più recenti). Un esempio di formula potrebbe essere:

$$\text{Ranking Score} = (\text{Qualità} \times 0.5) + (\text{Quantità} \times 0.3) + (\text{Attualità} \times 0.2)$$

2. **Gestione delle Notifiche**: Utilizzeremo il protocollo UDP e multicast per inviare notifiche agli utenti loggati quando cambia il primo classificato di un ranking locale.

Method

Architettura

L'architettura del sistema segue un modello client-server, in cui il client interagisce con il server per eseguire operazioni di registrazione, login, inserimento di recensioni, ricerca di hotel e visualizzazione di distintivi. Il server gestisce tutte le operazioni richieste dai client, memorizza i dati degli utenti e degli hotel, calcola i ranking locali e invia notifiche agli utenti.

Componenti

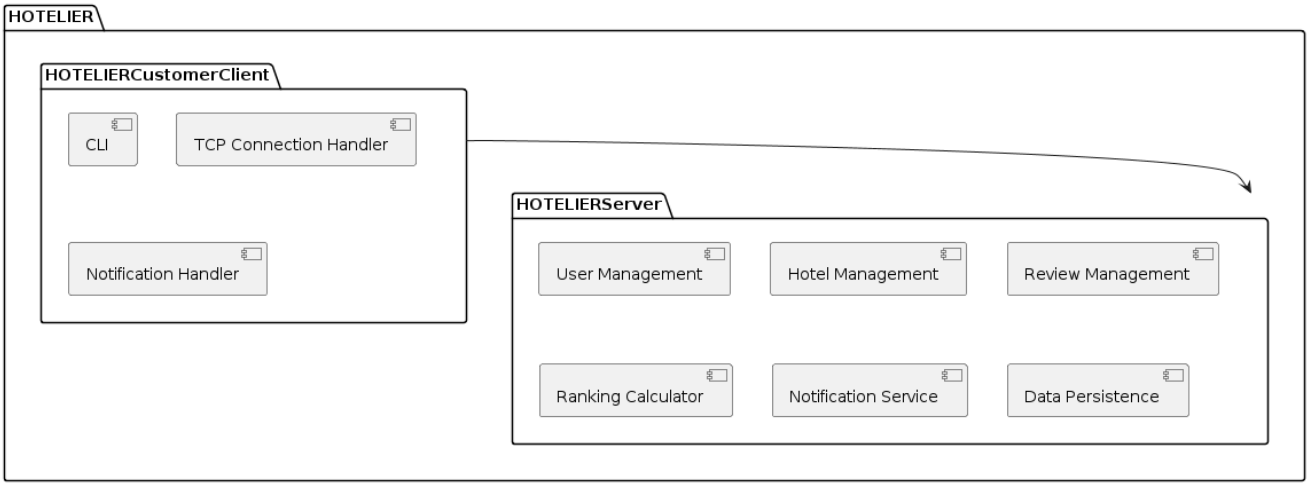
HOTELIERCustomerClient

- **CLI**: Gestisce l'interazione con l'utente tramite una linea di comando.
- **TCP Connection Handler**: Gestisce le connessioni TCP con il server per le operazioni di registrazione, login, logout, ricerca e inserimento recensioni.
- **Notification Handler**: Gestisce le notifiche inviate dal server in caso di aggiornamento dei ranking locali.

HOTELIERServer

- **User Management**: Gestisce la registrazione, login e logout degli utenti.
- **Hotel Management**: Gestisce le informazioni sugli hotel e le recensioni.
- **Review Management**: Riceve e memorizza le recensioni degli utenti, aggiornando il ranking degli hotel.
- **Ranking Calculator**: Calcola periodicamente il ranking degli hotel basandosi su qualità, quantità e attualità delle recensioni.
- **Notification Service**: Invia notifiche agli utenti registrati quando i ranking locali cambiano.
- **Data Persistence**: Memorizza periodicamente le informazioni degli utenti e degli hotel in formato JSON.

Design delle Componenti



Schemi di Database

Utilizzeremo strutture dati in formato JSON per memorizzare le informazioni degli utenti e degli hotel.

- **Struttura dati per gli utenti**:

```
{
    "username": "john_doe",
    "password": "hashed_password",
    "reviews": 5,
    "badge": "Contributore"
}
```

- **Struttura dati per gli hotel**:

```
{
    "name": "Hotel Roma",
    "city": "Rome",
    "globalScore": 4.5,
    "scores": {
        "position": 4,
        "cleanliness": 5,
        "service": 4,
        "price": 4
    },
    "reviews": 10,
    "rank": 1
}
```

Algoritmi

1. **Calcolo del Ranking**: L'algoritmo di ranking considera la qualità (media dei punteggi globali), la quantità (numero di recensioni) e l'attualità (peso maggiore alle recensioni

più recenti). Un esempio di formula potrebbe essere:

```
double calculateRankingScore(double quality, int quantity, double recencyWeight)
{
    return (quality * 0.5) + (quantity * 0.3) + (recencyWeight * 0.2);
}
```

2. **Gestione delle Notifiche**: Utilizzeremo il protocollo UDP e multicast per inviare notifiche agli utenti loggati quando cambia il primo classificato di un ranking locale.

Implementazione

Ecco i passi per implementare le funzionalità descritte:

1. **User Management**:
   - Implementare le operazioni di registrazione, login e logout gestendo le richieste dei client tramite connessioni TCP.
   - Memorizzare le informazioni degli utenti in un file JSON.
2. **Hotel Management**:
   - Caricare le informazioni sugli hotel da un file JSON all'avvio del server.
   - Gestire le operazioni di ricerca degli hotel richieste dai client.
3. **Review Management**:
   - Ricevere e memorizzare le recensioni degli utenti.
   - Aggiornare i punteggi degli hotel e ricalcolare i ranking locali.
4. **Ranking Calculator**:
   - Calcolare periodicamente il ranking degli hotel utilizzando l'algoritmo descritto.
   - Verificare se c'è un cambiamento nella prima posizione e inviare notifiche agli utenti loggati se necessario.
5. **Notification Service**:
   - Inviare notifiche agli utenti loggati tramite multicast UDP quando il primo classificato di un ranking locale cambia.
6. **Data Persistence**:
   - Memorizzare periodicamente le informazioni degli utenti e degli hotel in formato JSON.

Implementation

Passi per l'Implementazione

1. Preparazione dell'Ambiente di Sviluppo**:

- Installare il JDK (Java Development Kit).
- Configurare un ambiente di sviluppo come IntelliJ IDEA o Eclipse.

- Creare un progetto Java e impostare la struttura delle directory.

2. Definizione delle Strutture Dati:

- Creare classi per rappresentare utenti e hotel.
- Utilizzare librerie come Gson per serializzare e deserializzare le strutture dati in formato JSON.

3. Implementazione del Server:

- Creare una classe `HOTELIERServer` che contenga i seguenti componenti:
  - **User Management**: Metodi per registrazione, login e logout degli utenti.
  - **Hotel Management**: Metodi per gestire le informazioni sugli hotel e le recensioni.
  - **Review Management**: Metodi per ricevere e memorizzare le recensioni degli utenti.
  - **Ranking Calculator**: Metodi per calcolare periodicamente il ranking degli hotel.
  - **Notification Service**: Metodi per inviare notifiche agli utenti loggati.
  - **Data Persistence**: Metodi per memorizzare periodicamente le informazioni in JSON.

4. Implementazione del Client:

- Creare una classe `HOTELIERCustomerClient` che contenga i seguenti componenti:
  - **CLI**: Interfaccia a linea di comando per l'interazione con l'utente.
  - **TCP Connection Handler**: Gestisce le connessioni TCP con il server per le operazioni di registrazione, login, logout, ricerca e inserimento recensioni.
  - **Notification Handler**: Gestisce le notifiche inviate dal server.

5. Configurazione delle Connessioni di Rete:

- Utilizzare socket TCP per la comunicazione tra client e server.
- Implementare un server multithreaded per gestire più connessioni client contemporaneamente.
- Utilizzare UDP e multicast per inviare notifiche agli utenti loggati.

6. Persistenza dei Dati:

- Memorizzare periodicamente le informazioni degli utenti e degli hotel in file JSON.
- Caricare le informazioni dagli stessi file all'avvio del server.

7. Test e Debugging:

- Scrivere unit test per verificare il corretto funzionamento delle varie componenti.
- Utilizzare strumenti di debugging per individuare e risolvere eventuali problemi.

Struttura del codice

Classe `User`

```
public class User {
    private String username;
    private String password;
    private int reviews;
    private String badge;

    // Getters e setters
}
```

Classe `Hotel`

```
public class Hotel {
    private String name;
    private String city;
    private double globalScore;
    private Map<String, Integer> scores;
    private int reviews;
    private int rank;

    // Getters e setters
}
```

Classe `HOTELIERServer`

```
public class HOTELIERServer {
    private Map<String, User> users;
    private Map<String, Hotel> hotels;

    // Metodi per gestione utenti
    public synchronized String register(String username, String password) {
        // Implementazione
    }

    public synchronized String login(String username, String password) {
        // Implementazione
    }

    public synchronized String logout(String username) {
        // Implementazione
    }

    // Metodi per gestione hotel
```

```
    public List<Hotel> searchHotel(String nomeHotel, String città) {
        // Implementazione
    }

    public List<Hotel> searchAllHotels(String città) {
        // Implementazione
    }

    // Metodi per gestione recensioni
    public synchronized String insertReview(String nomeHotel, String nomeCittà,
double globalScore, Map<String, Integer> singleScores) {
        // Implementazione
    }

    // Metodi per calcolo ranking
    public void updateRankings() {
        // Implementazione
    }

    // Metodi per gestione notifiche
    public void sendNotifications() {
        // Implementazione
    }

    // Metodi per persistenza dati
    public void saveData() {
        // Implementazione
    }

    public void loadData() {
        // Implementazione
    }

    // Metodo main per avviare il server
    public static void main(String[] args) {
        HOTELIERServer server = new HOTELIERServer();
        server.loadData();
        server.start();
    }
}
```

Classe `HOTELIERCustomerClient`

```
public class HOTELIERCustomerClient {
    private Socket socket;
    private BufferedReader in;
    private PrintWriter out;
```

```
    public void start() {
        // Implementazione CLI
    }

    public void register(String username, String password) {
        // Implementazione
    }

    public void login(String username, String password) {
        // Implementazione
    }

    public void logout(String username) {
        // Implementazione
    }

    public void searchHotel(String nomeHotel, String città) {
        // Implementazione
    }

    public void searchAllHotels(String città) {
        // Implementazione
    }

    public void insertReview(String nomeHotel, String nomeCittà, double
globalScore, Map<String, Integer> singleScores) {
        // Implementazione
    }

    public void showMyBadges() {
        // Implementazione
    }

    // Metodo main per avviare il client
    public static void main(String[] args) {
        HOTELIERCustomerClient client = new HOTELIERCustomerClient();
        client.start();
    }
}
```

Configurazione delle connessioni

1. TCP Connection:

- Utilizzare socket TCP per la comunicazione client-server.
- Gestire la connessione TCP nel client e nel server.

2. UDP Multicast:

- Utilizzare socket UDP per inviare notifiche multicast agli utenti loggati.
- Gestire il gruppo multicast nel server e nel client.

Persistenza dei Dati

- **Caricamento Dati**: Caricare le informazioni sugli utenti e gli hotel da file JSON all'avvio del server.
- **Salvataggio Dati**: Salvare periodicamente le informazioni sugli utenti e gli hotel in file JSON.

Mileston

Fase 1: Preparazione dell'Ambiente di Sviluppo

1. Installazione del JDK (Java Development Kit).
2. Configurazione dell'ambiente di sviluppo (IntelliJ IDEA, Eclipse, etc.).
3. Creazione del progetto Java e impostazione della struttura delle directory.

Fase 2: Definizione delle Strutture Dati

1. Creazione delle classi `User` e `Hotel`.
2. Implementazione dei metodi di serializzazione e deserializzazione utilizzando la libreria Gson.

Fase 3: Implementazione del Server

1. Creazione della classe `HOTELIERServer`.
2. Implementazione dei componenti per la gestione degli utenti:
    - Registrazione degli utenti.
    - Login e logout degli utenti.
3. Implementazione dei componenti per la gestione degli hotel:
    - Caricamento delle informazioni sugli hotel da un file JSON.
    - Ricerca degli hotel per nome e città.
    - Ricerca di tutti gli hotel in una città ordinati per ranking.
4. Implementazione dei componenti per la gestione delle recensioni:
    - Inserimento delle recensioni degli utenti.
    - Aggiornamento dei punteggi e dei ranking degli hotel.
5. Implementazione del `Ranking Calculator`:
    - Calcolo periodico dei ranking degli hotel.
6. Implementazione del `Notification Service`:
    - Invio di notifiche agli utenti loggati.
7. Implementazione della `Data Persistence`:

- Memorizzazione periodica delle informazioni degli utenti e degli hotel in file JSON.

8. Implementazione del metodo main per avviare il server.

Fase 4: Implementazione del Client

1. Creazione della classe `HOTELIERCustomerClient`.
2. Implementazione della `CLI` per l'interazione con l'utente.
3. Implementazione del `TCP Connection Handler` per gestire le connessioni con il server.
4. Implementazione del `Notification Handler` per gestire le notifiche dal server.
5. Implementazione dei metodi per le operazioni di registrazione, login, logout, ricerca e inserimento recensioni.
6. Implementazione del metodo main per avviare il client.

Fase 5: Configurazione delle Connessioni di Rete

1. Implementazione delle connessioni TCP per la comunicazione client-server.
2. Implementazione di un server multithreaded per gestire più connessioni client contemporaneamente.
3. Implementazione delle notifiche UDP multicast per gli utenti loggati.

Fase 6: Persistenza dei Dati

1. Implementazione del caricamento dei dati da file JSON all'avvio del server.
2. Implementazione del salvataggio periodico dei dati in file JSON.

Fase 7: Test e Debugging

1. Scrittura di unit test per verificare il corretto funzionamento delle componenti.
2. Utilizzo di strumenti di debugging per individuare e risolvere eventuali problemi.

Fase 8: Documentazione e Consegna

1. Scrittura della documentazione del progetto, inclusa la relazione in formato PDF.
2. Preparazione del codice sorgente e dei file JAR eseguibili per la consegna.
3. Consegna del progetto su Moodle in un unico archivio compresso in formato ZIP.

Gathering Results

Valutazione dei Requisiti

Per valutare se i requisiti sono stati soddisfatti correttamente, verranno eseguite le seguenti attività:

1. Test di Funzionalità:

- Verifica che tutte le funzionalità elencate nella sezione dei requisiti siano state implementate correttamente.
- Esecuzione di casi di test specifici per ogni funzionalità:
  - **Registrazione**: Testare la registrazione di nuovi utenti con vari scenari (username unici, password vuote, etc.).
  - **Login e Logout**: Testare il processo di login e logout per utenti registrati.
  - **Ricerca Hotel**: Verificare la ricerca di hotel specifici per nome e città, e la ricerca di tutti gli hotel in una città ordinati per ranking.
  - **Inserimento Recensioni**: Verificare l'inserimento di recensioni con punteggi globali e specifici.
  - **Visualizzazione Distintivi**: Assicurarsi che gli utenti possano visualizzare correttamente i propri distintivi di esperienza.
  - **Notifiche**: Testare la ricezione di notifiche quando il ranking locale cambia.

2. Test di Usabilità:

- Verifica che l'interfaccia a linea di comando (CLI) sia intuitiva e facile da usare per gli utenti.
- Raccolta di feedback da parte di utenti reali per migliorare l'esperienza utente.

3. Test di Performance:

- Misurazione del tempo di risposta del server per le principali operazioni (registrazione, login, ricerca, inserimento recensioni).
- Valutazione delle prestazioni del sistema sotto carico, simulando l'accesso simultaneo di più utenti.

4. Test di Robustezza:

- Verifica del comportamento del sistema in situazioni di errore (connessioni perse, dati non validi, etc.).
- Assicurarsi che il sistema gestisca correttamente gli errori e mantenga l'integrità dei dati.

Analisi delle Prestazioni Post-Produzione

Dopo il deployment del sistema in un ambiente di produzione, verranno monitorate le seguenti metriche per valutare le prestazioni:

1. Disponibilità del Sistema:

- Monitoraggio del tempo di uptime del server.
- Verifica della disponibilità del servizio per gli utenti finali.

2. Utilizzo delle Risorse:

- Monitoraggio dell'utilizzo della CPU, memoria e disco del server.
- Ottimizzazione delle risorse per garantire una performance ottimale.

3. Feedback degli Utenti:

- Raccolta continua di feedback dagli utenti per identificare eventuali problemi o aree di miglioramento.
- Implementazione di miglioramenti basati sui suggerimenti degli utenti.

4. Manutenzione e Aggiornamenti:

- Pianificazione di interventi di manutenzione periodica per assicurare il corretto funzionamento del sistema.
- Aggiornamenti del software per correggere bug e aggiungere nuove funzionalità.

Con questi passi, sarà possibile valutare in modo accurato se il sistema HOTELIER soddisfa i requisiti specificati e se le prestazioni del sistema in ambiente di produzione sono adeguate.