DIPARTIMENTO DI INGEGNERIA ELETTRICA, ELETTRONICA E INFORMATICA

CORSO DI LAUREA MAGISTRALE IN: INGEGNERIA INFORMATICA

INGEGNERIA DEL SOFTWARE

Sistema di Gestione per un Hotel

Documentazione

Studenti:
Stefano Caramagno
Alessia Fichera
Mario Anthony Guerrera

DOCENTE:
PROF. ORAZIO TOMARCHIO

ANNO ACCADEMICO 2024-2025

SOMMARIO

1. FASE DI IDEAZIONE

1.1. DOCUMENTO DI VISIONE

1.1.1. INTRODUZIONE

Il progetto prevede la realizzazione di un **Sistema di Gestione per un Hotel**, un'applicazione innovativa progettata per ottimizzare e modernizzare la gestione operativa di strutture alberghiere. Il sistema è pensato per rispondere in modo completo alle esigenze di due principali categorie di utenti: il personale alberghiero e i clienti.

L'obiettivo dell'applicazione è fornire strumenti digitali avanzati per la gestione delle prenotazioni, il monitoraggio dello stato delle camere, l'organizzazione del personale e l'implementazione di servizi personalizzati per gli ospiti. Questo approccio integrato garantisce una migliore efficienza operativa, una riduzione degli errori e una significativa ottimizzazione del tempo, contribuendo così a migliorare la qualità complessiva dell'esperienza di soggiorno.

Il sistema si distingue per la sua flessibilità e scalabilità, rendendolo adatto a hotel di diversa dimensione e tipologia, da piccole strutture familiari a grandi catene alberghiere. Grazie alla possibilità di personalizzare le funzionalità e le regole di dominio, il software si adatta alle specifiche esigenze di ciascuna struttura, offrendo al contempo una soluzione tecnologica al passo con i tempi.

In un settore caratterizzato da una crescente competitività, il Sistema di Gestione per un Hotel mira a rappresentare uno strumento strategico per affrontare le sfide del mercato, migliorare i processi interni e soddisfare al meglio le aspettative degli ospiti.

1.1.2. POSIZIONAMENTO

Il Sistema di Gestione per un Hotel nasce per rispondere alle esigenze di ottimizzazione e digitalizzazione delle strutture alberghiere, garantendo una gestione efficiente delle operazioni quotidiane e migliorando l'esperienza complessiva degli utenti. La soluzione si pone come un ponte tra le esigenze del personale, che richiede strumenti organizzativi affidabili, e quelle dei clienti, sempre più orientati verso un servizio rapido e personalizzato.

L'analisi di mercato evidenzia una necessità crescente di sistemi in grado di sostituire i processi tradizionali, spesso manuali o scarsamente integrati, con soluzioni moderne e tecnologiche, in grado di rispondere alle sfide attuali e future del settore. Il progetto si configura come una risposta concreta a questa necessità, grazie a un approccio basato sull'innovazione e sull'attenzione alle specifiche esigenze delle strutture alberghiere.

1.1.2.1. OPPORTUNITA' DI BUSINESS

Il settore alberghiero si trova attualmente in una fase di profonda trasformazione, alimentata da fattori quali l'aumento della domanda turistica, l'evoluzione delle preferenze dei viaggiatori e la crescente competitività tra le strutture ricettive. In questo contesto, il Sistema di Gestione per un Hotel rappresenta una risposta innovativa e strategica alle sfide che il mercato impone.

Una delle principali **opportunità di business** risiede nella digitalizzazione dei processi gestionali interni. Molte strutture alberghiere, in particolare quelle di piccola e media dimensione, continuano a fare affidamento su strumenti manuali o scarsamente integrati per gestire operazioni essenziali come le prenotazioni, l'allocazione delle camere, la gestione del personale e il monitoraggio dei servizi offerti. Questo approccio tradizionale non solo rallenta le operazioni quotidiane, ma comporta anche un aumento del rischio di errori e una diminuzione della qualità del servizio percepita dai clienti.

Il sistema proposto permette di superare queste limitazioni, introducendo una piattaforma tecnologica integrata in grado di semplificare e ottimizzare i processi gestionali. Grazie a strumenti avanzati per la raccolta e l'analisi dei dati, l'applicazione consente una pianificazione più accurata e una gestione più efficace delle risorse, migliorando al contempo la capacità della struttura di rispondere in modo rapido e proattivo alle esigenze degli ospiti.

Un'altra opportunità di business è rappresentata dalla crescente importanza attribuita all'esperienza del cliente. I viaggiatori moderni si aspettano un servizio personalizzato e di alta qualità, che tenga conto delle loro preferenze e necessità specifiche. Il Sistema di Gestione per un Hotel permette di rispondere a queste aspettative offrendo funzionalità come la personalizzazione dei servizi, il check-in e check-out digitali, e l'accesso a un'interfaccia utente intuitiva per la gestione autonoma del proprio soggiorno. Questi elementi non solo migliorano la soddisfazione del cliente, ma rafforzano anche la fidelizzazione e l'immagine complessiva dell'hotel sul mercato.

Inoltre, il sistema si posiziona come un valido strumento per affrontare la crescente pressione sulle strutture alberghiere di operare in maniera sostenibile e conforme alle normative vigenti. Grazie a funzionalità che favoriscono una gestione più responsabile

delle risorse, il software supporta le strutture nel perseguire obiettivi di sostenibilità, sempre più centrali nelle strategie aziendali.

In conclusione, il Sistema di Gestione per un Hotel si propone come un'opportunità unica per le strutture alberghiere di differenziarsi nel mercato, migliorare la propria efficienza operativa e offrire un servizio superiore ai propri ospiti, affermandosi così come leader in un settore in continua evoluzione.

1.1.2.2. FORMULAZIONE DEL PROBLEMA

Nel contesto alberghiero contemporaneo, le strutture ricettive si trovano spesso a dover affrontare problematiche legate alla gestione inefficiente dei processi operativi e all'incapacità di rispondere in modo adeguato alle aspettative sempre più elevate dei clienti. Tali problematiche derivano principalmente dall'utilizzo di sistemi gestionali tradizionali, spesso caratterizzati da un basso grado di integrazione e automazione.

Un **primo problema** rilevante riguarda la gestione delle prenotazioni e l'allocazione delle camere. Nelle strutture che ancora si affidano a registri manuali o a software non progettati specificamente per il settore alberghiero, si riscontrano frequentemente errori nella pianificazione, come l'overbooking o la mancata ottimizzazione delle risorse disponibili. Tali inefficienze non solo compromettono l'organizzazione interna, ma generano anche insoddisfazione tra i clienti, con un conseguente impatto negativo sulla reputazione della struttura.

Un **ulteriore problema** è rappresentato dalla difficoltà di coordinare il personale e di monitorare in tempo reale le operazioni quotidiane, come la pulizia delle camere, la manutenzione e l'erogazione dei servizi aggiuntivi. L'assenza di un sistema centralizzato che consenta una visione d'insieme delle attività in corso limita la capacità del management di intervenire tempestivamente e di garantire standard di qualità costanti.

Dal punto di vista dell'esperienza del cliente, uno dei **principali problemi** è l'assenza di strumenti personalizzati che rispondano alle specifiche esigenze degli ospiti. Senza un sistema che raccolga e analizzi in modo strutturato i dati sui clienti, risulta difficile per la struttura offrire servizi mirati, come l'adattamento delle opzioni di soggiorno alle preferenze individuali. Questo limite si traduce in un'esperienza meno coinvolgente e, spesso, in una minore fidelizzazione.

Infine, va sottolineato che molte strutture alberghiere si trovano a operare in un contesto di crescente pressione normativa e di richieste legate alla sostenibilità. L'assenza di strumenti adeguati al monitoraggio e la gestione di parametri rende complesso per gli hotel conformarsi alle normative vigenti e adottare pratiche sostenibili, una condizione ormai essenziale per mantenere la competitività sul mercato.

La combinazione di questi fattori evidenzia l'urgenza di dotare le strutture alberghiere di un sistema gestionale moderno e integrato, capace di affrontare le problematiche operative e di migliorare al contempo l'esperienza del cliente. Il Sistema di Gestione per un Hotel si pone l'obiettivo di superare queste limitazioni, fornendo soluzioni tecnologiche avanzate che ottimizzino l'efficienza interna e rafforzino la posizione competitiva delle strutture sul mercato.

1.1.2.3. FORMULAZIONE DELLA POSIZIONE DEL PRODOTTO

Il Sistema di Gestione per un Hotel si posiziona come una soluzione tecnologica innovativa, progettata per soddisfare le esigenze sia del personale alberghiero sia dei clienti. Il prodotto rappresenta una risposta diretta e strategica alle inefficienze e ai limiti operativi tipici delle strutture ricettive che utilizzano metodi di gestione tradizionali o sistemi non integrati.

Pensato per strutture alberghiere di ogni dimensione, il sistema offre una piattaforma completa che integra funzionalità avanzate per la gestione delle operazioni quotidiane. Per il personale alberghiero, il sistema funge da strumento centrale per la gestione delle prenotazioni, l'allocazione delle camere, il coordinamento del personale e il monitoraggio dei servizi. La sua flessibilità consente di personalizzare le regole di dominio e adattarle alle specifiche necessità di ciascuna struttura, fornendo così una soluzione tailor-made che migliora l'efficienza organizzativa e riduce i margini di errore.

Dal lato dei clienti, il sistema mira a migliorare l'esperienza del soggiorno, offrendo strumenti digitali semplici e intuitivi. Attraverso un'interfaccia utente accessibile, gli ospiti possono gestire autonomamente le loro prenotazioni, accedere ai dettagli del soggiorno, effettuare il check-in e il check-out in modo rapido e personalizzare i servizi disponibili. Questa attenzione al cliente consente alle strutture alberghiere di aumentare la soddisfazione e la fidelizzazione degli ospiti, due elementi essenziali per il successo competitivo in un mercato sempre più orientato all'utente.

In termini di **posizionamento strategico**, il sistema si propone come una piattaforma all'avanguardia, ideale per le strutture alberghiere che intendono innovare e differenziarsi. Grazie alla combinazione di tecnologie avanzate, flessibilità operativa e attenzione al cliente, il prodotto rappresenta un punto di riferimento per il mercato, offrendo valore sia sotto il profilo operativo che strategico. In questo modo, il Sistema di Gestione per un Hotel non è solo un software gestionale, ma una leva per il miglioramento continuo e la competitività delle strutture alberghiere.

1.1.3. DESCRIZIONE DELLE PARTI INTERESSATE

Il successo del Sistema di Gestione per un Hotel dipende dalla sua capacità di rispondere efficacemente alle esigenze delle diverse parti interessate coinvolte nelle operazioni alberghiere. Ognuna di queste parti gioca un ruolo fondamentale nel garantire l'efficienza del sistema e nel promuovere un'esperienza di alto livello per gli ospiti.

Da un lato, il personale alberghiero, suddiviso in amministratori, addetti alla reception, responsabili delle pulizie e manutentori, necessita di strumenti che semplifichino il lavoro quotidiano, riducano gli errori operativi e migliorino il coordinamento tra i vari reparti. Dall'altro lato, i clienti, con le loro aspettative sempre più sofisticate, richiedono un'interazione fluida e personalizzata con il sistema, che permetta loro di vivere un soggiorno confortevole e privo di inconvenienti.

Una comprensione dettagliata delle esigenze, degli obiettivi e delle responsabilità di ciascuna parte interessata costituisce la base per progettare un sistema che non solo soddisfi le aspettative, ma che contribuisca a migliorare la qualità complessiva del servizio e a rafforzare la competitività delle strutture alberghiere.

1.1.3.1. OBIETTIVI A LIVELLO DELL'UTENTE

Gli **obiettivi degli utenti** del Sistema di Gestione per un Hotel sono strettamente legati ai ruoli e alle responsabilità che essi ricoprono all'interno dell'organizzazione alberghiera. Il sistema deve rispondere alle esigenze di efficienza, semplicità e personalizzazione per garantire un'esperienza positiva sia per il personale che per i clienti.

Receptionist

I receptionist sono il punto di contatto diretto con i clienti e necessitano di strumenti che semplifichino il loro lavoro quotidiano, tra cui:

- Gestione rapida del check-in e del check-out, con funzionalità digitali che minimizzino i tempi di attesa per gli ospiti.
- Accesso alle informazioni sui clienti per offrire un servizio personalizzato.
- Comunicazione efficace per garantire che tutte le richieste vengano soddisfatte tempestivamente.

Clienti

Dal punto di vista dei clienti, l'obiettivo principale è vivere un'esperienza di soggiorno semplice, confortevole e personalizzata. Il sistema deve quindi offrire:

- Un'interfaccia intuitiva per la gestione autonoma delle prenotazioni, inclusa la possibilità di modificare o cancellare le stesse in base alle necessità.
- Servizi personalizzati, come la possibilità di richiedere opzioni specifiche per le camere.
- Check-in e check-out digitali, che eliminino la necessità di lunghe attese alla reception.
- Accesso a informazioni utili sui servizi disponibili.

Staff Operativo

Lo staff operativo, composto da responsabile operativo e addetti alle pulizie, ha l'obiettivo di garantire un servizio impeccabile e un ambiente confortevole per gli ospiti. Per questo, il sistema deve offrire:

- Un sistema di assegnazione delle attività, che organizzi in modo chiaro le mansioni da svolgere.
- Aggiornamenti in tempo reale, per monitorare lo stato delle attività.
- Accesso a una piattaforma che consenta di visualizzare e completare le attività assegnate anche in movimento, migliorando la produttività e riducendo i tempi di risposta.

In sintesi, il Sistema di Gestione per un Hotel deve soddisfare un'ampia gamma di obiettivi specifici per ogni tipologia di utente, contribuendo a migliorare l'efficienza interna, la qualità del servizio e la soddisfazione complessiva dei clienti. Questo approccio mirato garantisce che ogni interazione con il sistema generi valore e rafforzi la competitività della struttura alberghiera.

1.1.4. RIEPILOGO DELLE CARATTERISTICHE DEL SISTEMA

Il **Sistema di Gestione per un Hotel** è concepito come una piattaforma completa e flessibile, capace di rispondere in modo efficace alle esigenze operative e strategiche delle strutture alberghiere. Le sue caratteristiche principali sono state progettate per garantire un miglioramento tangibile sia nelle operazioni quotidiane che nell'esperienza complessiva dei clienti, rendendolo uno strumento indispensabile per il settore.

Gestione delle Prenotazioni

- **Visualizzazione in tempo reale della disponibilità**: offre uno strumento aggiornato per monitorare lo stato delle camere, garantendo un'allocazione ottimale delle risorse.
- **Prevenzione dell'overbooking**: grazie a funzionalità di controllo automatico, il sistema minimizza gli errori nella gestione delle prenotazioni.

Amministrazione delle Camere e dei Servizi

- **Monitoraggio dello stato delle camere**: fornisce una visione aggiornata della disponibilità e dello stato di pulizia.
- **Configurazione personalizzabile dei servizi**: consente di associare specifiche caratteristiche o pacchetti a ogni camera.

Ottimizzazione delle Operazioni del Personale

- **Assegnazione automatica delle attività**: il sistema organizza e assegna compiti al personale, migliorando l'efficienza operativa.
- **Piattaforma per il personale**: permette di accedere al sistema e completare le attività assegnate in tempo reale, indipendentemente dalla posizione.

Esperienza Cliente

- **Portale cliente intuitivo**: i clienti possono gestire autonomamente le prenotazioni, effettuare il check-in e il check-out, e accedere a informazioni utili sui servizi disponibili.
- **Accesso a servizi digitali**: il sistema include opzioni per il pagamento online, la richiesta di servizi aggiuntivi e la gestione delle preferenze di soggiorno.

In conclusione, il **Sistema di Gestione per un Hotel** rappresenta una soluzione tecnologica di punta, progettata per ottimizzare ogni aspetto delle operazioni alberghiere. Grazie alla sua combinazione di funzionalità avanzate, flessibilità e attenzione alla sostenibilità, il sistema contribuisce a migliorare l'efficienza interna, elevare la qualità del servizio offerto e rafforzare la competitività della struttura nel mercato globale.

1.2. MODELLO DEI CASI D'USO

1.2.1. REQUISITI

Il Sistema di Gestione per un Hotel si configura come una piattaforma software ideata per rispondere alle necessità operative di strutture alberghiere moderne. La progettazione del sistema mira a ottimizzare i processi interni, garantire un'esperienza cliente superiore e supportare una gestione efficace e scalabile.

I **requisiti** funzionali principali del sistema sono:

- **Gestione delle prenotazioni**:
  - Inserimento di nuove prenotazioni da parte del personale o dei clienti.
  - Modifica e aggiornamento delle prenotazioni esistenti.
  - Controllo in tempo reale della disponibilità delle camere.
- **Gestione del check-in e check-out**:
  - Procedure digitali per velocizzare l'accoglienza e la partenza degli ospiti.
  - Registrazione automatica dei dati relativi agli ospiti.
- **Amministrazione delle camere**:
  - Monitoraggio dello stato delle camere.
  - Assegnazione delle camere ai clienti in base alle loro preferenze e richieste specifiche.
- **Coordinamento del personale**:
  - Assegnazione delle attività quotidiane.
  - Visualizzazione in tempo reale dello stato delle attività assegnate.
- **Gestione dei pagamenti**:
  - Registrazione dei pagamenti effettuati dagli ospiti.
  - Possibilità di effettuare il pagamento con diverse modalità.

Il sistema deve rispondere a requisiti di affidabilità, scalabilità e sicurezza, garantendo la protezione dei dati sensibili degli ospiti e la continuità operativa anche in caso di carichi elevati. La sua adozione punta a ottimizzare le operazioni alberghiere, migliorare la soddisfazione degli ospiti e rafforzare la competitività della struttura nel mercato globale.

1.2.2. OBIETTIVI E CASI D'USO

L'analisi dei requisiti ha permesso di identificare gli attori principali coinvolti nel sistema e gli obiettivi specifici che il Sistema di Gestione per un Hotel deve soddisfare. A partire da questi elementi, sono stati definiti i **casi d'uso** fondamentali che rappresentano le interazioni tra gli attori e il sistema. La seguente tabella riassume gli attori, gli obiettivi correlati e i casi d'uso principali individuati:

ATTORE	OBIETTIVO	CASO DI'USO
Cliente	Inviare una richiesta per prenotare una camera	UC1: Effettua richiesta di prenotazione
Receptionist	Accettare o annullare una richiesta di prenotazione ricevuta	UC2: Gestisci richiesta di prenotazione
Cliente	Effettuare il pagamento per prenotazioni.	UC3: Effettua pagamento
Cliente	Consultare i dettagli della prenotazione e i servizi inclusi	UC4: Visualizza dettagli della prenotazione
Cliente	Richiedere modifiche ai dettagli della prenotazione	UC5: Richiedi modifica alla prenotazione
Receptionist	Modificare i dettagli di una prenotazione su richiesta del cliente	UC6: Modifica dettagli della prenotazione
Cliente	Annullare una prenotazione per necessità del cliente.	UC7: Annulla prenotazione
Receptionist	Annullare una prenotazione per motivi interni legati all'hotel.	UC8: Cancella prenotazione
Cliente	Effettuare il check-in digitale	UC9: Check-in digitale
Cliente	Effettuare il check-out digitale	UC10: Check-out digitale
Responsabile Operativo	Assegnare attività di manutenzione o pulizia	UC11: Assegna attività al personale
Addetto alle Pulizie	Visualizzare e aggiornare lo stato delle camere	UC12: Gestisci stato delle camere

Dettagli del flusso cronologico

- **UC1** → **UC3**: Inizia con la richiesta di prenotazione e il pagamento da parte del cliente.
- **UC4** → **UC6**: Il cliente può consultare i dettagli, richiedere modifiche, e il receptionist può apportare modifiche se richiesto.
- **UC7** → **UC8**: La cancellazione della prenotazione può essere effettuata dal cliente o dal receptionist.
- **UC9** → **UC10**: Durante il soggiorno, il cliente effettua il check-in e il check-out digitale.
- **UC11** → **UC12**: Le operazioni interne di gestione delle camere e delle attività completano il ciclo.

Ogni caso d’uso rappresenta un’interazione chiave che garantisce il funzionamento fluido e completo del sistema, rispondendo in modo diretto alle esigenze di gestione e alle aspettative degli ospiti. Questi casi saranno sviluppati nei dettagli nelle sezioni successive per delineare gli scenari principali, le pre-condizioni, i passi fondamentali e le possibili estensioni.

1.2.3. CASI D'USO

I **casi d'uso** rappresentano uno strumento fondamentale per descrivere le interazioni tra gli attori e il sistema, evidenziando i flussi di operazioni che consentono di raggiungere specifici obiettivi operativi. Attraverso una descrizione strutturata e dettagliata, ogni caso d'uso definisce il contesto, i requisiti funzionali e i comportamenti attesi del sistema in risposta alle azioni degli attori coinvolti.

L'approccio adottato si basa sull'identificazione dei processi chiave che caratterizzano la gestione di un hotel, garantendo una rappresentazione chiara e completa delle funzionalità offerte. Ogni caso d'uso considera gli scenari principali e le eventuali eccezioni, assicurando che il sistema sia in grado di rispondere efficacemente anche a situazioni non standard.

Questa analisi dettagliata consente di stabilire una base solida per lo sviluppo e la validazione del sistema, favorendo una comprensione condivisa tra tutti gli stakeholder e assicurando che il prodotto finale soddisfi pienamente le esigenze degli utenti.

1.2.3.1. UC1: EFFETTUA UNA RICHIESTA DI PRENOTAZIONE

Questo caso d'uso descrive in dettaglio il processo che il cliente segue per inviare una richiesta di prenotazione.

Nome del caso d'uso	**UC1: Effettua una richiesta di prenotazione**
Portata	Sistema di Gestione per un Hotel
Livello	Obiettivo utente
Attore primario	Cliente
Parti interessate e interessi	- **Cliente**: desidera richiedere la prenotazione di una camera per uno specifico periodo. - **Receptionist**: vuole ricevere richieste di prenotazione in modo chiaro e completo per gestirle efficientemente.
Pre-condizioni	- Il cliente può effettuare l'accesso al portale di prenotazione.
Garanzia di successo	La richiesta di prenotazione è registrata nel sistema con tutte le informazioni necessarie.
Scenario principale di successo	**1.** Il cliente accede alla sezione dedicata ad effettuare richieste di prenotazione. **2.** Il sistema mostra le tipologie di camere offerte dalla struttura alberghiera. **3.** Il cliente sceglie la tipologia di camera, seleziona le date di arrivo e partenza e invia la richiesta di prenotazione. **4.** Il sistema registra la richiesta di prenotazione e invia una notifica al cliente di avvenuta richiesta.
Estensioni	-***a**. Il sistema fallisce arrestandosi improvvisamente: 1. Il cliente riavvia il software.

	2. Il sistema ripristina lo stato precedente. -**3a**. Il cliente non seleziona una data di arrivo o una data di partenza: 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°3. -**3b**. Il cliente seleziona una data di arrivo procedente alla data attuale. 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°3. -**3c**. Il cliente seleziona una data di arrivo e una data di partenza equivalenti. 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°3. -**3d**. Il cliente seleziona una data di partenza precedente alla data di arrivo. 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°3. -**3e**. Il cliente seleziona una data di arrivo e una data di partenza per cui non vi è disponibilità di camere. 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°3.
Requisiti speciali	- Interfaccia intuitiva per l'effettuazione di una richiesta di prenotazione. - Validazione automatica dei dati inseriti. - Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).
Frequenza di ripetizioni	Ogni volta che un cliente vuole effettuare una richiesta di prenotazione.
Varie	L'identificativo di ogni richiesta di prenotazione è un codice numerico univoco.

1.2.3.2. UC2: GESTISCI RICHIESTA DI PRENOTAZIONE

Questo caso d'uso illustra il processo che il receptionist segue per gestire le richieste di prenotazione, garantendo un flusso operativo chiaro ed efficiente.

Nome del caso d'uso	**UC2: Gestione richiesta di prenotazione**
Portata	Sistema di Gestione per un Hotel
Livello	Obiettivo utente
Attore primario	Receptionist
Parti interessate e interessi	- **Receptionist**: desidera gestire in modo efficiente le richieste di prenotazione

	ricevute. - **Cliente**: vuole essere informato tempestivamente sull'esito della propria richiesta di prenotazione.
Pre-condizioni	- Il receptionist può effettuare l'accesso al portale di prenotazione. - Esistono richieste di prenotazione registrate nel sistema.
Garanzia di successo	La richiesta di prenotazione è accettata o annullata.
Scenario principale di successo	**1.** Il receptionist accede alla sezione dedicata alla gestione delle richieste di prenotazione in attesa di approvazione. **2.** Il sistema elenca le richieste di prenotazione con i dettagli. **3**. Il receptionist sceglie una richiesta di prenotazione. **4.** Il receptionist accetta la richiesta di prenotazione. **5.** Se necessario, il receptionist può annullare la richiesta di prenotazione. **6.** Il sistema aggiorna lo stato della richiesta di prenotazione e invia al receptionist una notifica di conferma della richiesta di prenotazione gestita.
Estensioni	-***a**. Il sistema fallisce arrestandosi improvvisamente: 1. Il receptionist riavvia il software. 2. Il sistema ripristina lo stato precedente.
Requisiti speciali	- Interfaccia intuitiva per la gestione di una richiesta di prenotazione. - Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).
Frequenza di ripetizioni	Ogni volta che il cliente effettua una richiesta di prenotazione.
Varie	- L'identificativo di ogni richiesta di prenotazione è un codice numerico univoco. - Ogni richiesta gestita andando ad aggiornare lo stato.

1.2.3.3. UC3: EFFETTUA PAGAMENTO

Questo caso d'uso descrive il processo che consente al cliente di completare il pagamento in modo sicuro e senza interruzioni, garantendo la conferma della prenotazione o dei servizi richiesti.

Nome del caso d'uso	**UC3: Effettua pagamento**
Portata	Sistema di Gestione per un Hotel
Livello	Obiettivo utente
Attore primario	Cliente
Parti interessate e interessi	- **Cliente**: desidera completare il pagamento per confermare una prenotazione. - **Hotel**: vuole garantire la ricezione sicura e tracciabile del pagamento per le prenotazioni.
Pre-condizioni	- Il cliente può effettuare l'accesso al portale di prenotazione. - Il cliente ha una richiesta di prenotazione confermata.
Garanzia di successo	Il pagamento è completato con successo, e il cliente riceve la conferma definitiva della prenotazione.
Scenario principale di successo	**1**. Il cliente accede alla sezione dedicata al pagamento delle richieste di prenotazione approvate. **2.** Il sistema visualizza i dettagli delle richieste di prenotazione, accettate dal receptionist, inclusi l'importo totale e le eventuali modalità di pagamento disponibili. **3.** Il cliente sceglie la prenotazione che desidera pagare. **4.** Il cliente seleziona una modalità di pagamento. **5**. Il cliente inserisce i dati di pagamento richiesti. **6.** Il sistema invia i dati e attende una risposta. **7.** Il sistema registra il pagamento come completato e invia una notifica al cliente di avvenuto pagamento.
Estensioni	-***a**. Il sistema fallisce arrestandosi improvvisamente: 1. Il cliente riavvia il software. 2. Il sistema ripristina lo stato precedente. -**4a**. Il cliente non seleziona un metodo di pagamento: 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°4.

	-**5a**. Il cliente non inserisce il numero della carta: 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°5. -**5b**. Il cliente non inserisce la data di scadenza: 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°5. -**5c**. Il cliente inserisce una data di scadenza che non rispetta il formato atteso dal sistema: 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°5. -**5d**. Il cliente inserisce una data di scadenza che non rispetta il formato atteso dal sistema: 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°5. -**5e**. Il cliente inserisce una data di scadenza precedente alla data attuale: 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°5. -**5f**. Il cliente non inserisce il cvv: 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°5. -**5g**. Il cliente inserisce un cvv che non rispetta il formato atteso dal sistema: 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°5.
Requisiti speciali	- Interfaccia intuitiva per l'effettuazione del pagamento di una richiesta di prenotazione. - Validazione automatica dei dati inseriti. - Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).
Frequenza di ripetizioni	Ogni volta che un receptionist approva una richiesta di prenotazione.
Varie	L'identificativo di ogni pagamento è un codice numerico univoco.

1.2.3.4. UC4: VISUALIZZA DETTAGLI DELLA PRENOTAZIONE

Questo caso d'uso illustra il processo attraverso il quale il cliente può accedere ai dettagli delle proprie prenotazioni in modo chiaro e sicuro, migliorando l'esperienza utente e riducendo richieste al personale alberghiero.

Nome del caso d'uso	**UC4: Visualizza dettagli della prenotazione**
Portata	Sistema di Gestione per un Hotel
Livello	Obiettivo utente
Attore primario	Cliente
Parti interessate e interessi	- **Cliente**: desidera accedere ai dettagli della propria prenotazione per verificarne le informazioni e i servizi inclusi. - **Hotel**: vuole garantire che i clienti abbiano accesso rapido e chiaro ai dettagli della prenotazione.
Pre-condizioni	- Il cliente può effettuare l'accesso al portale di prenotazione. - Il cliente ha effettuato il pagamento di una richiesta di prenotazione. - La prenotazione è presente e accessibile nel database.
Garanzia di successo	Il cliente visualizza correttamente i dettagli della prenotazione.
Scenario principale di successo	**1.** Il cliente accede alla sezione dedicata alla visualizzazione delle prenotazioni pagate. **2.** Il sistema elenca le prenotazioni attive del cliente. **3.** Il sistema mostra i dettagli della prenotazione.
Estensioni	-***a**. Il sistema fallisce arrestandosi improvvisamente: 1. Il cliente riavvia il software. 2. Il sistema ripristina lo stato precedente.
Requisiti speciali	- Interfaccia intuitiva per la visualizzazione della prenotazione. - Validazione automatica dei dati inseriti. - Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).
Frequenza di ripetizioni	Ogni volta che il cliente desidera visualizzare i dettagli di una prenotazione.

Varie	L’identificativo di ogni prenotazione è un codice numerico univoco.

1.2.3.5. UC5: RICHIEDI MODIFICA ALLA PRENOTAZIONE

Questo caso d'uso descrive il processo che consente al cliente di richiedere modifiche alla propria prenotazione in modo chiaro e strutturato, garantendo che le richieste siano gestibili per l’hotel.

Nome del caso d’uso	**UC5: Richiedi modifica alla prenotazione**
Portata	Sistema di Gestione per un Hotel
Livello	Obiettivo utente
Attore primario	Cliente
Parti interessate e interessi	- **Cliente**: desidera modificare i dettagli della prenotazione per adattarli a nuove esigenze, come cambi di date o tipologia di camera - **Receptionist**: vuole ricevere richieste di modifica chiare e ben definite per garantire una gestione rapida e accurata.
Pre-condizioni	- Il cliente può effettuare l’accesso al portale di prenotazione. - La prenotazione è confermata e modificabile secondo i termini previsti.
Garanzia di successo	La richiesta di modifica viene inviata con successo al sistema e risulta visibile al receptionist per la successiva elaborazione.
Scenario principale di successo	**1.** Il cliente accede alla sezione dedicata alla richiesta di modifica alle prenotazioni. **2.** Il cliente seleziona la prenotazione da modificare. **3.** Il sistema presenta un modulo con i dettagli della prenotazione e i campi modificabili. **4.** Il cliente apporta le modifiche desiderate. **5.** Il sistema registra la richiesta di modifica della prenotazione e invia al cliente una notifica di avvenuta richiesta
Estensioni	-***a**. Il sistema fallisce arrestandosi improvvisamente: 1. Il cliente riavvia il software. 2. Il sistema ripristina lo stato precedente. -**4a**. Il cliente non seleziona una nuova data di arrivo: 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°4. -**4b**. Il cliente non seleziona una nuova data di partenza:

	1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°4. **-4c**. Il cliente seleziona una nuova data di arrivo procedente alla data attuale. 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°4. **-4d**. Il cliente seleziona una nuova data di arrivo e una data di partenza equivalenti. 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°4. **-4e**. Il cliente seleziona una nuova data di arrivo precedente alla data attuale. 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°4. **-4f**. Il cliente seleziona una nuova data di partenza precedente alla nuova data di arrivo: 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°4. **-4g**. Il cliente seleziona una data di arrivo e una data di partenza per cui non vi è disponibilità di camere. 1. Il sistema notifica il problema al cliente. 2. Il cliente ripete il passaggio n°4.
Requisiti speciali	- Interfaccia intuitiva per la richiesta di modifica della prenotazione pagata. - Validazione automatica dei dati inseriti. - Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).
Frequenza di ripetizioni	Ogni volta che un cliente vuole richiedere una modifica di una prenotazione
Varie	- L'identificativo di ogni prenotazione è un codice numerico univoco. - Ogni prenotazione è gestita andando ad aggiornare lo stato.

1.2.3.6. UC6: MODIFICA DETTAGLI DELLA PRENOTAZIONE

Questo caso d'uso descrive in dettaglio il processo attraverso il quale il receptionist può gestire e applicare le modifiche richieste dal cliente, garantendo precisione ed efficienza.

Nome del caso d'uso	**UC6: Modifica dettagli della prenotazione**
Portata	Sistema di Gestione per un Hotel

Livello	Obiettivo utente
Attore primario	Receptionist
Parti interessate e interessi	- **Receptionist**: desidera aggiornare i dettagli di una prenotazione in base alle richieste del cliente o per esigenze organizzative. - **Cliente**: vuole che le modifiche richieste vengano applicate in modo accurato e tempestivo.
Pre-condizioni	- Il receptionist può effettuare l'accesso al portale di prenotazione. - Esiste una prenotazione pagata, ma in stato di approvazione di modifica nel sistema. - La richiesta di modifica è stata ricevuta e verificata dal receptionist.
Garanzia di successo	La prenotazione viene aggiornata correttamente nel sistema.
Scenario principale di successo	**1.** Il receptionist accede alla sezione dedicata alla gestione delle richieste di modifica delle prenotazioni in attesa di approvazione. **2.** Il sistema elenca le richieste di modifica delle prenotazione con i dettagli. **3**. Il receptionist sceglie una richiesta di modifica. **4.** Il receptionist accetta la richiesta di modifica della prenotazione. **5.** Se necessario, il receptionist può annullare la richiesta di modifica della prenotazione. **7.** Il sistema aggiorna lo stato della prenotazione e invia al receptionist una notifica di conferma della richiesta di modifica della prenotazione gestita.
Estensioni	-***a**. Il sistema fallisce arrestandosi improvvisamente: 1. Il receptionist riavvia il software. 2. Il sistema ripristina lo stato precedente.
Requisiti speciali	- Interfaccia intuitiva per la modifica di una prenotazione. - Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).

Frequenza di ripetizioni	Ogni volta che un cliente effettua una richiesta di modifica della prenotazione.
Varie	- L'identificativo di ogni prenotazione è un codice numerico univoco. - Ogni prenotazione è gestita andando ad aggiornare lo stato.

1.2.3.7. UC7: ANNULLA PRENOTAZIONE

Questo caso d'uso descrive in modo dettagliato il processo che consente al cliente di annullare autonomamente una prenotazione, assicurando trasparenza e conformità ai termini previsti.

Nome del caso d'uso	**UC7: Annulla prenotazione**
Portata	Sistema di Gestione per un Hotel
Livello	Obiettivo utente
Attore primario	Cliente
Parti interessate e interessi	- **Cliente**: desidera annullare una prenotazione per motivi personali, rispettando i termini di cancellazione. - **Hotel**: vuole gestire le cancellazioni in modo efficiente e trasparente, riducendo l'impatto operativo ed economico.
Pre-condizioni	- Il cliente può effettuare l'accesso al portale di prenotazione. - La prenotazione esiste nel sistema ed è annullabile. - I termini di cancellazione sono definiti e visualizzabili dal cliente.
Garanzia di successo	La prenotazione è annullata con successo, e lo stato aggiornato nel sistema. Il cliente riceve conferma dell'annullamento.
Scenario principale di successo	**1.** Il cliente accede alla sezione dedicata all'annullamento delle prenotazioni. **2.** Il sistema elenca le prenotazioni del cliente. **3.** Il cliente sceglie la prenotazione da annullare. **4.** Il sistema verifica i termini di cancellazione. **5.** Il cliente conferma l'intenzione di annullare la prenotazione. **6.** Il sistema aggiorna lo stato della prenotazione e invia al cliente una notifica di conferma di annullamento.
Estensioni	-***a**. Il sistema fallisce arrestandosi improvvisamente: 1. Il cliente riavvia il software. 2. Il sistema ripristina lo stato precedente.
Requisiti speciali	- Interfaccia intuitiva per l'annullamento della prenotazione

	- Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).
Frequenza di ripetizioni	Ogni volta che un cliente deve annullare una prenotazione sulla base dei suoi imprevisti personali.
Varie	- L'identificativo di ogni prenotazione è un codice numerico univoco. - Ogni prenotazione è gestita andando ad aggiornare lo stato.

1.2.3.8. UC8: CANCELLA PRENOTAZIONE

Questo caso d'uso descrive il processo attraverso il quale il receptionist può annullare una prenotazione per motivi operativi, garantendo un'esperienza trasparente per il cliente e minimizzando l'impatto operativo sull'hotel.

Nome del caso d'uso	**UC8: Cancella prenotazione**
Portata	Sistema di Gestione per un Hotel
Livello	Obiettivo utente
Attore primario	Receptionist
Parti interessate e interessi	- **Receptionist**: desidera annullare una prenotazione per motivi interni all'hotel, - **Cliente**: vuole essere informato in caso di cancellazione non dipendente dalla propria volontà.
Pre-condizioni	- Esiste una prenotazione confermata nel sistema. - La cancellazione è dovuta a motivi operativi o organizzativi interni all'hotel.
Garanzia di successo	La prenotazione è cancellata con successo, il cliente riceve notifica della cancellazione e un rimborso completo.
Scenario principale di successo	**1.** Il receptionist accede alla sezione dedicata alla cancellazione delle prenotazioni. **2.** Il sistema elenca le prenotazioni di tutti i clienti. **3.** Il receptionist sceglie la prenotazione da cancellare. **4.** Il receptionist conferma la cancellazione. **5.** Il sistema cancella la prenotazione e invia al receptionist una notifica con la conferma della cancellazione.

Estensioni	**-*a**. Il sistema fallisce arrestandosi improvvisamente: 1. Il receptionist riavvia il software. 2. Il sistema ripristina lo stato precedente.
Requisiti speciali	- Interfaccia intuitiva per la cancellazione della prenotazione - Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).
Frequenza di ripetizioni	Ogni volta che la struttura alberghiera deve cancellare una prenotazione sulla base di imprevisti non programmati dell'hotel.
Varie	- L'identificativo di ogni prenotazione è un codice numerico univoco.

1.2.3.9. UC9: CHECK-IN IN DIGITALE

Questo caso d'uso descrive in modo chiaro il processo di check-in digitale, migliorando l'esperienza del cliente e ottimizzando le operazioni dell'hotel.

Nome del caso d'uso	**UC9: Check-in in digitale**
Portata	Sistema di Gestione per un Hotel
Livello	Obiettivo utente
Attore primario	Cliente
Parti interessate e interessi	- **Cliente**: desidera effettuare il check-in in modo rapido e autonomo, riducendo i tempi di attesa alla reception. - **Hotel**: vuole snellire le procedure di accoglienza, garantendo accuratezza e velocità nel processo di check-in.
Pre-condizioni	- Il cliente può effettuare l'accesso al portale di prenotazione. - La prenotazione è confermata e il pagamento è stato completato.
Garanzia di successo	Il check-in viene completato con successo, e il cliente riceve la conferma dell'operazione.
Scenario principale di successo	**1.** Il cliente accede alla sezione dedicata al check-in in digitale. **2.** Il sistema elenca le prenotazioni attive per il check-in. **3.** Il cliente sceglie la prenotazione da confermare.

	4. Il sistema aggiorna lo stato della prenotazione e invia al cliente una notifica di conferma del check-in.
Estensioni	-***a**. Il sistema fallisce arrestandosi improvvisamente: 1. Il cliente riavvia il software. 2. Il sistema ripristina lo stato precedente.
Requisiti speciali	- Interfaccia intuitiva per l'effettuazione del check-in in digitale. - Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).
Frequenza di ripetizioni	Ogni volta che ha inizio una prenotazione di un cliente.
Varie	- L'identificativo di ogni prenotazione è un codice numerico univoco. - Ogni prenotazione è gestita andando ad aggiornare lo stato.

1.2.3.10. UC10: CHECK-OUT IN DIGITALE

Questo caso d'uso descrive in modo chiaro e dettagliato il processo di check-out digitale, migliorando l'efficienza del sistema e garantendo un'esperienza cliente fluida.

Nome del caso d'uso	**UC10: Check-out in digitale**
Portata	Sistema di Gestione per un Hotel
Livello	Obiettivo utente
Attore primario	Cliente
Parti interessate e interessi	- **Cliente**: desidera completare il check-out in modo rapido e senza interazioni fisiche, evitando code o ritardi. - **Hotel**: vuole semplificare il processo di check-out per ottimizzare la gestione delle partenze e delle risorse.
Pre-condizioni	- Il cliente può effettuare l'accesso al portale di prenotazione. - Il cliente ha effettuato il pagamento per tutti i servizi utilizzati durante il soggiorno. - La prenotazione è attiva e può essere chiusa secondo i termini del soggiorno.
Garanzia di successo	Il check-out viene completato con successo, aggiornando lo stato della prenotazione e liberando la camera.
Scenario principale di successo	**1.** Il cliente accede alla sezione dedicata al check-out in digitale.

	2. Il sistema elenca le prenotazioni attive per il check-out. **3.** Il cliente sceglie la prenotazione da chiudere. **4.** Il sistema aggiorna lo stato della prenotazione e invia al cliente una notifica di conferma del check-out.
Estensioni	-***a**. Il sistema fallisce arrestandosi improvvisamente: 1. Il cliente riavvia il software. 2. Il sistema ripristina lo stato precedente.
Requisiti speciali	- Interfaccia intuitiva per l'effettuazione del check-in in digitale. - Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).
Frequenza di ripetizioni	Ogni volta che ha termine una prenotazione di un cliente.
Varie	- L'identificativo di ogni prenotazione è un codice numerico univoco. - Ogni prenotazione è gestita andando ad aggiornare lo stato.

1.2.3.11. UC11: ASSEGNA ATTIVITÀ AL PERSONALE

Questo caso d'uso descrive in dettaglio il processo attraverso il quale il responsabile operativo assegna le attività al personale, garantendo precisione e trasparenza per un'efficace gestione delle operazioni interne dell'hotel.

Nome del caso d'uso	**UC11: Assegna attività del personale**
Portata	Sistema di Gestione per un Hotel
Livello	Obiettivo utente
Attore primario	Responsabile operativo
Parti interessate e interessi	- **Responsabile operativo**: desidera assegnare in modo efficace le attività al personale, ottimizzando la gestione delle risorse disponibili. - **Addetto alle pulizie**: vuole ricevere istruzioni chiare e puntuali per eseguire le attività assegnate senza errori o sovrapposizioni.
Pre-condizioni	- Il responsabile operativo può effettuare l'accesso al portale di prenotazione. - L'addetto delle pulizie è registrato e attivo nel sistema.

Garanzia di successo	Le attività vengono assegnate con successo al personale.
Scenario principale di successo	**1.** Il responsabile operativo accede alla sezione dedicata all'assegnazione delle attività. **2.** Il responsabile operativo assegna l'attività selezionando un addetto del personale delle pulizie e una camera. **3.** Il sistema aggiorna lo stato di attività e invia al responsabile operativo una notifica di conferma assegnazione attività.
Estensioni	-***a**. Il sistema fallisce arrestandosi improvvisamente: 1. Il responsabile operativo riavvia il software. 2. Il sistema ripristina lo stato precedente. -**2a**. Il responsabile operativo non seleziona nessun addetto del personale delle pulizie: 1. Il sistema notifica il problema al cliente. 2. Il responsabile operativo ripete il passaggio n°2. -**2b**. Il responsabile operativo non seleziona nessun camera da pulire: 1. Il sistema notifica il problema al cliente. 2. Il responsabile operativo ripete il passaggio n°2.
Requisiti speciali	- Interfaccia intuitiva per l'effettuazione del check-in in digitale. - Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).
Frequenza di ripetizioni	Ogni volta che il responsabile operativo vuole assegnare un'attività al personale delle pulizia.
Varie	L'identificativo di ogni attività di pulizia è un codice numerico univoco.

1.2.3.12. UC12: GESTISCI STATO DELLE CAMERE

Questo caso d'uso descrive in dettaglio il processo che permette al personale delle pulizie di aggiornare lo stato delle camere in modo preciso ed efficiente, migliorando la coordinazione operativa dell'hotel.

Nome del caso d'uso	**UC12: Gestisci stato delle camere**
Portata	Sistema di Gestione per un Hotel
Livello	Obiettivo utente
Attore primario	Addetto alle pulizie
Parti interessate e interessi	- **Addetto alle pulizie**: desidera aggiornare lo stato delle camere per riflettere le attività completate. - **Hotel**: vuole avere una visione aggiornata e precisa dello stato delle camere per ottimizzare la gestione delle risorse e migliorare il servizio offerto ai clienti.
Pre-condizioni	- L'addetto delle pulizie può effettuare l'accesso al portale di prenotazione. - Le camere hanno stati predefiniti che possono essere aggiornati.
Garanzia di successo	Lo stato delle camere viene aggiornato correttamente nel sistema, ed è immediatamente visibile al personale responsabile.
Scenario principale di successo	**1.** Il responsabile operativo accede alla sezione dedicata alla gestione dello stato delle camere. **2.** Il sistema elenca le camere assegnate con i rispettivi stati attuali. **3.** L'addetto alle pulizie modifica lo stato della camera. **4.** Il sistema aggiorna lo stato della camera e invia una notifica all'addetto delle pulizie di conferma della modifica gestita.
Estensioni	-***a**. Il sistema fallisce arrestandosi improvvisamente: 1. L'addetto alle pulizie riavvia il software. 2. Il sistema ripristina lo stato precedente.
Requisiti speciali	- Interfaccia intuitiva per l'effettuazione del check-in in digitale. - Visualizzazione ottimizzata per dispositivi mobili e desktop. - Sistema di notifica automatica. - Sicurezza dei dati personali durante la consultazione.
Elenco delle varianti tecnologiche e dei dati	- Accesso al sistema tramite interfaccia web. - Supporto in lingua internazionale (inglese).

Frequenza di ripetizioni	Ogni volta che un responsabile operativo assegna un'attività ad un membro del personale delle pulizie.
Varie	- L'identificativo di ogni attività di pulizia è un codice numerico univoco. - Ogni attività di pulizia è gestita andando ad aggiornare lo stato.

1.3. REGOLE DI DOMINIO

1.3.1. ELENCO REGOLE

Le **regole di dominio** rappresentano un insieme di vincoli e linee guida che disciplinano il funzionamento del sistema, garantendo che le operazioni e le interazioni siano conformi alle politiche aziendali e alle necessità operative. Esse derivano dai requisiti identificati e dai casi d'uso precedentemente analizzati, e hanno lo scopo di tradurre i principi gestionali in direttive implementabili.

Ogni regola è definita in base a tre dimensioni principali:

- **ID**, che ne identifica univocamente l'ordine;
- **Modificabilità**, che valuta la probabilità di revisione o modifica della regola;
- **Sorgente**, che descrive la provenienza o il razionale della regola.

L'adozione di queste regole garantisce un'esperienza fluida per i clienti e un'operatività efficiente per il personale alberghiero.

ID	REGOLA	MODIFICABILITA'	SORGENTE
R1	I clienti possono richiedere una richiesta di prenotazione per una determinata tipologia di camera solo in un intervallo di date disponibile per quella tipologia.	Alta	Politica operativa dell'hotel
R2	Il receptionist può approvare o rifiutare una richiesta di prenotazione del cliente.	Alta	Politica operativa dell'hotel
R3	I clienti possono visualizzare le richieste di prenotazione solo dopo la loro approvazione da parte del receptionist.	Media	Politica operativa dell'hotel
R4	I clienti possono effettuare il pagamento di una richiesta di prenotazione solo dopo che essa è stata approvata dal receptionist.	Alta	Politica operativa dell'hotel

R5	Il prezzo totale di una prenotazione deve essere pari al prezzo della tipologia della camera per il numero di giorni richiesti.	Alta	Politica operativa dell'hotel
R6	I clienti possono visualizzare tutte le loro prenotazioni pagate solo se esse: - non sono in attesa di un approvazione di richiesta di modifica; - non sono state annullate dal cliente; - non sono cancellate dal receptionist - non sono passate da un processo di check-out.	Media	Politica interna dell'hotel
R7	I clienti possono richiedere una richiesta di modifica della prenotazione scegliendo un nuovo intervallo di date disponibile per quella tipologia di camera.	Alta	Politica interna dell'hotel
R8	I clienti possono effettuare il check-in digitale il giorno in cui inizia la prenotazione	Alta	Politica interna dell'hotel
R9	I clienti possono effettuare il check-out digitale il giorno in cui termina la prenotazione	Alta	Politica interna dell'hotel
R10	Il responsabile operativo può assegnare al personale delle pulizie solo camere non pulite.	Media	Politica interna dell'hotel

R11	Il cliente non può avere accesso alle informazioni riguardanti le prenotazioni di altri clienti.	Alta	Politica interna dell'hotel
R12	L'addetto alle pulizie non può avere accesso alle informazioni riguardanti le attività assegnate ad altri membri del personale delle pulizie.	Alta	Politica interna dell'hotel

La tabella fornisce una visione complessiva delle regole che disciplinano le operazioni, ordinandole cronologicamente in base agli eventi legati ai casi d'uso analizzati. Ogni regola rappresenta un punto cardine per l'efficace funzionamento del sistema e il rispetto delle politiche aziendali.

1.4. GLOSSARIO

1.4.1. DEFINIZIONI

Le **definizioni** costituiscono un elemento cruciale per garantire una comprensione univoca dei termini utilizzati nel progetto. Attraverso una chiara formulazione, il glossario facilita la comunicazione tra i diversi attori coinvolti e assicura che ogni termine abbia un significato condiviso, riducendo ambiguità e malintesi.

La seguente tabella presenta i termini più rilevanti derivati dall'analisi dei casi d'uso e delle regole di dominio, organizzati in ordine cronologico rispetto agli eventi descritti. Ogni definizione è strutturata per essere comprensibile, rigorosa e applicabile in modo consistente.

TERMINE	DEFINIZIONE
Richiesta di Prenotazione	Proposta inviata dal cliente per riservare una camera in un determinato periodo.
Prenotazione	Conferma di una richiesta di prenotazione da parte dell'hotel, dopo il pagamento effettuato.
Receptionist	Operatore responsabile della gestione delle prenotazioni, modifiche e cancellazioni.
Cliente	Utente del sistema che interagisce per prenotare camere, effettuare pagamenti e richiedere servizi.
Notifica	Messaggio automatico inviato dal sistema per informare il cliente o il personale su azioni rilevanti.
Identificativo Unico	Codice alfanumerico generato automaticamente per identificare una richiesta, prenotazione o attività.
Stato della Prenotazione	Condizione corrente di una prenotazione, espressa tramite valori come "In sospeso", "Confermata", "Annullata".
Check-in Digitale	Procedura che consente al cliente di registrarsi autonomamente al sistema per accedere alla camera.
Check-out Digitale	Procedura che permette al cliente di concludere il soggiorno e liberare la camera in modo digitale.
Camera	Unità abitativa prenotabile nell'hotel, identificata da un numero univoco e specifiche caratteristiche.
Stato della Camera	Informazione relativa alla disponibilità o condizione della camera (es. "Pulita", "Sporca").
Addetto alle Pulizie	Operatore incaricato di gestire la pulizia e la manutenzione delle camere.

Responsabile Operativo	Figura incaricata di assegnare le attività al personale e monitorare l'efficienza operativa.
Attività	Compito assegnato al personale, come pulizia, manutenzione o accoglienza, con priorità definita.
Disponibilità Camere	Informazione sul numero di camere libere in un determinato periodo, visibile a clienti e receptionist.
Modifica Prenotazione	Processo di aggiornamento di uno o più dettagli di una prenotazione esistente, su richiesta del cliente.

2. FASE DI ELABORAZIONE

2.1. TECNOLOGIE UTILIZZATE

2.1.1. INTRODUZIONE

Il **Sistema di Gestione per un Hotel** è stato progettato con un ecosistema tecnologico moderno, selezionato per garantire scalabilità, sicurezza e manutenibilità. L'architettura segue il paradigma **MVC (Model-View-Controller)**, separando la logica di presentazione, di controllo e di accesso ai dati, e si basa su una combinazione di tecnologie consolidate nel settore dello sviluppo software.

2.1.2. FRONT-END

L'interfaccia utente è stata realizzata utilizzando **HTML, CSS e JavaScript**, tecnologie fondamentali per il web development, supportate dall'adozione di **TailwindCSS**. Quest'ultimo, un framework CSS utility-first, consente di costruire interfacce moderne e responsive con uno stile altamente personalizzabile e prestazioni ottimizzate.
L'utilizzo di **JavaScript** permette di implementare un'esperienza dinamica e interattiva, garantendo un'interfaccia fluida e reattiva, migliorata dall'uso di tecniche asincrone per la comunicazione con il backend tramite **RESTful API**.

2.1.3. BACK-END

Il cuore applicativo del sistema è sviluppato in **Java**, sfruttando il framework **Spring Boot** per garantire un'architettura modulare, sicura ed efficiente. **Spring Boot** offre una serie di funzionalità avanzate, tra cui **gestione delle dipendenze**, **configurazioni automatiche**, e un ambiente scalabile ideale per applicazioni di livello enterprise.
La progettazione del backend segue il paradigma **RESTful**, esponendo API strutturate per la comunicazione con il frontend e garantendo interoperabilità con altri sistemi.

L'architettura è progettata secondo il modello **MVC (Model-View-Controller)**, una scelta che facilita la separazione delle responsabilità, migliorando la leggibilità del codice e semplificando la manutenzione. La gestione delle dipendenze e il build system sono affidati a **Maven**, uno strumento standard per la configurazione e il packaging delle applicazioni Java.

2.1.4. DATABASE E ORM

Per la gestione dei dati, il sistema utilizza **MySQL**, un database relazionale ampiamente diffuso e apprezzato per le sue prestazioni e affidabilità. L'interazione tra l'applicazione e il database è gestita tramite **Hibernate** e **JPA (Java Persistence API)**, che insieme consentono di adottare un paradigma **ORM (Object-Relational Mapping)**. Questo approccio facilita la persistenza dei dati, riduce la necessità di scrivere query SQL manualmente e garantisce maggiore portabilità tra database relazionali.

2.1.5. TESTING E QUALITA' DEL CODICE

Per assicurare l'affidabilità e la robustezza del software, il sistema implementa test automatici utilizzando **JUnit**, uno dei framework più diffusi per il testing in Java. I test

unitari, supportati da **Mockito** per la simulazione delle dipendenze, verificano il corretto funzionamento delle singole componenti in modo isolato. L'adozione di un processo di testing continuo permette di individuare tempestivamente eventuali regressioni, contribuendo a mantenere elevata la stabilità e la qualità complessiva del codice.

2.1.6. VERSION CONTROL E STRUMENTI DI SVILUPPO

Il codice sorgente è gestito con **Git**, un sistema di controllo versione distribuito che permette di tracciare ogni modifica e facilitare il lavoro collaborativo. Il repository è ospitato su **GitHub**, che offre strumenti avanzati per la gestione del versionamento, il code review e l'integrazione continua.
L'ambiente di sviluppo principale utilizzato per la scrittura e la gestione del codice è **Visual Studio Code (VSCode)**, scelto per la sua leggerezza, versatilità e il supporto a un'ampia gamma di estensioni dedicate allo sviluppo Java e web.

2.1.7. VANTAGGI DI QUESTE TECNOLOGIE NEL PROGETTO

L'adozione di queste tecnologie consente di ottenere un sistema altamente **modulare, scalabile e sicuro**, capace di rispondere alle esigenze di un ambiente alberghiero in continua evoluzione. L'integrazione di strumenti moderni per lo sviluppo, il testing e il versionamento assicura che il software sia facilmente **manutenibile ed estensibile**, offrendo un'esperienza utente fluida e un'infrastruttura backend robusta e performante.

2.1.8. STRUMENTI DI MODELLAZIONE

Per la realizzazione dei diagrammi impiegati nel progetto, è stato utilizzato **Astah**, uno strumento di modellazione **UML** ampiamente riconosciuto per la sua versatilità e facilità d'uso. L'impiego di Astah ha permesso di creare diagrammi chiari e strutturati, fondamentali per rappresentare le diverse componenti del sistema e le loro interazioni in modo efficace.
La scelta di questo strumento è stata dettata dalla necessità di garantire coerenza e precisione nella documentazione visiva, facilitando la comprensione dell'architettura.
I diagrammi generati includono rappresentazioni delle classi, delle sequenze e dei casi d'uso, offrendo una panoramica completa della struttura e del funzionamento del sistema.

2.2. MVC (MODEL-VIEW-CONTROLLER)

2.2.1. INTRODUZIONE

L'architettura **Model-View-Controller (MVC)** è un paradigma di progettazione software ampiamente adottato per lo sviluppo di applicazioni modulari e scalabili. Il **Sistema di Gestione per un Hotel** si basa su un'implementazione estesa di MVC, che integra anche i livelli **Service** e **Repository**, al fine di garantire una chiara separazione delle responsabilità, una gestione efficiente della logica applicativa e un'interazione strutturata con il database.

Grazie a questa suddivisione, il sistema può evolversi facilmente, supportando la **manutenibilità a lungo termine**, l'**indipendenza tra le componenti** e una **chiara organizzazione del codice**. Questo approccio consente di ridurre la complessità dello

sviluppo, migliorare la collaborazione tra team e ottimizzare le prestazioni dell'applicazione.

L'architettura adottata è composta da **cinque componenti principali**:

- **Model**: definisce la struttura dei dati e la logica di dominio.
- **View**: gestisce la presentazione e l'interazione con l'utente.
- **Controller**: coordina la comunicazione tra View e Service.
- **Service**: implementa la logica di business dell'applicazione.
- **Repository**: gestisce l'accesso ai dati e la persistenza nel database.

Questa configurazione permette di **ottimizzare la modularità**, garantendo che ogni componente sia altamente coeso e indipendente dagli altri, facilitando **l'estensibilità del sistema** e la sua **integrazione con nuove funzionalità**.

2.2.2. MODEL: LA STRUTTURA DEI DATI E LA LOGICA DEL DOMINIO

Il **Model** è il livello dell'applicazione che definisce la struttura dei dati e la logica di dominio. In questo progetto, il Model è implementato in **Java** utilizzando **JPA (Java Persistence API)** e **Hibernate** per l'Object-Relational Mapping (**ORM**), facilitando così la gestione delle entità del database.

Ogni entità all'interno del sistema è una rappresentazione di una tabella nel database **MySQL**, con annotazioni che definiscono le relazioni e i vincoli di integrità. Questo approccio permette di evitare scrittura manuale di query SQL, migliorando la coerenza e la sicurezza dei dati.

L'utilizzo del **Model** consente di astrarre la persistenza dei dati, lasciando ai livelli superiori (Service e Controller) la gestione della logica applicativa e delle interazioni con l'utente.

2.2.3. VIEW: LA PRESENTAZIONE GRAFICA E L'INTERFACCIA UTENTE

La **View** è il livello responsabile della presentazione grafica e dell'interazione con l'utente. Nel sistema, l'interfaccia utente è sviluppata con **HTML, CSS e JavaScript**, utilizzando **TailwindCSS** per garantire uno stile moderno e responsivo.

L'approccio adottato per la View è completamente **decoupled dal backend**, consentendo una comunicazione efficiente tramite **RESTful API**. Questo modello garantisce una maggiore flessibilità, permettendo di migliorare o sostituire il frontend senza impattare la logica di business.

Le principali caratteristiche della **View** includono:

- **Responsive design**, ottimizzato per dispositivi desktop e mobile.
- **Interazione dinamica**, con aggiornamenti asincroni tramite JavaScript e chiamate API.
- **Esperienza utente intuitiva**, con una chiara separazione tra layout e dati dinamici.

Questa struttura facilita l'integrazione con eventuali future applicazioni mobile o altri sistemi client.

2.2.4. CONTROLLER: IL GESTORE DELLE RICHIESTE E DELLE RISPOSTE

Il **Controller** è il livello intermedio tra la View e i livelli applicativi, responsabile della gestione delle richieste HTTP e della loro elaborazione. Nel sistema, i controller sono implementati con **Spring Boot**, sfruttando le funzionalità di **Spring MVC** per definire endpoint RESTful.

Le principali responsabilità del **Controller** sono:

- **Ricevere le richieste HTTP** dal frontend ed elaborarle.
- **Delegare la logica applicativa al Service**, evitando che la logica sia scritta direttamente nel controller.
- **Restituire risposte formattate in JSON**, compatibili con il frontend.

L'adozione di questo livello consente di mantenere il codice pulito e organizzato, garantendo una maggiore chiarezza e coesione tra le diverse componenti.

2.2.5. SERVICE: IL CUORE DELLA LOGICA APPLICATIVA

Il livello **Service** è un'estensione dell'architettura MVC tradizionale, introdotta per separare la logica di business dai Controller. Questo strato intermedio assicura che il **Controller** si limiti alla gestione delle richieste e delega l'elaborazione dei dati e l'interazione con il database ai **Service**.

I principali vantaggi di un **Service Layer** includono:

- **Separazione delle responsabilità**, garantendo che il Controller sia leggero e focalizzato sulla gestione delle richieste.
- **Riutilizzabilità del codice**, permettendo di centralizzare la logica e riutilizzarla in più punti dell'applicazione.
- **Maggiore modularità**, facilitando future espansioni senza impattare direttamente il Controller o il Repository.

I **Service** interagiscono con i **Repository**, ottenendo i dati necessari e applicando le regole di business prima di restituire le risposte ai Controller.

2.2.6. REPOSITORY: LA GESTIONE DELLA PERSISTENZA E L'ACCESSO DEI DATI

Il livello **Repository** è responsabile della comunicazione con il database. Nel sistema, è implementato utilizzando **Spring Data JPA**, che fornisce un'astrazione potente per le operazioni di persistenza.

I **Repository** offrono una serie di funzionalità chiave:

- **Astrazione delle query SQL**, permettendo di eseguire operazioni CRUD senza dover scrivere SQL manualmente.
- **Maggiore efficienza nella gestione dei dati**, con supporto per caching e ottimizzazioni delle query.
- **Possibilità di personalizzare le query**, definendo metodi specifici per esigenze avanzate.

Grazie all'uso di **Spring Data JPA**, il sistema può eseguire operazioni complesse con poche righe di codice, garantendo efficienza e sicurezza nella gestione della persistenza.

2.2.7. VANTAGGI DELL'ARCHITETTURA MVC NEL PROGETTO

L'adozione del pattern **MVC** nel **Sistema di Gestione per un Hotel** ha portato numerosi vantaggi, tra cui:

- **Separazione delle responsabilità**: permette di sviluppare e manutenere ogni componente in modo indipendente, migliorando l'organizzazione del codice.
- **Scalabilità**: la struttura modulare consente di estendere facilmente il sistema con nuove funzionalità senza compromettere il codice esistente.
- **Facilità di debugging e testing**: la divisione tra Model, View e Controller facilita il testing delle singole componenti, garantendo una maggiore affidabilità del software.
- **Manutenibilità**: L'applicazione può essere aggiornata e migliorata con interventi mirati senza dover riscrivere l'intero sistema.

L'implementazione di **MVC con Spring Boot** rappresenta dunque una soluzione ideale per un progetto enterprise come questo, assicurando un **design solido, flessibile e facilmente estensibile**, capace di adattarsi alle esigenze dinamiche del settore alberghiero.

2.3. ORM (OBJECT-RELATIONAL MAPPING)

2.3.1. INTRODUZIONE

Nello sviluppo di applicazioni basate su database relazionali, la gestione della persistenza dei dati rappresenta una sfida fondamentale. La necessità di interfacciarsi con il database in modo strutturato, evitando codice SQL complesso e ridondante, ha portato all'adozione del paradigma **Object-Relational Mapping (ORM)**.

Il **Sistema di Gestione per un Hotel** sfrutta il potente framework **Hibernate**, in combinazione con **JPA (Java Persistence API)**, per fornire un'astrazione completa rispetto al database **MySQL**. Questo approccio consente di modellare le entità applicative in modo naturale, mantenendo un elevato livello di coesione tra la logica di business e la persistenza dei dati, senza compromettere le prestazioni.

L'ORM semplifica lo sviluppo riducendo la dipendenza da query SQL scritte manualmente, migliorando la leggibilità e la manutenibilità del codice, oltre a garantire un'integrazione efficiente tra le entità applicative e il database.

2.3.2. HIBERNATE E JPA: LA GESTIONE AVANZATA DELLA PERSISTENZA

Il **Sistema di Gestione per un Hotel** implementa la persistenza dei dati attraverso **Hibernate**, il principale framework ORM per Java, sfruttando l'interfaccia standard di **JPA**. Hibernate consente di trasformare gli oggetti Java in rappresentazioni persistenti, che vengono poi mappate sulle tabelle del database in modo trasparente ed efficiente.

L’uso combinato di Hibernate e JPA garantisce numerosi vantaggi:

- **Astrazione dal database**: il codice non è vincolato a un DBMS specifico, facilitando eventuali migrazioni future.
- **Automazione delle operazioni CRUD**: grazie all’integrazione con **Spring Data JPA**, la gestione delle operazioni di lettura, scrittura, modifica ed eliminazione dei dati avviene in modo automatico.
- **Supporto per relazioni complesse**: il framework semplifica la gestione di associazioni come **uno-a-molti**, **molti-a-uno** e **molti-a-molti**, riducendo il rischio di errori e migliorando l’organizzazione del codice.
- **Ottimizzazione delle query**: grazie a tecniche avanzate come il **lazy loading**, la cache di secondo livello e il batching, il sistema minimizza il numero di operazioni sul database, migliorando significativamente le prestazioni.

2.3.3. MAPPATURA DELLE ENTITA’ E GESTIONE DELLE RELAZIONI

Il **modello di dati** nel **Sistema di Gestione per un Hotel** è progettato per garantire una rappresentazione chiara e coerente delle informazioni gestite dall’applicazione. Le classi Java che definiscono il dominio applicativo sono annotate con le specifiche di **JPA**, consentendo una gestione efficace della persistenza.

L’uso delle annotazioni ***@Entity***, ***@Table***, ***@Id***, ***@Column***, ***@GeneratedValue*** permette di configurare facilmente la corrispondenza tra classi e tabelle del database. Inoltre, l’adozione di annotazioni come ***@OneToMany***, ***@ManyToOne*** **e** ***@ManyToMany*** facilita la gestione delle relazioni tra le diverse entità, eliminando la necessità di scrivere query SQL complesse.

Questo approccio garantisce che la logica di business possa operare direttamente su oggetti di alto livello, senza dover interagire con la struttura relazionale sottostante.

2.3.4. STRATEGIE DI CARICAMENTO E INTERROGAZIONE DEI DATI

Uno degli aspetti più critici della gestione della persistenza è l’ottimizzazione delle prestazioni. Hibernate mette a disposizione diverse strategie per migliorare l’efficienza delle operazioni sul database:

- **Lazy Loading & Eager Fetching**: permette di controllare il caricamento dei dati in modo selettivo, riducendo il numero di query eseguite.
- **JPQL (Java Persistence Query Language)**: fornisce un linguaggio più espressivo e flessibile rispetto all’SQL, facilitando l’interrogazione delle entità.

Grazie a queste ottimizzazioni, il sistema garantisce **prestazioni elevate, riduzione dei tempi di risposta e scalabilità**, anche in scenari con un alto volume di dati.

2.3.5. RUOLO DEL REPOSITORY E INTEGRAZIONE CON SPRING DATA JPA

Nel sistema, il livello **Repository** svolge un ruolo cruciale nella gestione della persistenza, fungendo da intermediario tra la logica applicativa e il database. Grazie a **Spring Data JPA**, le operazioni CRUD vengono eseguite in modo automatico, riducendo la quantità di codice necessario per interagire con il database.

L’uso di interfacce estende le funzionalità fornite da JPA, consentendo di definire metodi personalizzati per interrogazioni più complesse. Questo approccio migliora la **modularità** del sistema e permette un migliore isolamento della logica di accesso ai dati.

2.3.6. VANTAGGI DEL PARADIGMA ORM NEL PROGETTO

L’adozione del paradigma **ORM** nel **Sistema di Gestione per un Hotel** ha permesso di ottenere un’architettura flessibile, scalabile e altamente manutenibile. Grazie a **Hibernate e JPA**, il sistema beneficia di un’astrazione completa rispetto al database, riducendo significativamente la complessità della gestione dei dati e garantendo al contempo elevata efficienza e sicurezza.

- **Maggiore produttività**: eliminazione della necessità di scrivere SQL manualmente, con operazioni CRUD automatizzate.
- **Indipendenza dal database**: facilita la migrazione tra diversi DBMS senza modificare la logica applicativa.
- **Modularità e riusabilità**: il codice ORM è più pulito, organizzato e facilmente riutilizzabile.

L'integrazione di **Hibernate e JPA** in combinazione con **Spring Boot e Spring Data JPA** rende il **Sistema di Gestione per un Hotel** un prodotto altamente **performante, sicuro e pronto per eventuali evoluzioni future**, garantendo una gestione avanzata ed efficace della persistenza dei dati.

2.4. APPROCCIO ITERATIVO

Lo sviluppo del progetto ha seguito un approccio iterativo, suddividendo le funzionalità in cicli di sviluppo progressivi, ciascuno finalizzato all’implementazione di requisiti fondamentali specifici. Questo metodo ha permesso una costruzione incrementale del sistema, facilitando il monitoraggio continuo dei progressi e consentendo una gestione efficace delle priorità e delle eventuali modifiche durante il ciclo di vita del progetto.

1° Iterazione
La fase iniziale si è concentrata sulla configurazione delle risorse e dei servizi essenziali del sistema, garantendo una base solida per le successive implementazioni. In questa iterazione, sono stati sviluppati i casi d’uso principali legati alla gestione delle prenotazioni, ovvero:

- **Caso d’uso di avviamento:** Configura le risorse e i servizi essenziali del sistema.
- **Caso d’uso UC1**: Effettua richiesta di prenotazione.
- **Caso d’uso UC2**: Gestisci richiesta di prenotazione.
- **Caso d’uso UC3**: Effettua pagamento.
- **Caso d’uso UC4**: Visualizza dettagli della prenotazione.

2° Iterazione
In questa fase, l’attenzione si è spostata sulla gestione delle modifiche e delle cancellazioni delle prenotazioni, aggiungendo funzionalità più avanzate per migliorare l’esperienza utente e la flessibilità del sistema:

- **Caso d’uso UC5**: Richiedi modifica alla prenotazione.
- **Caso d’uso UC6**: Modifica dettagli della prenotazione.

- **Caso d’uso UC7**: Annulla prenotazione.
- **Caso d’uso UC8**: Cancella prenotazione.

3° Iterazione
Questa iterazione ha introdotto i processi di check-in e check-out digitali, con l’obiettivo di semplificare l’esperienza dei clienti e ridurre le operazioni manuali:

- **Caso d’uso UC9**: Check-in digitale.
- **Caso d’uso UC10**: Check-out digitale.

4° Iterazione
L’ultima iterazione si è focalizzata sulla gestione operativa interna dell’hotel, con l’obiettivo di migliorare l’efficienza delle attività del personale e la gestione delle risorse:

- **Caso d’uso UC11**: Assegna attività del personale.
- **Caso d’uso UC12**: Gestisci stato delle camere.

Questo approccio iterativo ha garantito uno sviluppo graduale e controllato, permettendo di validare ogni fase attraverso test e feedback continui. Ogni iterazione ha consolidato le funzionalità precedenti, assicurando un’evoluzione coerente e progressiva del sistema, in linea con gli obiettivi del progetto.

2.5. ANALISI

2.5.1. INTRODUZIONE

L’obiettivo della **fase di analisi** è definire in modo formale e dettagliato i componenti strutturali e funzionali del sistema, garantendo una chiara separazione tra i diversi elementi del dominio e le interazioni tra gli attori coinvolti.

A tal fine, in questa fase vengono sviluppati i seguenti elementi:

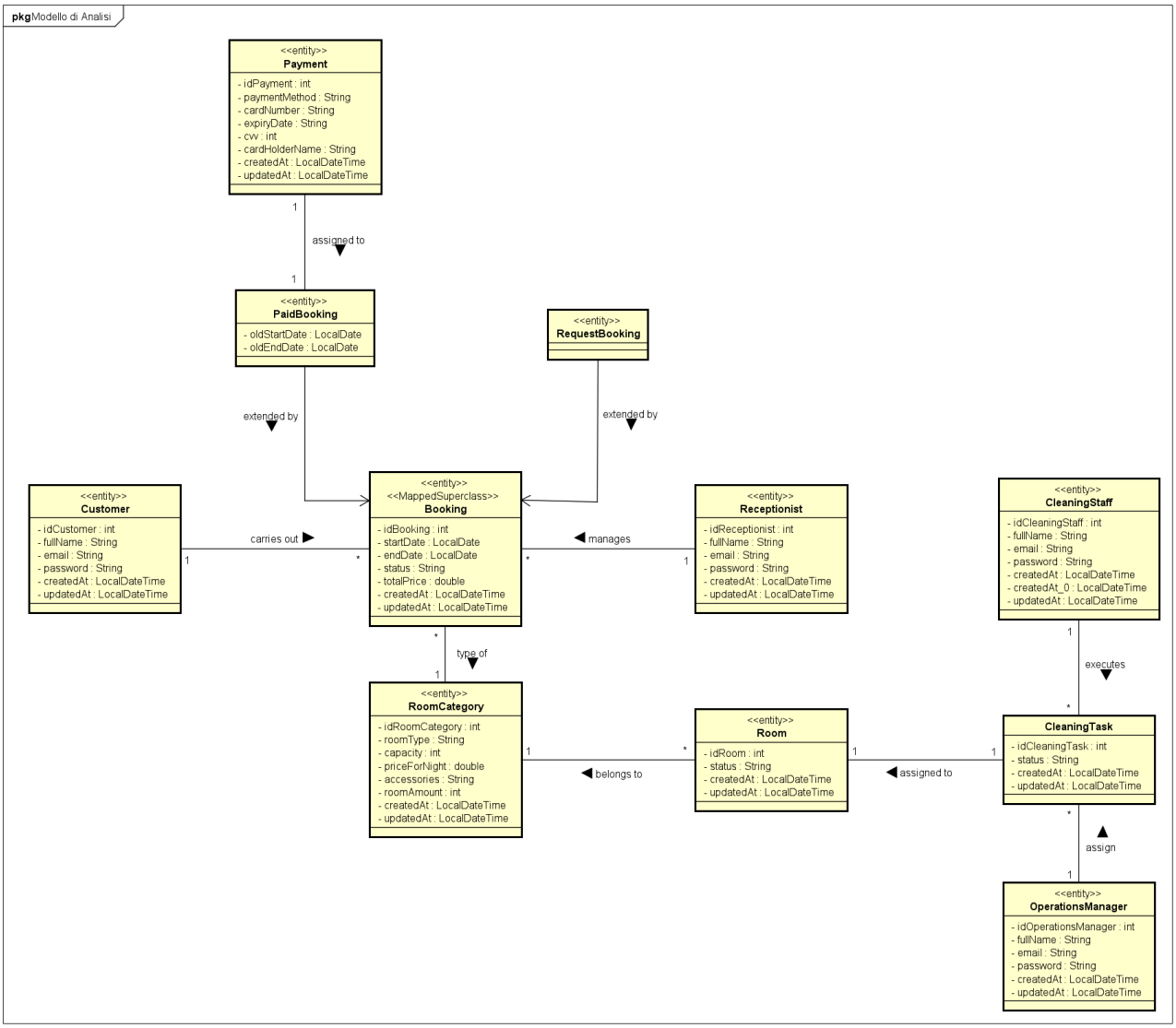
- **Modello di Dominio**: rappresenta la struttura concettuale del sistema, delineando le principali entità coinvolte e le loro relazioni. Questo modello costituisce la base per la progettazione e lo sviluppo del software, garantendo la coerenza dei dati e delle operazioni previste.
- **Diagrammi di Sequenza di Sistema (SSD)**: descrivono le interazioni tra gli attori e il sistema nei vari scenari d’uso, fornendo una rappresentazione dettagliata del flusso degli eventi e dei messaggi scambiati.
- **Contratti delle Operazioni**: formalizzano il comportamento atteso delle principali operazioni del sistema, specificando pre-condizioni e post-condizioni, al fine di garantire la consistenza del sistema e la corretta gestione delle informazioni.

Questa fase di analisi è essenziale per stabilire una solida base metodologica per lo sviluppo del sistema, assicurando che la successiva fase di progettazione sia aderente ai requisiti identificati e che il sistema risponda in modo efficace alle esigenze degli utenti finali.

2.5.2. MODELLO DI DOMINIO

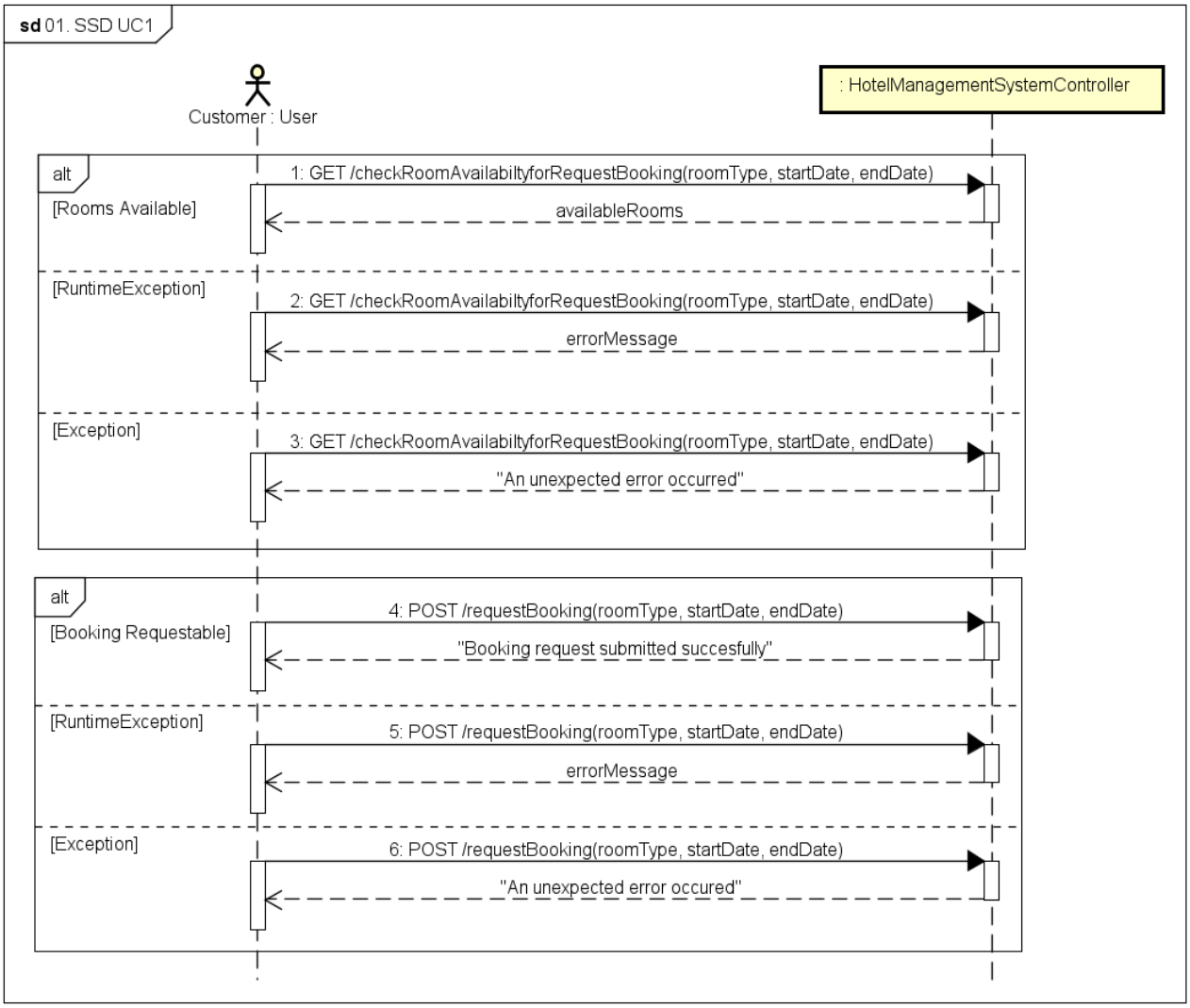
È possibile identificare le seguenti classi concettuali:

- **Customer**: rappresenta un cliente del sistema che effettua una prenotazione presso l'hotel.
- **Receptionist**: rappresenta un receptionist dell'hotel, responsabile della gestione delle prenotazioni e delle richieste di modifica inviate dai clienti.
- **OperationsManager**: rappresenta il manager delle operazioni dell'hotel, responsabile dell'assegnazione delle attività di pulizia al personale delle pulizie.
- **CleaningStaff**: rappresenta un membro del personale delle pulizie, responsabile della pulizia delle stanze.
- **RoomCategory**: rappresenta una categoria di stanza disponibile nell'hotel, specificando il tipo e il numero di stanze disponibili.
- **Room**: rappresenta una stanza specifica all'interno dell'hotel, appartenente a una determinata categoria di stanza.
- **Booking**: rappresenta una prenotazione generica all'interno del sistema, che può essere una richiesta di prenotazione o una prenotazione pagata.
- **RequestBooking**: rappresenta una richiesta di prenotazione inviata dal cliente.
- **PaidBooking**: rappresenta una prenotazione pagata dal cliente.
- **Payment**: rappresenta un pagamento effettuato da un cliente per confermare una prenotazione.
- **CleaningTask**: rappresenta un'attività di pulizia assegnata a un membro del personale delle pulizie.


pkgModello di Analisi
<<entity>>
Payment
- idPayment : int
- paymentMethod : String
- cardNumber : String
- expiryDate : String
- cvv : int
- cardHolderName : String
- createdAt : LocalDateTime
- updatedAt : LocalDateTime
1
assigned to
1
<<entity>>
PaidBooking
- oldStartDate : LocalDate
- oldEndDate : LocalDate
<<entity>>
RequestBooking
extended by
extended by
<<entity>>
Customer
- idCustomer : int
- fullName : String
- email : String
- password : String
- createdAt : LocalDateTime
- updatedAt : LocalDateTime
carries out
1
*
<<entity>>
<<MappedSuperclass>>
Booking
- idBooking : int
- startDate : LocalDate
- endDate : LocalDate
- status : String
- totalPrice : double
- createdAt : LocalDateTime
- updatedAt : LocalDateTime
manages
*
1
<<entity>>
Receptionist
- idReceptionist : int
- fullName : String
- email : String
- password : String
- createdAt : LocalDateTime
- updatedAt : LocalDateTime
<<entity>>
CleaningStaff
- idCleaningStaff : int
- fullName : String
- email : String
- password : String
- createdAt : LocalDateTime
- createdAt_0 : LocalDateTime
- updatedAt : LocalDateTime
*
1
type of
1
executes
*
<<entity>>
RoomCategory
- idRoomCategory : int
- roomType : String
- capacity : int
- priceForNight : double
- accessories : String
- roomAmount : int
- createdAt : LocalDateTime
- updatedAt : LocalDateTime
1
*
belongs to
<<entity>>
Room
- idRoom : int
- status : String
- createdAt : LocalDateTime
- updatedAt : LocalDateTime
1
1
assigned to
CleaningTask
- idCleaningTask : int
- status : String
- createdAt : LocalDateTime
- updatedAt : LocalDateTime
*
1
assign
<<entity>>
OperationsManager
- idOperationsManager : int
- fullName : String
- email : String
- password : String
- createdAt : LocalDateTime
- updatedAt : LocalDateTime

2.5.3. CASO D'USO UC1, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC1 è il seguente:



2.5.4. CASO D'USO UC1, CONTRATTI DELLE OPERAZIONI

2.5.4.1. *checkRoomAvailabiltyforRequestBooking*

L'operazione di sistema *checkRoomAvailabiltyforRequestBooking* consente di calcolare il numero di stanza disponibili per un determinato tipo di stanza e un intervallo di date.

Operazione:
GET /checkRoomAvailabiltyforRequestBooking(roomType, startDate, endDate)

Riferimenti:
Caso d'uso UC1: Effettua una richiesta di Prenotazione.

Pre-condizioni:

- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.

- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esistono istanze *roomCategories* dell'entità *RoomCategory*.

Post-condizioni:
- Se ci sono state stanze disponibili per la *roomCategory* selezionata dall'utente nell'intervallo di date scelto, il sistema ha restituito il *availableRooms*.
- Se non ci sono state stanze disponibili per la *roomCategory* selezionata dall'utente nell'intervallo di date scelto, il sistema ha restituito un *availableRooms* pari a "0".
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.4.2. *requestBooking*

L'operazione di sistema *requestBooking* consente di creare una richiesta di prenotazione.

Operazione:
POST /requestBooking (roomType, startDate, endDate)

Riferimenti:
Caso d'uso UC1: Effettua una richiesta di prenotazione.

Pre-condizioni:
- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un c*ustomer* ha effettuato l'accesso al portale di prenotazione.
- Il *customer* ha selezionato un range di date per una determinata *roomCategory*.

Post-condizioni:
- Se la *roomCategory* è risultata disponibile:
    - È stata creata una nuova istanza *requestBooking* dell'entità *RequestBooking*.
    - Gli attributi di *requestBooking* sono stati inizializzati tramite i loro setter, in particolare per due attributi:
        - *totalPrice:* è stato calcolato sulla base dei giorni inseriti nel form di richiesta di prenotazione (*priceForNight * days*).
        - *status*: e stato settato come: "*Pending Approval"*.
    - È stata salvata l'istanza *requestBooking* come record nella tabella del database associata all'entità *RequestBooking.*
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.5. CASO D'USO UC2, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC2 è il seguente:

2.5.6. CASO D'USO UC2, CONTRATTI DELLE OPERAZIONI

2.5.6.1. *getRequestBooking*

L'operazione di sistema *getRequestBooking* consente di visualizzare la lista delle richieste di prenotazione.

Operazione:
GET /getRequestBooking()

Riferimenti:
Caso d'uso UC2: Gestione richiesta di prenotazione.

Pre-condizioni:

- Esiste un record *receptionist* nella tabella del database associata all'entità *Receptionist*.
- Un *receptionist* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *requestBooking* nella tabella del database associata all'entità *RequestBooking* con un valore *status* settato come: *"Pending Approval"*.

Post-condizioni:

- Sono stati restituiti tutti i record *requestBooking* della tabella del database associata all'entità *RequestBooking* con un valore *status* settato come: "*Pending Approval"*.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.6.2. *manageRequestBooking*

L'operazione di sistema *manageRequestBooking* consente di gestire una richiesta di prenotazione.

Operazione:
POST /manageRequestBooking(status, idBooking)

Riferimenti:
Caso d'uso UC2: Gestione richiesta di prenotazione.

Pre-condizioni:

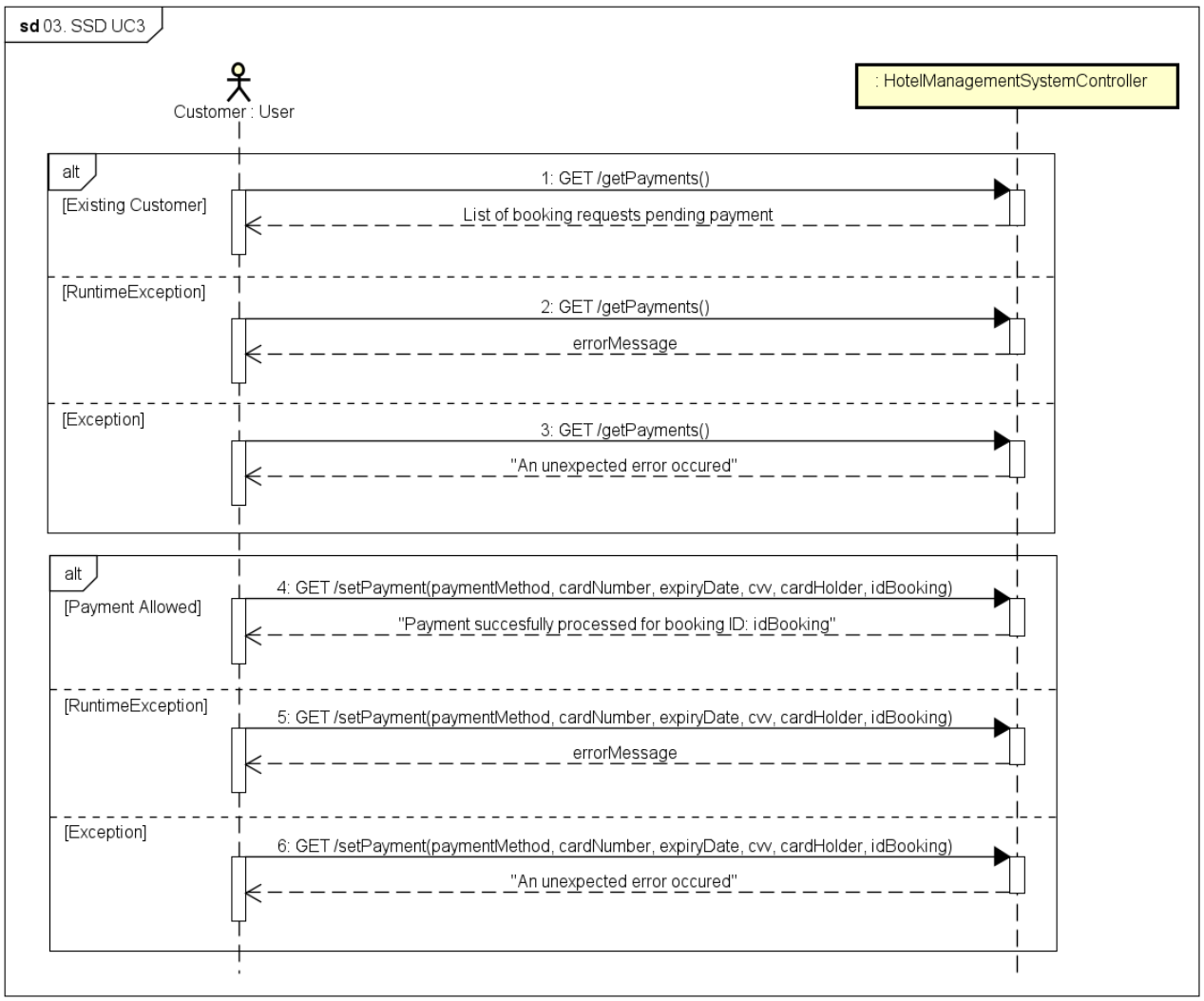
- Esiste un record *receptionist* nella tabella del database associata all'entità *Receptionist*.
- Un *receptionist* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *requestBooking* nella tabella del database associata all'entità *RequestBooking* con un valore *status* settato come: *"Pending Approval"*.

Post-condizioni:

- Il *receptionist* ha selezionato una *requestBooking* e ha ridefinito il suo attributo *status* tramite il suo setter, in particolare:
    - *status*: è stato settato come: *"Approved"* o come "*Rejected*".
- È stata salvata la modifica dell'istanza *requestBooking* con conseguente aggiornamento del record corrispondente nella tabella del database associata all'entità *RequestBooking.*
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.7. CASO D'USO UC3, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC3 è il seguente:



2.5.8. CASO D'USO UC3, CONTRATTI DELLE OPERAZIONI

2.5.8.1. *getPayments*

L'operazione di sistema *getPayments* consente di visualizzare la lista delle richieste di prenotazione valide per il pagamento.

Operazione:
GET /getPayments()

Riferimenti:
Caso d'uso UC3: Effettua pagamento.

Pre-condizioni:

- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.

- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *requestBooking* nella tabella del database associata all'entità *RequestBooking* con un valore *status* settato come: *"Approved"* e associato al *customer* attualmente loggato nel portale di prenotazione.

Post-condizioni:
- Sono stati restituiti tutti i record *requestBooking* della tabella del database associata all'entità *RequestBooking* con un valore *status* settato come: *"Approved"* associati al *customer* attualmente loggato nel portale di prenotazione.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.8.2. *setPayment*

L'operazione di sistema *setPayment* consente di effettuare il pagamento di una richiesta di prenotazione.

Operazione:
GET /setPayment(paymentMethod, cardNumber, expiryDate, cvv, cardHolder, idBooking)

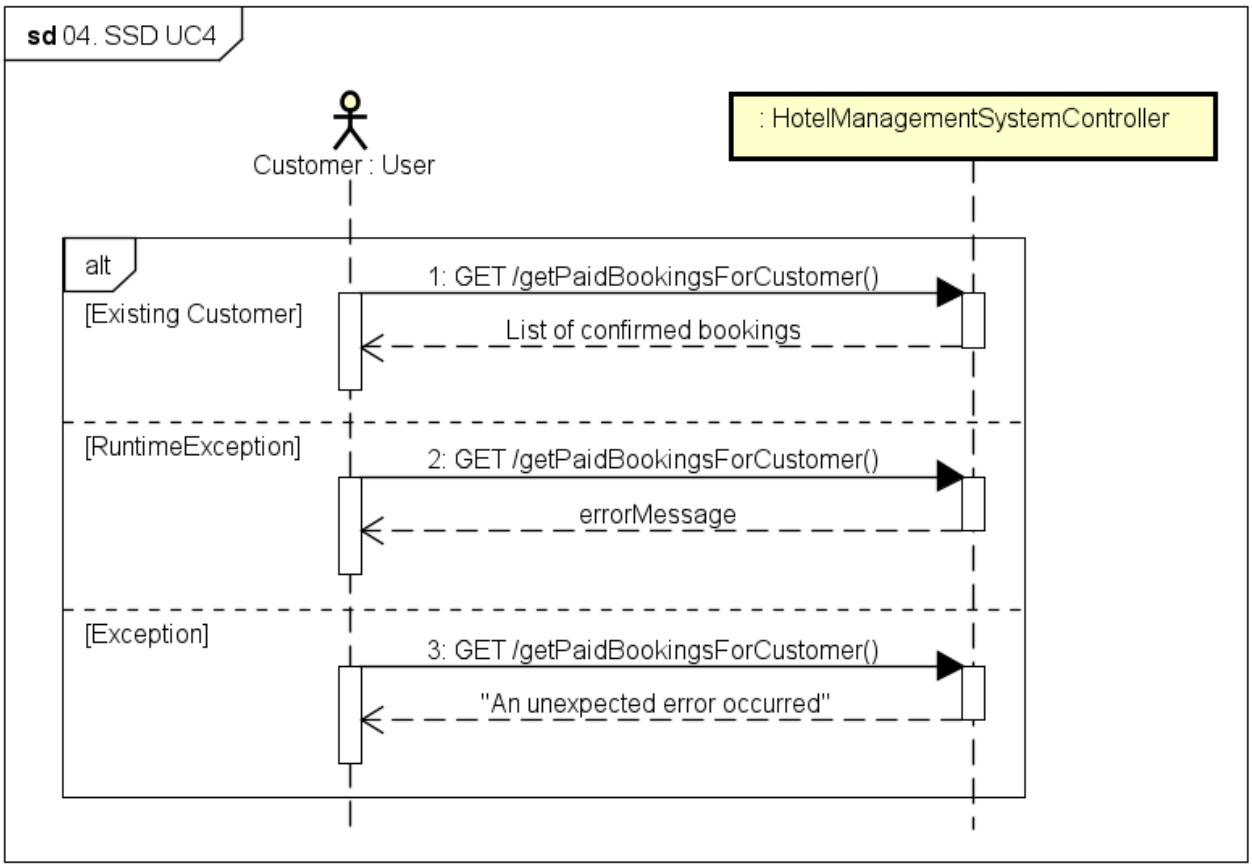
Riferimenti:
Caso d'uso UC3: Effettua pagamento.

Pre-condizioni:
- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *requestBooking* nella tabella del database associata all'entità *RequestBooking* con un valore *status* settato come: *"Approved"* e associato al *customer* attualmente loggato nel portale di prenotazione.

Post-condizioni:
- Il *customer* ha selezionato una *requestBooking*:
  - È stata creata una nuova istanza *paidBooking* dell'entità *paidBooking*.
  - Gli attributi di *paidBooking* sono stati inizializzati tramite i loro setter.
  - È stata salvata l'istanza *paidBooking* come record nella tabella del database associata all'entità *PaidBooking.*
  - È stato eliminato il record requestBooking nella tabella del database associata all'entità RequestBooking.
  - È stata creata una nuova istanza *payment* dell'entità *Payment*.
  - Gli attributi di *payment* sono stati inizializzati tramite i loro setter.
  - È stata salvata l'istanza *payment* come record nella tabella del database associata all'entità *Payment.*
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.9. CASO D'USO UC4, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC4 è il seguente:



2.5.10. CASO D'USO UC4, CONTRATTI DELLE OPERAZIONI

2.5.10.1. *getPaidBookingForCustomer*

L'operazione di sistema *getPaidBookingForCustomer* consente di visualizzare la lista delle prenotazioni confermate da un pagamento precedente.

Operazione:
GET /getPaidBookingForCustomer()

Riferimenti:
Caso d'uso UC4: Visualizza dettagli della prenotazione.

Pre-condizioni:

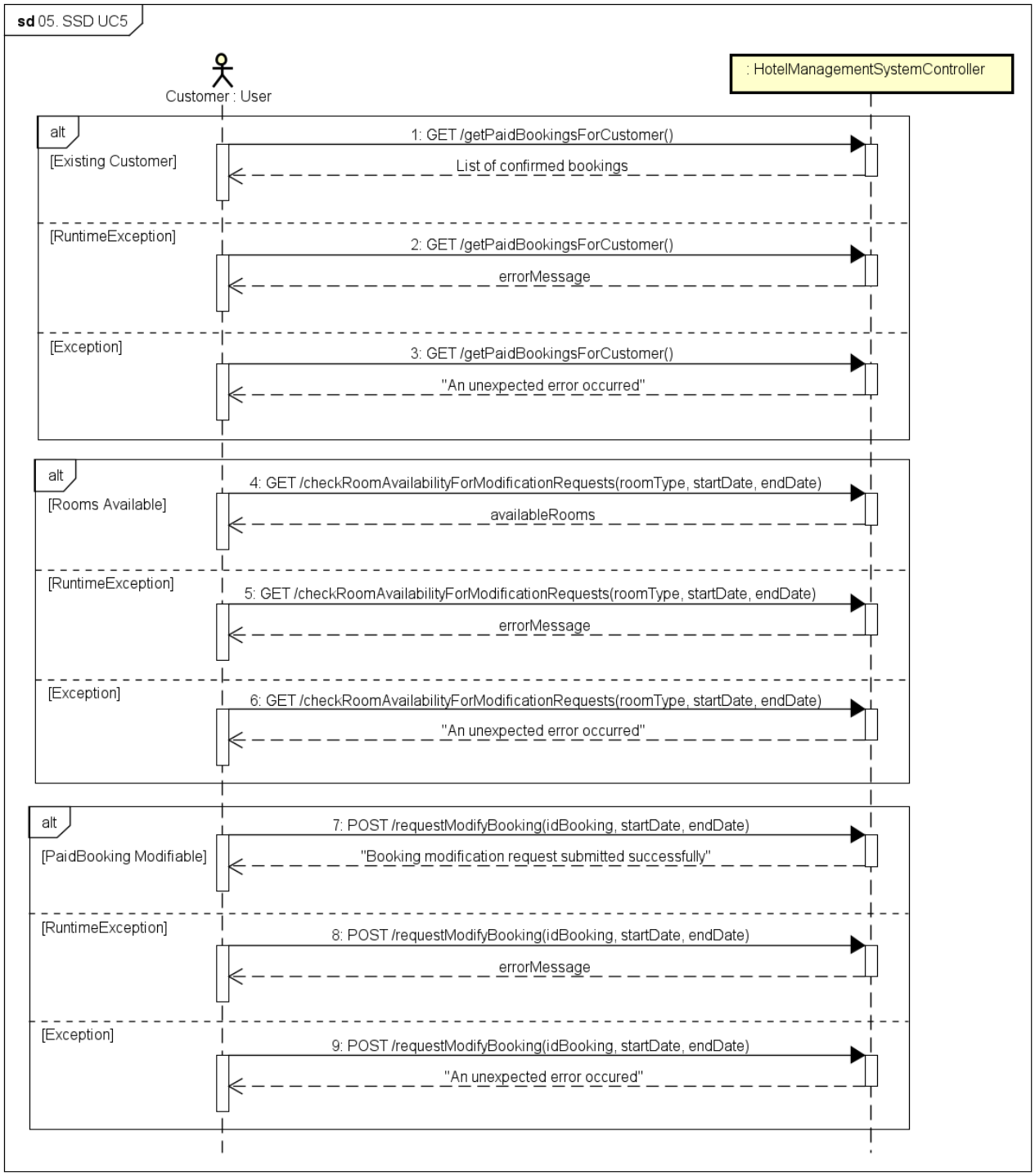
- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"* e associato al *customer* attualmente loggato nel portale di prenotazione.

Post-condizioni:

- Sono stati restituiti tutti i record *paidBooking* della tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"* associati al *customer* attualmente loggato nel portale di prenotazione.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.11. CASO D'USO UC5, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC5 è il seguente:



2.5.12. CASO D'USO UC5, CONTRATTI DELLE OPERAZIONI

2.5.12.1. *getPaidBookingForCustomer*

L'operazione di sistema *getPaidBookingForCustomer* consente di visualizzare la lista delle prenotazioni confermate da un pagamento precedente.

Operazione:

GET /getPaidBookingForCustomer()

Riferimenti:
Caso d'uso UC4: Visualizza dettagli della prenotazione.

Pre-condizioni:

- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"* e associato al *customer* attualmente loggato nel portale di prenotazione.

Post-condizioni:

- Sono stati restituiti tutti i record *paidBooking* della tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"* associati al *customer* attualmente loggato nel portale di prenotazione.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.12.2. *checkRoomAvailabilityForModificationRequest*

L'operazione di sistema *checkRoomAvailabilityForModificationRequest* consente di calcolare il numero di stanze disponibili per un determinato tipo di stanza e un intervallo di date.

Operazione:
GET /checkRoomAvailabilityForModificationRequest(roomType, startDate, endDate)

Riferimenti:
Caso d'uso UC5: Richiedi modifica alla prenotazione.

Pre-condizioni:

- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esistono istanze *roomCategories* dell'entità *RoomCategory*.

Post-condizioni:

- Se ci sono state stanze disponibili per la *roomCategory* selezionata dall'utente nell'intervallo di date scelto, il sistema ha restituito il *availableRooms*.
- Se non ci sono state stanze disponibili per la *roomCategory* selezionata dall'utente nell'intervallo di date scelto, il sistema ha restituito un *availableRooms* pari a "0".
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.12.3. *requestModifyBooking*

L'operazione di sistema *requestModifyBooking* consente di richiedere una modifica di una prenotazione confermata da un pagamento precedente.

Operazione:
POST /requestModifyBooking(idBooking, startDate, endDate)

Riferimenti:
Caso d'uso UC5: Richiedi modifica alla prenotazione.

Pre-condizioni:

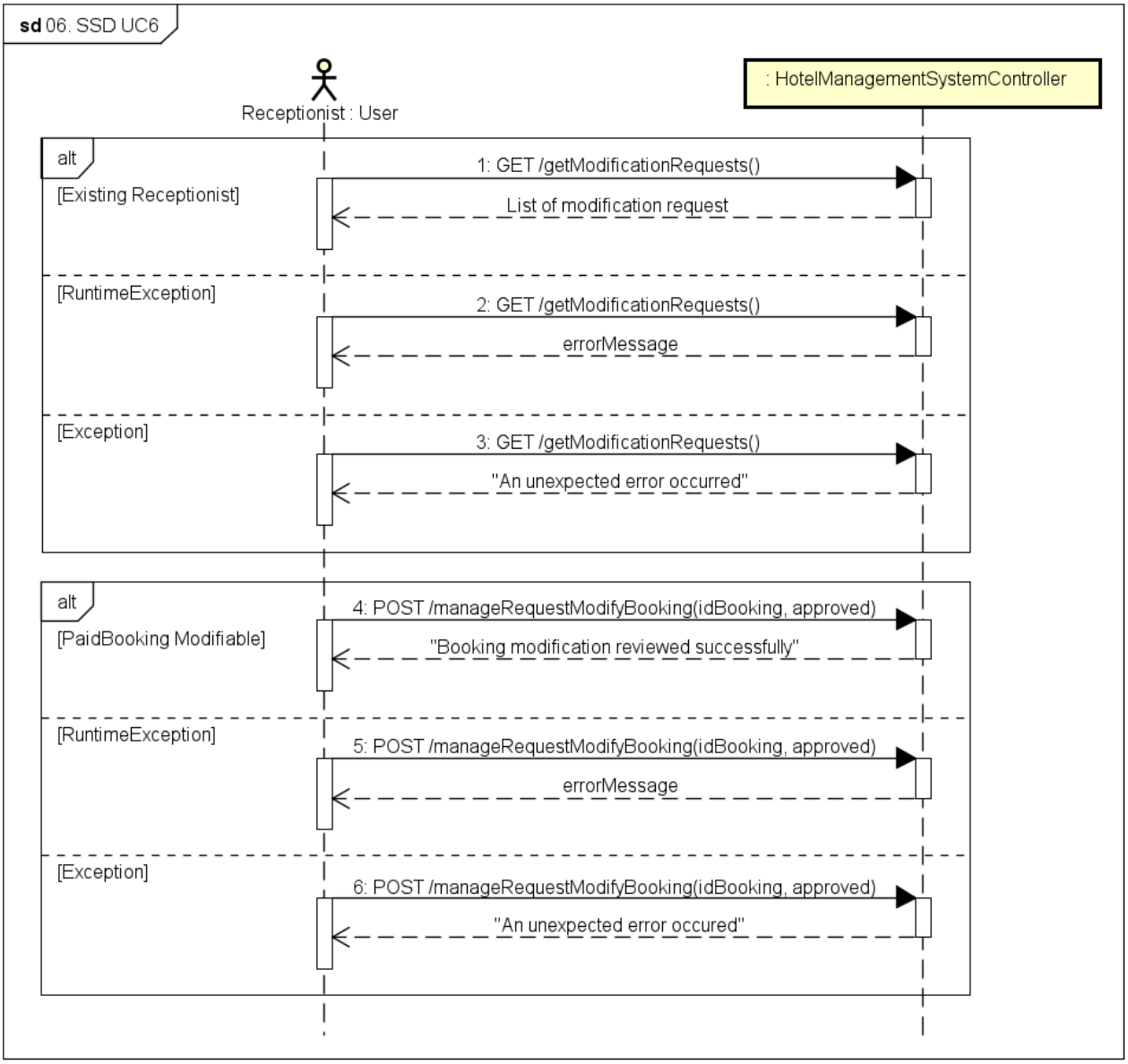
- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"* e associato al *customer* attualmente loggato nel portale di prenotazione.
- Il *customer* ha selezionato un nuovo range di date per una determinata *roomCategories*.

Post-condizioni:

- Se la roomCategory è risultata disponibile:
  - È stata recuperata un' istanza *paidBooking* dell'entità *PaidBooking*.
  - Alcuni attributi di *paidBooking* sono stati aggiornati tramite i loro setter, in particolare quattro attributi:
    - *oldStartDate:* è stato settato con il valore attuale *startDate*.
    - *oldStartDate*: è stato settato con il valore attuale *oldDate*.
    - *startDate:* è stato settato con il valore nuovo scelto passato come parametro alla funzione.
    - *endDate:* è stato settato con il valore nuovo scelto passato come parametro alla funzione.
    - *status*: è stato settato come: *"Pending Modification"*.
  - È stata salvata l'istanza *paidBooking* in modo da sovrascrivere il precedente record nella tabella del database associata all'entità *PaidBooking.*
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.13. CASO D'USO UC6, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC6 è il seguente:



2.5.14. CASO D'USO UC6, CONTRATTI DELLE OPERAZIONI

2.5.14.1. *getModificationRequests*

L'operazione di sistema *getModificationRequests* consente di ottenere la lista delle richieste di modifica delle prenotazioni.

Operazione:
GET /getModificationRequests()

Riferimenti:
Caso d'uso UC6: Modifica dettagli della prenotazione.

Pre-condizioni:

- Esiste un record *receptionist* nella tabella del database associata all'entità *Receptionist*.
- Un *receptionist* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Pending Modification"*.

Post-condizioni:

- Il sistema ha restituito una lista contenente tutte le istanze *paidBooking* dell'entità *PaidBooking* settate con status *"Pending Modification".*
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.14.2. *manageRequestModifyBooking*

L'operazione di sistema *manageRequestModifyBooking* consente di gestire una richiesta di modifica di una prenotazione confermata da un pagamento precedente.

Operazione:
POST /manageRequestModifyBooking(idBooking, approved)

Riferimenti:
Caso d'uso UC6: Modifica dettagli della prenotazione.

Pre-condizioni:

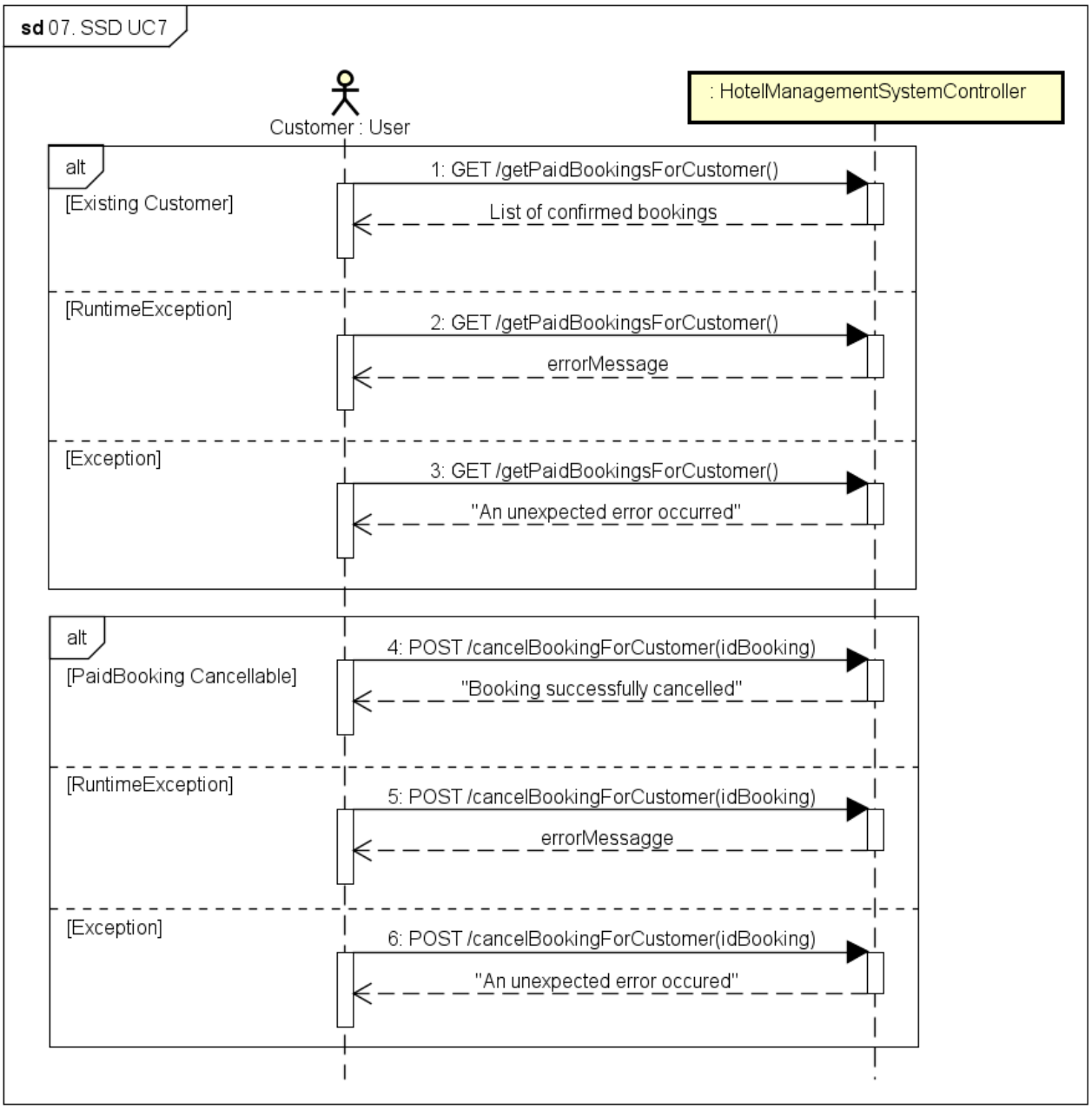
- Esiste un record *receptionist* nella tabella del database associata all'entità *Receptionist*.
- Un *receptionist* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Pending Modification"*.

Post-condizioni:

- Il *receptionist* ha selezionato una *paidBooking*:
    - Se la richiesta di modifica è stata approvata, ha ridefinito il suo attributo *status* tramite il suo setter, in particolare:
        - *status*: viene settato come: *"Paid"*.
    - Se la richiesta di modifica non è stata approvata, alcuni attributi sono stati aggiornati tramite i loro setter, in particolare quattro attributi:
        - *startDate:* è stato settato *oldStartDate*.
        - *endDate:* è stato settato *oldEndDate*.
        - *oldStartDate*: è stato settato con "*null".*
        - *oldEndDate*: è stato settato con "*null".*
        - *status*: è stato settato come: *"Paid"*.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.15. CASO D'USO UC7, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC7 è il seguente:



2.5.16. CASO D'USO UC7, CONTRATTI DELLE OPERAZIONI

2.5.16.1. *getPaidBookingForCustomer*

L'operazione di sistema *getPaidBookingForCustomer* consente di visualizzare la lista delle prenotazioni confermate da un pagamento precedente.

Operazione:
GET /getPaidBookingForCustomer()

Riferimenti:
Caso d'uso UC4: Visualizza dettagli della prenotazione.

Pre-condizioni:

- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"* e associato al *customer* attualmente loggato nel portale di prenotazione.

Post-condizioni:

- Sono stati restituiti tutti i record *paidBooking* della tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"* associati al *customer* attualmente loggato nel portale di prenotazione.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.16.2. *cancelBookingForCustomer*

L'operazione di sistema *cancelBookingForCustomer* consente di annullare una prenotazione confermata da un pagamento precedente.

Operazione:
POST /cancelBookingForCustomer(idBooking)

Riferimenti:
Caso d'uso UC7: Annulla prenotazione.

Pre-condizioni:

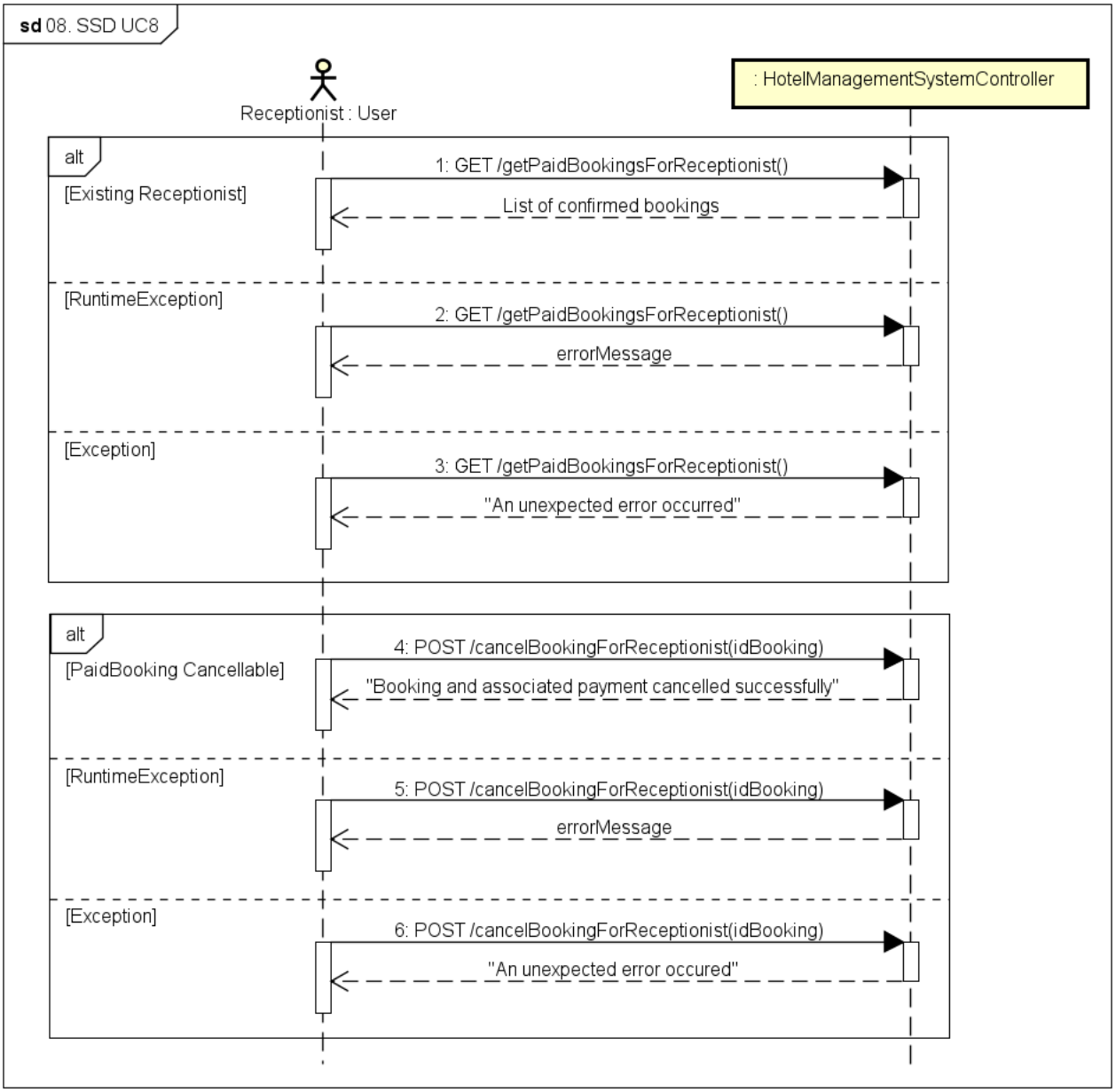
- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"*.

Post-condizioni:

- Il *customer* ha selezionato una *paidBooking* e ha ridefinito il suo attributo *status* tramite il suo setter, in particolare:
  - *status*: è stato settato come: *"Cancelled"*.
- È stata salvata la modifica dell'istanza *paidBooking* con conseguente aggiornamento del record corrispondente nella tabella del database associata all'entità *PaidBooking.*
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.17. CASO D'USO UC8, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC8 è il seguente:



2.5.18. CASO D'USO UC8, CONTRATTI DELLE OPERAZIONI

2.5.18.1. *getPaidBookingsForReceptionist*

L'operazione di sistema *getPaidBookingsForReceptionist* consente di visualizzare la lista delle prenotazioni confermate da un pagamento precedente.

Operazione:
GET /getPaidBookingsForReceptionist()

Riferimenti:
Caso d'uso UC8: Cancella prenotazione.

Pre-condizioni:

- Esiste un record *receptionist* nella tabella del database associata all'entità *Receptionist*.
- Un *receptionist* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"*.

Post-condizioni:

- Sono stati restituiti tutti i record *paidBooking* della tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"* associati al *customer* attualmente loggato nel portale di prenotazione.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.18.2. *cancelBookingForReceptionist*

L'operazione di sistema *cancelBookingForReceptionist* consente di cancellare una prenotazione confermata da un pagamento precedente.

Operazione:
POST /cancelBookingForReceptionist(idBooking)

Riferimenti:
Caso d'uso UC8: Cancella prenotazione.

Pre-condizioni:

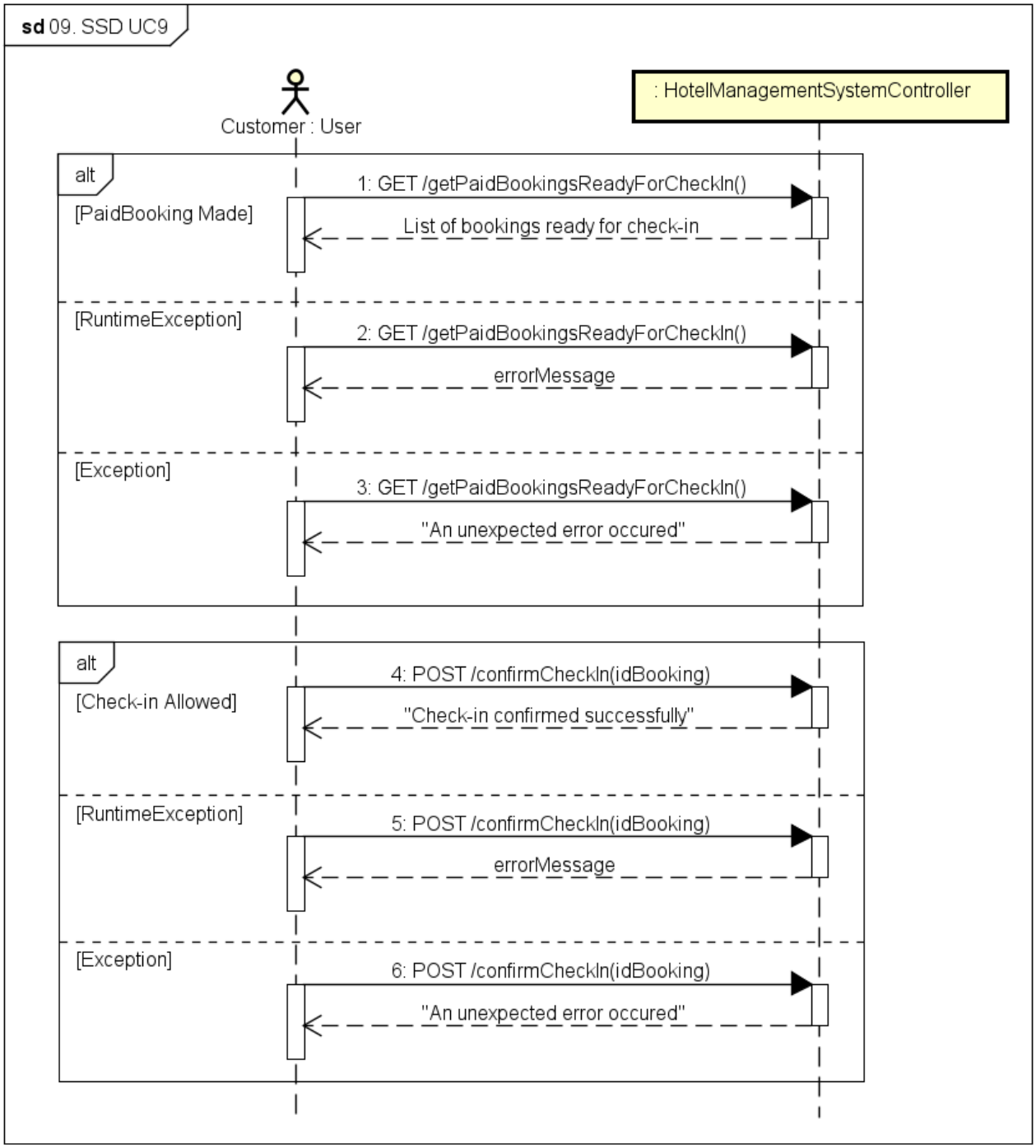
- Esiste un record *receptionist* nella tabella del database associata all'entità *Receptionist*.
- Un *receptionist* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"*.

Post-condizioni:

- Il *receptionist* ha selezionato una *paidBooking* ed elimina il suo record nella tabella del database associata all'entità *PaidBooking* andando ad eliminare anche il record del relativo *payment* nella tabella associata all'entità *Payment*.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.19. CASO D'USO UC9, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC9 è il seguente:



2.5.20. CASO D'USO UC9, CONTRATTI DELLE OPERAZIONI

2.5.20.1. *getPaidBookingsReadyForCheckIn*

L'operazione di sistema *getPaidBookingsReadyForCheckIn* consente di visualizzare la lista delle prenotazioni confermate da un pagamento precedente e abilitate al check-in.

Operazione:
GET /getPaidBookingsReadyForCheckIn()

Riferimenti:
Caso d'uso UC9: Check-In in digitale.

Pre-condizioni:
- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"*.

Post-condizioni:
- Sono stati restituiti tutti i record *paidBooking* della tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"* e con un valore *startDate* pari alla data attuale associati al *customer* attualmente loggato nel portale di prenotazione.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.20.2. *confirmCheckIn*

L'operazione di sistema *confirmCheckIn* consente di effettuare il check-in di una prenotazione confermata da un pagamento precedente.

Operazione:
POST /confirmCheckIn(idBooking)

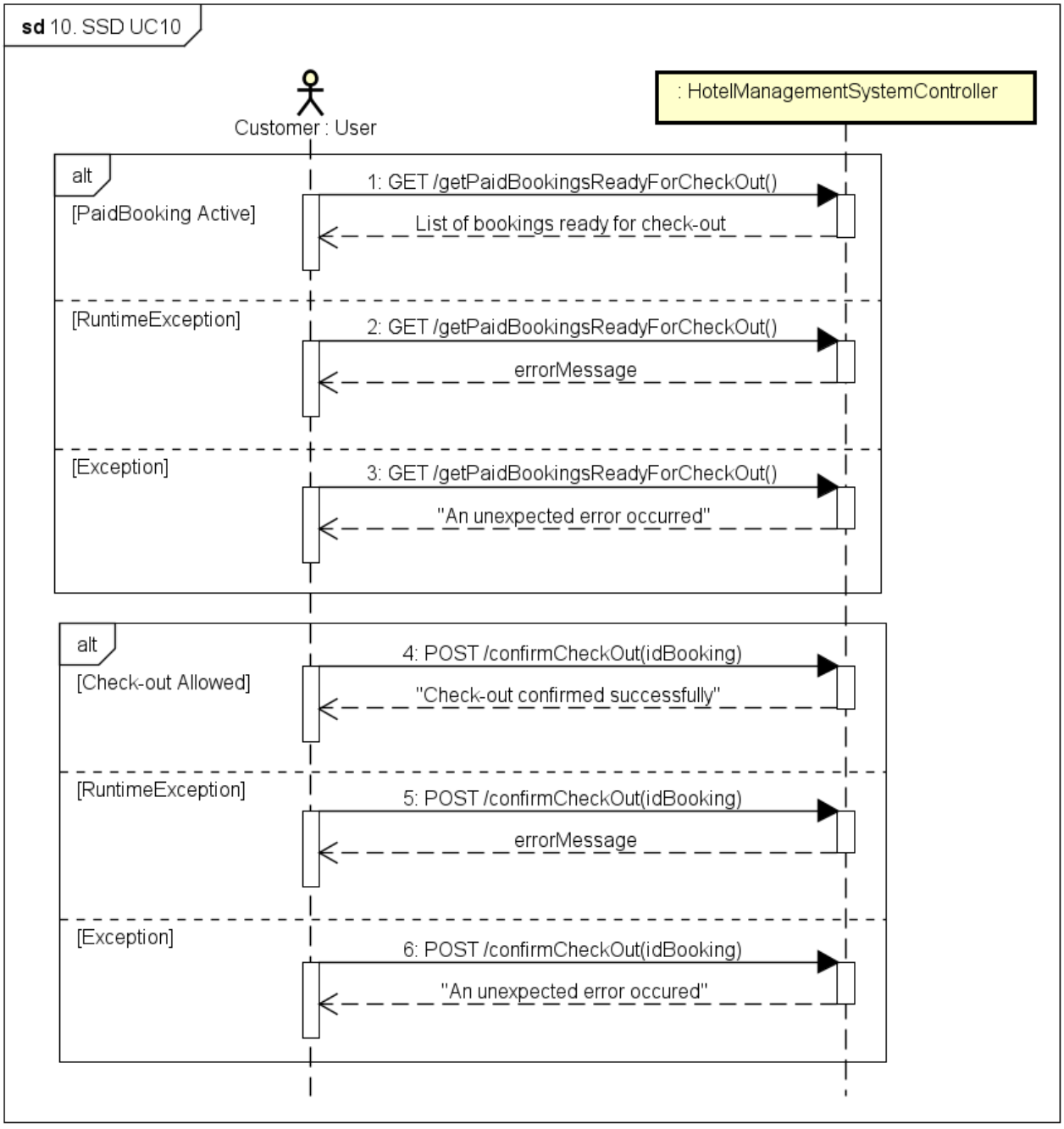
Referment:
Caso d'uso UC9: Check-In in digitale.

Pre-condizioni:
- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"*.

Post-condizioni:
- Il *customer* ha selezionato una *paidBooking* e ha ridefinito il suo attributo *status* tramite il suo setter, in particolare:
  - *status*: è stato settato come: *"Check-In Confirmed"*.
- È stata salvata la modifica dell'istanza *paidBooking* con conseguente aggiornamento del record corrispondente nella tabella del database associata all'entità *PaidBooking*.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.21. CASO D'USO UC10, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC10 è il seguente:



2.5.22. CASO D'USO UC10, CONTRATTI DELLE OPERAZIONI

2.5.22.1. *getPaidBookingsReadyForCheckOut*

L'operazione di sistema *getPaidBookingsReadyForCheckOut* consente di visualizzare la lista delle prenotazioni confermate da un pagamento precedente e abilitate al check-out.

Operazione:
GET /getPaidBookingsReadyForCheckOut()

Riferimenti:
Caso d'uso UC10: Check-out in digitale.

Pre-condizioni:
- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Check-In Confirmed"*.

Post-condizioni:
- Sono stati restituiti tutti i record *paidBooking* della tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Paid"* e con un valore *endDate* pari alla data attuale associati al *customer* attualmente loggato nel portale di prenotazione.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.22.2. confirmCheckOut

L'operazione di sistema *confirmCheckOut* consente di effettuare il check-out di una prenotazione confermata da un pagamento precedente.

Operazione:
POST /confirmCheckOut(idBooking)

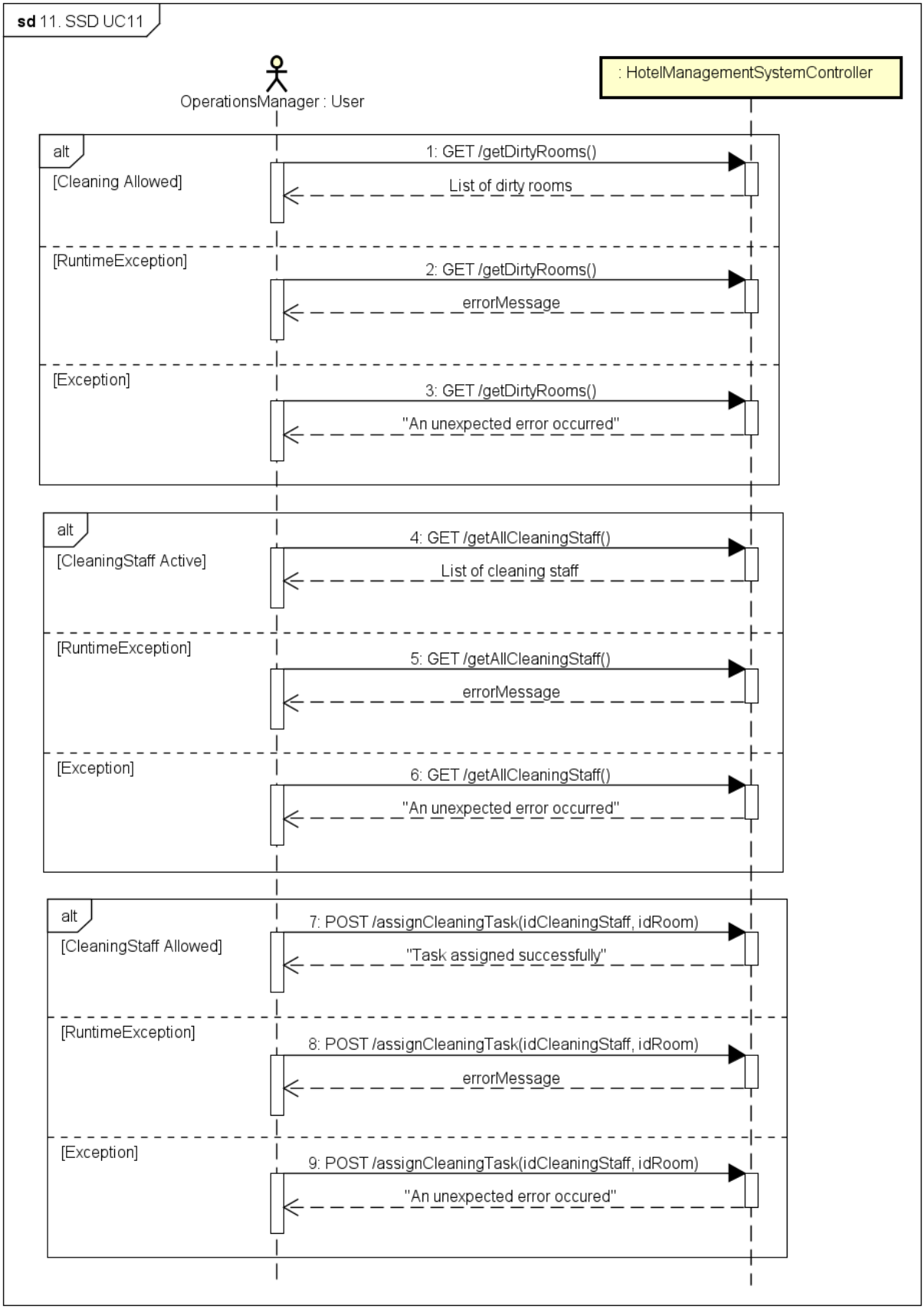
Riferimenti:
Caso d'uso UC10: Check-out in digitale.

Pre-condizioni:
- Esiste almeno un record *customer* nella tabella del database associata all'entità *Customer*.
- Un *customer* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *paidBooking* nella tabella del database associata all'entità *PaidBooking* con un valore *status* settato come: *"Check-In Confirmed"*.

Post-condizioni:
- Il *customer* ha selezionato una *paidBooking* e ha ridefinito il suo attributo *status* tramite il suo setter, in particolare:
  - *status*: è stato settato come: *"Check-Out Confirmed"*.
- È stata salvata la modifica dell'istanza *paidBooking* con conseguente aggiornamento del record corrispondente nella tabella del database associata all'entità *PaidBooking*.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.23. CASO D'USO UC11, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC11 è il seguente:



2.5.24. CASO D'USO UC11, CONTRATTI DELLE OPERAZIONI

2.5.24.1. *getDirtyRooms*

L'operazione di sistema *getDirtyRooms* consente di visualizzare la lista delle stanze sporche.

Operazione:
GET /getDirtyRooms()

Riferimenti:
Caso d'uso UC11: Assegna attività del personale.

Pre-condizioni:
- Esiste un record *operationsManager* nella tabella del database associata all'entità *OperationsManager*.
- Un *operationsManager* ha effettuato l'accesso al portale di prenotazione.
- Esistono istanze *room* dell'entità *Room* con un valore status settato come: "*Dirty*"

Post-condizioni:
- Sono stati restituiti tutti i record *room* della tabella del database associata all'entità *Room* con un valore *status* settato come: *"Dirty"*.
- Se si verifica un'eccezione, il sistema restituisce un messaggio di errore.

2.5.24.2. *getAllCleaningStaff*

L'operazione di sistema *getAllCleaningStaff* consente di visualizzare la lista dei cleaning staff.

Operazione:
GET /getAllCleaningStaff()

Riferimenti:
Caso d'uso UC11: Assegna attività del personale.

Pre-condizioni:
- Esiste un record *operationsManager* nella tabella del database associata all'entità *OperationsManager*.
- Un *operationsManager* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *cleaningStaff* nella tabella del database associata all'entità *CleaningStaff*.

Post-condizioni:
- Sono stati restituiti tutti i record *cleaningStaff* della tabella del database associata all'entità *CleaningStaff*.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.24.3. *assignCleaningTask*

L'operazione di sistema *assignCleaningTask* consente di assegnare un task di pulizia.

Operazione:
POST /assignCleaningTask(idCleaningStaff, idRoom)

Riferimenti:
Caso d'uso UC11: Assegna attività del personale.

Pre-condizioni:

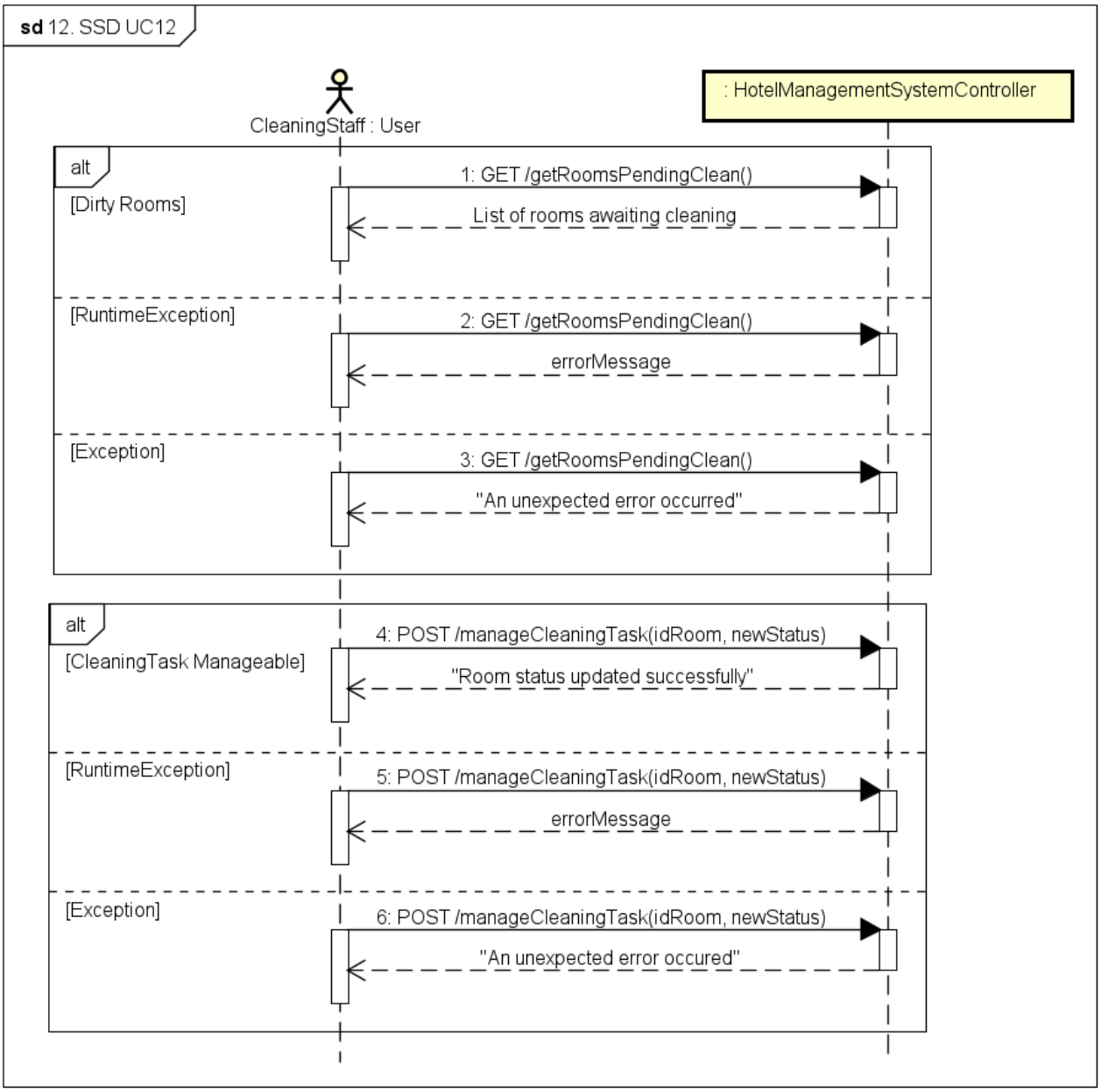
- Esiste un record *operationsManager* nella tabella del database associata all'entità *OperationsManager*.
- Un *operationsManager* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *cleaningStaff* nella tabella del database associata all'entità *CleaningStaff*.

Post-condizioni:

- L'*OperationsManager* ha selezionato un record *room* della tabella del database associato all'entità *Room* e un record *cleaningStaff* della tabella del database associata all'entità *CleaningStaff*:
    - È stata creata una nuova istanza *cleaningTask* dell'entità *CleaningTask*.
    - Gli attributi di *cleaningTask* sono stati inizializzati tramite i loro setter, in particolare per un attributo:
        - *status*: è stato settato come: "*Assigned"*.
    - È stato ridefinito l'attributo *status* della *room* assegnata tramite il suo setter, in particolare:
        - *status*: è stato settato come: *"Pending Clean"*.
    - È stata salvata l'istanza *cleaningTask* come record nella tabella del database associata all'entità *CleaningTask.*
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.25. CASO D'USO UC12, DIAGRAMMA DI SEQUENZA DEL SISTEMA

Il diagramma di sequenza di sistema per il caso d'uso UC12 è il seguente:



2.5.26. CASO D'USO UC12, CONTRATTI DELLE OPERAZIONI

2.5.26.1. *getRoomsPendingClean*

L'operazione di sistema *getRoomsPendingClean* consente di visualizzare la lista dei task di pulizia.

Operazione:
GET /getRoomsPendingClean()

Riferimenti:
Caso d'uso UC12: Gestisci stato delle camere.

Pre-condizioni:

- Esiste un record *cleaningStaff* nella tabella del database associata all'entità *CleaningStaff*.
- Un *cleaningStaff* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *cleaningTask* nella tabella del database associato all'entità *CleaningTask* con un valore *status* settato come: *"Assigned".*

Post-condizioni:

- Sono stati restituiti tutti i record *cleaningTask* della tabella del database associata all'entità *CleaningTask* con un valore *status* settato come: *"Assigned"* associata al *cleaningStaff* attualmente loggato nel portale di prenotazione.
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.5.26.2. *manageCleaningTask*

L'operazione di sistema *manageCleaningTask* consente di gestire l'assegnazione di un task di pulizia.

Operazione:
POST /manageCleaningTask(idRoom, newStatus)

Riferimenti:
Caso d'uso UC12: Gestisci stato delle camere.

Pre-condizioni:

- Esiste un record *cleaningStaff* nella tabella del database associata all'entità *CleaningStaff*.
- Un *cleaningStaff* ha effettuato l'accesso al portale di prenotazione.
- Esiste almeno un record *cleaningTask* nella tabella del database associato all'entità *CleaningTask* con un valore *status* settato come: *"Assigned".*

Post-condizioni:

- Il *cleaningStaff* ha selezionato una *cleaningTask* e ha ridefinito l'attributo *status* della *room* assegnata tramite il suo setter, in particolare:
  - *status*: è stato settato come: *"Clean"*.
- È stata salvata la modifica dell'istanza *room* con conseguente aggiornamento del record corrispondente nella tabella del database associata all'entità *Room.*
- Se si è verificata un'eccezione, il sistema ha restituito un messaggio di errore.

2.6. PROGETTAZIONE

2.6.1. INTRODUZIONE

La **fase di progettazione** ha l'obiettivo di tradurre i requisiti individuati nella fase di analisi in un'architettura software strutturata e coerente, garantendo un design modulare, scalabile ed efficiente.

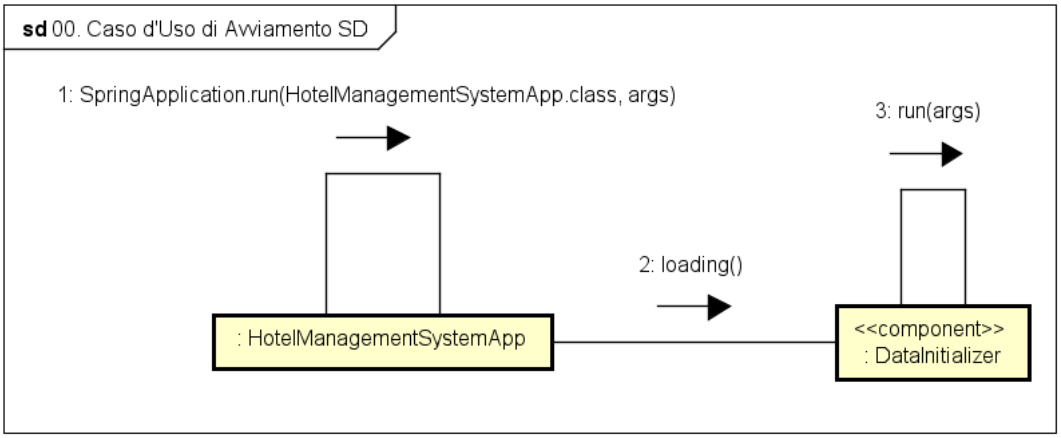
In particolare, questa iterazione si concentra sulla definizione dei componenti principali del sistema e sulle loro interazioni, assicurando un'implementazione conforme ai requisiti funzionali e non funzionali.

In questa fase vengono sviluppati i seguenti elementi:

- **Caso d'uso di avviamento - Diagramma di Comunicazione**: rappresenta la configurazione delle risorse e dei servizi essenziali del sistema, necessari per garantire il corretto funzionamento della piattaforma di gestione alberghiera sin dalla sua attivazione. Questo diagramma descrive le interazioni tra i componenti coinvolti nel processo di avvio del sistema, evidenziando lo scambio di messaggi tra gli oggetti.
- **Diagrammi di Sequenza (SD)**: illustrano il flusso di esecuzione delle operazioni del sistema, evidenziando l'ordine temporale dei messaggi scambiati tra gli oggetti o i componenti coinvolti. Questi diagrammi affinano le interazioni già descritte nei Diagrammi di Sequenza di Sistema (SSD) della fase di analisi, fornendo una specifica più dettagliata del comportamento interno del sistema.
- **Diagramma delle Classi di Progetto**: descrive la struttura del software mediante la definizione delle classi, dei loro attributi e metodi, nonché delle relazioni tra di esse. Questo diagramma costituisce un riferimento essenziale per la successiva fase di implementazione, garantendo una chiara separazione delle responsabilità tra i diversi componenti.

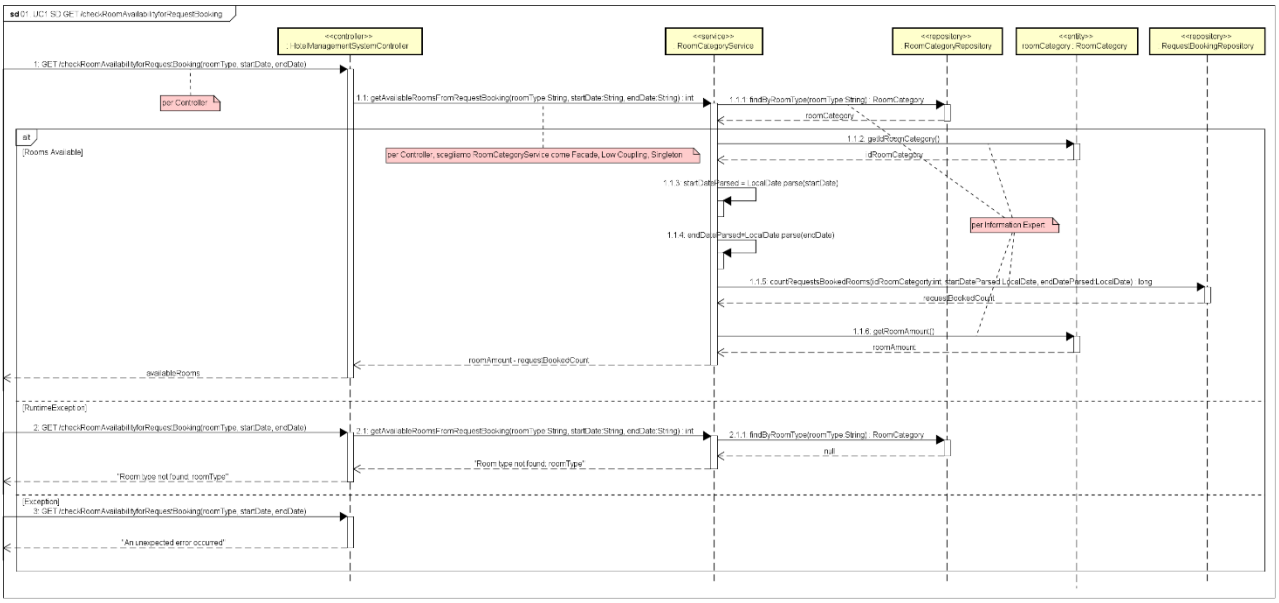
L'approccio adottato nella progettazione si basa sui principi della modularità e della separazione delle responsabilità, con l'obiettivo di migliorare la manutenibilità e l'evoluzione futura del sistema. I modelli sviluppati in questa fase costituiranno la base per l'implementazione, assicurando una transizione fluida e priva di ambiguità tra la progettazione concettuale e la realizzazione concreta del software.

2.6.2. CASO D'USO DI AVVIAMENTO, DIAGRAMMA DI COMUNICAZIONE

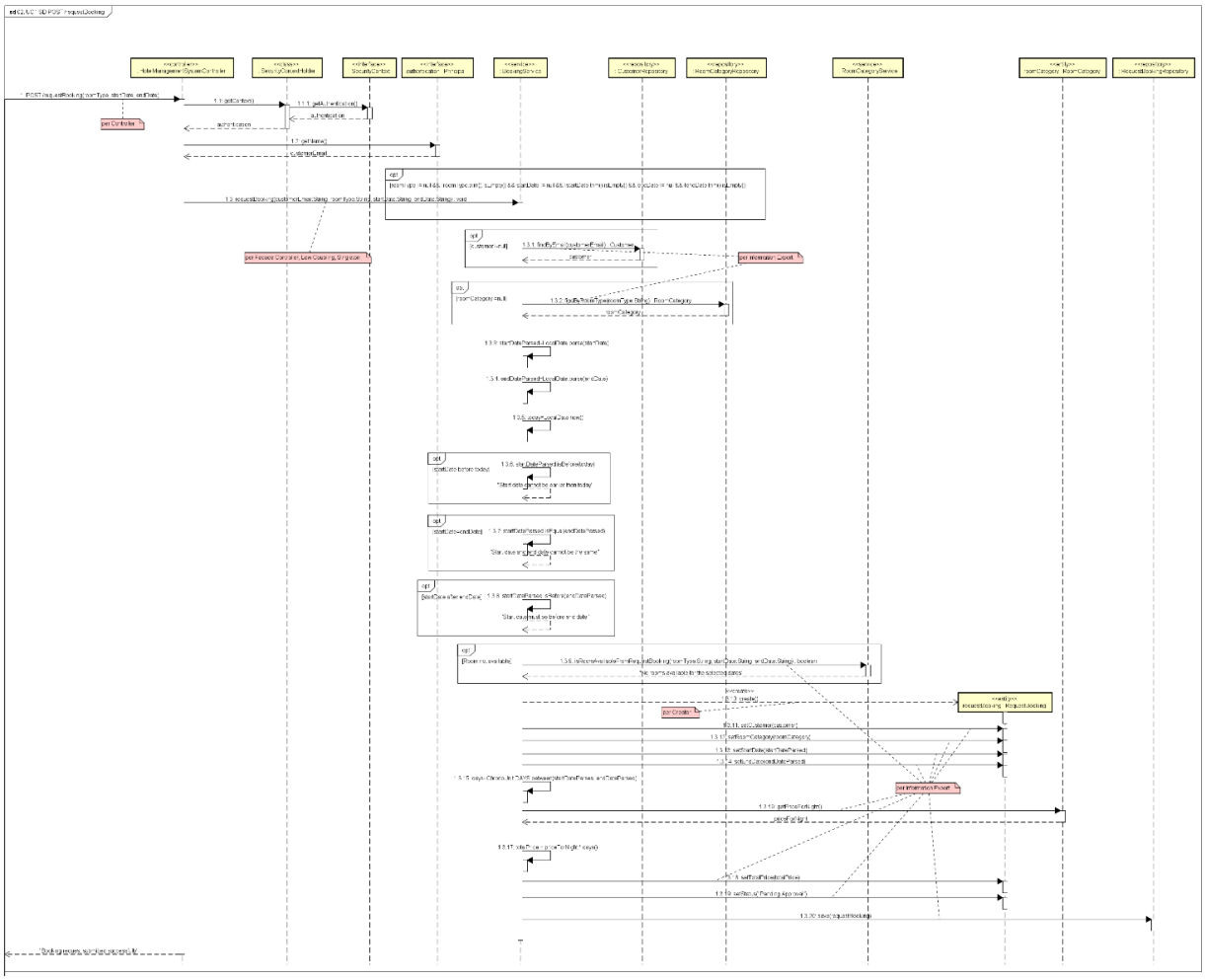


2.6.3. CASO D'USO UC1, DIAGRAMMI DI SEQUENZA

2.6.3.1. *GET /checkRoomAvailabiltyforRequestBooking(roomType, startDate, endDate)*

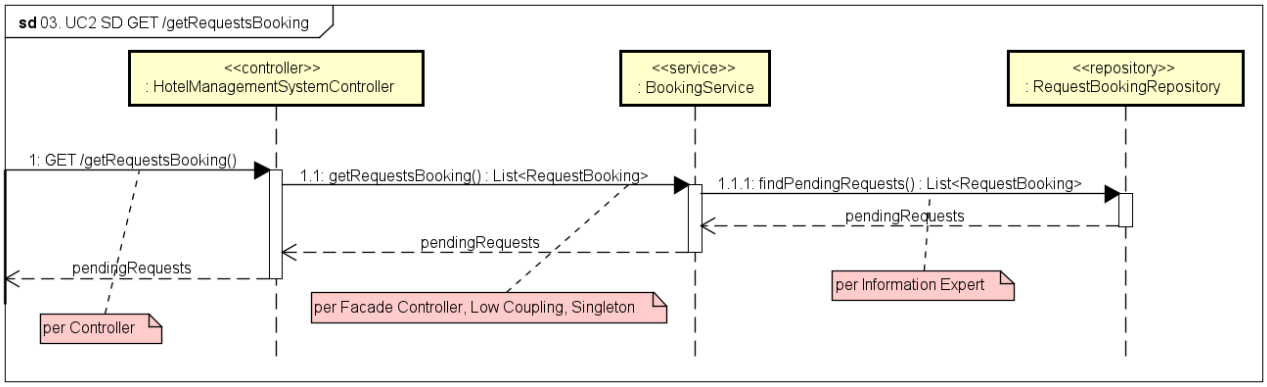


2.6.3.2. *POST /requestBooking(roomType, startDate, endDate)*

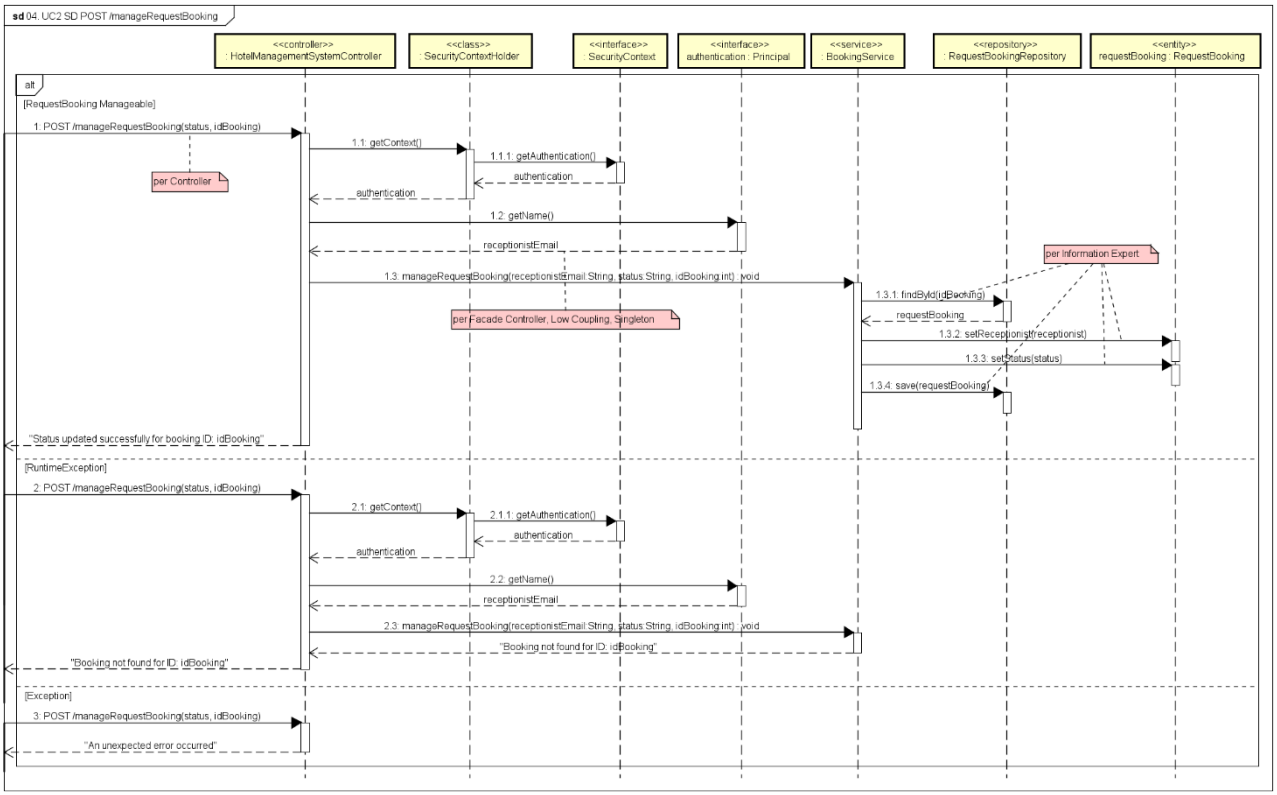


2.6.4. CASO D'USO UC2, DIAGRAMMI DI SEQUENZA

2.6.4.1. *POST /requestBooking (roomType, startDate, endDate)*

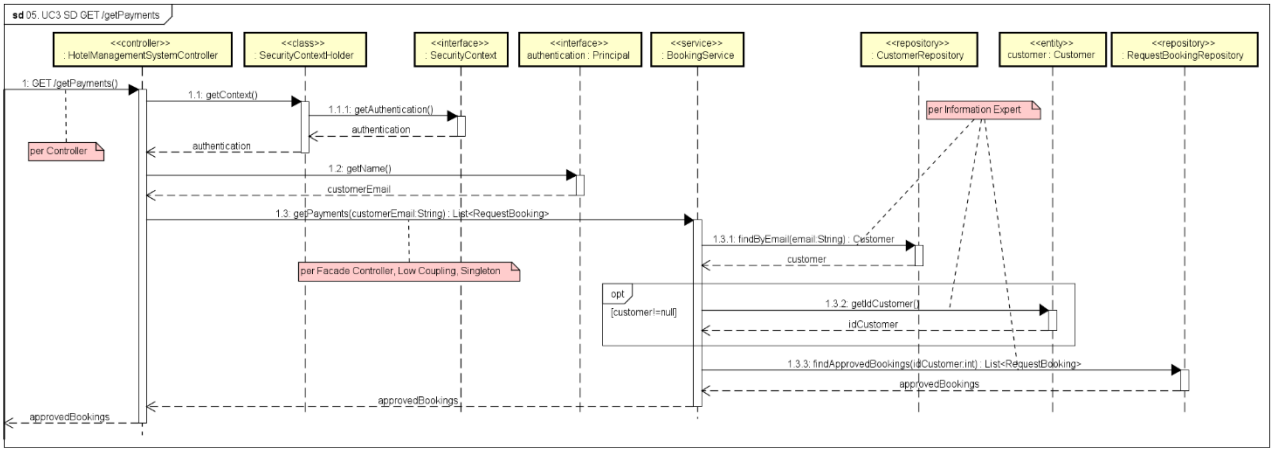


2.6.4.2. *POST /manageRequestBooking(status, idBooking)*

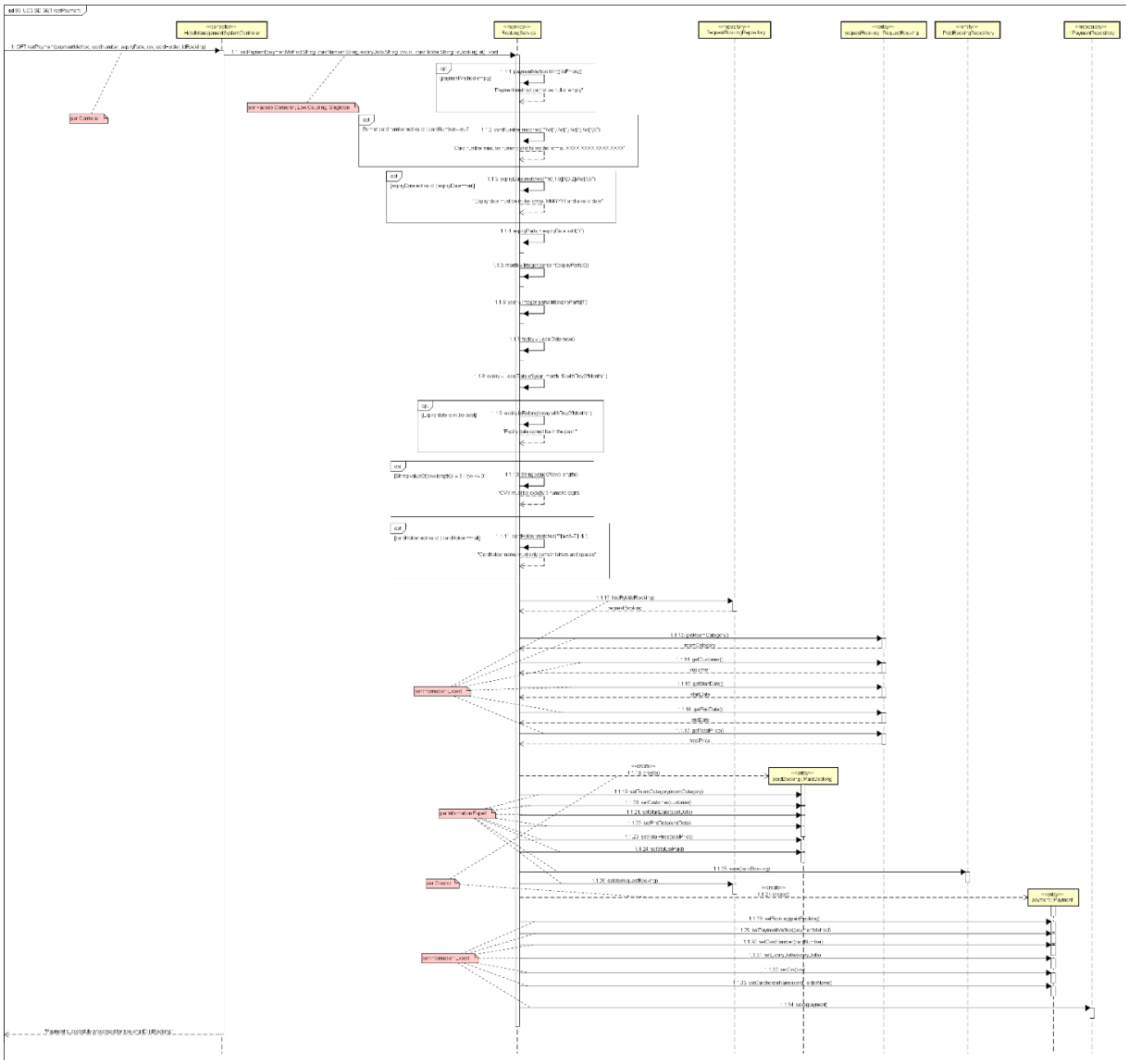


2.6.5. CASO D'USO UC3, DIAGRAMMI DI SEQUENZA

2.6.5.1. *GET /getPayments()*

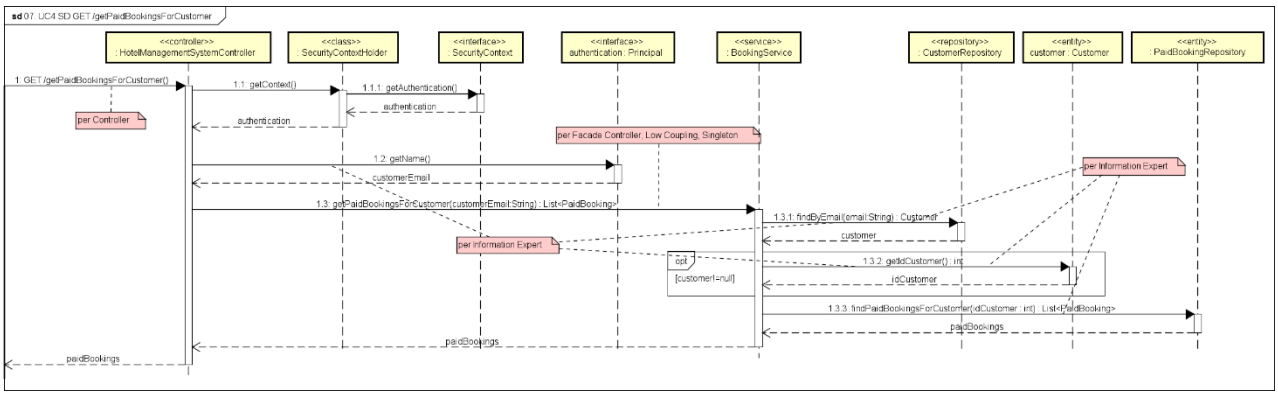


2.6.5.2. *GET /setPayment(paymentMethod, cardNumber, expiryDate, cvv, cardHolder, idBooking)*



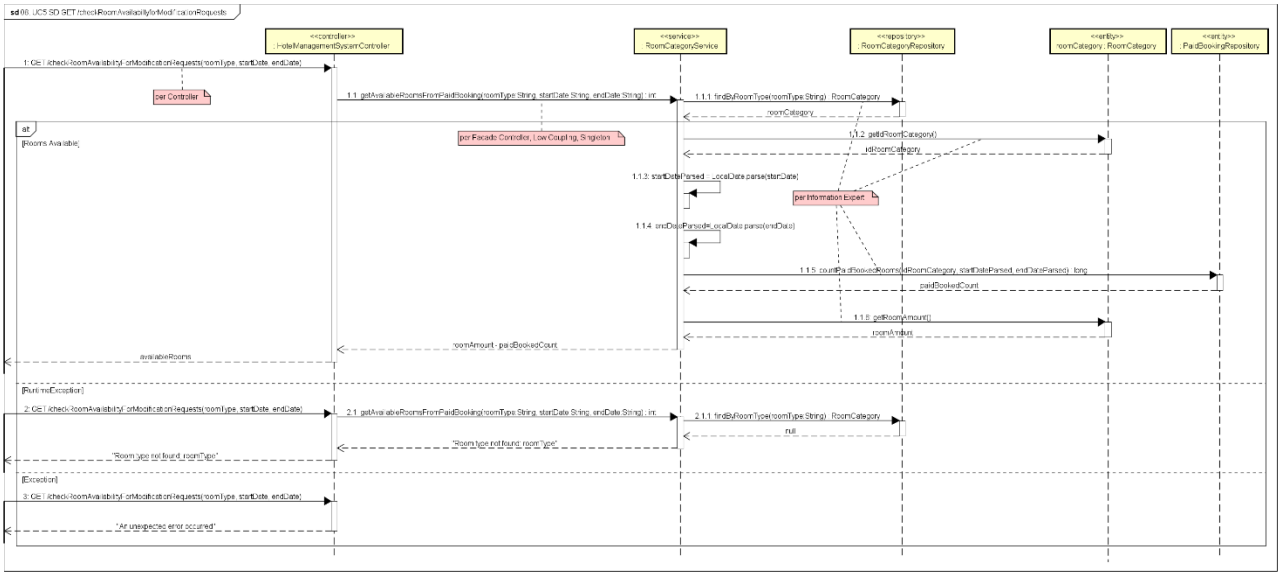
2.6.6. CASO D'USO UC4, DIAGRAMMI DI SEQUENZA

2.6.6.1. *GET /getPaidBookingForCustomer()*

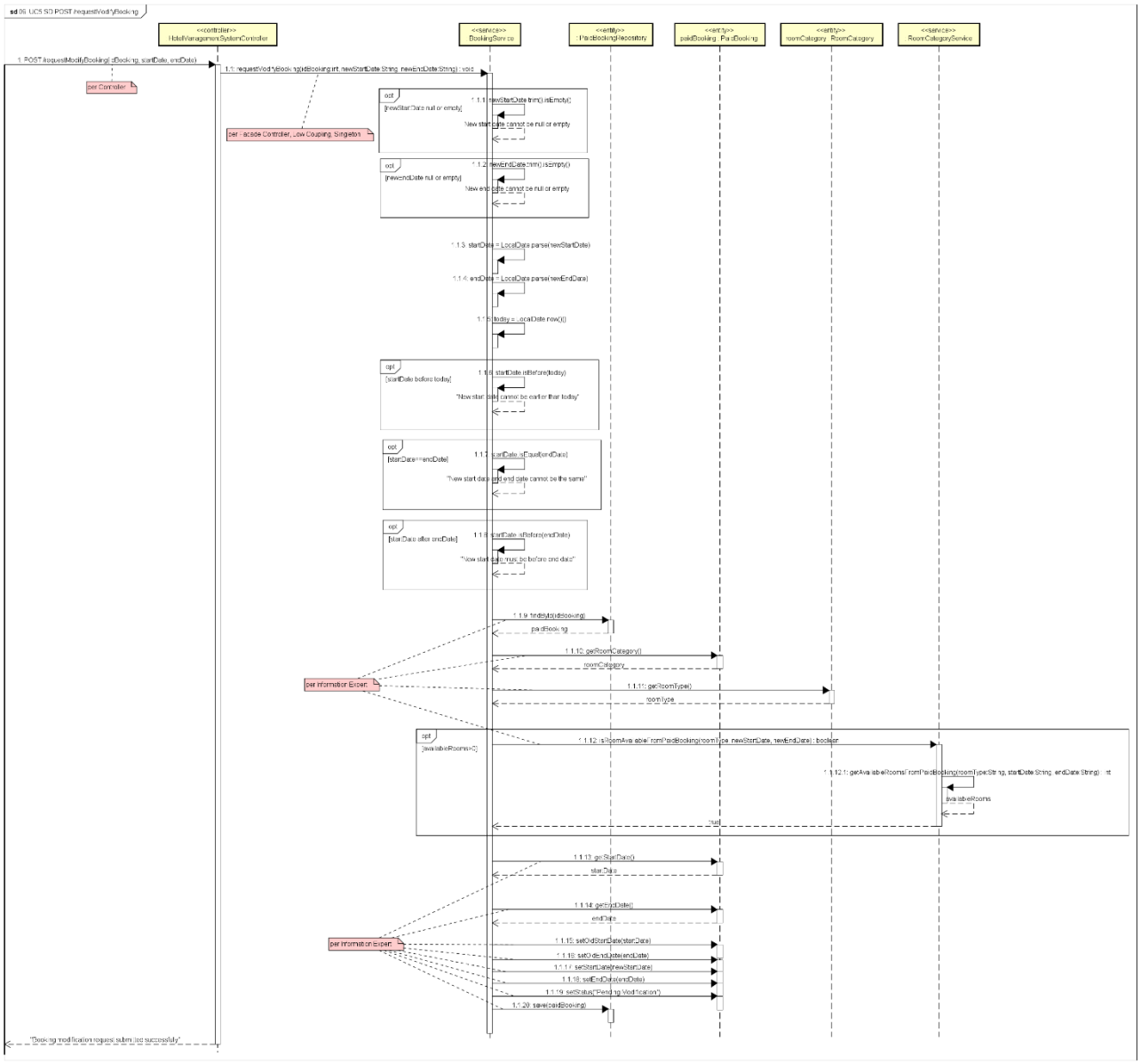


2.6.7. CASO D'USO UC5, DIAGRAMMI DI SEQUENZA

2.6.7.1. GET /checkRoomAvailabilityForModificationRequest(roomType, startDate, endDate)

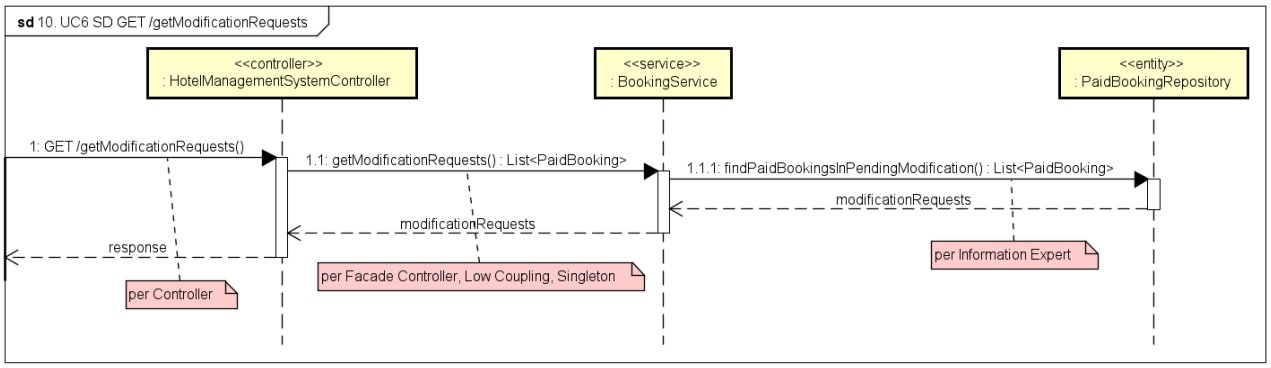


2.6.7.2. *POST /requestModifyBooking(idBooking, startDate, endDate)*

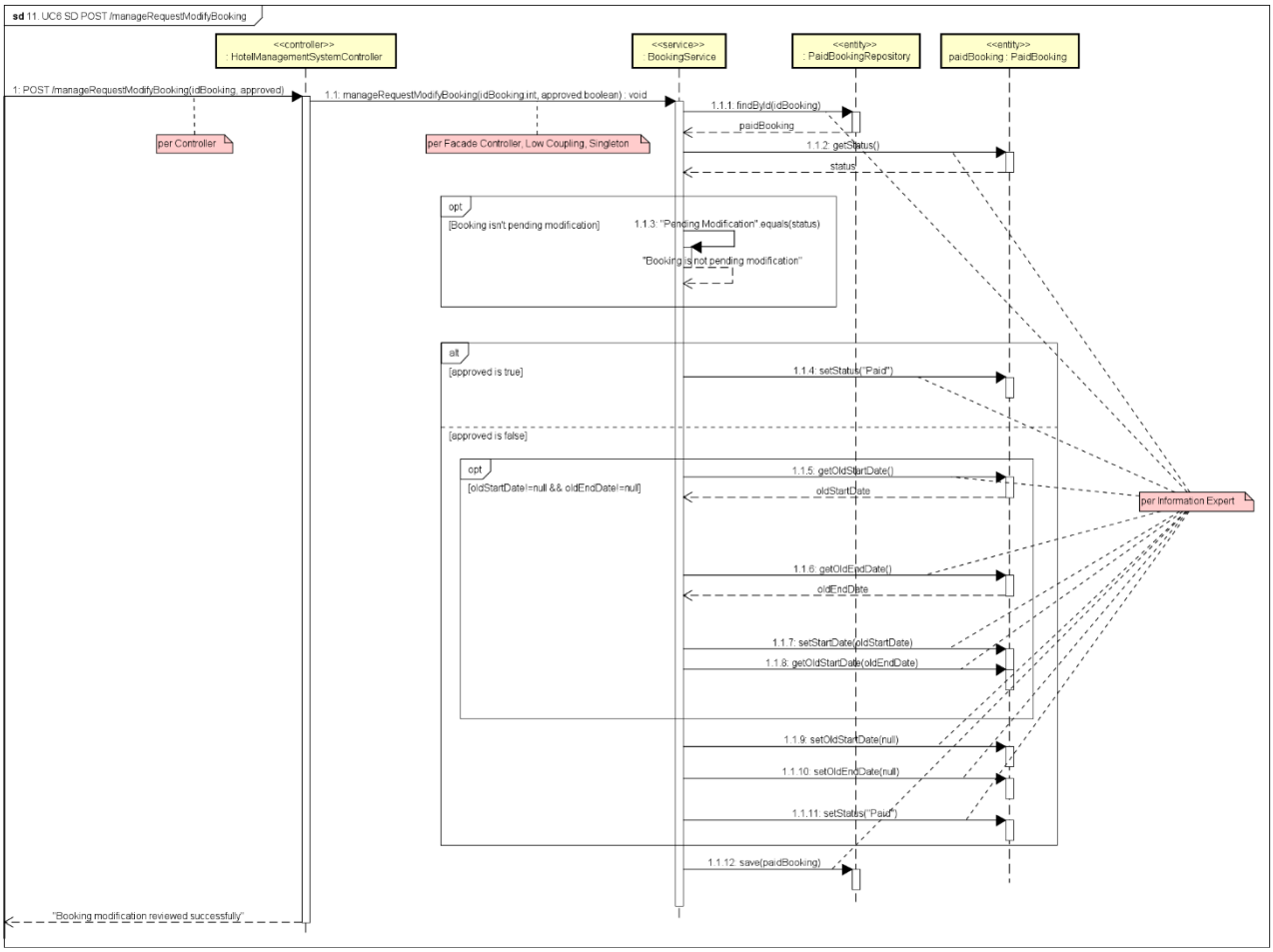


2.6.8. CASO D'USO UC6, DIAGRAMMI DI SEQUENZA

2.6.8.1. *GET /getModificationRequests()*

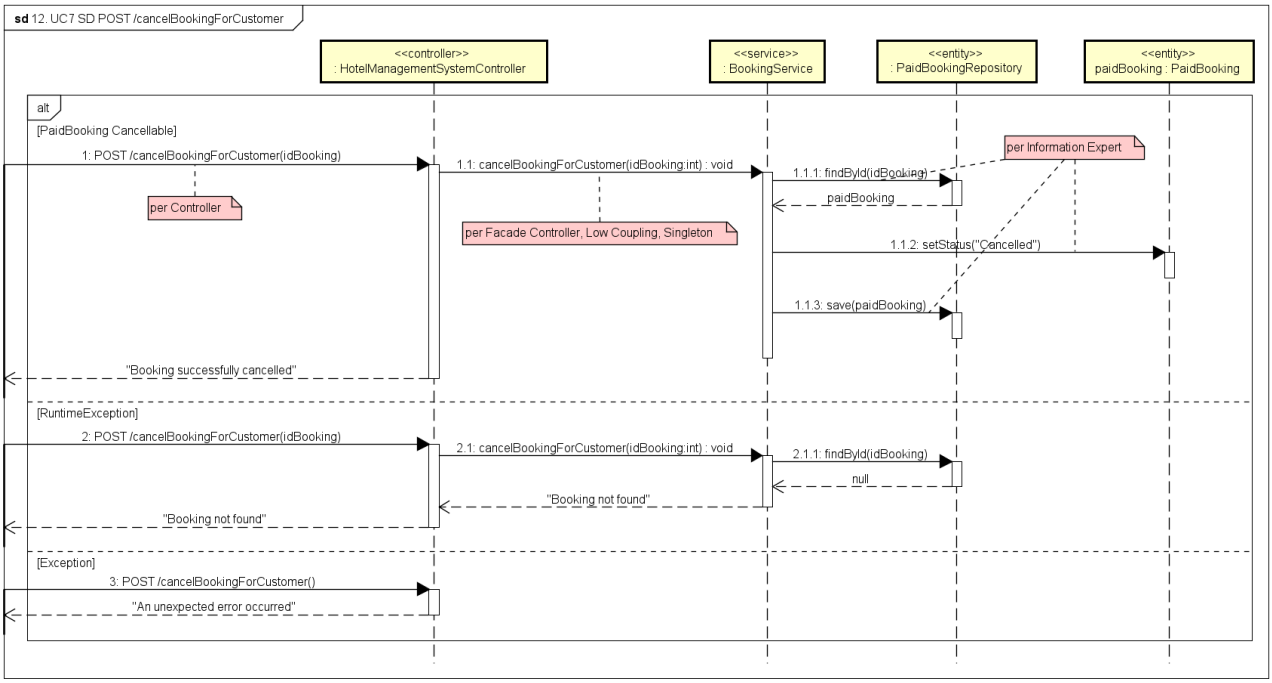


2.6.8.2. *POST /manageRequestModifyBooking(idBooking, approved)*



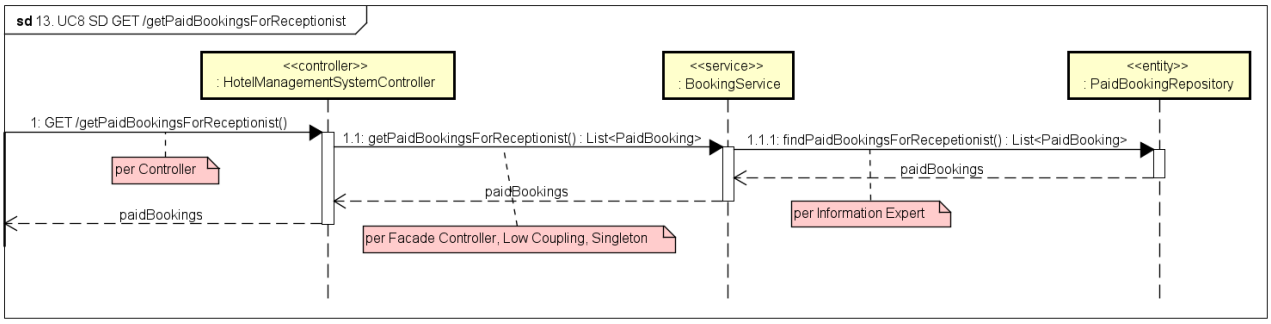
2.6.9. CASO D'USO UC7, DIAGRAMMI DI SEQUENZA

2.6.9.1. *POST /cancelBookingForCustomer(idBooking)*

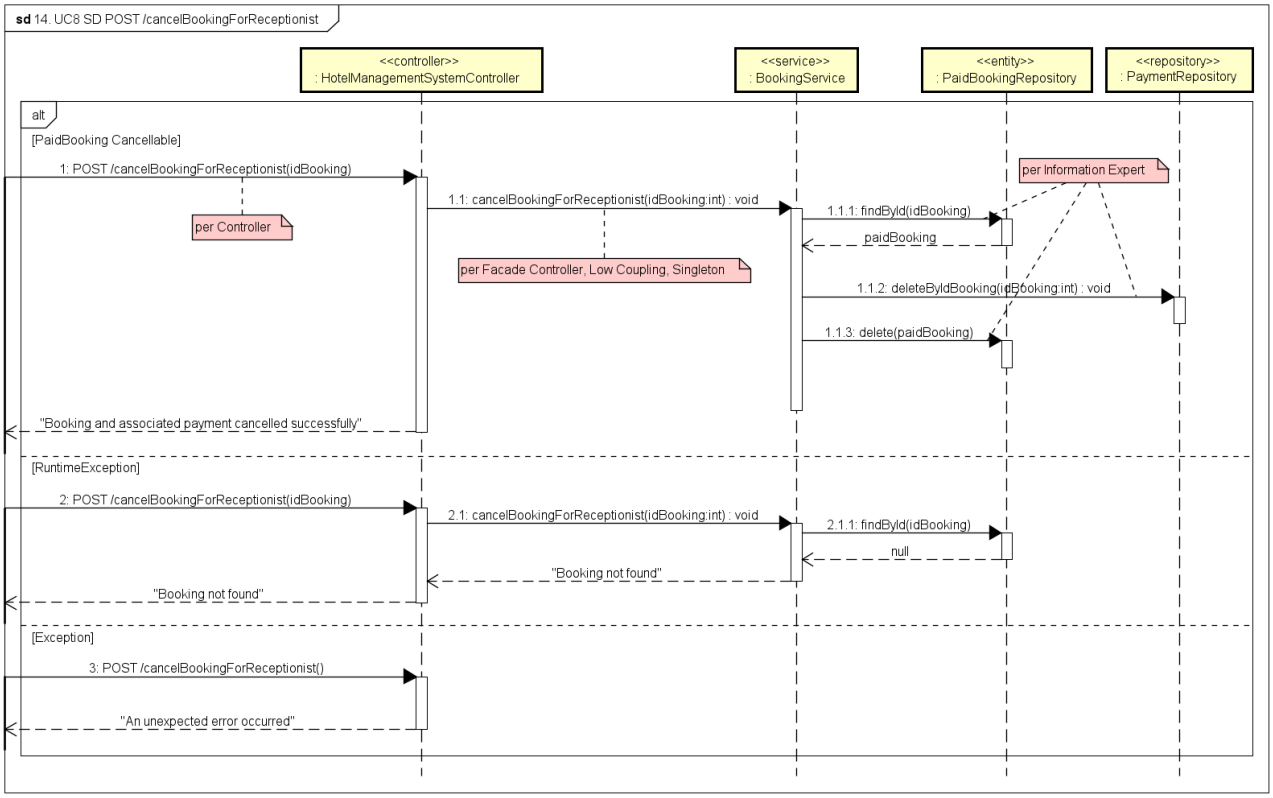


2.6.10. CASO D'USO UC8, DIAGRAMMI DI SEQUENZA

2.6.10.1. *GET /getPaidBookingsForReceptionist()*

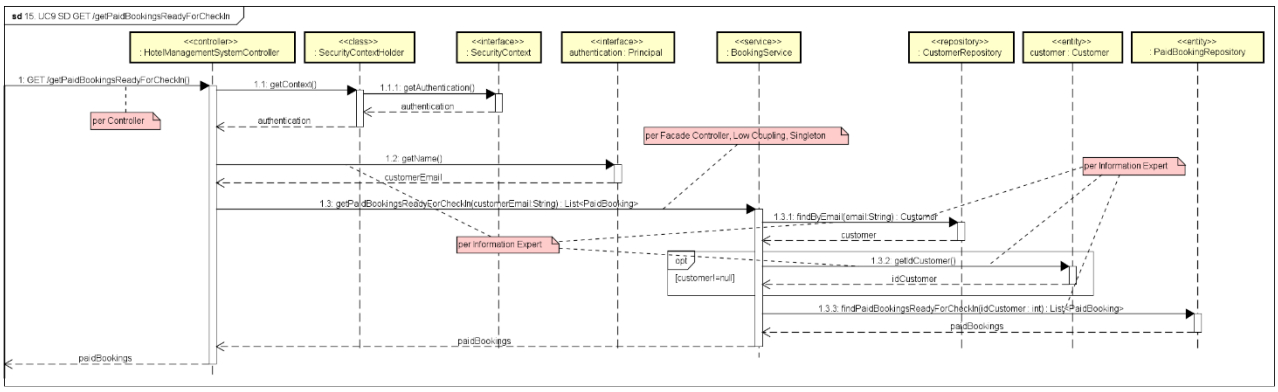


2.6.10.2. *POST /cancelBookingForReceptionist(idBooking)*

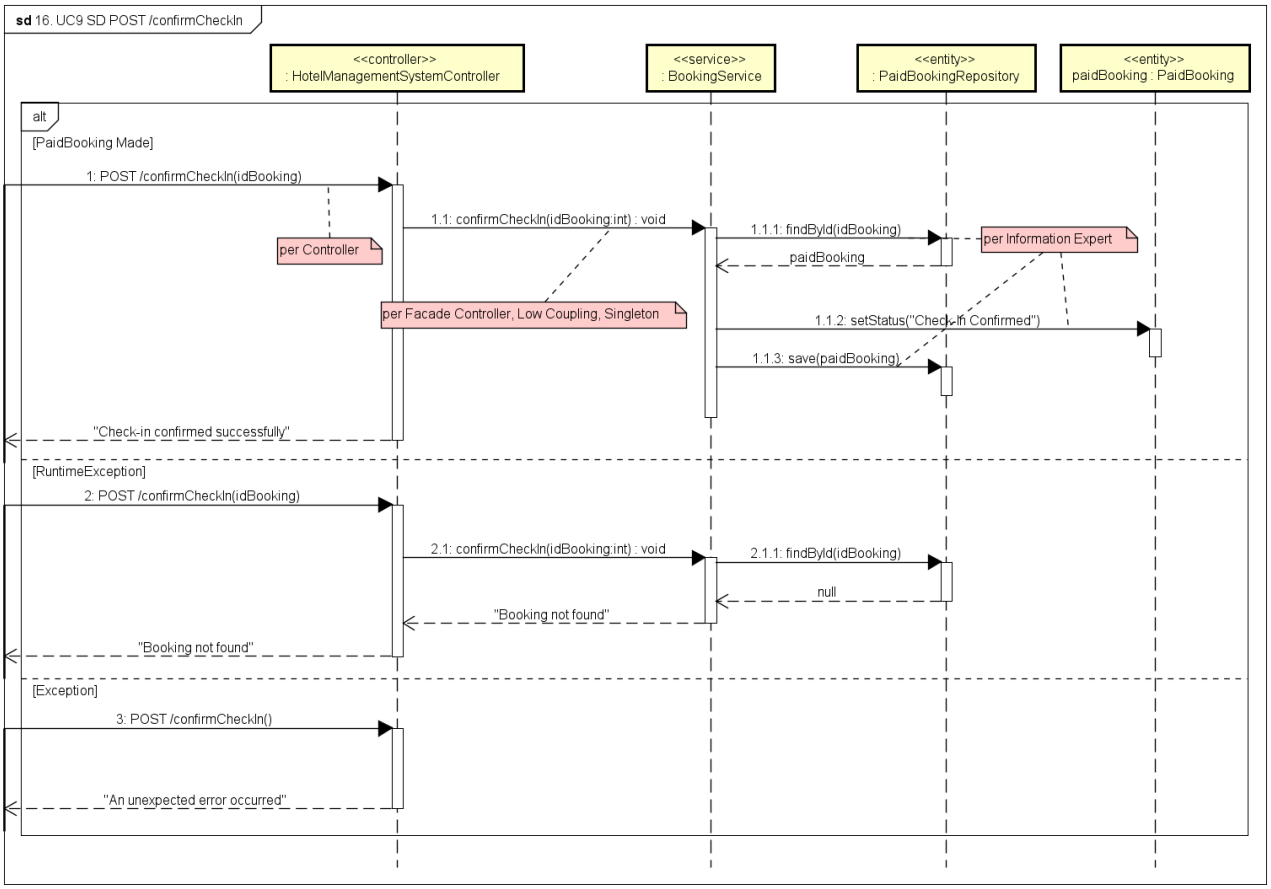


2.6.11. CASO D'USO UC9, DIAGRAMMI DI SEQUENZA

2.6.11.1. *GET /getPaidBookingsReadyForCheckIn()*

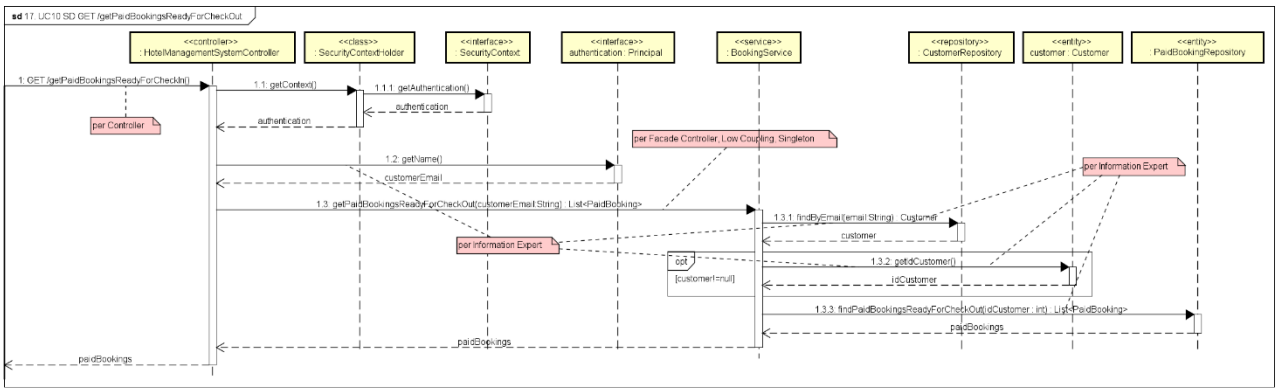


2.6.11.2. *POST /confirmCheckIn(idBooking)*

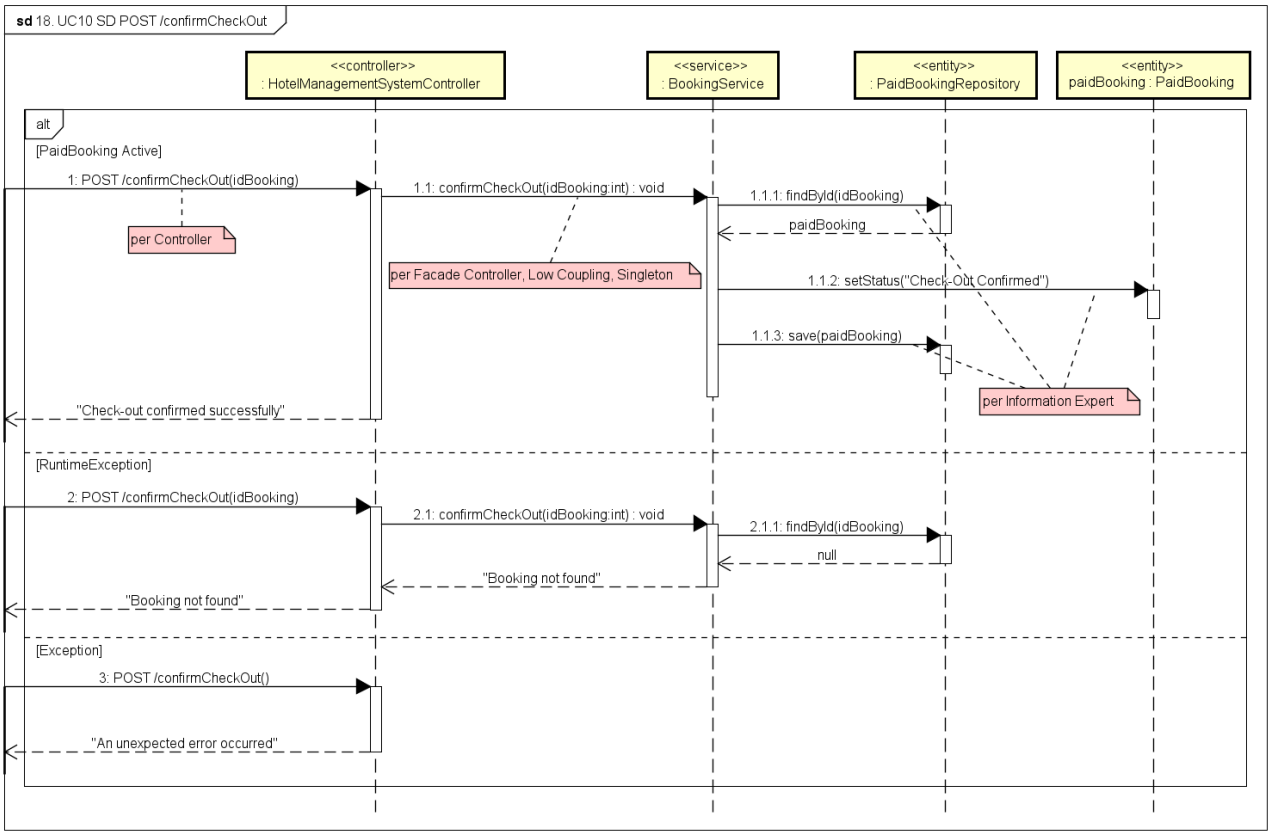


2.6.12. CASO D'USO UC10, DIAGRAMMI DI SEQUENZA

2.6.12.1. *GET /getPaidBookingsReadyForCheckOut()*

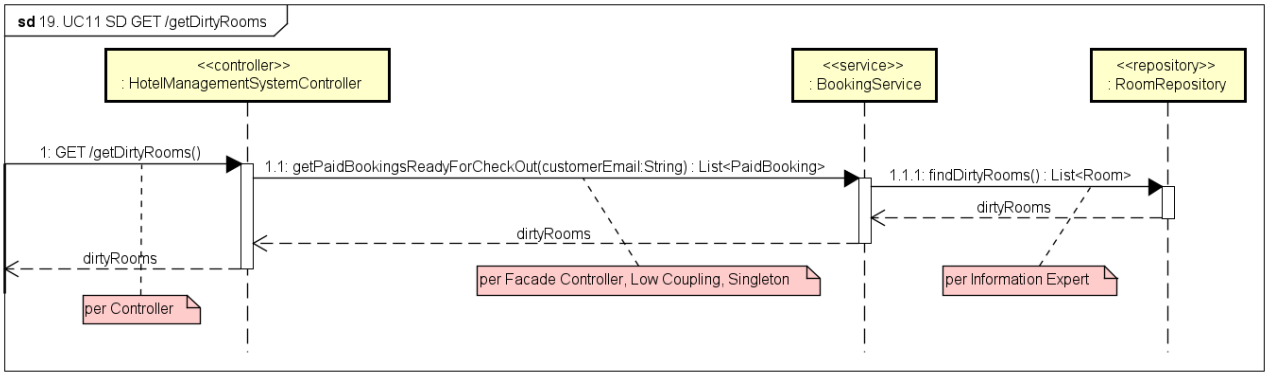


2.6.12.2. *POST /confirmCheckOut(idBooking)*

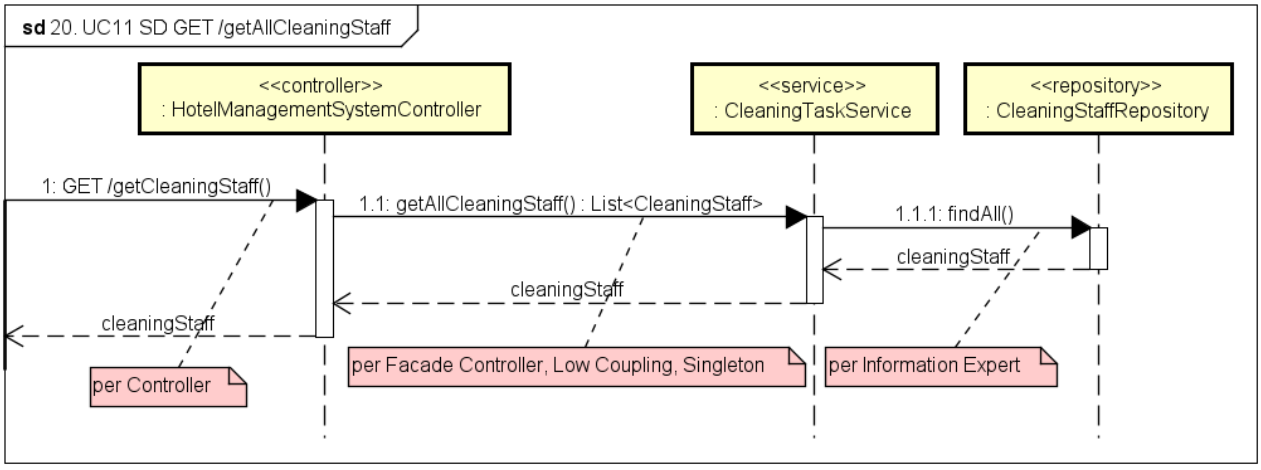


2.6.13. CASO D'USO UC11, DIAGRAMMI DI SEQUENZA

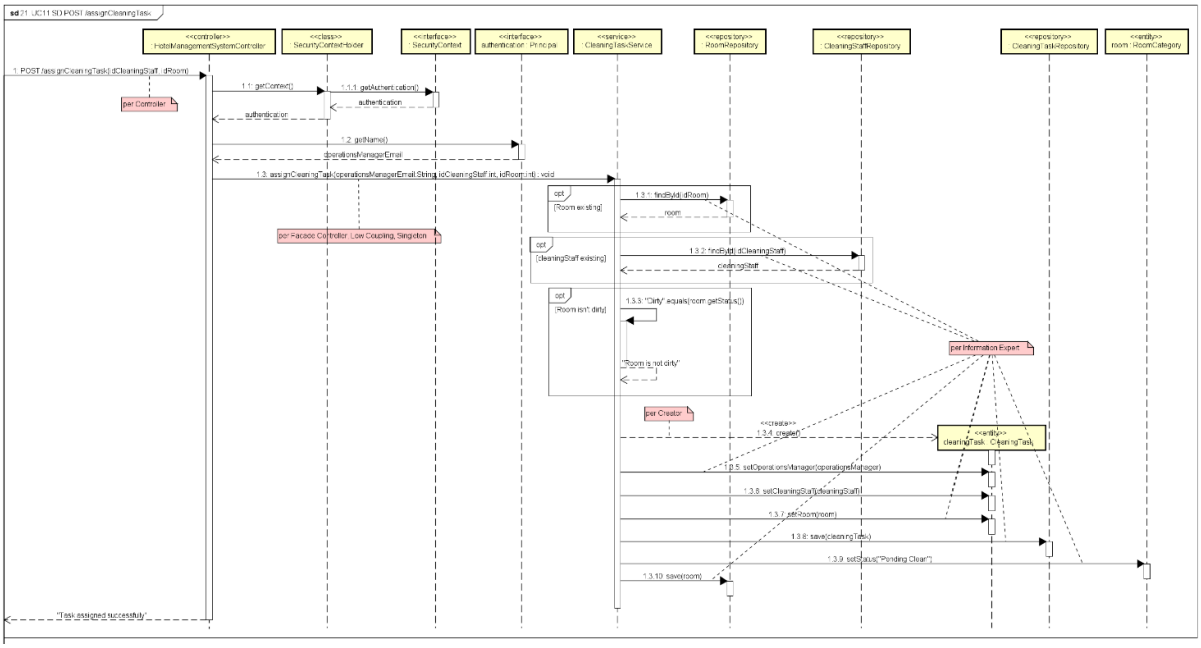
2.6.13.1. *GET /getDirtyRooms()*



2.6.13.2. *GET /getAllCleaningStaff()*

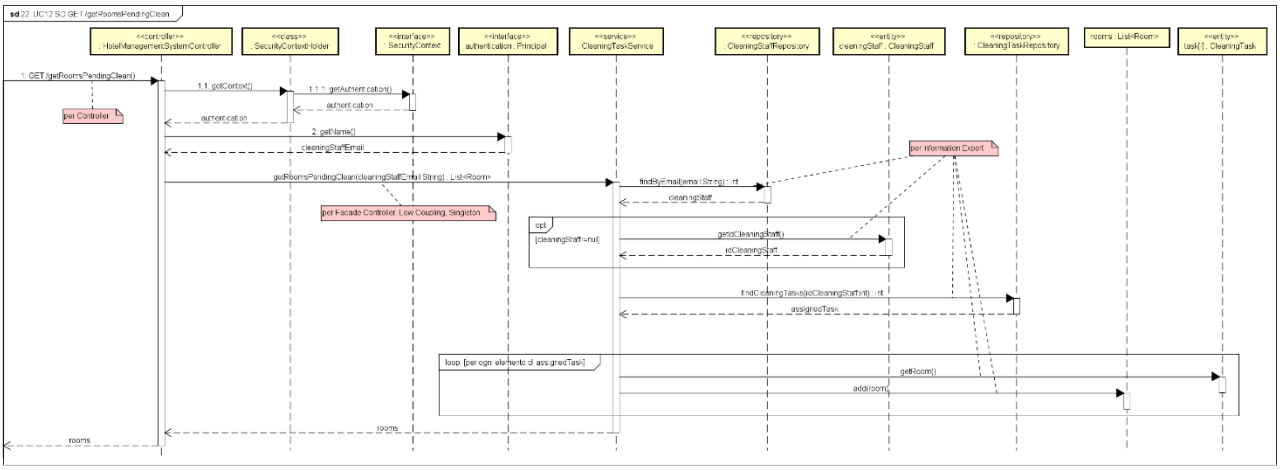


2.6.13.3. *POST /assignCleaningTask(idCleaningStaff, idRoom)*

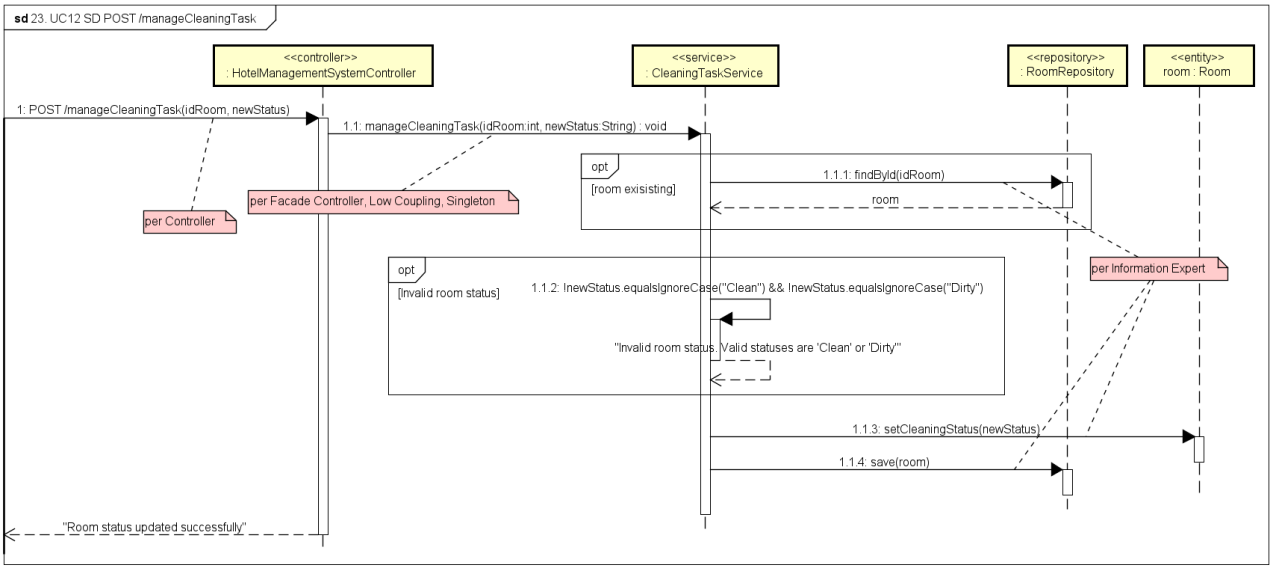


2.6.14. CASO D'USO UC12, DIAGRAMMI DI SEQUENZA

2.6.14.1. *GET /getRoomsPendingClean()*



2.6.14.2. *POST /manageCleaningTask(idRoom, newStatus)*



2.5.15. DIAGRAMMA DELLE CLASSI DI PROGETTO

L'architettura del sistema adottata segue il pattern **Model-View-Controller** (**MVC**), garantendo una chiara separazione tra i livelli applicativi e favorendo una progettazione modulare e scalabile.

Il sistema implementa inoltre un approccio **ORM** (**Object-Relational Mapping**) per la gestione della persistenza dei dati, consentendo di interagire con un **database relazionale** in modo strutturato ed efficiente.

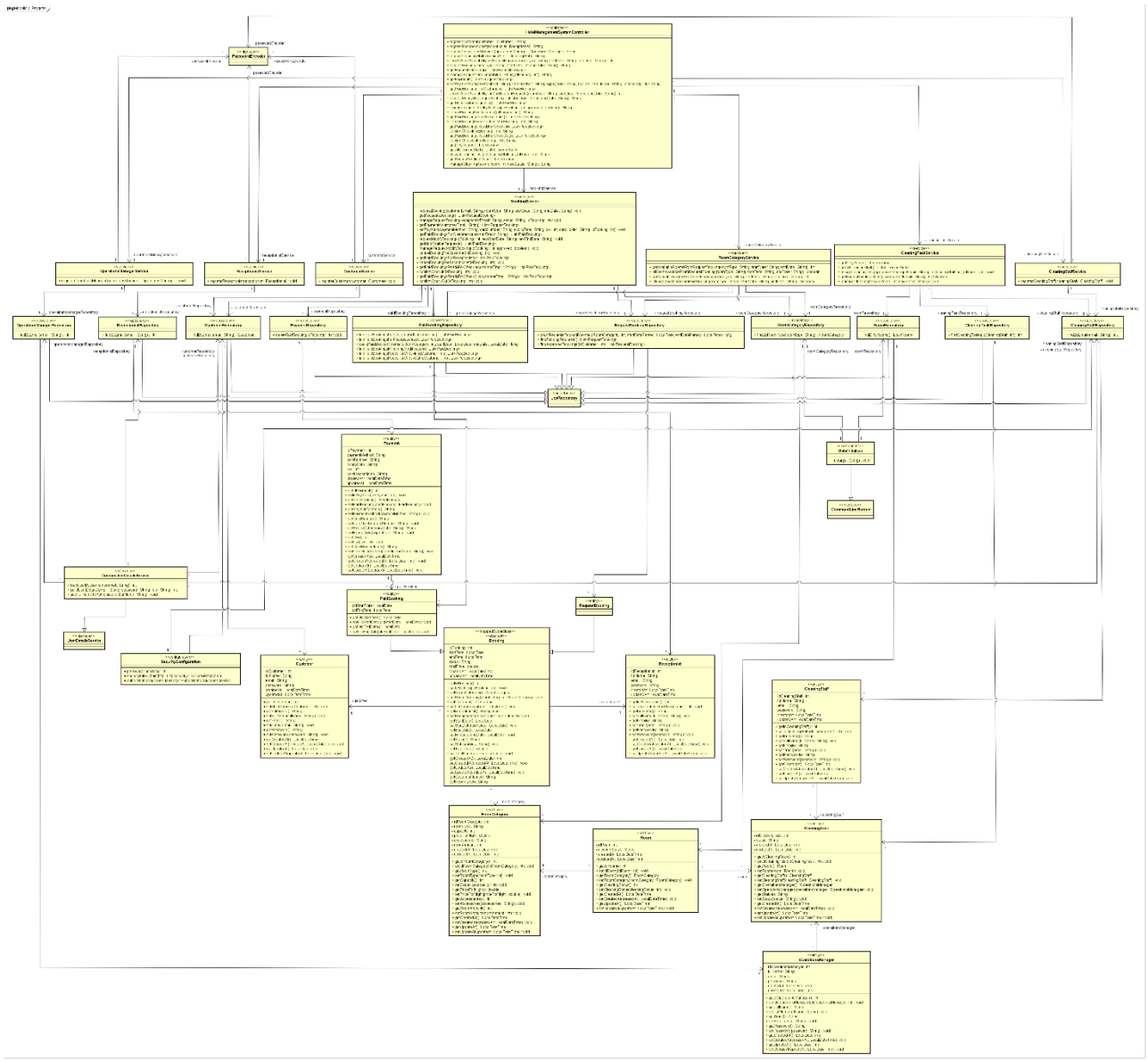
Il diagramma delle classi di progetto presentato in questa iterazione include le seguenti categorie di classi:

- **Model Classes**: rappresentano le entità del dominio e definiscono la struttura dei dati gestiti dal sistema. Queste classi modellano gli oggetti di interesse

dell’applicazione, includendo attributi e relazioni che consentono di mantenere la coerenza tra la logica applicativa e la rappresentazione dei dati nel database.

- **Controller Classes**: fungono da interfaccia tra il sistema e gli utenti, gestendo le richieste in ingresso e orchestrando le operazioni necessarie per generare le risposte appropriate. Queste classi operano come endpoint, ricevendo input, validando i dati e delegando l’elaborazione al livello di servizio.
- **Service Classes**: incapsulano la logica di business e costituiscono il livello intermedio tra i controller e la persistenza. Il loro ruolo è quello di garantire il rispetto delle regole applicative e di centralizzare l’implementazione delle operazioni fondamentali del sistema, evitando la duplicazione di codice e migliorando la manutenibilità.
- **Repository Interfaces**: gestiscono l’accesso ai dati e l’interazione con il database relazionale. Oltre a fornire metodi CRUD standard, queste interfacce includono query personalizzate, ottimizzate per le esigenze specifiche del sistema. L’integrazione di interrogazioni mirate consente di migliorare le prestazioni e di rispondere a scenari complessi che non possono essere gestiti esclusivamente con operazioni di base.

L’adozione di questa struttura architetturale assicura una chiara suddivisione delle responsabilità e una gestione efficiente delle risorse applicative, ponendo le basi per un sistema scalabile, estensibile e facilmente manutenibile nelle iterazioni successive.



2.7. TESTING

2.7.1. INTRODUZIONE

Il processo di **testing** ha rappresentato una fase cruciale nello sviluppo del **Sistema di Gestione per un Hotel**, finalizzato a garantire la **correttezza funzionale**, la **stabilità operativa** e l'**affidabilità complessiva** dell'applicazione. Attraverso l'utilizzo di strumenti avanzati come **JUnit** e **Mockito**, sono stati effettuati controlli approfonditi sulle principali funzionalità del sistema, con un'attenzione particolare alle singole **componenti** e alle loro **interazioni dirette**.

La strategia di testing adottata ha previsto un focus mirato sui seguenti aspetti:

- La **logica di business** dei servizi.
- La corretta gestione delle **API REST**.
- L'**esperienza utente** attraverso le interfacce grafiche e i flussi di navigazione.

2.7.2. VERIFICA DELLE FUNZIONALITÀ DEI SERVIZI

I **servizi principali** sono stati sottoposti a un'accurata attività di testing, mirata a validare il corretto funzionamento delle operazioni di **gestione degli utenti**, **prenotazioni** e **task operativi**.

I test hanno riguardato le seguenti componenti:

- **Gestione utenti**: Le funzionalità relative alla registrazione di **clienti**, **receptionist**, **operations manager** e **cleaning staff** sono state verificate attraverso l'uso di **mocking** per isolare le dipendenze esterne, come il database e i meccanismi di **hashing** delle credenziali.
- **Gestione delle prenotazioni**: Sono stati testati i metodi relativi alla creazione, modifica, cancellazione e gestione delle prenotazioni, inclusi i processi di pagamento e verifica della disponibilità delle camere.
- **Assegnazione delle attività**: La logica di assegnazione e monitoraggio delle attività operative per il personale di pulizia è stata testata per garantire la corretta distribuzione e gestione delle stanze.

Questa fase ha permesso di individuare e risolvere potenziali malfunzionamenti nelle operazioni critiche del sistema, migliorando la robustezza complessiva del codice.

2.7.3. VALIDAZIONE DEI CONTROLLER E DELLE API REST

I controller sono stati sottoposti a una serie di test approfonditi per verificare la corretta gestione delle **richieste HTTP** e delle risposte restituite agli utenti.

I test principali hanno riguardato:

- La **gestione degli endpoint** per le funzionalità chiave del sistema, tra cui la registrazione degli utenti, la gestione delle prenotazioni e le operazioni di pagamento.
- La verifica delle **risposte HTTP**, assicurando che ogni endpoint restituisse i codici di stato corretti e messaggi coerenti con le operazioni richieste.

- La simulazione dell'**autenticazione utente** tramite **SecurityContextHolder**, per testare scenari realistici di accesso alle funzionalità riservate a specifici ruoli.

Questi test hanno assicurato la **corretta esposizione** delle API REST e la gestione delle interazioni tra il sistema e i client esterni.

2.7.4. TESTING DELLE INTERFACCE UTENTE E DEI FLUSSI DI NAVIGAZIONE

L'esperienza utente è stata verificata attraverso test specifici sui **controller delle viste**, assicurando la corretta gestione delle interfacce grafiche e dei flussi di navigazione. I test hanno incluso:

- La verifica dei **reindirizzamenti** tra le diverse pagine, in base ai diversi ruoli utente (cliente, receptionist, operations manager, personale di pulizia).
- Il controllo della **coerenza delle viste**, con particolare attenzione alle sezioni personali e alle funzionalità specifiche per ciascun ruolo.
- La simulazione di scenari realistici di interazione, con l'inserimento di dati di esempio e la verifica dell'integrità delle informazioni mostrate.

L'esecuzione di questi test ha contribuito a migliorare la **usabilità** dell'applicazione e a prevenire eventuali errori di navigazione.

2.7.5. CONSIDERAZIONI FINALI

L'approccio adottato ha previsto una copertura completa delle funzionalità principali del sistema, con un focus particolare sulla verifica della **logica di business**, delle **API REST** e delle **interfacce utente**.

Grazie all'utilizzo di **JUnit** e **Mockito**, è stato possibile simulare scenari complessi, testare le dipendenze in modo isolato e garantire la **stabilità complessiva** del sistema senza dover ricorrere a un caricamento completo del contesto di **Spring Boot**.

Il risultato è un sistema robusto, in grado di gestire correttamente le diverse operazioni richieste dagli utenti, con una particolare attenzione alla **sicurezza** e all'**efficienza** del codice.