**Hotel Service - Comprehensive Documentation**
**Turcu Flavius CTI Ro anul 3 Seria A grupa 30235**

## ç Table of Contents



---

## Project Overview

### What is Hotel Service?

The Hotel Service is a comprehensive microservice designed to manage all aspects of hotel operations within a hotel chain. It serves as the backbone for hotel management, providing different interfaces for various types of users while maintaining data consistency and business logic integrity.

### Core Purpose

This system addresses the complex needs of modern hotel management by providing:

- **Centralized hotel and room management** for administrators
- **Property-specific operations** for hotel managers
- **Daily operational support** for hotel employees
- **Public browsing and booking interfaces** for guests

### Key Business Value

- **Operational Efficiency**: Streamlines hotel operations from room status updates to guest reviews
- **Multi-Role Support**: Accommodates different user types with appropriate access levels
- **Data Consistency**: Ensures reliable data management across all hotel properties
- **Scalability**: Built to handle multiple hotels with thousands of rooms
- **Guest Experience**: Provides seamless browsing and review capabilities for customers

---

## System Architecture

### Domain-Driven Design Philosophy

The Hotel Service follows Domain-Driven Design (DDD) principles, organizing the system into four distinct layers that separate business concerns from technical implementation:

**Domain Layer - The Business Heart** This layer contains the core business entities that represent real-world concepts:

- Hotels with their locations, amenities, and descriptions
- Rooms with pricing, facilities, and availability status
- Guest reviews with ratings and comments
- Room images for marketing and presentation

The domain layer also defines repository interfaces that specify how data should be accessed without dictating the technical implementation.

**Application Layer - Business Logic Orchestration** This layer contains services that implement business rules and coordinate between different domain entities:

- Hotel Service manages hotel creation, updates, and business validation
- Room Service handles room operations, availability logic, and image management
- Review Service manages the rating system with validation rules (1-5 stars)
- Authentication Service ensures proper access control across user roles

**Infrastructure Layer - Technical Implementation** This layer provides concrete implementations of domain interfaces:

- SQLite database repositories that handle data persistence
- Database models that map domain entities to database tables
- Connection management and transaction handling
- External service integrations

**Interface Layer - External Communication** This layer handles all external interactions:

- REST API controllers that expose functionality to clients
- Data Transfer Objects (DTOs) that shape request and response formats
- Error handling and HTTP status code management
- Authentication middleware and request validation

**Architectural Benefits**
The layered architecture provides several key advantages:

- **Testability**: Each layer can be tested independently
- **Maintainability**: Changes in one layer don't affect others
- **Flexibility**: Database or API changes don't impact business logic
- **Clarity**: Business rules are clearly separated from technical concerns

## 🍸 User Workflows

The system supports four distinct user types, each with tailored workflows and capabilities:

### Admin User Journey

Administrators have complete system oversight and can manage the entire hotel chain:

**Hotel Management Flow**: Admins can create new hotel properties by providing essential details like location, address, and amenities. They can view all hotels across the chain, update hotel information, and remove properties when necessary. The system validates all hotel data and maintains referential integrity when hotels are deleted.

**Room Operations Flow**: Admins can add rooms to any hotel, specifying room types, pricing, and facilities. They can update room details, manage availability status, and handle room images. The system ensures unique room numbers within each hotel and validates pricing rules.

**Review Oversight Flow**: Admins can view all reviews across the hotel chain, moderate inappropriate content, and analyze review trends. They can create reviews for testing purposes and delete problematic reviews when necessary.

### Manager User Journey

Hotel managers focus on their specific property with enhanced operational capabilities:

**Property Management Flow**: Managers can update their hotel's description and amenities to reflect current offerings. They can modify hotel policies and ensure accurate representation of their property's features.

**Room Strategy Flow**: Managers can adjust room availability and pricing based on demand patterns. They can update room facilities, manage seasonal pricing, and handle room images to maintain an attractive online presence.

**Performance Monitoring Flow**: Managers can access detailed analytics about their property including occupancy rates, revenue metrics, and guest satisfaction scores. They can monitor recent reviews and respond to guest feedback.

**Guest Relations Flow**: Managers can track guest reviews and identify areas for improvement. They can analyze feedback patterns and implement service enhancements based on guest suggestions.

### Employee User Journey

Hotel employees handle day-to-day operational tasks with focused, task-specific access:

**Room Status Management Flow**: Employees can update room availability as guests check in and out. They can mark rooms for cleaning or maintenance and track housekeeping progress throughout the day.

**Guest Service Flow**: Employees can access guest information for checked-in rooms to provide personalized service. They can view guest preferences and special requests to enhance the guest experience.

**Maintenance Coordination Flow**: Employees can check maintenance schedules and update task completion status. They can flag urgent issues and coordinate with maintenance teams.

**Review Handling Flow**: Employees can monitor guest reviews for immediate concerns and escalate issues to management when necessary. They can flag reviews that require management attention.

**Client User Journey**
Guests and potential customers can browse and interact with hotel services through public interfaces:

**Hotel Discovery Flow**: Clients can search for hotels by location and filter results based on amenities like pools, WiFi, or fitness centers. They can view detailed hotel information including descriptions and available facilities.

**Room Exploration Flow**: Clients can browse available rooms in their desired hotels, comparing prices, room types, and facilities. They can view room images and read detailed descriptions of room features.

**Review Research Flow**: Clients can read reviews from previous guests to make informed decisions. They can filter reviews by rating and read recent feedback to understand current service quality.

**Feedback Sharing Flow**: After staying at a property, clients can leave reviews and ratings to help future guests. The system validates that users can only review rooms where they have actually stayed.

---

## Database Design
The database structure reflects real-world hotel operations through four interconnected tables:

**Core Entity Relationships**
**Hotels Table** serves as the foundation, storing essential property information including name, location, complete address, and marketing descriptions. The amenities field uses JSON format to flexibly store various hotel features like WiFi availability, pool access, gym facilities, and spa services.

**Rooms Table** connects directly to hotels through foreign key relationships. Each room has a unique identifier within its hotel, specified room type (like Standard, Deluxe, or Suite), current pricing, and position details (such as ocean view or garden view). The facilities field stores room-specific features like bed types, balcony access, and mini-bar availability.

**Room Images Table** supports marketing efforts by storing multiple images per room. Each image includes a URL reference and optional description, allowing hotels to showcase their best features and help guests make informed decisions.

**Reviews Table** captures guest feedback with ratings constrained to 1-5 stars and optional comment text. Reviews link to specific rooms and include user identification for accountability while maintaining guest privacy in public displays.

**Data Integrity and Performance**
The database design ensures data consistency through foreign key constraints that prevent orphaned records. When a hotel is deleted, the system handles cascading effects appropriately. Similarly, room deletions properly manage associated images and reviews.
Performance optimization comes through strategic indexing on frequently searched fields like hotel location, room pricing, and review ratings. The system can quickly locate hotels in specific cities, find rooms within price ranges, and filter reviews by rating levels.
The JSON fields for amenities and facilities provide flexibility while maintaining query performance. Hotels can add new amenities without schema changes, and the system can efficiently search for hotels with specific features.

---

F **Component Structure**
The component diagram illustrates how different parts of the system work together while maintaining clear separation of responsibilities:

**Interface Components**
These components handle external communication and user interactions:

**Controllers** manage HTTP requests and responses for different user types. The Hotel Controller handles administrative hotel management, while the Client Controller provides public browsing interfaces. Each controller validates input data and formats appropriate responses.

**Data Transfer Objects (DTOs)** shape the data flowing in and out of the system. Request DTOs define the structure for creating and updating entities, while Response DTOs control what information is shared with different user types.

**Business Logic Components**
These components implement core business rules and coordination:

**Services** contain the primary business logic for each domain area. The Hotel Service manages hotel creation and validation rules, the Room Service handles availability logic and pricing rules, and the Review Service enforces rating constraints and comment policies.

**Authentication Service** ensures that users can only perform actions appropriate to their roles. It validates permissions and enforces business rules about who can access what information.

**Domain Components**
These components represent the core business concepts:

**Entities** model real-world objects like hotels, rooms, and reviews. They contain business rules and validation logic that applies regardless of how the data is stored or accessed.

**Repository Interfaces** define how the system should interact with data without specifying technical implementation details. This allows the business logic to remain independent of database choices.

**Infrastructure Components**
These components handle technical implementation details:

**Repository Implementations** provide concrete data access using SQLite database technology. They translate between domain entities and database records while maintaining data integrity.

**Database Models** define the structure of database tables and their relationships. They handle the technical aspects of data storage while supporting the business requirements defined in the domain layer.

---

### 📝 Deployment Architecture
The deployment diagram shows how the system operates in real-world environments across development, testing, and production phases:

**Production Environment Architecture**
**Load Balancing Strategy**: The production environment uses Nginx as a load balancer to distribute incoming requests across multiple application servers. This ensures high availability and handles traffic spikes effectively. SSL termination occurs at the load balancer level, securing all client communications.

**Application Server Cluster**: Multiple Docker containers run the Hotel Service application, providing redundancy and horizontal scaling capabilities. Each container operates independently, allowing for rolling updates and maintenance without service interruption.

**Database Management**: The SQLite database operates in Write-Ahead Logging (WAL) mode to support concurrent access from multiple application servers. Regular automated backups ensure data protection, with backup files stored both locally and in cloud storage.

**Monitoring Infrastructure**: Dedicated monitoring servers run Prometheus for metrics collection, Grafana for visualization, and the ELK stack for log analysis. This provides comprehensive visibility into system performance and health.

**File Storage System**: A separate storage server handles room images and static files, with synchronization to cloud storage services for redundancy and content delivery network (CDN) distribution.

**Development and Testing Environments**
**Development Setup**: Developers work with local instances running in Docker containers, providing consistent environments across different machines. Hot reload capabilities enable rapid development cycles, while local SQLite databases allow independent development.

**Continuous Integration Pipeline**: GitHub Actions automatically runs tests, builds Docker images, and deploys to staging environments when code changes are pushed. This ensures code quality and deployment reliability.

**Staging Environment**: Production-like staging servers allow final testing before deployment. These servers use similar configurations to production but with separate databases and monitoring to avoid affecting live services.

**Client Access Patterns**
**Multi-Device Support**: The system serves different types of client devices, from admin desktops for comprehensive management to mobile devices for guest interactions. Each interface is optimized for its intended use case.

**Security and Performance**: All client communications use HTTPS encryption, and the load balancer implements rate limiting to prevent abuse. Content delivery networks serve static assets efficiently to global users.

**Scalability and Reliability**
**Horizontal Scaling**: The containerized architecture allows adding more application servers during peak demand periods. Load balancing ensures even distribution of requests across available servers.

**Backup and Recovery**: Automated backup systems protect against data loss, with both local and cloud storage options. Recovery procedures are regularly tested to ensure business continuity.

**Monitoring and Alerting**: Comprehensive monitoring covers application performance, infrastructure health, and business metrics. Automated alerts notify administrators of issues before they impact users.

---

## Features s Capabilities

### Multi-Role Access Control

The system provides carefully designed access levels for different user types:

**Administrative Capabilities** include complete system oversight with the ability to create, modify, and delete any hotel or room. Admins can access comprehensive analytics across all properties and manage user accounts and permissions.

**Management Capabilities** focus on specific properties with enhanced operational control over room pricing, availability, and hotel amenities. Managers can access detailed performance reports for their properties and respond to guest feedback.

**Employee Capabilities** support daily operations with room status updates, guest information access, and maintenance coordination. The system provides task-focused interfaces that streamline common operational workflows.

**Guest Capabilities** enable comprehensive browsing and search functionality with the ability to leave reviews and feedback. The public interface prioritizes ease of use and information discovery.

### Advanced Search and Filtering

The system provides sophisticated search capabilities:

**Location-Based Search** allows users to find hotels in specific cities or regions, with support for proximity-based results and geographic filtering.

**Amenity Filtering** enables searches for hotels with specific features like pools, fitness centers, business facilities, or pet-friendly policies.

**Room-Specific Search** supports finding rooms by type, price range, availability, and special features like ocean views or balcony access.

**Review-Based Filtering** allows users to find highly-rated properties and filter by guest satisfaction levels.

**Comprehensive Review System**
The review system maintains data quality while encouraging guest feedback:

**Rating Validation** ensures all ratings fall within the 1-5 star range with clear validation messages for invalid inputs.

**Verified Reviews** only allow reviews from guests who have actually stayed at the property, maintaining review authenticity.

**Moderation Capabilities** enable managers and admins to address inappropriate content while preserving legitimate feedback.

**Statistical Analysis** provides property-level metrics including average ratings, review counts, and trend analysis over time.

**Image Management System**
Professional presentation capabilities support marketing efforts:

**Multiple Images per Room** allow comprehensive visual representation of accommodations with support for different room angles and features.

**Organized Storage** maintains logical organization of images with proper categorization and easy retrieval.

**Performance Optimization** ensures fast image loading through appropriate sizing and content delivery network integration.

---

## ⊖ Technical Specifications

### Architecture Patterns

The system implements several proven architectural patterns:

**Domain-Driven Design (DDD)** provides clear separation between business logic and technical implementation, making the system easier to understand, maintain, and extend.

**Repository Pattern** abstracts data access operations, allowing business logic to remain independent of specific database technologies or data storage approaches.

**Service Layer Pattern** encapsulates business operations and coordinates between different domain entities, providing clear interfaces for complex business workflows.

**Dependency Injection** promotes loose coupling between components and enables comprehensive testing by allowing easy substitution of dependencies.

**Data Management Strategy**
The system uses SQLite as the primary database with several performance optimizations:

**Write-Ahead Logging (WAL)** mode enables concurrent read access while maintaining data consistency and improving performance for multi-user scenarios.

**Strategic Indexing** on frequently queried fields like location, price, and ratings ensures fast search performance even with large datasets.

**JSON Field Support** for amenities and facilities provides flexibility while maintaining query performance and data structure.

**Foreign Key Constraints** ensure referential integrity and prevent data inconsistencies across related entities.

**API Design Principles**
The REST API follows industry best practices:

**Resource-Based URLs** provide intuitive endpoints that map clearly to business entities and operations.

**HTTP Status Codes** accurately reflect operation results with consistent error handling and meaningful error messages.

**Request/Response Consistency** ensures predictable data formats and validation rules across all endpoints.

**Version Management** supports API evolution while maintaining backward compatibility for existing clients.

**Security Considerations**
The system implements multiple security layers:

**Role-Based Access Control** ensures users can only access appropriate functionality based on their assigned roles and responsibilities.

**Input Validation** prevents malicious data from entering the system through comprehensive validation at multiple layers.

**Error Handling** provides helpful messages without exposing sensitive system information that could be exploited.

**Audit Logging** tracks significant system operations for security monitoring and compliance requirements.

**Performance and Scalability**
The system is designed for growth and high performance:

**Horizontal Scaling** through containerization allows adding capacity during peak demand periods without service interruption.

**Database Optimization** includes proper indexing, query optimization, and connection management for efficient data access.

**Caching Strategies** reduce database load and improve response times for frequently accessed data.

**Load Balancing** distributes traffic evenly across multiple application instances for optimal resource utilization.
This comprehensive hotel management system provides a solid foundation for hotel chain operations while maintaining the flexibility to adapt to changing business requirements and scale with organizational growth.