# Documento de Arquitetura de Software

Versão 1.4

| *[CRM4SH] – [Costumer Relationship Management for Small Hotels]* | |
|---|---|
| **Orientador do Projeto** | **Equipe de Projeto (Documentação)** |
| Rosiberto dos Santos Gonçalves | Danilo Henrique Lira da Silva - 201502250861 |
| | Francisco Diego Farias Hilario - 201904044662 |
| | Jamile de Souza Alves - 202002543205 |
| | Júlio Paiva de Souza Filho - 201703162358 |
| | Mateus Luiz de Santos Oliveira - 201703072677 |

## *Objetivo deste Documento*

Este documento tem como objetivo descrever as principais decisões de projeto tomadas pela equipe de desenvolvimento e os critérios considerados durante a tomada destas decisões. Suas informações incluem a parte de *hardware* e *software* do sistema.

# HISTÓRICO DE ALTERAÇÕES

| Data | Versão | Descrição | Autor |
|---|---|---|---|
| 02/maiol/24 | 1.0 | Criação do documento de arquitetura | Danilo |
| 03/maio/24 | 1.0 | Ajustes, descrição e criação de conteúdo do documento | Danilo e Diego |
| 04/maio/24 | 1.1 | Casos de uso | Diego |
| 11/maio/24 | 1.2 | Visão Logica | Mateus |
| 12/maio/24 | 1.3 | Visão de Implantação | Jamile |
| 13/maio/24 | 1.4 | Visão de Implementação | Júlio |

# Tabela de Conteúdo

# 1. INTRODUÇÃO

## 1.1 Finalidade

Este documento fornece uma visão arquitetural abrangente do sistema CRM4SH usando diversas visões de arquitetura para **representar** diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.
O documento irá adotar uma estrutura baseada na visão “4+1” de modelo de arquitetura [KRU41].

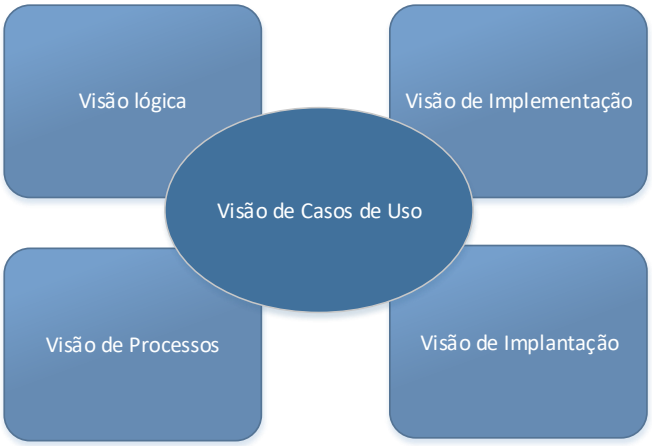



*Figura 1 – Arquitetura 4+1*

## 1.2 Escopo

Este Documento de Arquitetura de Software se aplica ao CRM4SH, que será desenvolvido pelas equipes de documentação, back-end, front-end, banco de dados e testes da Universidade
Estácio de Sá.

## 1.3 Definições, Acrônimos e Abreviações

QoS – Quality of Service, ou qualidade de serviço. Termo utilizado para descrever um conjunto de qualidades que descrevem as requisitos não-funcionais de um sistema, como performance, disponibilidade e escalabilidade[QOS].

### 1.4 Referências

[KRU41]: The “4+1” view model of software architecture, Philippe Kruchten, November 1995, https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf

[QOS] https://docs.oracle.com/cd/E19636-01/819-2326/6n4kfe7dj/index.html

## 2. REPRESENTAÇÃO ARQUITETURAL

Este documento irá detalhar as visões baseado no modelo “4+1” [KRU41], utilizando como referência os modelos definidos na MDS. As visões utilizadas no documento serão:

### 2.1 Visão Lógica (para Analistas)

• Responsabilidades: Realizar os casos de uso específicos para a hotelaria, entender a lógica do sistema e como os diferentes componentes interagem para atender aos requisitos funcionais do sistema.

• Conteúdo: Detalhes sobre as classes principais, como Quarto, Reserva, Hóspede, Pagamento, GestorReservas, e como elas se organizam em pacotes de serviços (como Gestão de Reservas, Gestão de Hóspedes, Gestão de Pagamentos) e subsistemas. Além disso, os diagramas de classes e sequência ilustrarão os relacionamentos entre essas classes e como elas realizam os casos de uso.

### 2.2 Visão de Processo (para Integradores)

• Responsabilidades: Garantir que o sistema de hotelaria atenda aos requisitos de performance, escalabilidade e concorrência.

• Conteúdo: Descrição dos processos de integração, dimensionamento de recursos (por exemplo, número de quartos e hóspedes suportados, transações de reserva por segundo), considerações de desempenho (otimização de consultas de banco de dados, cache de dados frequentemente acessados), e estratégias para lidar com a concorrência (como transações de reserva simultâneas).

### 2.3 Visão de Implementação (para Programadores)

• Responsabilidades: Desenvolver e implementar os componentes de software que compõem o sistema web de hotelaria.

• Conteúdo: Detalhes sobre a implementação de classes, métodos, APIs e frameworks utilizados no desenvolvimento do sistema, como Java Spring Framework para a camada de aplicação, Laravel Framework para a camada de apresentação, e Node.js para a camada de infraestrutura.

### 2.4 Visão de Implantação (para Gerência de Configuração)

• Responsabilidades: Gerenciar a implantação física do sistema de hotelaria web, incluindo servidores, redes e outros recursos de hardware.

• Conteúdo: Especificações dos nós físicos (por exemplo, servidores de aplicativos, banco de dados), configuração de servidores (como sistema operacional, recursos de hardware), rede (configuração de firewall, balanceamento de carga), e outros recursos de infraestrutura (como serviço de armazenamento em nuvem para backup de dados).

### 2.5 Visão de Caso de Uso (para todos)

• Responsabilidades: Capturar e entender os requisitos funcionais específicos da hotelaria para o sistema web.

• Conteúdo: Descrição dos casos de uso, como Efetuar Reserva de Quarto, Fazer Check-in de Hóspede, Realizar Pagamento, Gerenciar Reservas, e seus atores, fluxos principais e alternativos, requisitos associados e documentação relacionada.

### 2.6 Visão de Dados (para Especialistas e Administradores de Dados)

• Responsabilidades: Garantir a adequada persistência e manipulação dos dados do sistema de hotelaria web.

• Conteúdo: Modelo de dados (por exemplo, entidades como Quarto, Reserva, Hóspede, Pagamento, suas relações e atributos), esquemas de banco de dados (como tabelas, índices, chaves estrangeiras), considerações de segurança (criptografia de dados sensíveis, controle de acesso) e integridade dos dados (restrições de integridade referencial, transações de banco de dados).

## 3. REQUISITOS E RESTRIÇÕES ARQUITETURAIS

### 3.1 Requisitos Arquiteturais

#### 3.1.1 Desempenho

• O sistema deve ser capaz de lidar com um grande volume de acessos simultâneos, garantindo tempos de resposta rápidos para os usuários.
• As operações críticas, como reserva de quartos e check-in de hóspedes, devem ser executadas de forma eficiente para garantir uma experiência satisfatória do usuário.

#### 3.1.2 Escalabilidade

• A arquitetura do sistema deve ser escalável horizontalmente, permitindo a adição de novos servidores para lidar com o aumento da carga de trabalho conforme necessário.
• Deve ser possível escalar individualmente os diferentes componentes do sistema, como o servidor web, o servidor de aplicativos e o banco de dados.

#### 3.1.3 Segurança

• O sistema deve garantir a segurança dos dados dos clientes, utilizando práticas recomendadas de criptografia e autenticação.
• Deve haver controles de acesso adequados para proteger informações sensíveis, como dados de cartão de crédito e informações pessoais dos hóspedes.

#### 3.1.4 Usabilidade

• A interface do usuário deve ser intuitiva e fácil de usar, permitindo que os usuários naveguem facilmente pelo sistema e realizem tarefas como fazer reservas de quartos e gerenciar suas informações pessoais.
• O sistema deve ser responsivo e compatível com uma variedade de dispositivos e navegadores web para garantir uma experiência consistente em diferentes plataformas.

### 3.2 Restrições Arquiteturais

#### 3.2.1 Tecnologia

• O sistema deve ser desenvolvido utilizando tecnologias web modernas e amplamente adotadas, como HTML5, CSS3, JavaScript e frameworks de desenvolvimento web, como React.js ou AngularJS.
• Deve ser compatível com os padrões da indústria e seguir as melhores práticas de desenvolvimento web para garantir a manutenibilidade e a escalabilidade do código.

### 3.2.2 Interoperabilidade

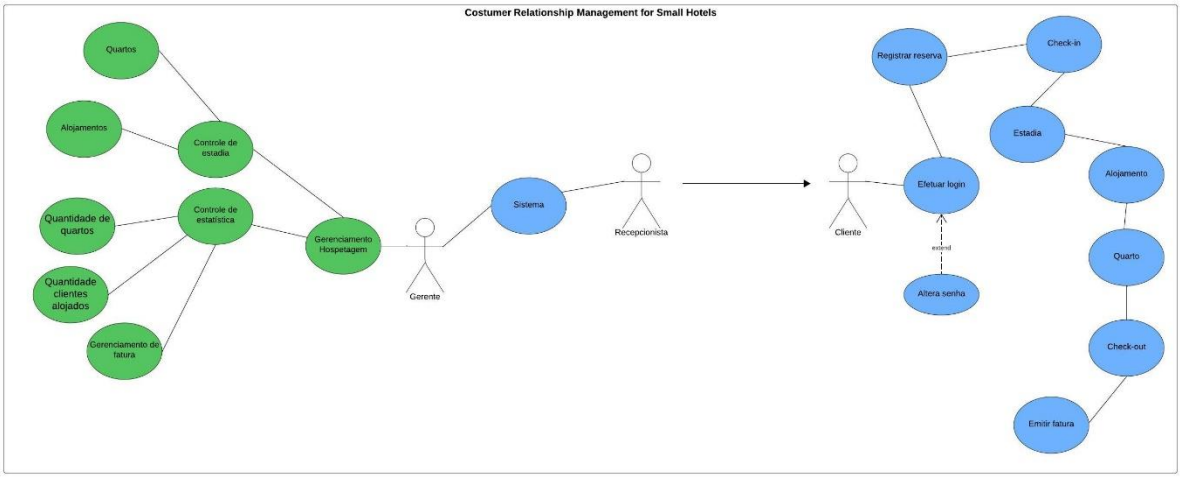
• O sistema deve ser capaz de integrar-se com sistemas externos, como sistemas de pagamento online, sistemas de gerenciamento de reservas de terceiros e sistemas de faturamento.
• Deve suportar protocolos de comunicação padrão, como HTTP/HTTPS e APIs RESTful, para facilitar a integração com outros sistemas.

### 3.2.3 Conformidade Regulatória

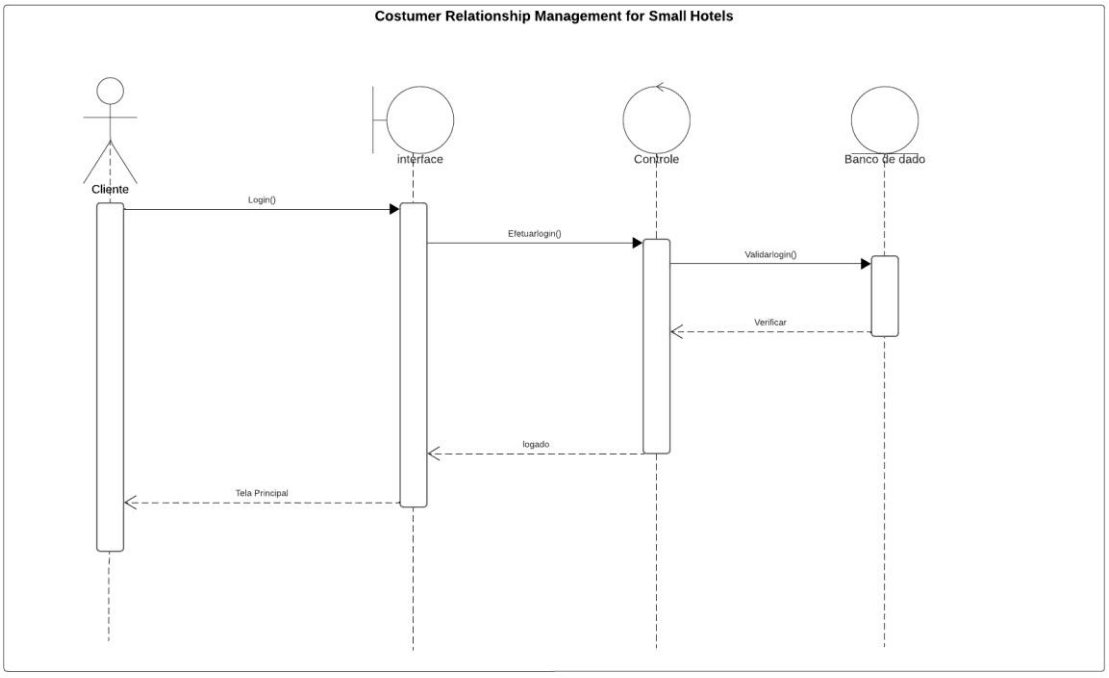
• O sistema deve estar em conformidade com as regulamentações de proteção de dados, como o Autoridade Nacional de Proteção de Dados (ANPD) órgão responsável para garantir a privacidade e segurança das informações dos usuários.
• Deve cumprir as regulamentações locais e internacionais relacionadas a transações financeiras online e armazenamento de dados sensíveis.

## 4. VISÃO DE CASOS DE USO

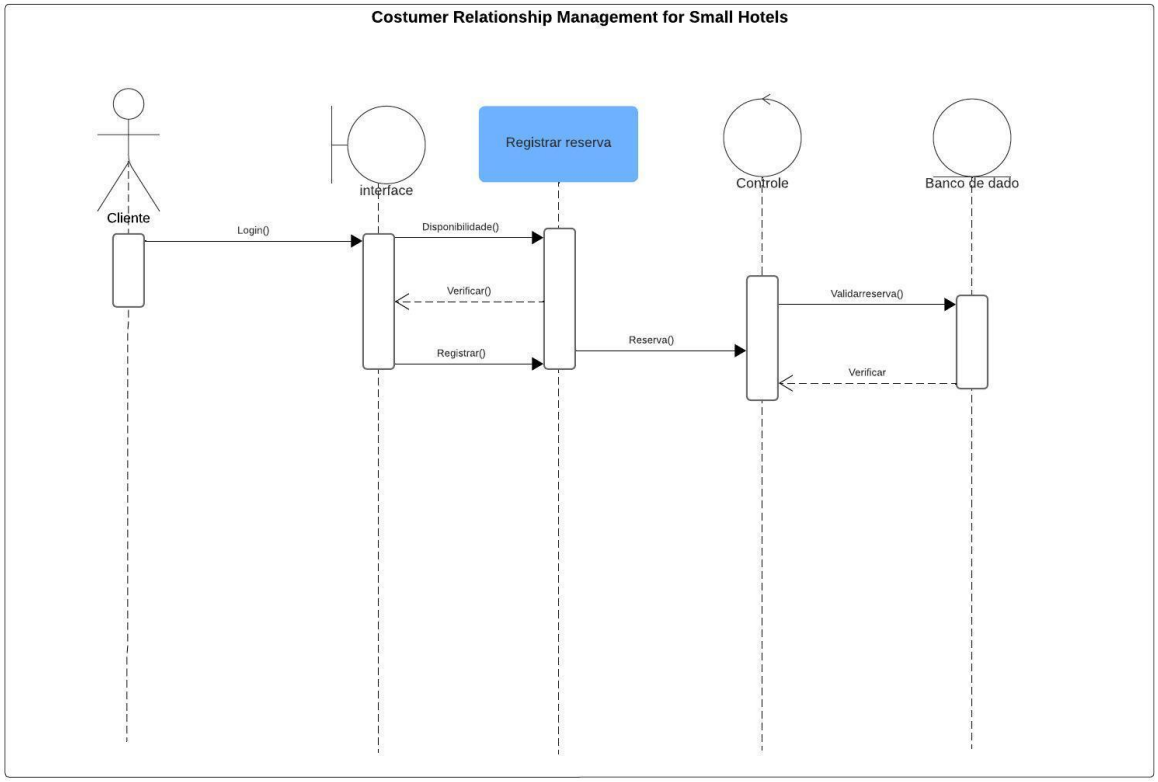
### 4.1 Casos de Uso significantes para a arquitetura



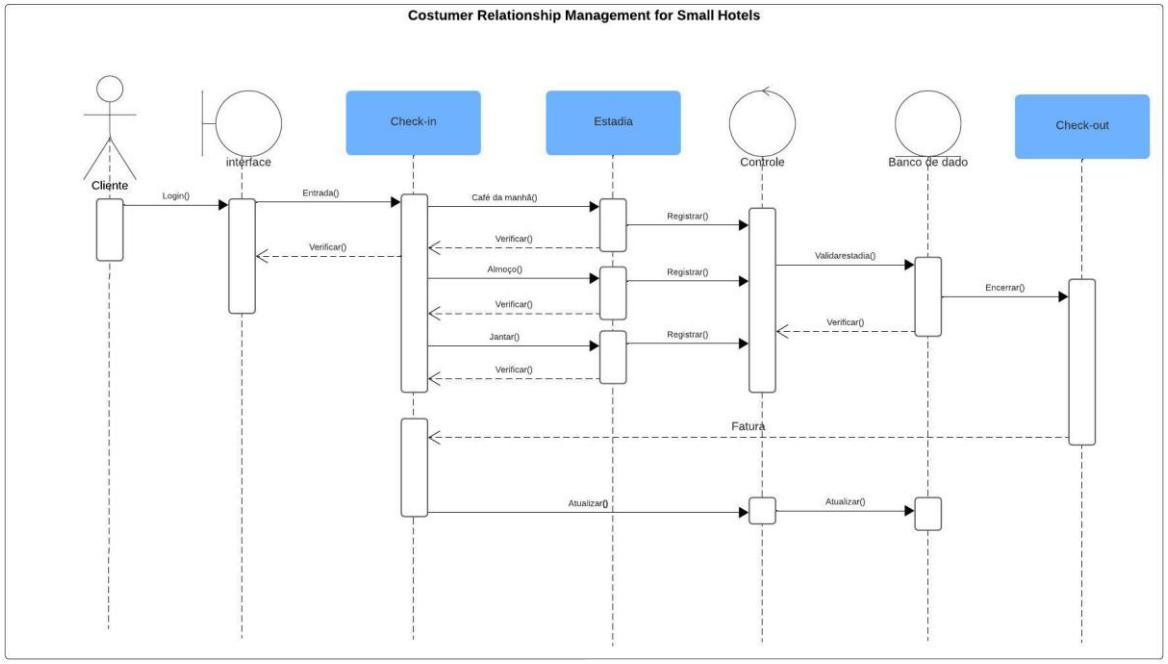

### 4.1.1 Login de Usuário



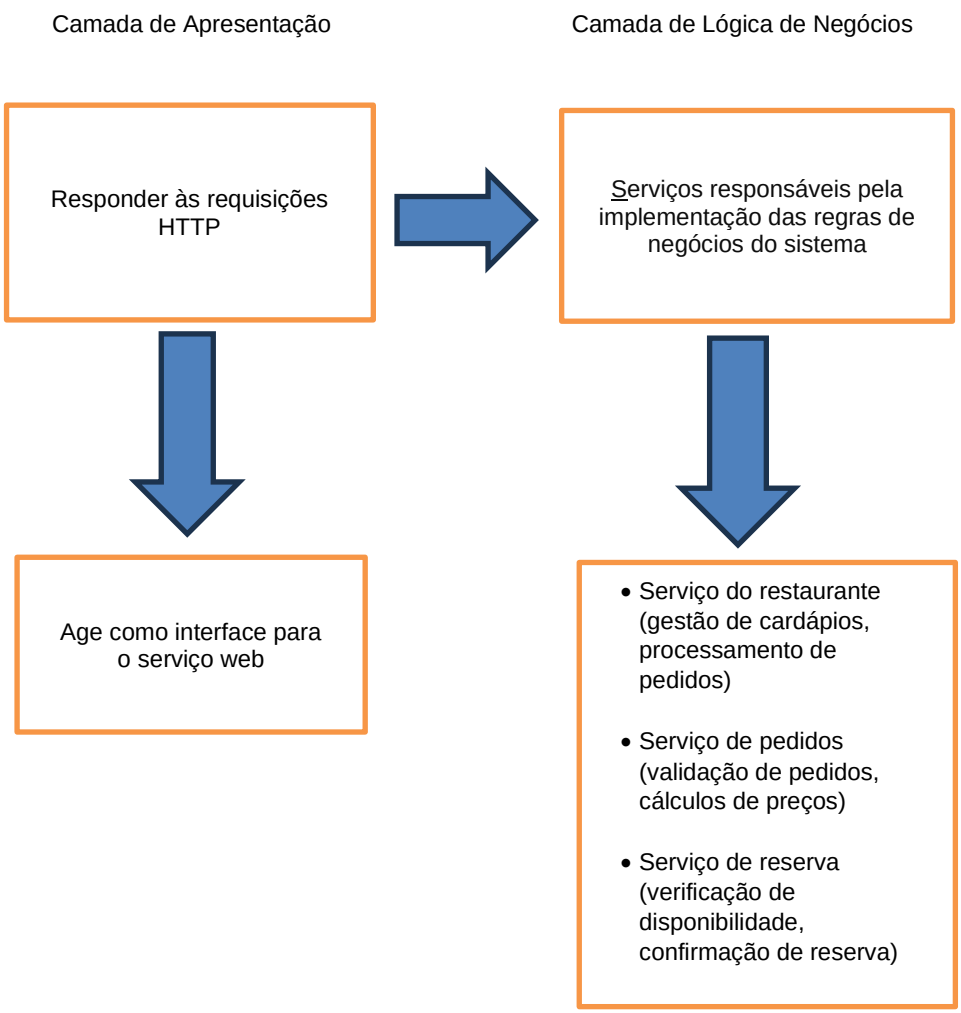
### 4.1.2 Registrar Reserva




### 4.1.2 Efetuar Check-in e Check-out



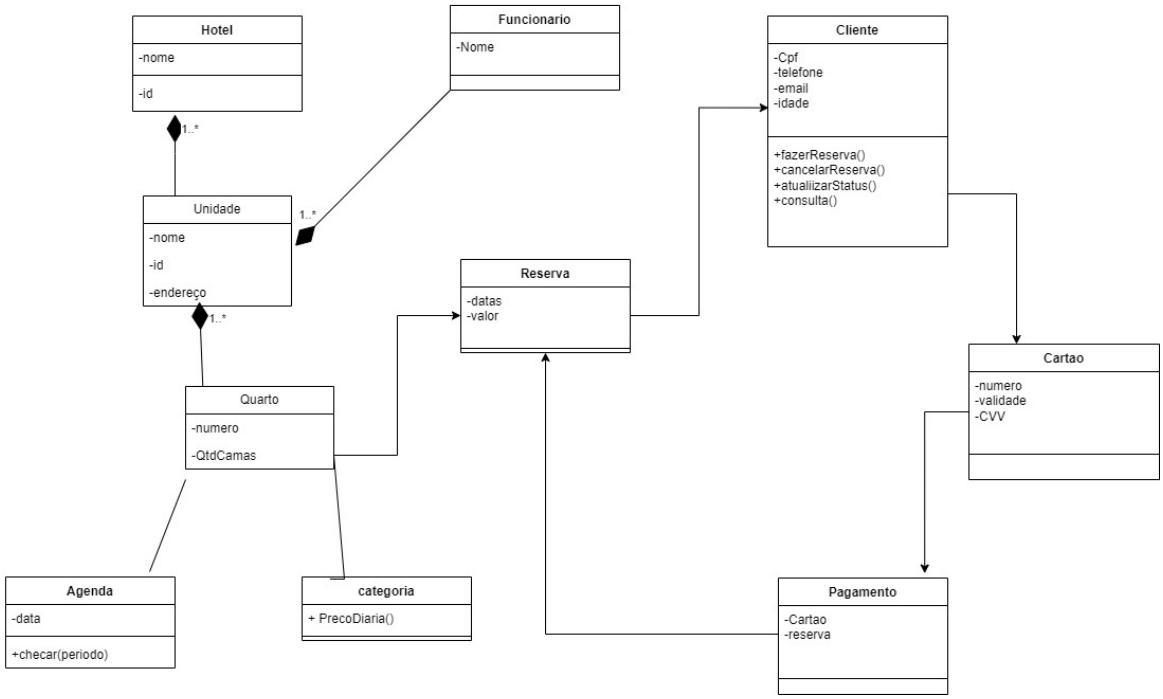
# 5. VISÃO LÓGICA




## 5.1 Visão Geral – pacotes e camadas

# 6. VISÃO DE IMPLEMENTAÇÃO

## 6.1 Diagrama de Classes

# 7. VISÃO DE IMPLANTAÇÃO

**Servidores Físicos:**

**Descrição:** Máquinas físicas dedicadas para hospedar os componentes do sistema.
**Configurações:**
**Processador:** Intel Xeon, 8 núcleos
**Memória RAM:** 32 GB
**Armazenamento:** SSD 500 GB
**Sistema Operacional:** Linux CentOS 7

**Banco de Dados:**

**Descrição:** Instância dedicada para armazenar os dados do sistema.
**Configurações:**
**Banco de Dados:** PostgreSQL 12
**Memória RAM:** 16 GB
**Armazenamento:** RAID 10 HDD 1 TB

**Balanceadores de Carga:**

**Descrição:** Dispositivos dedicados para distribuir o tráfego entre os servidores de aplicativos.
**Configurações:**
**Dispositivo:** F5 BIG-IP LTM
**Capacidade:** Até 10 Gbps
**Protocolos:** HTTP, HTTPS

**Servidores de Aplicativos:**

**Descrição:** Máquinas dedicadas para executar a lógica de negócios da aplicação.
**Configurações:**
**Servidor de Aplicativos:** Apache Tomcat 9
**Java Virtual Machine:** OpenJDK 11
**Memória RAM:** 8 GB
**Armazenamento:** SSD 250 GB
**Servidores de Monitoramento:**
**Descrição:** Máquinas dedicadas para monitorar o desempenho e a integridade do sistema.
**Configurações:**
**Ferramenta de Monitoramento:** Prometheus
**Memória RAM:** 4 GB
**Armazenamento:** HDD 100 GB

### Artefatos Implantados

**Código-fonte do Sistema:**
**Descrição:** Artefato contendo o código-fonte do sistema de reservas de hotel.
**Implantação:** Compilado e implantado nos servidores de aplicativos.

**Base de Dados:**
**Descrição:** Estrutura de banco de dados contendo tabelas, índices e procedimentos armazenados.
**Implantação:** Criada e gerenciada no servidor de banco de dados PostgreSQL.

**Arquivos de Configuração:**
**Descrição:** Arquivos de configuração do servidor de aplicativos, banco de dados e balanceadores de carga.
**Implantação:** Distribuídos e configurados nos respectivos nós físicos.

# 8. DIMENSIONAMENTO E PERFORMANCE

## 8.1 Volume

- Número de estimado usuários: 3000 usuários
- Número estimado de acessos diários: 500 usuários
- Número estimado de acessos por período: 5000 usuários em uma semana
- Tempo de sessão de um usuário: 30 minutos

## 8.2 Performance

- Tempo máximo para a execução de determinada transação: 1 minuto

# 9. QUALIDADE

## 9.1 Escalabilidade

O sistema deve ter capacidade de lidar com os possíveis aumentos de demanda, seja em termo de quantidade de usuário, volume de dados, carga de trabalho ou recursos computacionais necessários.

## 9.2 Confiabilidade

O sistema deve ter a capacidade de executar suas funções conforme esperado, sem falhas ou erros, mesmo diante de condições adversas, é fundamental para garantir uma experiência confiável aos usuários.

## 9.3 Disponibilidade

O sistema deve estar operacional e acessível sempre que necessário, garantindo que os usuários possam interagir com ele sem interrupções significativas, a alta disponibilidade é essencial para garantir continuidade dos serviços e evitar impactos negativos.

## 9.4 Portabilidade

O sistema deve ter a capacidade de ser facilmente transferido ou adaptado para diferentes ambientes de execução, plataformas de hardware, sistemas operacionais ou ambientes de desenvolvimento, o sistema deve permitir que seja implantado e executado em uma variedade de ambientes sem sofrer modificações significativas.

## 9.5 Segurança

O sistema deve ter a capacidade de proteger os seus dados, recursos e funcionalidades contra ameaças maliciosas, violações de segurança e acesso não autorizado, a implementação de autenticação e autorização, bem como controle de acesso e criptografia, são essenciais para garantir estes critérios.