Documento de Arquitetura de Software

Versão 1.6

[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
20/maio/24	1.5	Ajustes de linguagem e arquitetura baseado nas informações do Front e referências	Danilo
25/maio/24	1.6	Ajuste diagrama de classes passado pela equipe de Banco de Dados	Diego

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].

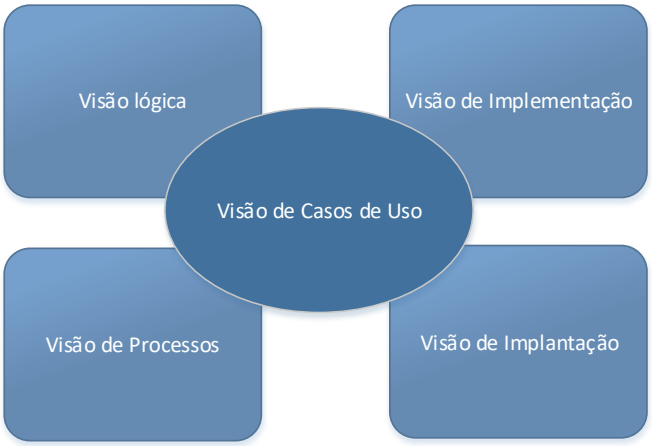


Figura 1 – Arquitetura 4+1

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].
MVC – Model View Controller.

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

[MVC] https://medium.com/@celionormando/arquitetura-mvc-e-princ%C3%ADpios-de-projeto-3d0b278ef910

[PHP] https://www.php.net/

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, tudo em PHP puro, utilizando o MVC.

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 PHP e MVC.
• 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

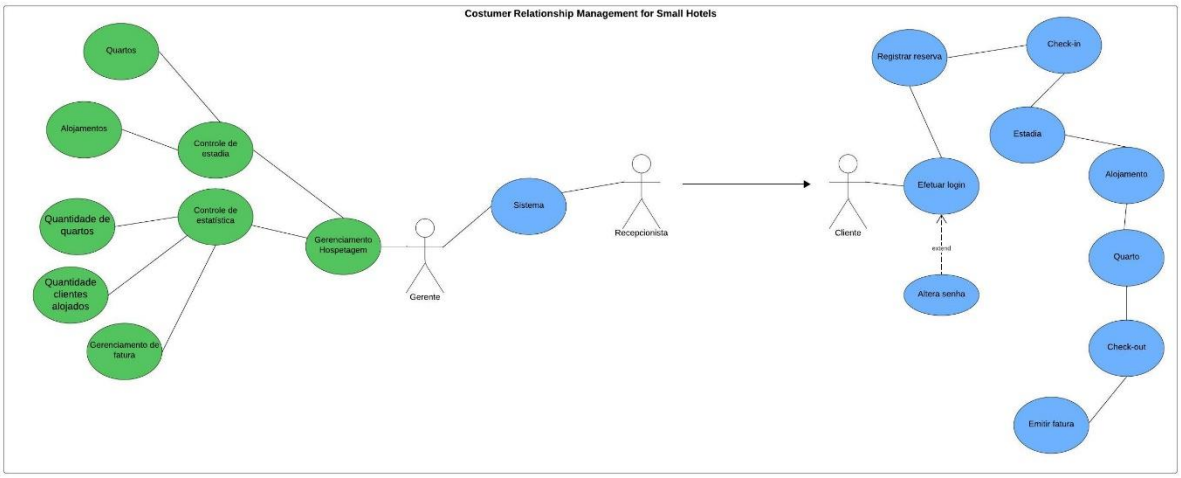
• O sistema deve ser capaz de integrar-se com sistemas externos, como sistemas de pagamento online, sistemas de gerenciamento de reservas de terceiros e sistemas de faturamento.
• Deve suportar protocolos de comunicação padrão, como HTTP/HTTPS e APIs RESTful, para facilitar a integração com outros sistemas.

3.2.3 Conformidade Regulatória

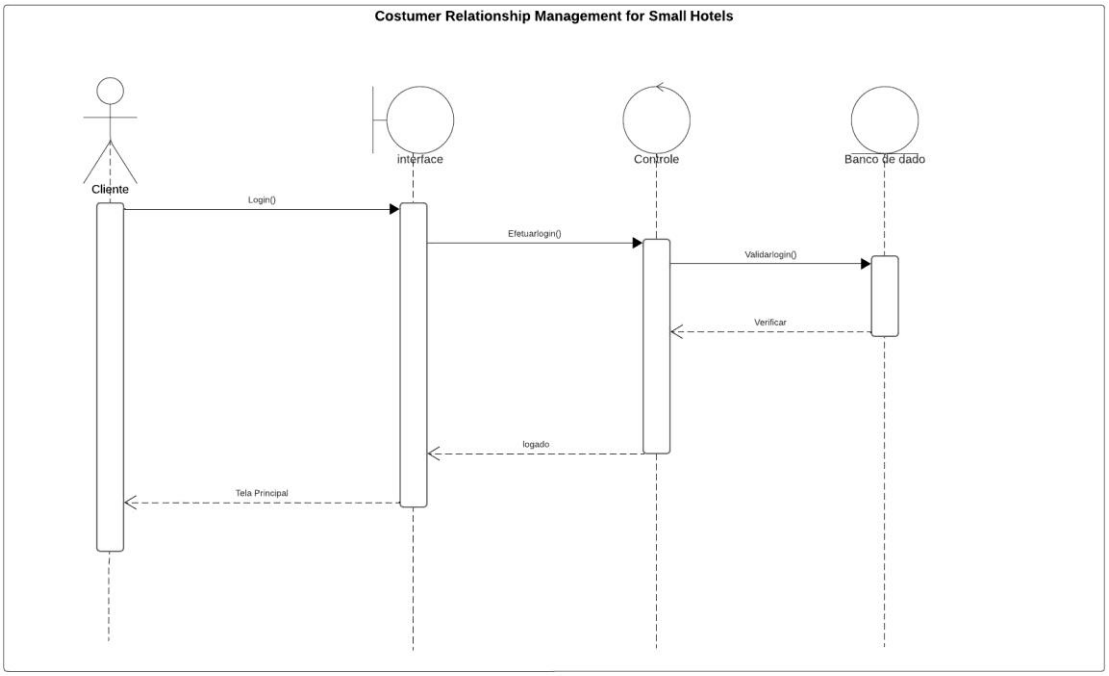
• O sistema deve estar em conformidade com as regulamentações de proteção de dados, como o Autoridade Nacional de Proteção de Dados (ANPD) órgão responsável para garantir a privacidade e segurança das informações dos usuários.
• Deve cumprir as regulamentações locais e internacionais relacionadas a transações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

4. VISÃO DE CASOS DE USO

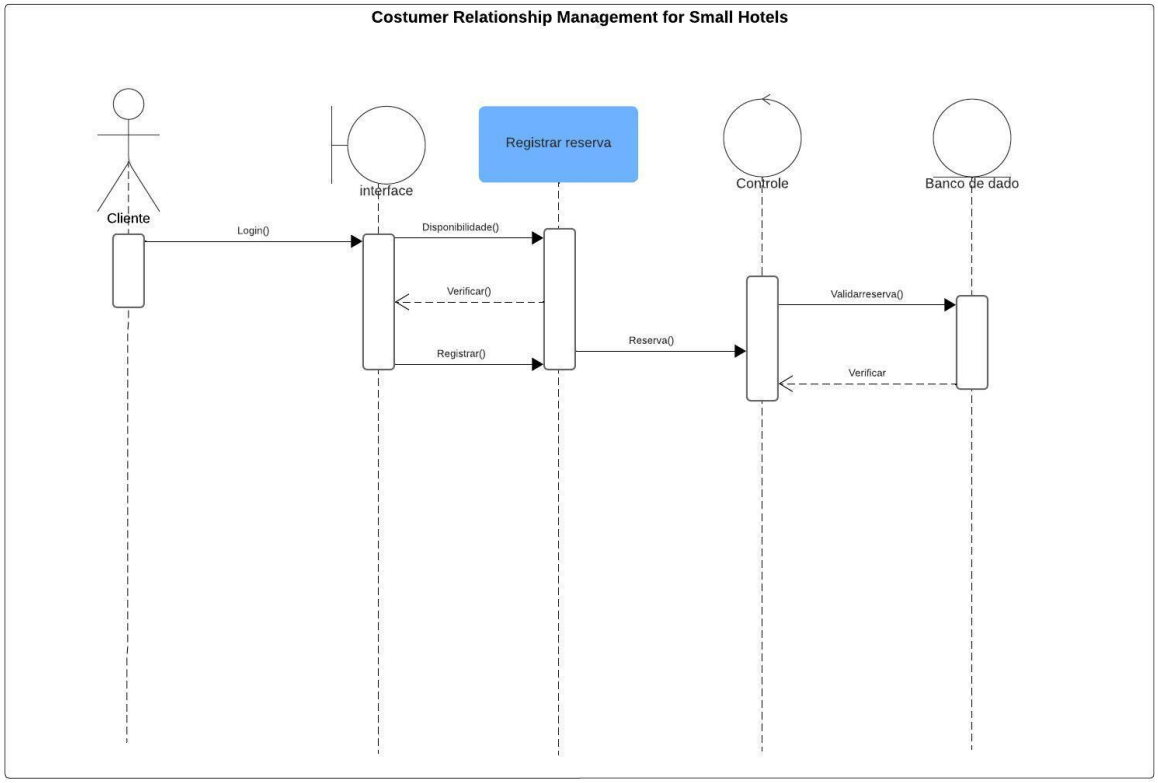
4.1 Casos de Uso significantes para a arquitetura



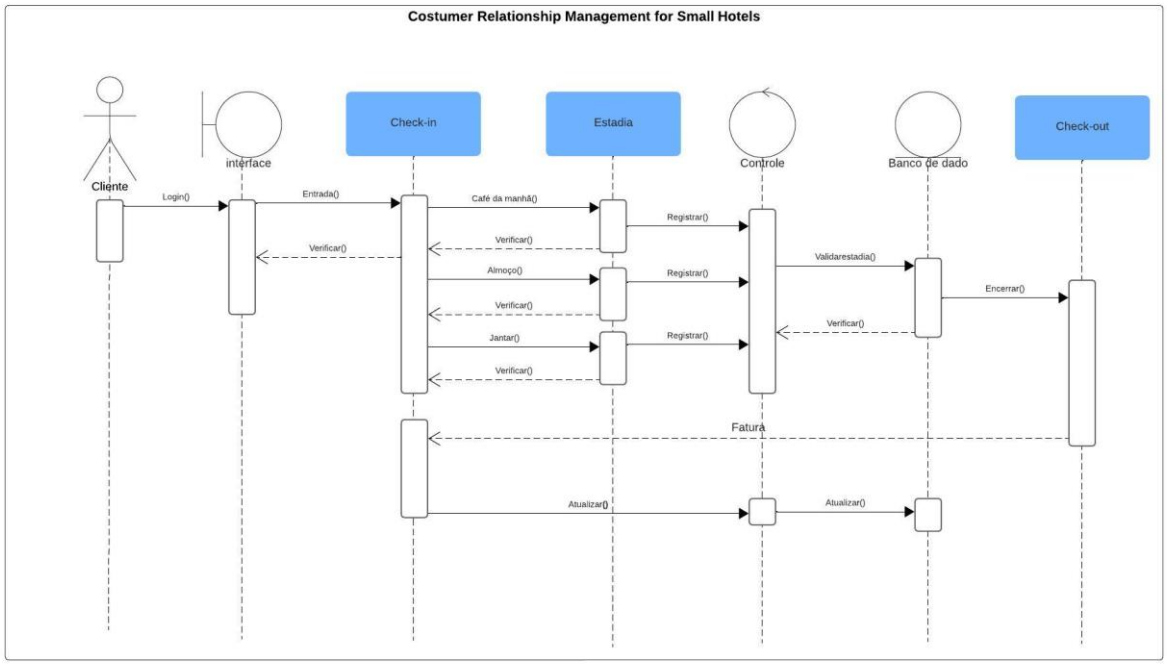
4.1.1 Login de Usuário



4.1.2 Registrar Reserva



4.1.2 Efetuar Check-in e Check-out



5. VISÃO LÓGICA

Camada de Apresentação

Camada de Lógica de Negócios

Responder às requisições HTTP

Serviços responsáveis pela implementação das regras de negócios do sistema

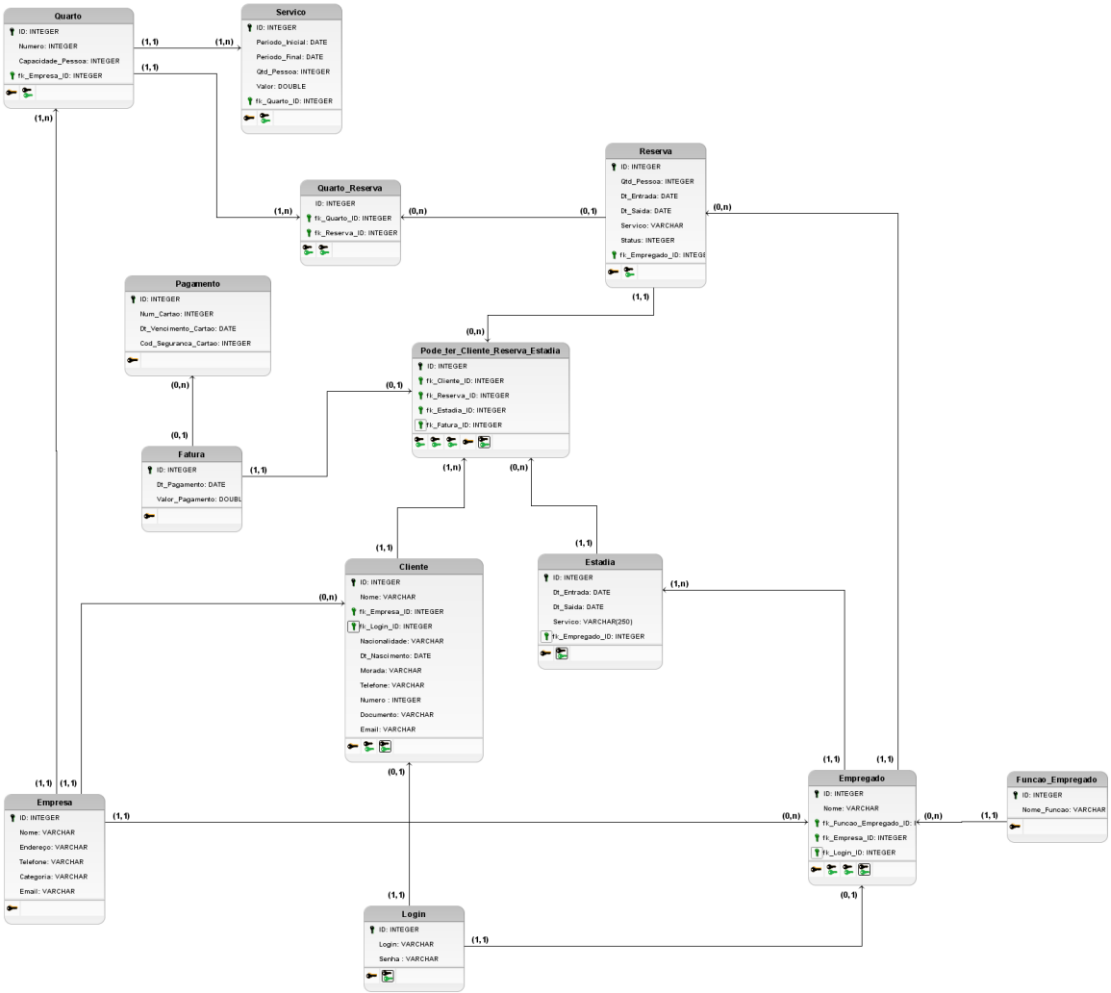
Age como interface para o serviço web

- Serviço do restaurante (gestão de cardápios, processamento de pedidos)
- Serviço de pedidos (validação de pedidos, cálculos de preços)
- Serviço de reserva (verificação de disponibilidade, confirmação de reserva)

5.1 Visão Geral – pacotes e camadas

6. VISÃO DE IMPLEMENTAÇÃO

6.1 Diagrama de Classes



Glossário
PK: Primary Key (Chave Primária);
FK: Foreign Key (Chave Estrangeira);
SQL: Structured Query Language;
ER: Entidade-Relacionamento;

Chaves e Índices

Chaves Primárias

- Empresa: id_empresa;
- Quarto: id_quarto;
- Empregado: id_empregado;
- Servico: id_servico;
- Funcao_Empregado: id_funcao_empregado;
- Cliente: id_cliente;
- Reserva: id_reserva;
- Estadia: id_estadia;
- Quarto_Reserva: id_quarto_reserva;
- Pode_ter_Cliente_Reserva_Estadia: id_cliente_reserva_estadia;
- Fatura: id_fatura;
- Login: id_login;
- Pagamento: id_pagamento;

Chaves Estrangeiras

- Quarto: fk_empresa_id referência a Empresa(id_empresa)
- Empregado:
  - fk_funcao_empregado_id referência a Funcao_Empregado(id_funcao_empregado)
  - fk_empresa_id referência a Empresa(id_empresa)
  - fk_login_id referência a Login(id_login)
- Servico: fk_quarto_id referência a Quarto(id_quarto)
- Cliente:
  - fk_empresa_id referência a Empresa(id_empresa)
  - fk_login_id referência a Login(id_login)
- Reserva: fk_empregado_id referência a Empregado(id_empregado)
- Estadia: fk_empregado_id referência a Empregado(id_empregado)
- Quarto_Reserva:
  - fk_quarto_id referência a Quarto(id_quarto)
  - fk_reserva_id referência a Reserva(id_reserva)
- Pode_ter_Cliente_Reserva_Estadia:
  - fk_cliente_id referência a Cliente(id_cliente)
  - fk_reserva_id referência a Reserva(id_reserva)
  - fk_estadia_id referência a Estadia(id_estadia)
  - fk_fatura_id referência a Fatura(id_fatura)

Relacionamentos 1:N

Empresa (1) - (N) **Quarto**
Empresa (1) - (N) **Empregado**
Empresa (1) - (N) **Cliente**
Quarto (1) - (N) **Servico**
Funcao_Empregado (1) - (N) **Empregado**
Login (1) - (N) **Empregado**
Login (1) - (N) **Cliente**
Empregado (1) - (N) **Reserva**
Empregado (1) - (N) **Estadia**

Relacionamentos N:M

Quarto (N) - (M) **Reserva** através de Quarto_Reserva
Cliente (N) - (M) **Reserva** através de Pode_ter_Cliente_Reserva_Estadia
Cliente (N) - (M) **Estadia** através de Pode_ter_Cliente_Reserva_Estadia

Documentação do Banco de Dados Hotel

Introdução ao Banco de Dados

Nome do Banco de Dados: Hotel

Descrição Geral: Este banco de dados é projetado para gerenciar informações sobre empresas, quartos, empregados, serviços, clientes, reservas, estadias e pagamentos em um sistema hoteleiro.

Tecnologia Utilizada: MySQL
Esquema do Banco de Dados
Diagrama ER (Um diagrama ER visual seria útil aqui, mas como estamos apenas com texto, vamos descrever a estrutura.)

Tabela: Empresa

Descrição: Armazena informações das empresas que possuem hotéis.
Colunas:
id_empresa (INT, PK): Identificador único da empresa.
nome (VARCHAR(40)): Nome da empresa.
endereco (VARCHAR(60)): Endereço da empresa.
telefone (VARCHAR(20)): Telefone de contato da empresa.
categoria (VARCHAR(20)): Categoria da empresa.
email (VARCHAR(60)): Email da empresa.

Relacionamentos:
1:N com Quarto
1:N com Empregado
1:N com Cliente

```
CREATE TABLE IF NOT EXISTS Empresa (
    id_empresa int PRIMARY KEY AUTO_INCREMENT,
    nome varchar(40),
    endereco varchar(60),
    telefone varchar(20),
    categoria varchar(20),
    email varchar(60)
);
```

Tabela: Quarto

Descrição: Armazena informações dos quartos disponíveis nos hotéis.
Colunas:
id_quarto (INT, PK): Identificador único do quarto.
numero (INT): Número do quarto.
capacidade_pessoa (INT): Capacidade máxima de pessoas no quarto**.**
fk_empresa_id (INT, FK): Identificador da empresa a que o quarto pertence.
Relacionamentos:
N:1 com Empresa
1:N com Servico
N:M com Reserva através de Quarto_Reserva

```
CREATE TABLE IF NOT EXISTS Quarto (
    id_quarto INTEGER PRIMARY KEY AUTO_INCREMENT,
    numero INTEGER,
    capacidade_pessoa INTEGER,
    fk_empresa_id INTEGER,
    FOREIGN KEY (fk_empresa_id) REFERENCES Empresa(id_empresa)
);
```

Tabela: Empregado

Descrição: Armazena informações dos empregados.
Colunas:
id_empregado (INT, PK): Identificador único do empregado.
nome (VARCHAR(40)): Nome do empregado.
fk_funcao_empregado_id (INT, FK): Identificador da função do empregado.
fk_empresa_id (INT, FK): Identificador da empresa a que o empregado pertence.
fk_login_id (INT, FK): Identificador do login do empregado.

Relacionamentos:
N:1 com Empresa
N:1 com Funcao_Empregado
N:1 com Login
1:N com Reserva
1:N com Estadia

```
CREATE TABLE IF NOT EXISTS Empregado (
    id_empregado INTEGER PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(40),
    fk_funcao_empregado_id INTEGER,
    FOREIGN KEY (fk_funcao_empregado_id) REFERENCES Funcao_Empregado(id_funcao_empregado),
    fk_empresa_id INTEGER,
    FOREIGN KEY (fk_empresa_id) REFERENCES Empresa(id_empresa),
    fk_login_id INTEGER,
    FOREIGN KEY (fk_login_id) REFERENCES Login(id_login)
);
```

Tabela: Servico
Descrição: Armazena informações sobre os serviços oferecidos nos quartos.
Colunas:
id_servico (INT, PK): Identificador único do serviço.
periodo_inicial (DATE): Data de início do serviço.
periodo_final (DATE): Data de término do serviço.
qtd_pessoa (INT): Quantidade de pessoas que podem usar o serviço.
valor (DOUBLE): Valor do serviço.
fk_quarto_id (INT, FK): Identificador do quarto onde o serviço é oferecido.
Relacionamentos:
N:1 com Quarto

```
CREATE TABLE IF NOT EXISTS Servico (
    id_servico INTEGER PRIMARY KEY AUTO_INCREMENT,
    periodo_inicial DATE,
    periodo_final DATE,
    qtd_pessoa INTEGER,
    valor DOUBLE,
    fk_quarto_id INTEGER,
    FOREIGN KEY (fk_quarto_id) REFERENCES Quarto(id_quarto)
);
```

Tabela: Funcao_Empregado

Descrição: Armazena as diferentes funções dos empregados.
Colunas:
id_funcao_empregado (INT, PK): Identificador único da função do empregado.
nome_funcao (VARCHAR(35)): Nome da função.

Relacionamentos:
1:N com Empregado

```
CREATE TABLE IF NOT EXISTS Funcao_Empregado (
    id_funcao_empregado INTEGER PRIMARY KEY AUTO_INCREMENT,
    nome_funcao VARCHAR(35)
);
```

Tabela: Cliente

Descrição: Armazena informações dos clientes.
Colunas:
id_cliente (INT, PK): Identificador único do cliente.
nome (VARCHAR(40)): Nome do cliente.
fk_empresa_id (INT, FK): Identificador da empresa relacionada ao cliente.
fk_login_id (INT, FK): Identificador do login do cliente.
nacionalidade (VARCHAR(18)): Nacionalidade do cliente.
dt_nascimento (DATE): Data de nascimento do cliente.
morada (VARCHAR(30)): Endereço do cliente.
telefone (VARCHAR(20)): Telefone do cliente.
numero (INT): Número da residência do cliente.
documento (VARCHAR(20)): Documento de identificação do cliente.
email (VARCHAR(60)): Email do cliente.

Relacionamentos:
N:1 com Empresa
N:1 com Login
N:M com Reserva através de Pode_ter_Cliente_Reserva_Estadia
N:M com Estadia através de Pode_ter_Cliente_Reserva_Estadia

```
CREATE TABLE IF NOT EXISTS Cliente (
    id_cliente INTEGER PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(40),
    fk_empresa_id INTEGER,
    fk_login_id INTEGER,
    nacionalidade VARCHAR(18),
    dt_nascimento DATE,
    morada VARCHAR(30),
    telefone VARCHAR(20),
    numero  INTEGER,
    documento VARCHAR(20),
    email varchar(60)
);
```

Tabela: Reserva

Descrição: Armazena informações das reservas feitas pelos clientes.
Colunas:
id_reserva (INT, PK): Identificador único da reserva.
qtd_pessoa (INT): Quantidade de pessoas na reserva.
dt_entrada (DATE): Data de entrada na reserva.
dt_saida (DATE): Data de saída da reserva.
servico (VARCHAR(250)): Descrição dos serviços incluídos na reserva.
status_reserva (INT): Status da reserva.
fk_empregado_id (INT, FK): Identificador do empregado responsável pela reserva

Relacionamentos:
N:1 com Empregado
N:M com Quarto através de Quarto_Reserva
N:M com Cliente através de Pode_ter_Cliente_Reserva_Estadia

```
CREATE TABLE IF NOT EXISTS Reserva (
    id_reserva INTEGER PRIMARY KEY AUTO_INCREMENT,
    qtd_pessoa INTEGER,
    dt_entrada DATE,
    dt_saida DATE,
    servico VARCHAR(250),
    status_reserva INTEGER,
    fk_empregado_id INTEGER,
    FOREIGN KEY (fk_empregado_id) REFERENCES Empregado(id_empregado)
);
```

Tabela: Estadia

Descrição: Armazena informações sobre as estadias dos clientes.
Colunas:
id_estadia (INT, PK): Identificador único da estadia.
dt_entrada (DATE): Data de entrada na estadia.
dt_saida (DATE): Data de saída da estadia.
servico (VARCHAR(250)): Descrição dos serviços incluídos na estadia.
fk_empregado_id (INT, FK): Identificador do empregado responsável pela estadia.
Sql

Relacionamentos:
N:1 com Empregado
N:M com Cliente através de Pode_ter_Cliente_Reserva_Estadia

```
CREATE TABLE IF NOT EXISTS Estadia (
    id_estadia INTEGER PRIMARY KEY AUTO_INCREMENT,
    dt_entrada DATE,
    dt_saida DATE,
    servico VARCHAR(250),
    fk_empregado_id INTEGER,
    FOREIGN KEY (fk_empregado_id) REFERENCES Empregado(id_empregado)
);
```

Tabela: Quarto_Reserva

Descrição: Tabela de relacionamento entre quartos e reservas.
Colunas:
id_quarto_reserva (INT, PK): Identificador único do relacionamento.
fk_quarto_id (INT, FK): Identificador do quarto.
fk_reserva_id (INT, FK): Identificador da reserva.

Relacionamentos:
N:1 com Quarto
N:1 com Reserva

```
CREATE TABLE IF NOT EXISTS Quarto_Reserva (
    id_quarto_reserva INTEGER PRIMARY KEY AUTO_INCREMENT,
    fk_quarto_id INTEGER,
    FOREIGN KEY (fk_quarto_id) REFERENCES Quarto(id_quarto),
    fk_reserva_id INTEGER,
    FOREIGN KEY (fk_reserva_id) REFERENCES Reserva(id_reserva)
);
```

Tabela: Pode_ter_Cliente_Reserva_Estadia

Descrição: Tabela de relacionamento entre clientes, reservas, estadias e faturas.
Colunas:
id_cliente_reserva_estadia (INT, PK): Identificador único do relacionamento.
fk_cliente_id (INT, FK): Identificador do cliente.
fk_reserva_id (INT, FK): Identificador da reserva.
fk_estadia_id (INT, FK): Identificador da estadia.
fk_fatura_id (INT, FK): Identificador da fatura.

Relacionamentos:
N:1 com Cliente
N:1 com Reserva
N:1 com Estadia
N:1 com Fatura

```
CREATE TABLE Pode_ter_Cliente_Reserva_Estadia (
    id_cliente_reserva_estadia INTEGER PRIMARY KEY AUTO_INCREMENT,
    fk_cliente_id INTEGER,
        FOREIGN KEY (fk_cliente_id) REFERENCES Cliente(id_cliente),
    fk_reserva_id INTEGER,
        FOREIGN KEY (fk_reserva_id) REFERENCES Reserva(id_reserva),
    fk_estadia_id INTEGER,
        FOREIGN KEY (fk_estadia_id) REFERENCES Estadia(id_estadia),
    fk_fatura_id INTEGER,
        FOREIGN KEY (fk_fatura_id) REFERENCES Fatura(id_fatura)
);
```

Tabela: Fatura

Descrição: Armazena informações sobre as faturas de pagamento.
Colunas:
id_fatura (INT, PK): Identificador único da fatura.
dt_pagamento (DATE): Data de pagamento da fatura.
valor_pagamento (DOUBLE): Valor do pagamento.

Relacionamentos:
1:N com Pode_ter_Cliente_Reserva_Estadia

```
CREATE TABLE IF NOT EXISTS Fatura (
    id_fatura INTEGER PRIMARY KEY AUTO_INCREMENT,
    dt_pagamento DATE,
    valor_pagamento DOUBLE
);
```

Tabela: Login

Descrição: Armazena informações de login para usuários.
Colunas:
id_login (INT, PK): Identificador único do login.
Login (VARCHAR(30)): Nome de usuário.
Senha (VARCHAR(12)): Senha de acesso

Relacionamentos:
1:N com Empregado
1:N com Cliente

.

```
CREATE TABLE IF NOT EXISTS Login (
    id_login INTEGER PRIMARY KEY AUTO_INCREMENT,
    Login VARCHAR(30),
    Senha  VARCHAR(12)
);
```

Tabela: Pagamento

Descrição: Armazena informações sobre pagamentos com cartão de crédito.
Colunas:
id_pagamento (INT, PK): Identificador único do pagamento.
num_cartao (INT): Número do cartão de crédito.
dt_vencimento_cartao (DATE): Data de vencimento do cartão de crédito.
cod_seguranca_cartao (INT): Código de segurança do cartão de crédito.

Relacionamentos:
Nenhum relacionamento direto com outras tabelas

```
CREATE TABLE IF NOT EXISTS Pagamento (
    id_pagamento INTEGER PRIMARY KEY AUTO_INCREMENT,
    num_cartao INTEGER,
    dt_vencimento_cartao DATE,
    cod_seguranca_cartao INTEGER
);
```

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.