

Documentação do Projeto de Banco de Dados

Aplicação para o restaurante japonês



Professora/orientadora:

- Gabrielle Karine Canalle

Responsáveis

- Adriana Rodrigues
- Pedro Villas Boas
- Vinicius Ventura

01. APRESENTAÇÃO DO PROJETO

Esta documentação descreve o projeto de desenvolvimento do site de vendas para o **Restaurante Japonês Portal Temaki**, da qual foram utilizadas diversas tecnologias e práticas com o uso destas.

O projeto envolveu o uso de SQL da linguagem de SQL e o MySQL como o sistema de gerenciamento de banco de dados relacional, que possui três lojas (Recife, Olinda e Paulista) e realiza vendas tanto nas lojas físicas quanto online.

Para os **dados**, foi definido SQL como a linguagem de programação e o MySQL como o sistema de gerenciamento de banco de dados relacional. Para o **backend**, foi utilizado Java como linguagem de programação. E para o **frontend**, foi utilizado HTML, CSS, Bootstrap e Javascript. E spring boot para conexão entre estes.

Uma etapa fundamental do projeto foi a realização da entrevista com a cliente. Essa entrevista teve como objetivo entender detalhadamente as necessidades e expectativas da cliente, permitindo a construção de um minimundo que represente os requisitos do sistemas. A partir desta entrevista, foram retiradas as principais funcionalidades e dados necessários para o desenvolvimento do banco de dados e do site.

01.1. Requisitos Identificados

- **Funcionalidades Principais**
 - Cadastro de funcionários e dependentes
 - Cadastro de clientes
 - Autenticação e login de usuários
 - Cadastro do cardápio do menu
 - Gerenciamento (cadastro e, alteração)
 - Sistema de pedidos e controle de atendimento do pedido

- **Outros Requisitos**
 - Interface amigável
 - Segurança dos dados

01.2. Repositório

A aplicação encontra-se disponível no repositório abaixo:

Github: <https://github.com/Dricalucia/BD--Projeto>

02. MINIMUNDO

O **Portal Temaki** é uma rede um restaurante com três unidades (uma matriz e duas filiais) e gostaria de ter um sistema de gerenciamento da operação de vendas on-line. Cada unidade possui as seguintes informações: CNPJ, IE, endereço, telefone, ponto de referência.

Cada loja tem seu time de funcionários, que podem ser: atendentes, ajudantes de cozinha, auxiliares de serviços gerais, supervisor e chefs. Cada funcionário é identificado por um número único de matrícula, nome, função, data contratação, dependentes, salário e horário de trabalho.

Os clientes, na maioria, são pessoas físicas, mas também atende pedidos de pessoas jurídicas. As informações registradas são: nome, endereço, telefone, email, contato, data cadastro, ponto referencia, observação. Os clientes que possuem cadastro, atualmente, são os que realizam pedidos por aplicativo.

Cada pedido realizado pelo cliente é gerado um número único, com data e hora do pedido, o tipo de pedido,, o número do cliente (para clientes cadastrados) e o status do pedido.

Através do site, os clientes têm acesso ao cardápio de menu, onde ele poderá visualizar os itens que deseja inserir no carrinho de compra. O pedido só pode ser confirmado se o cliente possuir cadastro no aplicativo e estiver logado.

Os itens do menu representam os itens disponíveis no cardápio do restaurante, como sushis, sashimis, tempurás, yakisobas, entre outros. Cada item de menu possui um código único, nome, descrição, preço e categoria.

Diariamente um produto do cardápio é colocado em promoção no site. O prazo de validade desta promoção é de 24 horas.

Deseja uma aplicação que tenha uma interface amigável para que os clientes possam utilizar sem dificuldade. É fundamental a segurança dos dados que estão sendo gerados e armazenados no banco de dados.

03. MODELAGEM

A modelagem do banco de dados foi dividida em três etapas principais: **conceitual, lógica e física.**

Utilizamos diagramas ER(Entidade-Relacionamento) para visualizar as entidades, atributos e relacionamentos entre elas

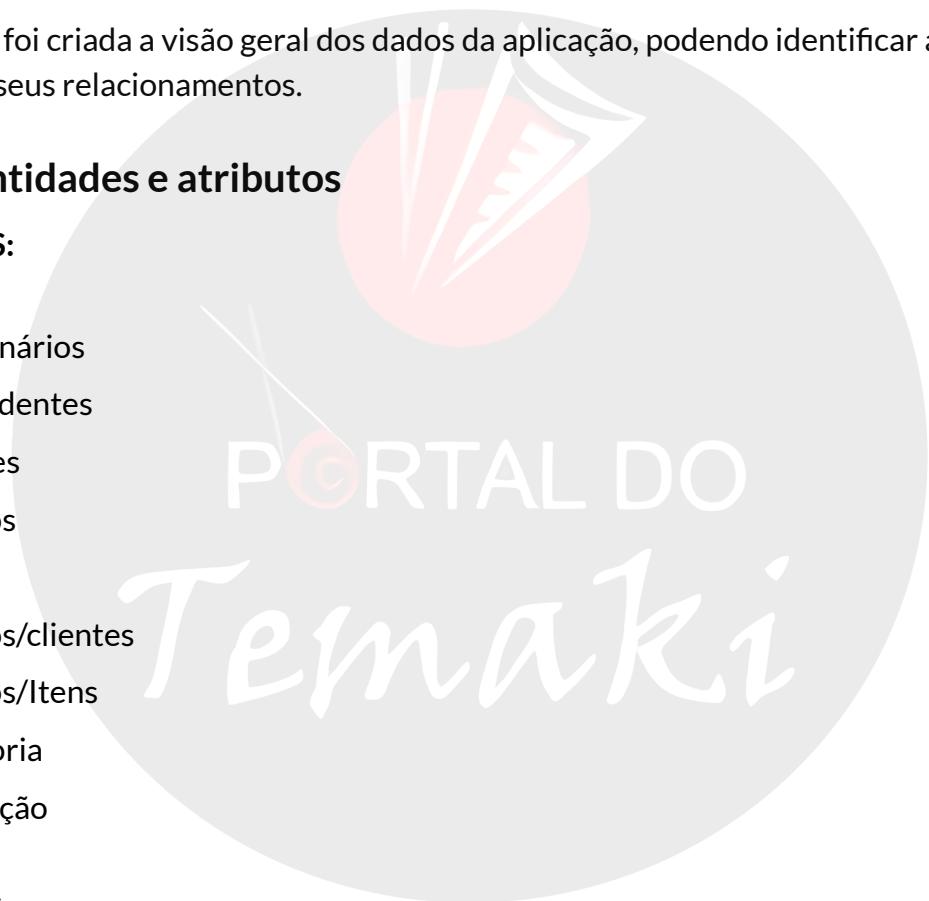
03.1. Modelagem Conceitual

Nessa etapa foi criada a visão geral dos dados da aplicação, podendo identificar as principais entidades e seus relacionamentos.

03.1.1. Entidades e atributos

ENTIDADES:

- Lojas
- Funcionários
- Dependentes
- Clientes
- Pedidos
- Itens
- Pedidos/clientes
- Pedidos/Itens
- Categoria
- Promoção

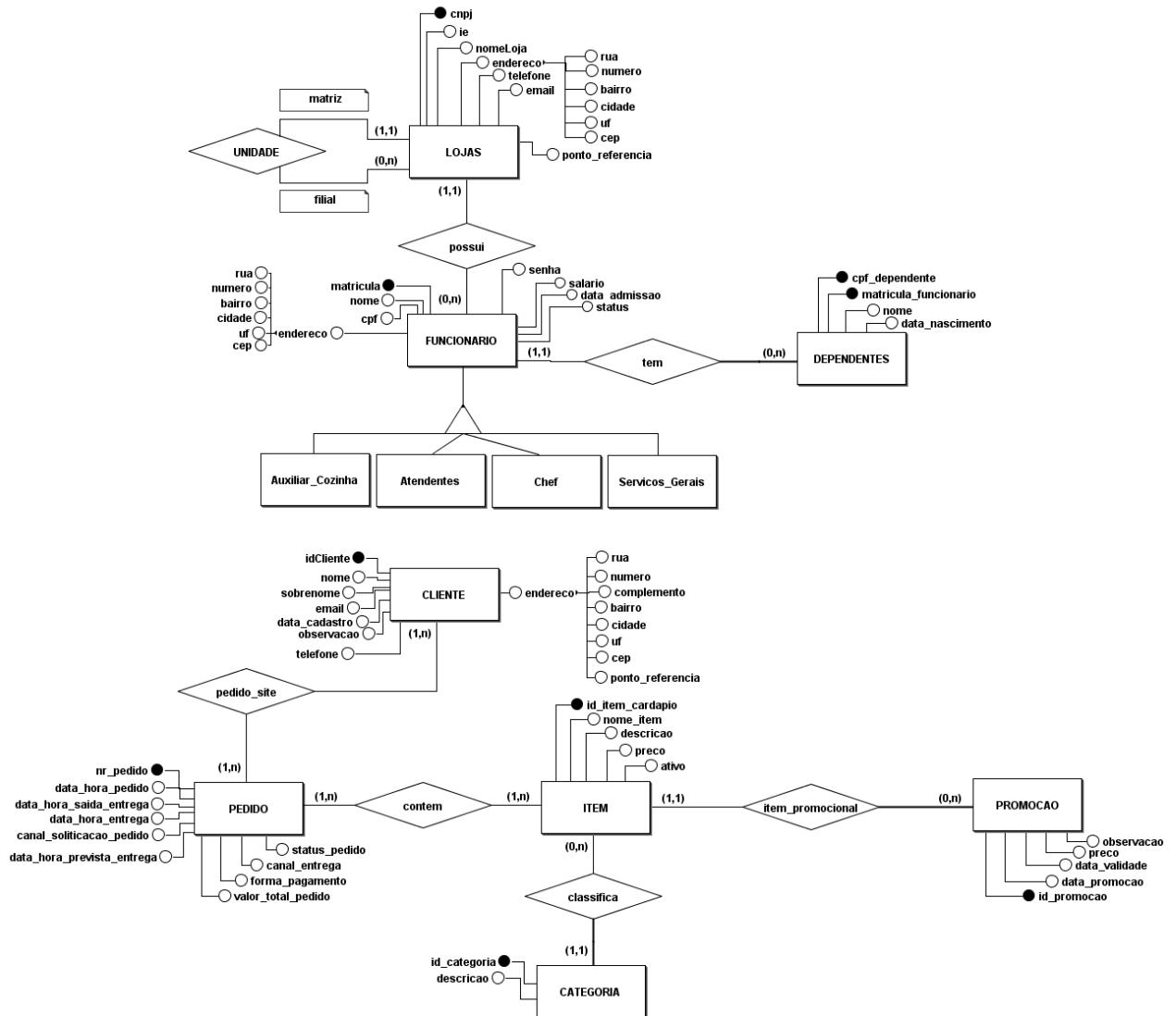


ATRIBUTOS:

- **Filial:** CNPJ, IE, endereço, telefone, contato, email
- **Funcionários:** Matrícula(chave única serial), nome, CPF, RG, dt_nascimento, endereço, telefone, email, salário e horário de trabalho
- **Dependentes:** IdDependentes (chave única serial), nome, dt_nascimento, CPF
- **Clientes:** IDCliente(chave única serial), nome, endereço, telefone, email, ponto referencia, observação
- **Menu (itens):** IdMenu(chave única serial), nome, descrição, preço, categoria (multivvalorado), Ingredientes (multivvalorado)

- Pedidos: IDPedido(chave única serial), IDCliente, dtPedido, hrPedido, TipoPedido (multivvalorad), IdItensMenu, quantidade, preço, status
- Estoque: idProduto, nome, estoqueMinimo, estoqueAtual
- Fornecedores: CNPJ, IE, nome, endereço, telefone, email, contato

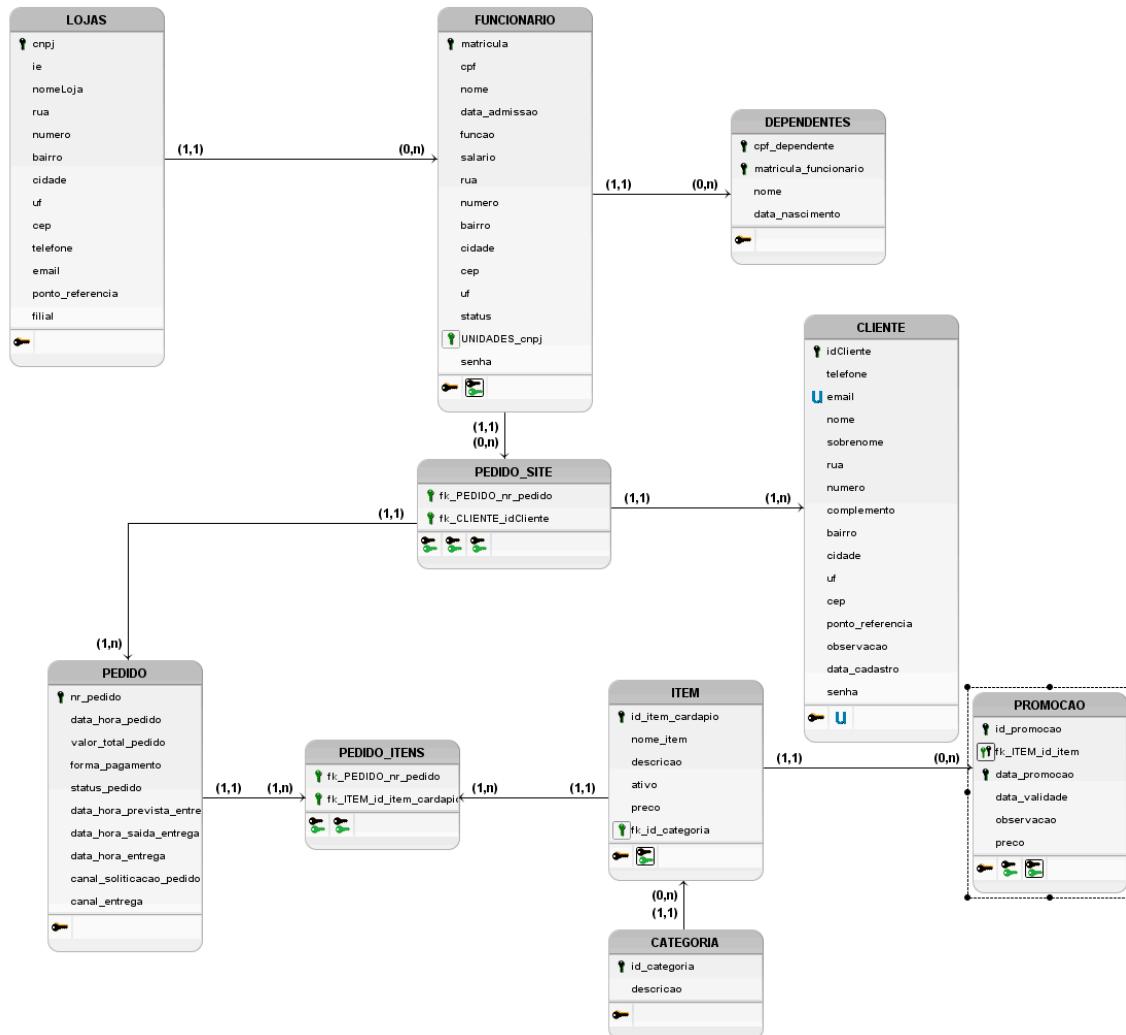
03.1.2. Diagrama Entidade-Relacionamento (DER)



03.2. Modelagem Lógica

Com base no modelo conceitual, foi elaborado o modelo lógico, que incluiu a definição das tabelas, colunas, tipos de dados e relacionamento entre as tabelas. Essa fase visou garantir que o banco de dados atendesse aos requisitos de armazenamento e recuperação de dados de maneira eficiente.

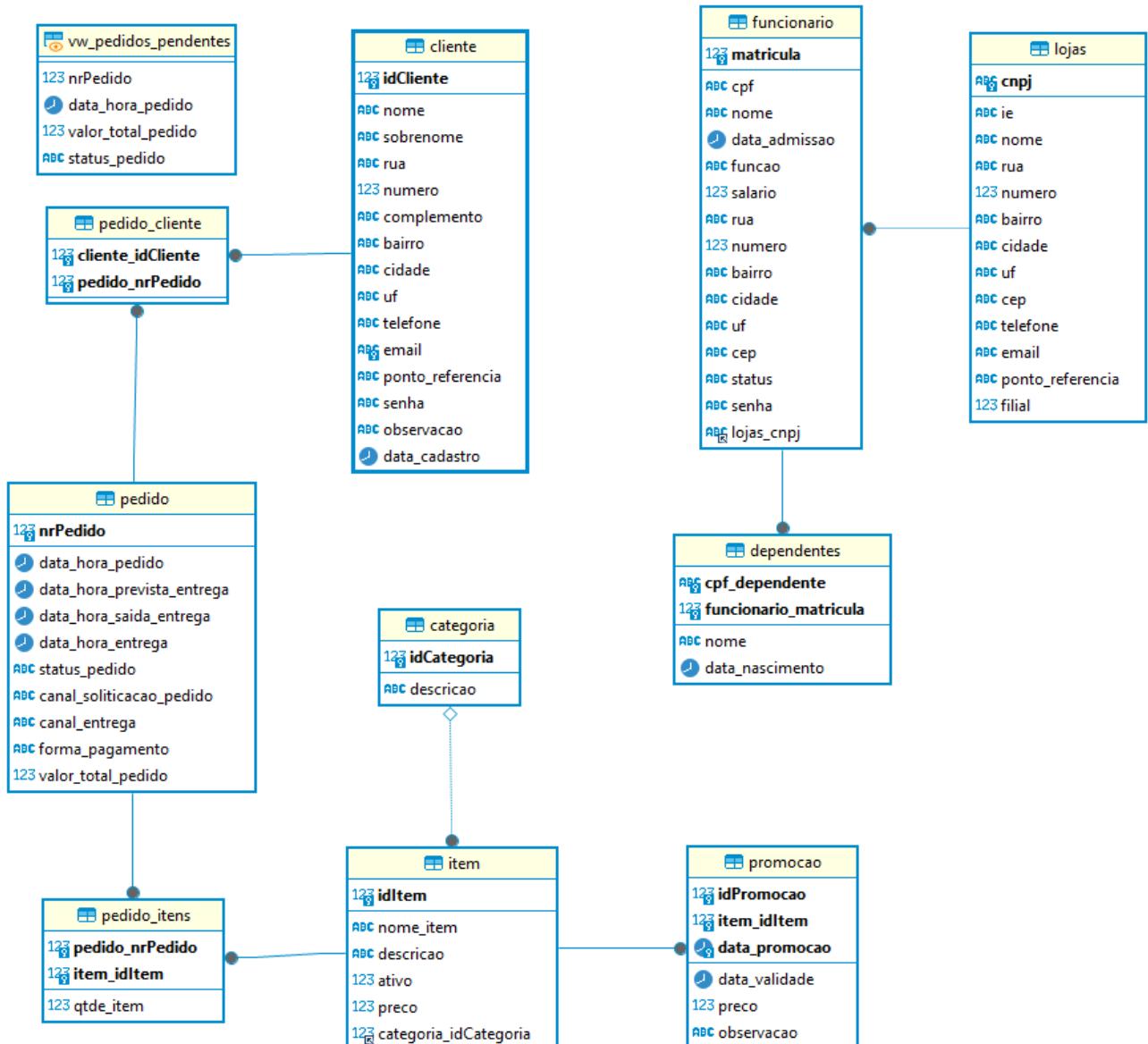
03.2.1. Diagrama Entidade-Relacionamento (DER)



03.3. Modelagem Física

A modelagem física traduziu o modelo lógico na implementação real no SQL. Inclui a criação de scripts SQL para criação das tabelas e outros objetos do banco de dados

03.3.1. Diagrama Entidade-Relacionamento (DER)



03.3.2. Script SQL

```

/* Projeto BD Restaurante */
create database restaurante;
use restaurante;
create table lojas (
  cnpj varchar(15) primary key,
  ie varchar(15) not null,
  nome varchar(10) not null,
  rua varchar(40) not null,
  numero integer,
  bairro varchar(15) not null,
  cidade varchar(15) not null,
  uf varchar(2) not null,
  cep varchar(8) not null,
  telefone varchar(11),
  email varchar(50),
  ponto_referencia text,
  filial boolean
);

create table funcionario (
  matricula integer primary key auto_increment,
  cpf varchar(11) not null,
  nome varchar(50) not null,
  data_admissao datetime not null default current_timestamp,
  funcao varchar(20) not null,
  salario decimal(10,2),
  rua varchar(50) not null,
  numero integer,
  bairro varchar(20) not null,
  cidade varchar(20) not null,
  uf varchar(2) not null,
  cep varchar(8) not null,
  status char(1) not null,
  lojas_cnpj varchar(15) not null,
  constraint fk_lojas_cnpj foreign key (lojas_cnpj) references lojas(cnpj)
);

create table dependentes (
  cpf_dependente varchar(11),
  funcionario_matricula integer,
  nome varchar(50) not null,
  data_nascimento date not null,
  constraint fk_funcionario_matricula foreign key (funcionario_matricula) references funcionario(matricula),
  primary key (cpf_dependente, funcionario_matricula)
);

create table cliente (
  idCliente integer primary key auto_increment,
  nome varchar(10) not null,
  sobrenome varchar(10) not null,

```

```

rua varchar(40) not null,
numero integer,
complemento varchar(30),
bairro varchar(15) not null,
cidade varchar(15) not null,
uf varchar(2) not null,
telefone varchar(11) unique not null,
email varchar(50),
ponto_referencia text not null,
observacao text,
data_cadastro datetime not null default current_timestamp
);

create table item (                                # temaki salmão, urumaki, jôjô
  idItem integer primary key auto_increment,
  nome_item varchar(100),
  descricao varchar(255),
  ativo boolean,
  preco decimal(10,2),
  categoria_idCategoria integer,
  constraint fk_categoria_idCategoria foreign key (categoria_idCategoria) references categoria(idCategoria)
);
create table categoria (                         # Bebidas, Proteinas, Temperos, Carboidrato
  idCategoria integer primary key auto_increment,
  descricao varchar(20)
);
create table promocao (
  idPromocao integer auto_increment,
  item_idItem integer,
  data_promocao datetime not null default current_timestamp,
  data_validade datetime not null,
  preco decimal(10,2),
  observacao varchar(255),
  constraint fk_item_idItem_promocao foreign key (item_idItem) references item(idItem),
  primary key (idPromocao, item_idItem, data_promocao)
);

create table pedido (
  nrPedido integer primary key auto_increment,
  data_hora_pedido datetime not null default current_timestamp,
  data_hora_prevista_entrega datetime not null,
  data_hora_saida_entrega datetime,
  data_hora_entrega datetime,
  status_pedido varchar(20),
  canal_solicitacao_pedido varchar(20) default 'site', #Padrão será site
  canal_entrega varchar(20) default 'domicilio',           #em domicilio, retirada na loja
  forma_pagamento varchar(20) default 'cartao credito',    #cartão credito, cartão débito, vale alimentação, dinheiro
  valor_total_pedido decimal(10,2)
);

create table pedido_cliente (
  cliente_idCliente integer,

```

```

pedido_nrPedido integer,
constraint fk_cliente_idCliente_pedidos foreign key (cliente_idCliente) references cliente(idCliente),
constraint fk_pedido_nrPedido_pedidos foreign key (pedido_nrPedido) references pedido(nrPedido),
primary key (cliente_idCliente, pedido_nrPedido)
);
create table pedido_itens (
  pedido_nrPedido integer,
  item_idItem integer,
  qtde_item integer,
  constraint fk_pedido_nrPedido_itens foreign key (pedido_nrPedido) references pedido(nrPedido),
  constraint fk_item_idItem_itens foreign key (item_idItem) references item(idItem),
  primary key(pedido_nrPedido, item_idItem)
);

```

/ / /

```

/* Scripts SQL Avançado - RELATORIOS */
/* View para listar todos os pedidos que estão com status 'em aberto' */
/* para ser usado na tela de funcionários */
create view vw_pedidos_pendentes as
select
  nrPedido,
  data_hora_pedido,
  valor_total_pedido,
  status_pedido
from pedido
where status_pedido <> 'Concluido' -- definir os níveis de status: Em aberto, Em preparo, Saída para entrega, Concluído
order by nrPedido;

select * from vw_pedidos_pendentes; -- chamada da view

```

```

/* Visualizar os detalhes do pedido */
select
  p.nrPedido,
  p.data_hora_pedido,
  p.valor_total_pedido,
  c.nome as nome_cliente,
  c.sobrenome as sobrenome_cliente,
  i.nome_item as item,
  pit.qtde_item as qtde,
  i.preco as Preco,
  p.data_hora_prevista_entrega,
  p.data_hora_saida_entrega,
  p.data_hora_entrega
from pedido p
join pedido_cliente pc on p.nrPedido = pc.pedido_nrPedido
join cliente c on pc.cliente_idCliente = c.idCliente
join pedido_itens pit on p.nrPedido = pit.pedido_nrPedido
join item i on pit.item_idItem = i.idItem
where p.nrPedido = ?; -- substituir '?' pelo parâmetro do nrpedido, que será passado na lista de pedido

```

```

/* Função para exibir a lista de pedidos por determinado período
* data inicial e final devem ser passados como parâmetros */
delimiter $$
```

```

create function fn_pedidos_por_periodo(@data_inicial date, @data_final date)
returns table
as
begin
  return (
    select
      p.nrPedido as Nr_pedido,
      p.data_hora_pedido as Data_Hora_Pedido,
      p.valor_total_pedido as Total_Pedido,
      c.nome as nome_cliente,
      c.sobrenome as sobrenome_cliente
    from pedido p
    join pedido_cliente pc on p.nrPedido = pc.pedido_nrPedido
      join cliente c on pc.cliente_idCliente = c.idCliente
    where p.data_hora_pedido between @data_inicial and @data_final
  );
end $$
delimiter ;

select *
from fn_pedidos_por_periodo(?, ?); -- substituir '?' por variaveis que tenha a data inicial e final da seleção de pedidos

/* Função para exibir a lista de pedidos por cliente e determinado periodo
 * data inicial e final devem ser passados como parametros */
delimiter $$
create function fn_pedidos_cliente_por_periodo(@data_inicial date, @data_final date, @cliente_id int)
returns table
as
begin
  return (
    select
      p.nrPedido as nr_Pedido,
      p.data_hora_pedido as data_hora_pedido,
      p.valor_total_pedido as total_pedido,
      c.nome as nome_cliente,
      c.sobrenome as sobrenome_cliente
    from pedido p
    join pedido_cliente pc on p.nrPedido = pc.pedido_nrPedido
      join cliente c on pc.cliente_idCliente = c.idCliente
    where p.data_hora_pedido between @data_inicial and @data_final
      and c.idCliente = @cliente_id
  );
end $$
delimiter ;

-- chamada da função
select *
from fn_pedidos_cliente_por_periodo(?, ?, ?); -- substituir '?' por variaveis que tenha a data inicial e final e o codigo do cliente

/* listar o pedido de maior valor realizado em determinado periodo */

```

```

select p.data_hora_pedido, pc pedido_nrPedido, c.nome as nome_cliente, c.sobrenome as sobrenome_cliente,
p.valor_total_pedido
from pedido_cliente pc
join cliente c on pc.cliente_idCliente = c.idCliente
join pedido p on pc.pedido_nrPedido = p.nrPedido
where p.valor_total_pedido =
  select max(valor_total_pedido)
  from pedido
  where data_hora_pedido between ? and ?      -- substituir '?' por variaveis que tenha a data inicial e final
)
limit 1;

/* Listar por cliente a quantidade de pedidos realizados em determinado periodo */
select c.nome as nome_cliente, c.sobrenome as sobrenome_cliente, count(pc.pedido_nrPedido) as quantidade_pedidos
from pedido_cliente pc
join cliente c on pc.cliente_idCliente = c.idCliente
join pedido p on pc.pedido_nrPedido = p.nrPedido
where p.data_hora_pedido between ? and ?      -- substituir '?' por variaveis que tenha a data inicial e final
group by c.nome;

/* Listar o total de vendas por item determinado periodo */
select pit.item_idItem, i.nome_item, sum(pit.qtde_item) as total_vendas
from pedido_itens pit
join item i on pit.item_idItem = i.idItem
join pedido p on pit.pedido_nrPedido = p.nrPedido
where p.data_hora_pedido between ? and ?      -- substituir '?' por variaveis que tenha a data inicial e final
group by pit.item_idItem, i.nome_item;

-- Populando tabela lojas
INSERT INTO lojas (cnpj, ie, nome, rua, numero, bairro, cidade, uf, cep, telefone, email, ponto_referencia, filial)
VALUES
('', '', '1', '', '', '', '', '', '', TRUE);

-- Populando tabela funcionario
INSERT INTO funcionario (matricula, cpf, nome, data_admissao, funcao, salario, rua, numero, bairro, cidade, uf, cep,
status, senha, lojas_cnpj)
VALUES
(1, '', NOW(), '1.00', 1, '', '', '', '', '', '');

-- Populando tabela dependentes
INSERT INTO dependentes (cpf_dependente, funcionario_matricula, nome, data_nascimento)
VALUES
('1', '1990-01-01');

-- Populando tabela cliente
INSERT INTO cliente (idCliente, nome, sobrenome, rua, numero, complemento, bairro, cidade, uf, telefone, email,
ponto_referencia, senha, observacao, dataCadastro)
VALUES
(1, '', '1', '', '', '', '', '', '', '', NOW());

-- Populando tabela pedido
INSERT INTO pedido (nrPedido, data_hora_pedido, data_hora_prevista_entrega, data_hora_saida_entrega,
data_hora_entrega, status_pedido, canal_solicicacao_pedido, canal_entrega, forma_pagamento, valor_total_pedido)

```

VALUES

```
(1, NOW(), DATE_ADD(NOW(), INTERVAL 1 HOUR), NOW(), NOW(), '', '', '', '0.00);
```

-- Populando tabela categoria

```
INSERT INTO categoria (idCategoria, descricao)
```

VALUES

```
(1, ''');
```

-- Populando tabela item

```
INSERT INTO item (idItem, nome_item, descricao, ativo, preco, categoria_idCategoria)
```

VALUES

```
(1, "", TRUE, 0.00, 1);
```

-- Populando tabela promocao

```
INSERT INTO promocao (idPromocao, item_idItem, data_promocao, data_validade, preco, observacao)
```

VALUES

```
(1, 1, NOW(), NOW(), 0.00, ''');
```

-- Populando tabela pedido_itens

```
INSERT INTO pedido_itens (pedido_nrPedido, item_idItem)
```

VALUES

```
(1, 1);
```

-- Populando tabela pedido_cliente

```
INSERT INTO pedido_cliente (cliente_idCliente, pedido_nrPedido, funcionario_matricula)
```

VALUES

```
(1, 1, 1);
```

03.4. Normalização

Para garantir a integridade e eficiência do banco de dados, as tabelas foram normalizadas até a terceira forma norma (3FN). A normalização envolveu a eliminação de redundância e a organização das tabelas em formas normais, garantindo a consistência do banco de dados.

03.5. Dicionário de Dados

- Tabela: Loja

Field	Type	Null	Key	Default	Extra
cnpj	varchar(15)	NO	PRI		
ie	varchar(15)	NO			
nome	varchar(10)	NO			
rua	varchar(40)	NO			
numero	int	YES			
bairro	varchar(15)	NO			
cidade	varchar(15)	NO			
uf	varchar(2)	NO			
cep	varchar(8)	NO			
telefone	varchar(11)	YES			
email	varchar(50)	YES			
ponto_referencia	text	YES			
filial	tinyint(1)	YES			

- Tabela: Funcionario

Field	Type	Null	Key	Default	Extra
matricula	int	NO	PRI		auto_increment
cpf	varchar(11)	NO			
nome	varchar(50)	NO			
data_admissao	datetime	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED
funcao	varchar(20)	NO			
salario	decimal(10,2)	YES			
rua	varchar(50)	NO			
numero	int	YES			
bairro	varchar(20)	NO			
cidade	varchar(20)	NO			
uf	varchar(2)	NO			

cep	varchar(8)	NO			
status	char(1)	NO			
senha	varchar(20)	NO			
lojas_cnpj	varchar(15)	NO	MUL		

- Tabela: **Dependente**

Field	Type	Null	Key	Default	Extra
cpf_dependente	varchar(11)	NO	PRI		
funcionario_matricula	int	NO	PRI		
nome	varchar(50)	NO			
data_nascimento	date	NO			

- Tabela: **Cliente**

Field	Type	Null	Key	Default	Extra
idCliente	int	NO	PRI		auto_increment
nome	varchar(10)	NO			
sobrenome	varchar(10)	NO			
rua	varchar(40)	NO			
numero	int	YES			
complemento	varchar(30)	YES			
bairro	varchar(15)	NO			
cidade	varchar(15)	NO			
uf	varchar(2)	NO			
telefone	varchar(11)	NO			
email	varchar(50)	NO	UNI		
ponto_referencia	text	NO			
senha	varchar(20)	NO			
observacao	text	YES			
dataCadastro	datetime	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED

- Tabela: Pedido

Field	Type	Null	Key	Default	Extra
nrPedido	int	NO	PRI		auto_increment
data_hora_pedido	datetime	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED
data_hora_prevista_entrega	datetime	NO			
data_hora_saida_entrega	datetime	YES			
data_hora_entrega	datetime	YES			
status_pedido	varchar(20)	YES			
canal_solicitacao_pedido	varchar(20)	YES		site	
canal_entrega	varchar(20)	YES		domicílio	
forma_pagamento	varchar(20)	YES		cartao credito	
valor_total_pedido	decimal(10,2)	YES			

- Tabela: Pedido_cliente

Field	Type	Null	Key	Default	Extra
cliente_idCliente	int	NO	PRI		
pedido_nrPedido	int	NO	PRI		

- Tabela: Pedido_item

Field	Type	Null	Key	Default	Extra
pedido_nrPedido	int	NO	PRI		
item_idItem	int	NO	PRI		
qtde_item	int	YES			

- Tabela: Categoria

Field	Type	Null	Key	Default	Extra
idCategoria	int	NO	PRI		auto_increment
descricao	varchar(20)	YES			

- Tabela: **Item**

Field	Type	Null	Key	Default	Extra
idItem	int	NO	PRI		auto_increment
nome_item	varchar(100)	YES			
descricao	varchar(255)	YES			
ativo	tinyint(1)	YES			
preco	decimal(10,2)	YES			
categoria_idCategoria	int	YES	MUL		

- Tabela: **Promoção**

Field	Type	Null	Key	Default	Extra
idPromocao	int	NO	PRI		auto_increment
item_idItem	int	NO	PRI		
data_promocao	datetime	NO	PRI	CURRENT_TIMESTAMP	DEFAULT_GENERATED
data_validade	datetime	NO			
preco	decimal(10,2)	YES			
observacao	varchar(255)	YES			

04. Desenvolvimento do Backend

O backend foi desenvolvido utilizando a linguagem Java e o framework Spring Boot, que facilitou a criação de APIs RESTful e a integração com o banco de dados MySQL.

Principais Componentes:

- Controladores (Controllers): Gerenciam as requisições HTTP.
- Serviços (Services): Contêm a lógica de negócios.
- Repositórios (Repositories): Interagem com o banco de dados.

05. Desenvolvimento do Frontend

O frontend foi desenvolvido utilizando HTML, CSS, Bootstrap e React, garantindo uma interface responsiva e amigável.

Estrutura:

- HTML: Estrutura básica das páginas.
- CSS: Estilos personalizados.
- Bootstrap: Layout responsivo e componentes prontos.
- React: Biblioteca JavaScript para a construção de interfaces de usuário interativas.

06. Integração com Spring Boot

Utilizamos o Spring Boot para integrar o frontend, backend e o banco de dados. O Spring Data JPA facilitou a persistência de dados no MySQL.

07. Conclusão

O projeto resultou numa aplicação funcional, com uma interface amigável e um sistema de gerenciamento de dados.

A entrevista com o cliente foi essencial para entendermos as necessidades e desenvolver um sistema que atendeu as expectativas.

A modelagem conceitual, lógica e física garantiu um banco de dados bem estruturado e normalizado. A utilização do Spring Boot facilitou a integração e desenvolvimento da aplicação.

Este documento serve como um registro completo das atividades e decisões tomadas durante o projeto, garantindo a transparência e facilitando futuras manutenções e evoluções do sistema.

Este projeto foi um exemplo da aplicação das boas práticas em desenvolvimento de software e gerenciamento de banco de dados.



PORTAL DO
Temaki