DOCUMENTAÇÃO PARA DATABASE DO PROJETO 1 – 2018.2
+
FRONT-END, REST API E INSTRUÇÕES DE INSTALAÇÃO

Repositório do Projeto: https://github.com/SabrinaSimao/Projeto1-MegaDados
Membros: Sabrina Simão, Hugo Mendes

Resumo

- **SQL Tables**
- **Stored proceadures, functions, views e triggers**
- **Diagrama Entidade Relacionamento**
- **Arquitetura Implementada**
- **Dicionario de Dados**
- **Instruções de Instalação do Projeto** → final do documento

SQL Tables

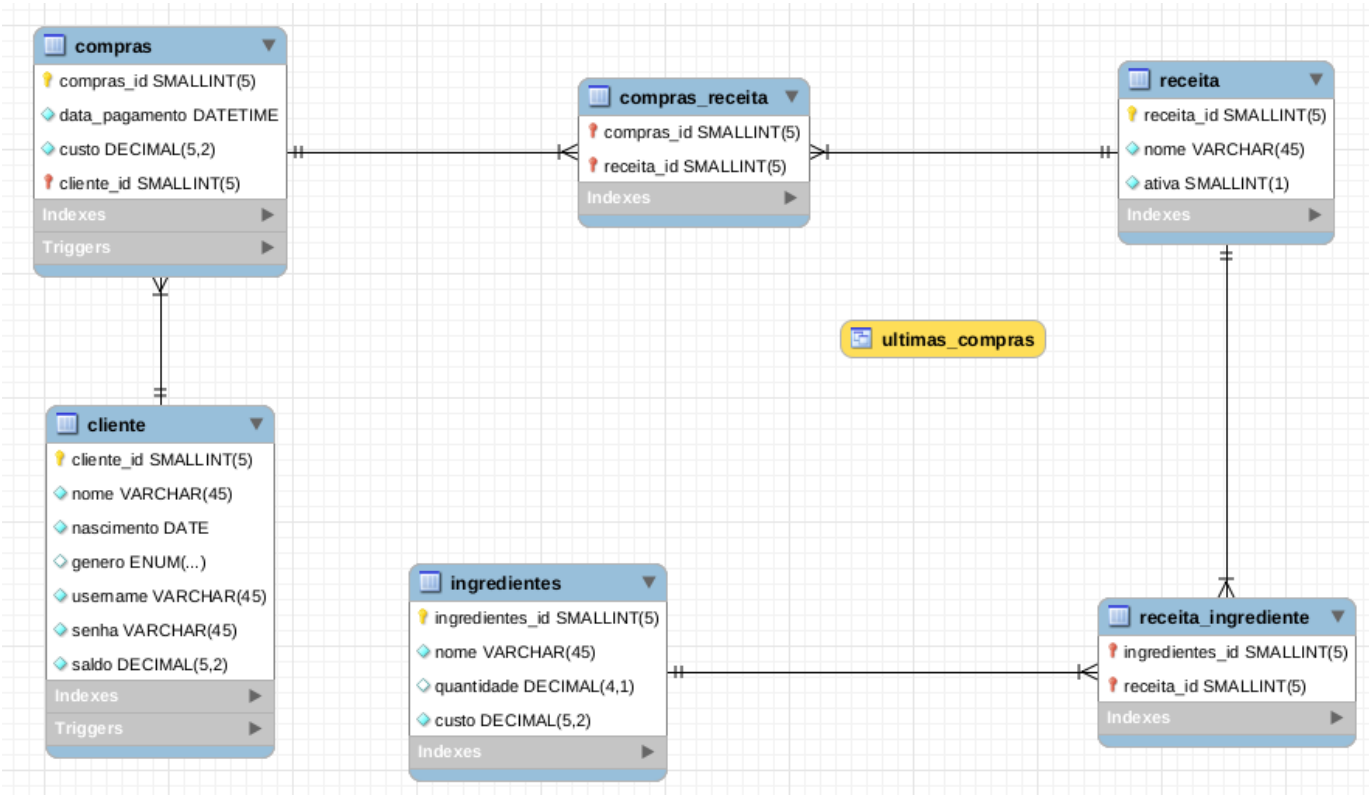
Nome	Tipo	Argumentos	Subtipo
ingredientes	tabela	ingredientes_id PK nome quantidade custo	Small int unsigned not null auto increment varchar(45) not null decimal (4,1) default 0 decimal (5,2) not null default 14.99
receita	tabela	receita_id PK nome ativa	Smallint unsigned not null auto increment varchar(45) not null small int (1) not null default 1
receita_ingrediente	tabela	ingredientes_id PK receita_id PK	Small int unsigned not null Small int unsigned not null
compras	tabela	compras_id PK data_pagamento custo cliente_id PK	Smallint unsigned not null auto increment Datetime not null decimal (5,2) not null default 0.0 Small int unsigned not null
compras_receita	tabela	compras_id PK receita_id PK	Small int unsigned not null Small int unsigned not null
cliente	tabela	cliente_id PK nome nascimento genero username senha saldo	Smallint unsigned not null auto increment Varchar(45) not null Date not null Enum('F','M','OTHER') default 'OTHER' Varchar(45) unique not null Varchar(45) not null Decimal (5,2) not null default 0.0

Stored Procedures e Funções

Nome	Tipo	Argumentos	Resumo
adiciona_usuario	Procedure	nome, nascimento, genero, username, senha	Adiciona usuário na tabela cliente
altera_senha	Procedure	Senha cliente_id	Altera a senha do usuario e confere na database
compras_valor	Function	id	Função para checar o valor das compras do cliente
check_senha	Procedure	username senha	Checa se a senha colocada no site é a mesma da database e retorna informações do cliente

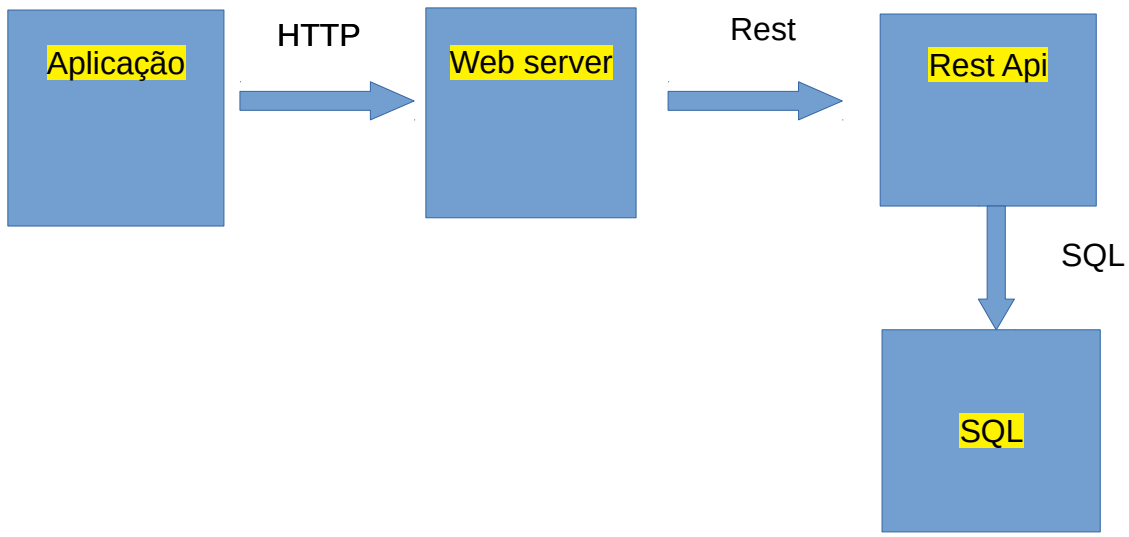
Views e Triggers

Nome	Tipo	Resumo
ultimas_compras	View	View para ver as ultimas 5 compras da loja
trig_compras	Trigger	Atualiza saldo do cliente após compra ser feita no site
trig_saldo_insuficiente	Trigger	Checa se cliente tem saldo antes de realizar compras no site



Arquitetura Implementada

Diagrama:



Descrição:

Na parte do **front-end** estamos usando um framework javascript chamado React.js, moderno e recente vem sendo uma ferramenta muito utilzada no mercado web. Permite rapido desenvolvimento e tem uma estrutura baseada em componentes reutilizaveis, gerando códigos modulares e escalaveis se implementados conforme a filosofia react.

A **rest-api** foi feita em node.js (também javascript), novamente utilizando-se uma framework também muito utilizada pelo mercado conhecida como express, esta que fornece ferramentas suficientes para criação de uma api robusta.

A conexão com o **SQL** é feita a partir de uma biblioteca existente para node, mas o management do que acontece com o banco é puramente 'escrita sql'.

Estrutura da rest:

A rest está sendo montada conforme um desing bem interessante e com alto potencial de escalabilidade que pode ser encontrado nesse artigo:

https://medium.com/studioarmix/learn-restful-api-design-ideals-c5ec915a430f

Em resumo (com o que foi implementado até então), temos um endpoint por tabela, e para cada endpoint diferentes verbos http são aceitos. Os verbos que vão ser utilizados serão somente o get, post e patch. A ausência do put e do delete se deve ao fato de que adotamos as praticas do que foi descrito no artigo, e o put é para updates em objetos inteiros, e o delete para remoções. No nosso caso os updates são bem específicos e mexem com um ou dois atributos, nesse caso recomenda-se o uso do patch, mesmo para o delete o que fazemos é mudar uma flag para true or false.

Exemplificando, quando queremos fazer alterações nas tabelas sql a rest possui um endpoint com o nome daquela tabela e uma implementação para cada tipo de verbo http que você quiser usar. A rest esta no mesmo repositório da aplicação para facilitar as

coisas, entretanto roda de maneira independente da aplicação, ou seja, se quiser startar a rest sem nem mexer com o front é possível.

As repostas da rest também seguem um padrão, 401 para erro de autenticação, 400 para erro interno da rest, e 200 se ocorreu tudo okay.

Especificação da REST

- Receitas (endpoint: /recipes):
  - Criar → POST
  - Ler → GET
  - Atualizar → PATCH
- Ingredientes (endpoint: /ingredients):
  - Ler → GET
- Clientes (endpoint: /clientes):
  - Ler → GET
  - Criar → POST
- Compras (endpoint:/compras):
  - Criar → POST
  - Ler → GET
- Receitas_Ingredientes (endpoint: /receitas_ingredientes):
  - Ler → GET
  - Criar → POST
- Compras_Receitas (endpoint: /compras_receita):
  - Ler → GET
  - Criar → POST

Dicionário de dados

Tabela ingredientes:

ingrediente_id → chave primária que identifica um ingrediente
nome → nome do ingrediente
disponível → boolean que diz se esse ingrediente está disponivel no estoque ou não.
custo → valor desse ingrediente (reais)

Receitas:

receita_id → chave primária que identifica uma receita
nome → nome da receita
ativa → se foi excluída ou não
cliente_id → a qual cliente essa receita pertence

Compras:

compra_id → chave primaria que identifica uma compra
data_pagamento → quando a compra foi paga , data fo calendario gregoriano
custo → o valor total dessa compra (reais)
cliente_id → chave estrangeira que diz qual cliente fez essa compra

Cliente:

cliente_id → chave primaria que identifica um cliente
nome → nome do cliente (pode ser primeiro nome ou completo)
nascimento → data de nascimento do cliente, calendario gregoriano
genêro → genêro com o qual o cliente se classifica como

Receita Ingrediente:

ingrediente_id→ chave estrangeira, parte da primária. Identifica qual ingrediente pertence à essa relação.

receita_id→ chave estrangeira, parte da primária. Identifica qual receita pertence à essa relação.

Compras Receita:

compra_id→ chave estrangeira, parte da primária. Identifica qual compra pertence à essa relação.

receita_id→ chave estrangeira, parte da primária. Identifica qual receita pertence à essa relação.

INSTRUÇÕES

- **Rodar script schema.sql**
- **Rodar script inserts.sql** → todos na pasta SQL
- **Rodar script stored_proceadures.sql**
- **Tenha node instalado no computador** → https://nodejs.org/en/download/
- **No folder restapi/api/models ponha sua autenticação sql no objeto ‘con’.**
- **Pelo terminal entre na pasta restApi**
- **Insira o comando npm install**
- **Insira o comando npm start**
- **Pelo terminal entre na pasta application/my-app**
- **Insira o comando npm install**
- **Insira o comando npm start**
- **Abra o endereço localhost:3000 no seu browser.**