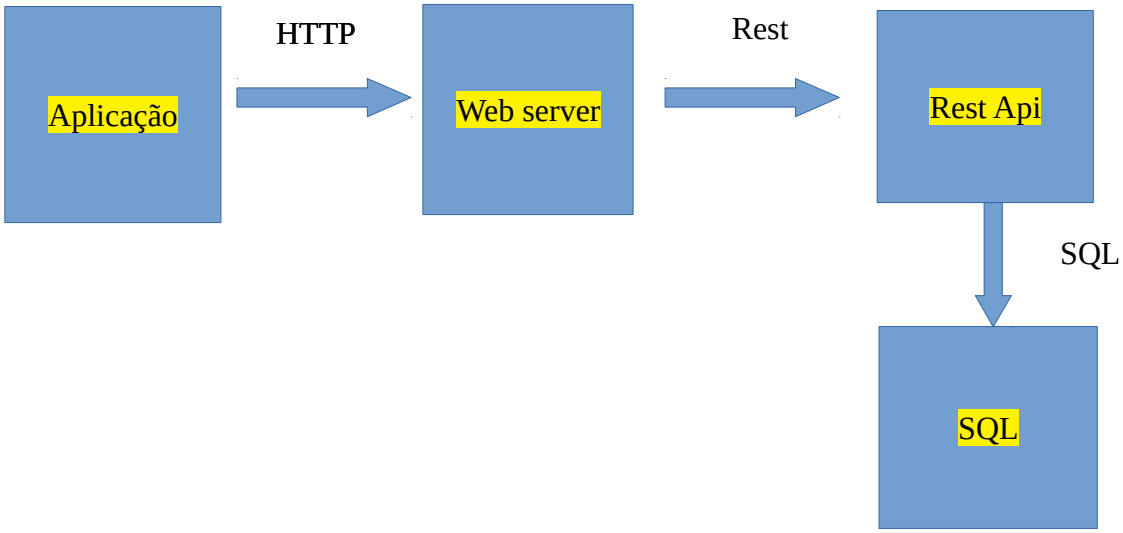
Insper

Big Data: Projeto 1
Primeira Entrega

Hugo Mendes, Sabrina Simão
Professor Fabio Ayres
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.

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

Receitas:

receita_id → chave primária que identifica uma receita
nome → nome da receita

Compras:

compra_id → chave primaria que identifica uma compra
data_pagamento → quando a compra foi paga , data no calendário romano
custo → o valor total dessa compra
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
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.

Observações Finais

Para rodar o front ou a rest basta entrar em suas respectivas pastas (front-big-data, RestApi), e rodar os comandos npm install seguido do comando npm run start. É necessario ter node instalado na maquina e um DB chamado ‘project_1’ em seu mysql local e alterar seu mysql user e password. Ainda não há praticamente nada nesses dois diretórios só uma quick start para testar as conexões (ver console log).