Base de dados com uma API Rest para fornecimento de dados sobre receitas caseiras

Alunos: Franklin Carvalho (franklincbc@gmail.com)
Rafael Douglas (rafael.dougllas@gmail.com)

Descrição do projeto

- **Domínio da Aplicação (Problema real, sem ser software)**

O fato de existir pouquíssimas bases de dados abertas sobre receitas caseiras brasileiras na web foi um dos motivos que mostrou a necessidade de se criar uma base consolidada, grátis e disponível na web.

Diversos aplicativos relacionados a área de cozinha, utilizam informações de bases estrangeiras ou que são nacionais mas pagas e isso dificulta a criação de aplicações relacionadas a esta área.

O crescimento da procura por aprendizado na área gastronômica, seja para fazer comidas diferenciadas ou simplesmente para aprender a cozinhar o básico, vêm mostrando que esta é uma área que é pouca explorada por softwares (Aplicativos) e que pode ser de grande progresso para os que desejam desenvolver soluções para a área.

Outro fator importante que devemos ressaltar, é o fato de que geralmente as pessoas estão dispostas a compartilhar receitas que fizeram e dão certo, logo a base estaria sendo preenchida/atualizada constantemente por usuários que têm permissão.

- **Solução do sistema (Utilização do software)**

Criação de uma base de dados em MongoDB que possua acesso por uma API Rest que dê opções de acesso, inserção e alteração dos dados da base.

Para ter uma padronização no acesso às informações utilizaremos o conceito de Rest, devido ele ignorar os detalhes da implementação de componente e a sintaxe de protocolo com o objetivo de focar nos papéis dos componentes, nas restrições sobre sua interação com outros componentes e na sua interpretação de elementos de dados significantes. Ele usa HTTP para se comunicar através do que já é definido no protocolo sem precisar "inventar" novos protocolos específicos para a aplicação.

Preferimos utilizar um banco não relacional (MongoDB) pela facilidade das consultas, escalabilidade possível em caso de crescimento exacerbado da base e consultas a base, pela habilidade de shard dos dados entre máquinas e a possibilidade de armazenar arquivos em GridFS tirando partido de replicação e sharding.

Detalhamento da arquitetura adotada

Justificativa

- **Modelo de distribuição dos elementos do sistema**

Utilizaremos uma arquitetura baseada em camadas pois possui a vantagem de tornar as partes do sistema independentes, pode-se fazer alterações em uma parte sem afetar as demais, também é útil para que várias pessoas possam trabalhar na mesma aplicação cada um cuidando da parte em que foi alocada e fica mais fácil de se aplicar testes.

Desejamos utilizar para gerenciar as requisições, o rabbitMQ, para criar e controlar as filas de requisição, ele servirá como broker, cuja a principal função é pegar a mensagem do remetente e garantir que ela chegue até o receptor que pode tratá-la.

O rabbitMQ implementa o Advanced Message Queuing Protocol (AMQP). O AMQP dá suporte a vários tipos de comunicação, como point-to-point e publish-subscribe.

- **<u>Dos protocolos de comunicação adotados</u>**

Utilizaremos http pois existem casos em que o acesso do usuário só pode ser feito pela web, pois alguns usuário possuem restrições na rede para acessar conteúdos externos como medida de segurança que é imposto em algumas empresas por meio de firewall. Seguiremos como padrão de manipulação apenas duas classes, uma que conterá informações dos usuários e outra das receitas.

Modelos:

1. Usuario:
    - id
    - nome
    - dataCriacao
    - administrador
2. Receita
    - id
    - titulo
    - ingredientes
    - modo de preparo
    - usuarioId
    - dataCriacao
    - custo

Os pontos de acesso(ENDPOINTS) serão listados abaixo e retornarão os conteúdos em forma de JSON.

GET	/usuarios	- Lista todos os usuários
POST	/usuarios	- Cria um novo usuário
GET	/usuarios/:id	- Retorna um usuário específico
PUT	/usuarios/:id	- Atualiza um usuário específico
PATCH	/usuarios/:id	- Atualiza um usuário específico por campo
DELETE	/usuarios/:id	- Remove um usuário específico
GET	/usuarios/:id/receitas	- Lista todas as receitas de um usuário
POST	/usuarios/:id/receitas	- Adiciona uma receita para um usuário
GET	/receitas	- Lista todas as receitas
POST	/receitas	- Cria uma nova receita
GET	/receitas/:id	- Retorna uma receita específica
PUT	/receitas/:id	- Atualiza uma receita específica
PATCH	/receitas/:id	- Atualiza campos de uma receita específica
DELETE	/receitas/:id	- Remove uma receita de um usuário específico

Para apresentação e utilização dos endpoints da nossa api, utilizaremos uma aplicação desenvolvida para android, cuja finalidade é consumir os dados da nossa api e mostrar os dados de uma forma mais familiar para o usuário final, com esta aplicação, o usuário poderá inserir/atualizar/deletar novas informações da nossa api por meio de uma interface amigável.