UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET
CÂMPUS GUARAPUAVA

DONIZETY BALTOKOSKI

SISTEMA WEB COLABORATIVO DE RECEITAS COM BUSCA POR INGREDIENTES

TRABALHO DE CONCLUSÃO DE CURSO

GUARAPUAVA
2019

DONIZETY BALTOKOSKI

SISTEMA WEB COLABORATIVO DE RECEITAS COM BUSCA POR INGREDIENTES

Monografia de Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2 do Curso Superior de Tecnologia em Sistemas para Internet - TSI - da Universidade Federal Tecnológica do Paraná - UTFPR - Câmpus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo em Sistemas para a Internet.

Orientador: Prof. Me. Paulo André Filipak
Universidade Tecnológica Federal do Paraná

GUARAPUAVA
2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Guarapuava
Curso Superior de Tecnologia em Sistemas para Internet

ATA DE DEFESA DE MONOGRAFIA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE TECNOLOGOIA EM SISTEMAS PARA INTERNET

No dia **5 de julho de 2019**, às 13:30 horas, em sessão pública nas dependências da Universidade Tecnológica Federal do Paraná Câmpus Guarapuava, ocorreu a banca de defesa da de Trabalho de Conclusão de Curso intitulada: **"Sistema Web Colaborativo de Receitas com Busca por Ingredientes"** do acadêmico **Donizety Baltokoski** sob orientação do professor **Prof. Me. Paulo André Filipak** do curso de Tecnologia em Sistemas para Internet.

Banca Avaliadora	
Membro	Nome
Orientador	Prof. Me. Paulo André Filipak
Avaliador 1	Prof. Dr. Roni Fabio Banaszewski
Avaliador 2	Prof. Me. Guilherme da Costa Silva

Situação do Trabalho	
Situação	(x) Aprovado ( ) Aprovado com ressalvas ( ) Reprovado ( ) Não compareceu
Encaminhamento do trabalho para biblioteca	(x) Autoriza o encaminhado para biblioteca ( ) Manter sigilo para publicação ou geração de patente

Guarapuava, 5 de julho de 2019.

A folha de aprovação assinada encontra-se na coordenação do curso (ou programa).

The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; premature optimization is the root of all evil (or at least most of it) in programming. (KNUTH, Donald, 1974).

RESUMO

BALTOKOSKI, Donizety. Sistema web colaborativo de receitas com busca por ingredientes. 2019. 42 f. Trabalho de Conclusão de Curso – Câmpus Guarapuava, Universidade Tecnológica Federal do Paraná. Guarapuava, 2019.

O Brasil está entre os países que mais desperdiçam comida no mundo, perdendo aproximadamente 41 mil toneladas de alimento ao longo de toda cadeia alimentar que começa no plantio e termina no consumo final, pensando em contribuir para evitar o desperdício, este trabalho propõe o desenvolvimento de um sistema *web* de receitas colaborativas onde todos podem compartilhar suas receitas favoritas, além de proporcionar um mecanismo de avaliação, área de comentários e busca de receitas possíveis de fazer a partir dos ingredientes informados. O sistema deve proporcionar ao consumidor a utilização de receitas a partir dos ingredientes que tem à disposição, evitando desperdício e contribuindo com o meio ambiente.

Palavras-chave: Sistemas de computação. Sistemas hipertexto. Sustentabilidade. Desperdício (Economia).

ABSTRACT

BALTOKOSKI, Donizety. Collaborative recipe web system with a search for ingredients. 2019. 42 f. Trabalho de Conclusão de Curso – Câmpus Guarapuava, Universidade Tecnológica Federal do Paraná. Guarapuava, 2019.

Brazil is among the countries that waste most food in the world, losing approximately 41,000 tons of food throughout the food chain that begins at the planting and ends with the end customer. This paper proposes the development of a collaborative web system of recipes where everyone can share their favorite recipes, as well as providing an evaluation mechanism, comment area and search for recipes possible to cook from the informed ingredients. The system should provide consumers with recipes possibles to do from the ingredients they have available, avoiding waste and contributing to the environment.

Keywords: Computer systems. Hypertext systems. Sustainability. Waste (Economics).

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
BDD	Behaviour Driven Development
CSS	Cascading Style Sheets
GPC	Google Cloud Platform
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model, View, Controller
REST	Representational State Transfer
SPA	Single-Page Applications
SQL	Structured Query Language
TDD	Test Drive Development
URI	Uniform Resource Identifier
XP	Extreme Programming

SUMÁRIO

1 INTRODUÇÃO

Farinha de trigo, cevada selvagem, raízes de plantas trituradas e água são os ingredientes de uma das mais antigas receitas de pão que se tem conhecimento (G1, 2018). Esta receita tem aproximadamente 14 mil anos, situando-a antes do início da agricultura, podendo ser um dos motivos que levou a humanidade a cultivar cereais, dando início a agricultura. Outro aspecto desta receita é que ela data de antes da invenção da escrita, tornando impossível transmiti-la por pergaminhos ou livros, limitando-se apenas à transmissão oral, tradição que em menor grau perdura até os dias atuais. O advento da escrita e posteriormente da prensa tornou possível a disseminação de receitas em grande escala, capacidade rivalizada apenas pelo surgimento e popularização da Internet, permitindo que uma receita recém elaborada possa ser compartilhada com pessoas em qualquer parte do mundo com um dispositivo com acesso à Internet.

O passar do tempo não mudou apenas a forma de transmitir receitas. O surgimento da agricultura e da industrialização massificou a produção de alimentos, tornando-os extremamente acessíveis, criando um novo problema: desperdício de alimentos. O artigo *Global food losses and food waste – Extent, causes and prevention* traz dados alarmantes: cerca de 1,3 bilhão de tonelada de alimento é descartado todo ano - um terço de toda produção; em países desenvolvidos, uma pessoa desperdiça entre 95Kg e 115Kg de alimento por ano (FAO, 2011). Este desperdício se estende por toda a cadeia de produção, e contribui para o aumento do aquecimento global. Se toda comida desperdiçada fosse um país ele ficaria atrás apenas da China e dos E.U.A. na emissão de CO2 equivalente (o potencial de efeito estufa de outros gases são estimados em CO2 e chamados de CO2e ou CO2 equivalente) (WRI, 2015). É estimado que em 2050 o planeta terá 9 bilhões de habitantes, e para alimentar todo mundo, será necessário produzir mais 60% de caloria alimentícia. Se o desperdício for reduzido pela metade será necessário produzir apenas mais 22% (LIPINSKI et al., 2013).

O desperdício de alimentos afeta a todos, direta ou indiretamente, e não deve ser ignorado. Valendo-se desta premissa o SABRE (Sistema Avançado de Busca de Receitas), sistema *web* de receitas colaborativo visa facilitar o aproveitamento de ingredientes que o usuário já possui, colaborando com a redução de alimento que vira lixo no final da cadeia de produção, o consumidor. O sistema conta com uma funcionalidade de busca de receitas com base em ingredientes fornecidos, assim o usuário informa os ingredientes que tem à disposição e o sistema exibe as receitas possíveis de fazer apenas com os ingredientes informados. Este recurso não está presente em aplicações similares na língua portuguesa, dificultando a busca de receitas por brasileiros que se possa fazer apenas com os itens que já possui, geralmente tendo que adquirir mais ingredientes para poder preparar uma receita, que pode desestimular o uso dos itens ou aumentar a quantidade de ingredientes sem uso. O SABRE também tem os recursos mais comuns entre os sistemas similares: busca por nome, área de comentários e

avaliação de receita.

O principal desafio deste projeto está em como relacionar os ingredientes e as receitas, selecionando apenas aquelas que contêm alguns ou todos os ingredientes da lista e não necessite de nenhum outro item. Durante o desenvolvimento é necessário analisar algoritmos que façam esta correlação e escolher o que melhor se adéque ao contexto. Este algoritmo deve receber um conjunto de ingredientes e verificar em cada receita se o conjunto de ingredientes desta está contido no conjunto fornecido pelo usuário. Neste projeto, durante o prazo de desenvolvimento do TCC foi verificado a possibilidade de automatizar este relacionamento direto via banco de dados, fazendo uso de técnicas como *Full Text Search* respeitando as limitações impostas pelas tecnologias escolhidas. Também foram analisadas brevemente a possibilidade de identificar os ingredientes previamente e fazer este relacionamento via comandos diretamente no banco de dados e relacionar diretamente na camada de aplicação.

O SABRE tem como objetivo o auxílio na otimização do uso de alimentos evitando desperdícios, resultando em economia para o consumidor e redução do impacto ambiental, já que pode diminuir a possibilidade de descarte de alimento próprio para consumo.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver um sistema *web* que permita busca de receitas por meio de ingredientes disponíveis.

1.1.2 Objetivos Específicos

- Estudar e definir um algoritmo para selecionar receitas de acordo com conjunto de ingredientes.
- Desenvolver componentes para listar as receitas.
- Desenvolver um sistema de busca simples (procurar receitas de acordo com um termo fornecido pelo usuário, verificando os nomes e ingredientes das receitas).
- Desenvolver formulários de cadastro e autenticação de usuários.
- Desenvolver formulários de cadastro de receitas.
- Desenvolver sistema de categorias.
- Desenvolver sistema de avaliação de receitas.
- Desenvolver sistema de comentários.
- Desenvolver área administrativa.

2 SISTEMAS CORRELATOS

Este capítulo lista alguns sistemas existentes com funcionalidades similares ao SABRE.

2.1 TUDOGOSTOSO

TudoGostoso (Figura 1) é um site colaborativo de receitas pioneiro na área com um grande acervo. Permite encontrá-las por palavra-chave no nome ou ingrediente. Possui lista de favoritos, área de comentários, categorias, envio de receitas, envio de fotos, avaliação de receitas e *ranking* de usuários que mais postam receitas durante a semana.

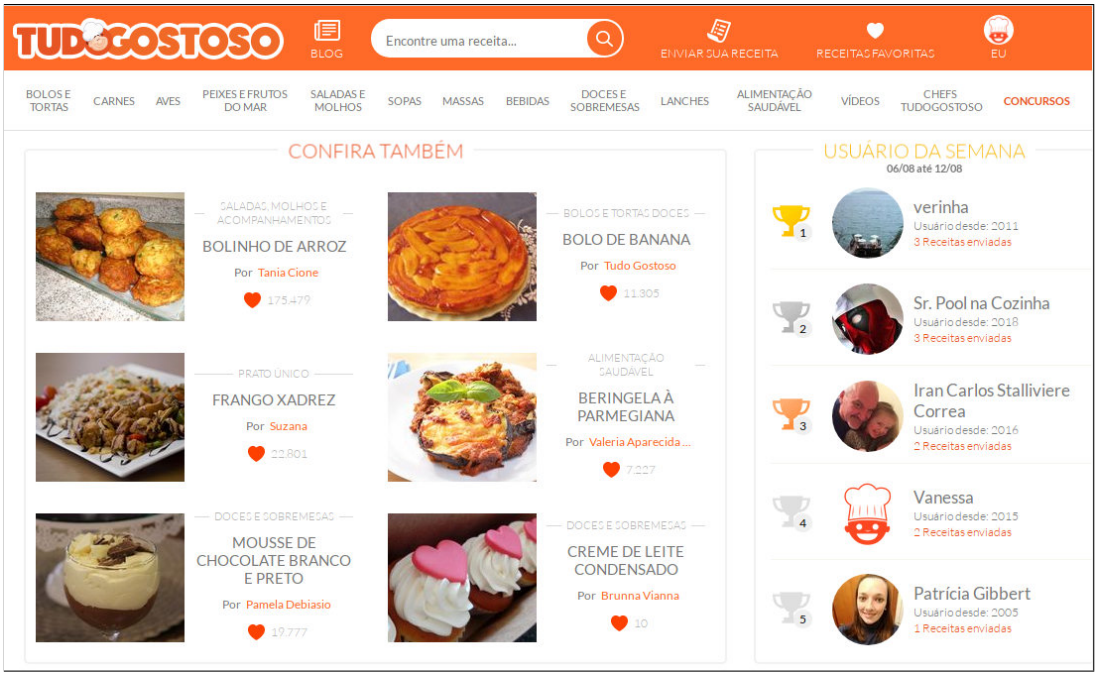

Figura 1 – TudoGostoso

Fonte: TudoGostoso (2018)

2.2 VOVÓ PALMIRINHA

Assim como TudoGostoso, Vovó Palmirinha (Figura 2) é um site colaborativo de receitas com um grande acervo. Permite encontrá-las por nome, além de possuir área de comentários, categorias e envio de fotos. Não possui sistema de qualificação para avaliar receitas, favoritos ou busca por ingredientes.

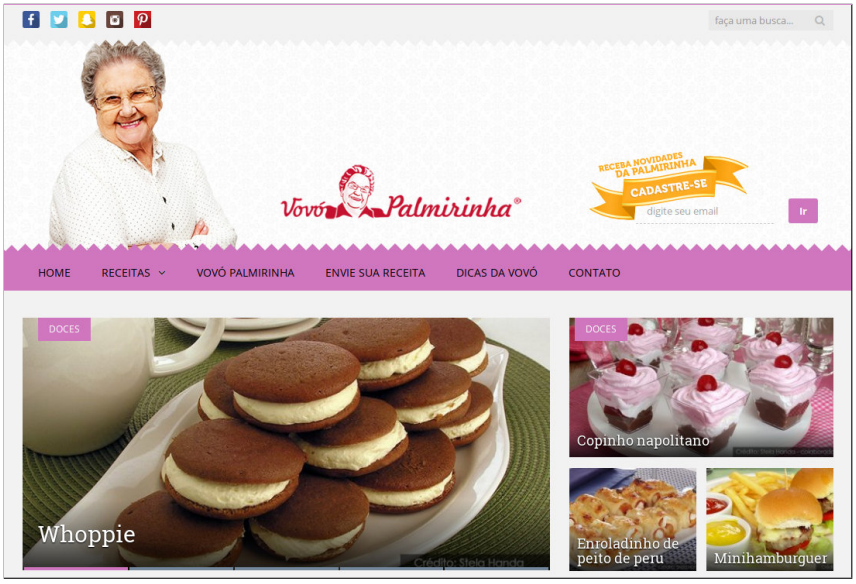

Figura 2 – Vovó Palmirinha

Fonte: Palmirinha (2018)

2.3 RECEITAS CYBERCOOK

Assim como TudoGostoso e Vovó Palmirinha, Receitas CyberCook (Figura 3) é um site colaborativo de receitas. Possui área de comentários, categorias, envio de receitas, fotos, sistema de qualificação para avaliar as mesmas, favoritos e busca por voz e texto - apenas nome. Não possui busca por ingredientes.

Figura 3 – Receitas CyberCook

Fonte: CyberCook (2018)

2.4 RECEITAS GSHOW

Diferente dos anteriores, Receitas GSHOW (Figura 4) não é um site colaborativo de receitas, pois seu foco é disponibilizar as que aparecem em programas de TV da Rede Globo, sendo exibidas como texto ou por meio de vídeo. Permite encontrá-las por nome, contudo não contém um mecanismo de avaliação, favoritos, área de comentários, envio de receitas e fotos, categorias ou busca por ingredientes.

Figura 4 – Receitas GSHOW

Fonte: Gshow (2018)

2.5 SUPERCOOK

SuperCook (Figura 5) é um site de receitas internacional que promete procurar receitas por ingredientes. Tem *design* focado em ingredientes, não tem idioma em português, não permite enviar receitas, fotos, comentários ou avaliações. Tem opção de favoritos. A busca por ingredientes retorna receitas possíveis de fazer com os ingredientes informados e receitas que precisam de mais ingredientes, mas em cada receita tem informação de qual ingrediente falta. Ele não armazena receitas - a busca mostra resultados de outros sites como allrecipes.com e food.com.

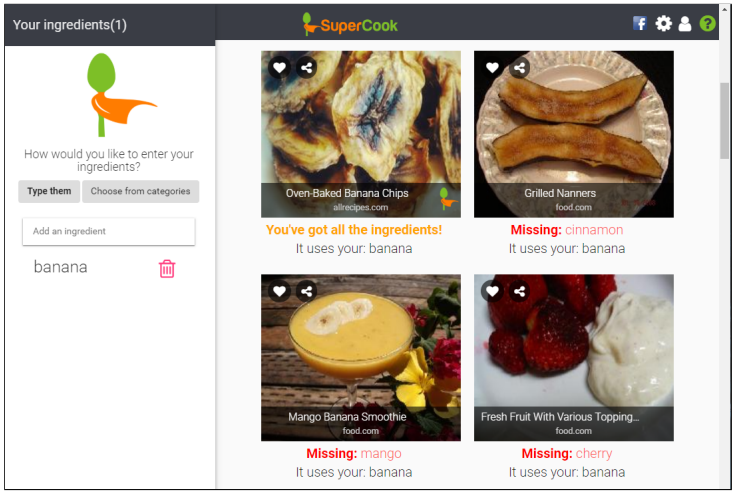

Figura 5 – SuperCook

Fonte: SuperCook (2019)

2.6 MYRECIPES

MyRecipes (Figura 6) é um sistema internacional de receitas com opção de busca por ingredientes. Assim como SuperCook ele mostra nos resultados receitas possíveis de fazer com os ingredientes informados e também receitas que precisam de mais ingredientes, mas diferente do SuperCook ele não dá dicas antes de abrir a receita. Possui sistema de análise e área com dicas para cozinheiros. Não possui favoritos, envio de receitas, fotos, comentários ou interface em português.

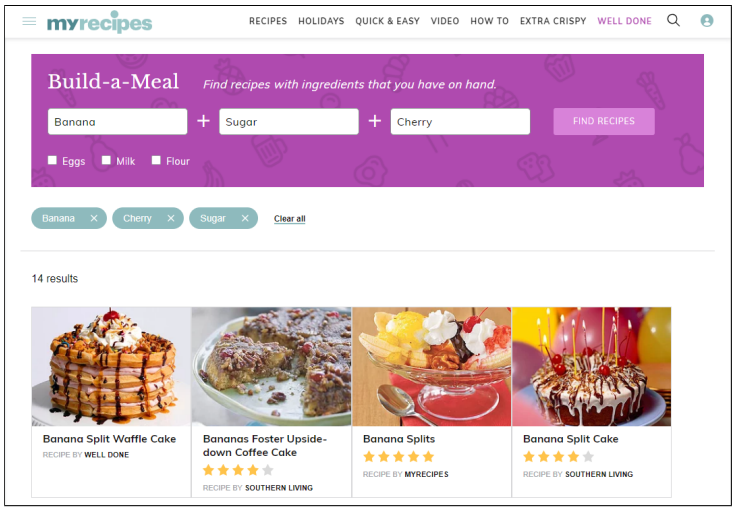

Figura 6 – MyRecipes

Fonte: MyRecipes (2019)

2.7 MYFRIDGEFOOD

MyFridgeFood (Figura 7) é outro sistema que promete encontrar receitas por ingredientes. Diferente dos demais ele não permite digitar o nome dos ingredientes - o usuário tem que escolher de uma lista disponível. A seleção tem um comportamento que pode ser indesejável para quem quer aproveitar o que já tem: ao selecionar uma fruta qualquer (como maçã) ele automaticamente seleciona uma opção para frutas em geral, embora seja possível remover da lista. Nos resultados também aparecem receitas que precisam de mais ingredientes. Possui comentários, sistema de envio de receitas e favoritos. Não possui sistema de avaliação ou idioma em português.

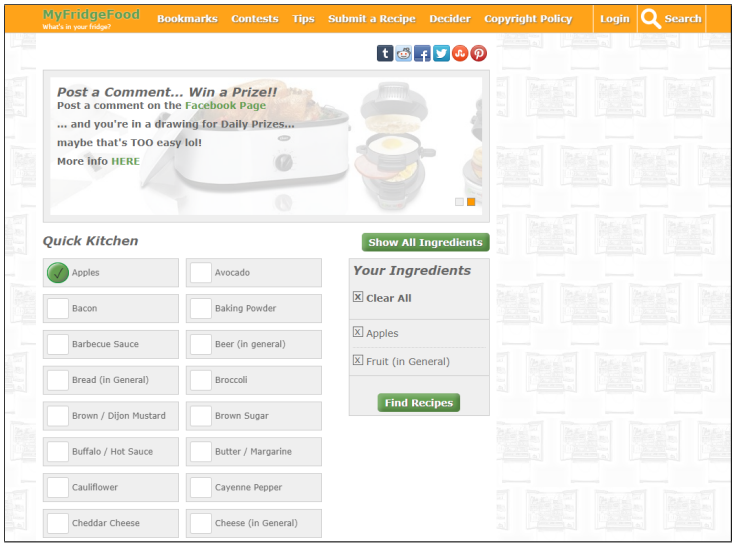

Figura 7 – MyFridgeFood

Fonte: MyFridgeFood (2019)

2.8 DIFERENCIAL TECNOLÓGICO

O SABRE propõe a característica distinta dos demais em encontrar receitas possíveis de preparar usando apenas os ingredientes disponíveis. Dentre sistemas nacionais similares existentes e apresentados nesta seção, apenas o site TudoGostoso chega a ter um mecanismo que leva os ingredientes em consideração, mas diferente do SABRE sua busca contempla outras receitas que exigem ingredientes não disponíveis, não tendo um filtro semelhante do sistema proposto. Já entre os internacionais o SuperCook consegue mostrar receitas possíveis de fazer com ingredientes fornecidos, mas não tem nenhum outro recurso como cadastro de receitas, avaliação, comentários e interface em português. Os demais, mesmo prometendo, não cumprem com eficiência este requisito.

3 TECNOLOGIAS USADAS NO DESENVOLVIMENTO

Este capítulo apresenta as principais tecnologias selecionadas para serem usadas no desenvolvimento.

3.1 REST E JSON

A arquitetura REST (*Representational State Transfer*) foi descrita pela primeira vez por Roy Thomas Fielding em sua tese de doutorado. É uma arquitetura que faz uso dos padrões URI e HTTP e define uma série de restrições e propriedades que visam performance de interação, simplicidade, modificabilidade, portabilidade e confiabilidade (FIELDING, 2000, cap. 2 e 5). As definições mais significativas para este projeto são: arquitetura cliente-servidor (Figura 8), *stateless* (cada requisição deve conter toda informação necessária, não fazendo uso de nenhum contexto salvo previamente no servidor) e interface uniforme.

JSON (*JavaScript Object Notation*) é um formato leve para troca de dados, fácil para humanos lerem e para máquinas gerarem e interpretarem. Embora seja baseado em um subconjunto da linguagem JavaScript seu formato é independente de linguagem (JSON.ORG, 2018). É usado nas trocas de informação entre o servidor e cliente.

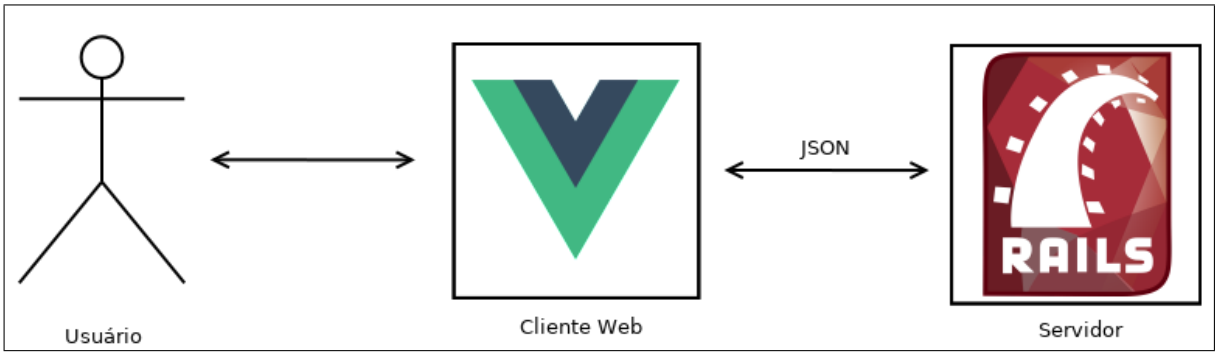

Figura 8 – Arquitetura Cliente-Servidor do SABRE

3.2 POSTGRESQL

É um sistema de banco de dados objeto-relacional de código aberto, que usa e estende a linguagem SQL combinando com vários recursos para armazenar os dados com segurança (POSTGRESQL, 2018). Tem a função de armazenar usuários, receitas, comentários e qualquer outra informação que precise ser persistida.

3.3 RUBY E RUBY ON RAILS

Ruby é uma linguagem de programação criada por Yukihiro "Matz" Matsumoto que mistura partes de suas linguagens favoritas (Perl, Smalltalk, Eiffel, Ada e Lisp) para formar uma nova linguagem que balance os paradigmas funcionais e imperativos (RUBY, 2018).

Ruby on Rails (também chamado apenas de Rails) é um *framework* de desenvolvimento *web* feito em Ruby. Rails segue o padrão MVC (*Model, View, Controller*) e tem suporte nativo a banco de dados relacional, APIs REST e outros recursos comuns em sistemas *web* (RAILSGUIDES, 2018).

Estas tecnologias são usadas no servidor do SABRE.

3.4 PGSEARCH

PgSearch é um *pluguin* Rails que adiciona a funcionalidade de busca *Full Text Search* do PostgreSQL, recurso não presente nativamente no Rails. Com ele é possível usar o *Full Text Search* com as extensões *Trigram*, *Unaccent* e *Fuzzystrmatch*, que enriquecem a busca usando similaridades em sequencias de caracteres, ignorando acentos e por aproximação fonética (PGSEARCH, 2018). É usado para realizar busca simples nas receitas, usando termo fornecido pelo usuário.

3.5 RSPEC

RSpec é uma ferramenta de testes para aplicativos criados na linguagem Ruby, feito para facilitar o desenvolvimento em BDD (*Behaviour Driven Development*) e TDD (RSPEC, 2018). É usado essencialmente para testes unitários e de requisição no servidor do projeto.

3.6 RUBOCOP, REEK, BRAKEMAN, BULLET, TRACEROUTE, BUNDLER-AUDIT e GUARD

RuboCop é um analisador de código fonte Ruby. Ele verifica se o código segue o *Ruby Style Guide*, procura por erros, consegue corrigir alguns problemas automaticamente e é altamente personalizável. (RUBOCOP, 2018).

Reek é outro analisador de código fonte Ruby, com foco em *Code Smells* (código duplicado, inalcançável, variáveis não descritivas, etc). Não corrige o erro, apenas exibe o problema. (REEK, 2019)

Brakeman é uma ferramenta de análise que procura por vulnerabilidades de segurança em aplicações Ruby on Rails - que valida do uso de variáveis até instruções SQL. (BRAKEMAN, 2019)

Bullet analisa as interações com o banco de dados e sugere alterações para melhorar a performance da aplicação. (BULLET, 2019)

Traceroute ajuda a encontrar rotas problemáticas nas aplicações Ruby on Rails. Esta ferramenta informa rotas que não podem ser alcançadas e ações não acessíveis. (TRACEROUTE, 2019)

Bundler-Audit verifica o arquivo Gemfile.lock e informa se a aplicação Ruby estiver usando uma biblioteca que deixe o sistema vulnerável. Exibe um aviso e sugestão de melhoria sempre que um problema é encontrado. (AUDIT, 2019)

Guard é uma ferramenta para automatizar tarefas. Ela verifica alterações em arquivos e executa tarefas personalizadas (GUARD, 2019). No projeto do servidor ela deverá executar as ferramentas de análise e teste sempre que um arquivo pertinente for alterado.

3.7 APACHE JMETER

Apache jMeter é uma software de código aberto feito para realizar testes de carga em aplicações *web* e mensurar sua performance (JMETER, 2019). É usado inicialmente para teste de carga nas rotas públicas do servidor.

3.8 HTML E CSS

HTML (*Hypertext Markup Language*) é a linguagem que descreve a estrutura de páginas *web*. Com ela é possível publicar documentos com cabeçalhos, textos, tabelas, fotos, etc. A estrutura das páginas é definida usando marcações que rotulam conteúdos como "formulário", "paragrafo", etc (W3C, 2018).

CSS (*Cascading Style Sheets*) é a linguagem que define a apresentação de páginas *web*, tais como cores, posição de conteúdo e fontes tipográficas. Com ela é possível adaptar o conteúdo em diferentes dispositivos, como telas grandes, pequenas ou para impressoras (W3C, 2018).

Estas duas linguagens são fundamentais no desenvolvimento do cliente *web* do SABRE.

3.9 JAVASCRIPT, VUE.JS e AXIOS

JavaScript® (comumente abreviado para JS) é uma linguagem interpretada, multi-paradigma, baseada em protótipos, com suporte à orientação a objetos, programação imperativa e declarativa. Inicialmente criada para proporcionar dinamismo as páginas *web*, agora ela pode ser usada em outras áreas como servidores e aplicativos não *web* (MOZILLA, 2018).

Vue.js é um *framework* JavaScript progressivo para criar interfaces de usuário. Seu núcleo é focado na parte visual, enquanto outras bibliotecas oficiais e não oficiais são fornecidas para adicionar outras funcionalidades, como temas, rotas, gerenciamento de estado, etc. Em conjunto com outras ferramentas e bibliotecas Vue tem a capacidade de criar SPA (*Single-Page Applications*) (VUE, 2018).

Axios é um cliente HTTP feito em JavaScript. Suporta requisições GET, POST, DELETE, PATCH e PUT, além de converter automaticamente os dados em JSON (AXIOS, 2018).

O cliente *web* do projeto é uma SPA com o conteúdo gerenciado dinamicamente pelo Vue.js e as trocas de informação com o servidor feitas através do axios.

3.10 KARMA

É uma ferramenta que testa código JavaScript em diversos navegadores. Ele funciona carregando um servidor e um ou mais navegadores, em seguida executa os testes em cada navegador. O Resultado é exibido no terminal. Extensível com *plugins* que adicionam suporte a mais navegadores e recursos extras de teste (KARMA, 2018).

3.11 ESLINT

ESLint é um *linter* JavaScript de código aberto. Um *linter* é uma ferramenta que faz análise estática de código fonte, procurando por padrões problemáticos ou códigos que não respeitam as regras de estilo. ESLint é um utilitário flexível, podendo ativar, desativar ou alterar regras pré-definidas, além da possibilidade de adicionar novas regras personalizadas (ESLINT, 2018).

3.12 GIT e GIGHUB

Git é um sistema livre e gratuito de versionamento de código projetado para lidar com projetos pequenos ou grandes com velocidade e eficiência (GIT, 2018).

GitHub é uma plataforma de desenvolvimento onde é possível hospedar e revisar código, gerenciar projetos e construir aplicativos de forma colaborativa (GITHUB, 2018).

Os códigos dos projetos cliente e servidor estão nos repositórios públicos <https://github.com/DoniB/sabre_server> e <https://github.com/DoniB/sabre_web> do GitHub. Tanto o módulo servidor quanto o cliente estão disponíveis sob a licença livre *BSD 3-Clause*

3.13 DOCKER, DOCKER COMPOSE e KUBERNETES

Docker é uma tecnologia de contêineres para empacotar produtos de software junto com suas dependências para facilitar o desenvolvimento e implantação (DOCKER, 2019b).

Docker Compose é uma ferramenta para configurar e executar aplicações multi contêineres Docker. Com um único arquivo de configuração é possível orquestrar o funcionamento de vários serviços. Pode ser usado durante o desenvolvimento, implantação, teste e integração contínua (DOCKER, 2019a).

Kubernetes (comumente escrito como k8s) é um sistema feito para automatizar a implantação, escalonamento e gerenciamento de contêineres. Automatiza tarefas como

balanceamento de carga, armazenamento, autorreparo, escalonamento horizontal e execução em lote (KUBERNETES, 2019).

3.14 GOOGLE CLOUD PLATFORM

GPC (Google Cloud Platform) é um conjunto de produtos e serviços feitos para facilitar e agilizar o desenvolvimento e implantação de sistemas de software (GOOGLE, 2019). Os produtos selecionados para usar no SABRE são Cloud Storage para armazenar fotos das receitas e persistir os dados do banco de dados, além do Google Kubernetes Engine, para gerenciar a execução do servidor, cliente *web* e gerenciamento de carga.

3.15 TRAVIS CI

Travis CI é uma ferramenta para automatização de teste e implantação que se integra com facilidade ao Github (TRAVIS-CI, 2019). Sua função no SABRE é testar toda alteração no código enviada ao Github. Em caso de sucesso ele gera uma imagem Docker com o novo código e suas dependências, em seguida notificando o Google Kubernetes Engine para atualizar seus contêineres com a nova versão.

3.16 TRELLO

Trello (Figura 9) é uma ferramenta *web* para organizar projetos e ideias. Nele é possível ter listas e cartões organizados facilmente e visualmente. Os projetos podem ter vários membros, que podem interagir com as listas e cartões, além de comentar nos mesmos (TRELLO, 2018).

Seu principal papel é de organizar o projeto e registrar o progresso do desenvolvimento do SABRE.

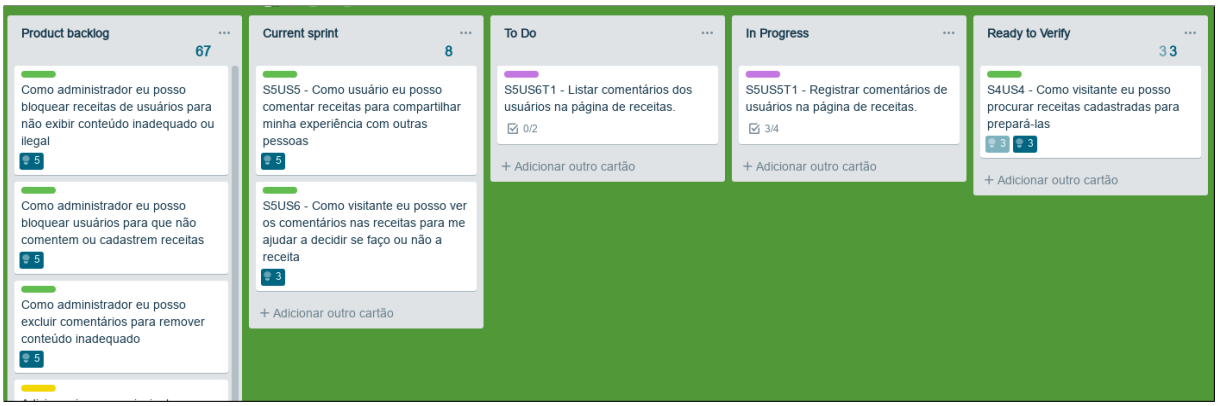

Figura 9 – Trello do projeto SABRE durante a *Sprint* 5

4 ANÁLISE E PROJETO DO SISTEMA

Este projeto foi desenvolvido seguindo fortemente os preceitos de desenvolvimento ágil definido por Fowler (2005) como: adaptativo em vez de preditivo e orientado a pessoas em vez de processos. Neste tipo de projeto evita-se a produção de artefatos que representam a visão final do produto visto que o desejo e necessidade do cliente é mais importante que os processos, e este pode mudar de ideia ou perceber que a funcionalidade pedida não corresponde as suas necessidades. Sua percepção pode mudar a cada entrega de funcionalidades e todo planejamento pode ser descartado para dar origem a um projeto diferente. Para fins didáticos e de cumprimento de requisitos da disciplina de TCC esta seção contém alguns artefatos que ilustram o resultado final do projeto de acordo com a percepção do sistema durante a modelagem inicial. Este capítulo também apresenta metodologia usada no desenvolvimento e as histórias de usuário (*User Stories*).

4.1 METODOLOGIA DE DESENVOLVIMENTO DO SISTEMA

O desenvolvimento do SABRE fez-se de maneira incremental e iterativa. A cada iteração foram entregues novas funcionalidades até que no último ciclo proposto foi desenvolvido no prazo estabelecido (ver detalhes na Seção 5.1). Para o desenvolvimento foram selecionados elementos dos modelos de desenvolvimento TDD (*Test Drive Development*) e *Scrum*, sendo que o aluno atuou nos papéis de usuário e desenvolvedor.

Os requisitos do sistema foram extraídos do item 1.1.2 deste documento e convertidos em *User Story* (História do Usuário, em tradução livre) e devem responder as perguntas "Quem?", "Por quê?"e "O Quê?". Por exemplo, para o módulo de cadastro de receitas: "Como usuário eu posso cadastrar receitas para que outras pessoas possam fazê-las" (ver 4.2).

O fluxo de trabalho foi organizado em quadros e cartões, categorizados em *Product Backlog*, *Current Sprint*, *To do*, *In Progress*, *Review* e *Done*. *Product Backlog* contém todas as *User Story* ordenadas por prioridade e movidas para *Current Sprint* quando seu desenvolvimento é iniciado. As histórias marcadas para desenvolver têm suas tarefas definidas em *To Do* e movidas para *In Progress* quando em desenvolvimento, seguindo para *Review* e *Done* após revisão. Uma tarefa marcada como *Done* deve passar pelo ciclo: Requisito, *Product Backlog*, *Current Sprint*, *To Do*, *In Progress*, Teste e Revisão. Após a revisão e verificação de conformidade com requisito solicitado a tarefa passa para o estado de *Done*.

No método *Scrum*, uma *Sprint* é uma unidade de tempo definida para ter uma entrega de requisitos convertidos em produto (SCHWABER; SUTHERLAND, 2017), cada *Sprint* teve duração de uma semana durante o projeto. Este método também define várias reuniões que foram simplificadas no desenvolvimento. Houve apenas uma reunião semanal onde era discutido o que foi feito na última *Sprint*, o que seria feito na próxima e os possíveis problemas encontrados

durante o último ciclo. Em algumas reuniões também houve *Sprint Retrospective*, evento onde a metodologia é discutida e adaptada visando aumentar a produtividade - principalmente para ajustar o tempo necessário de cada *User Story*.

No modelo TDD o desenvolvedor primeiro escreve um teste automatizado para a funcionalidade que desenvolverá, este teste deve falhar enquanto a funcionalidade não estiver funcionando propriamente. Em seguida é escrito o código da funcionalidade para que passe no teste. Por último o código é refatorado seguindo boas práticas como simplicidade e não duplicidade (AGILEALLIANCE, 2015). A cada nova funcionalidade o sistema acumula mais códigos de teste que verificam o funcionamento de todas as funcionalidades implementadas. Este procedimento foi adotado durante todo o desenvolvimento do projeto do servidor onde as tarefas são facilmente definidas antes do desenvolvimento, já no desenvolvimento do cliente *web* esta atividade foi ignorada pela falta de experiência nas tecnologias utilizadas.

4.2 USER STORIES

Uma *User Story* é um artefato de alto nível em metodologias ágeis como *Scrum* e *Extreme Programming* (XP), ela deve conter informação suficiente para o desenvolvedor estimar um prazo de desenvolvimento (AGILEMODELING, 2013). As *User Stories* definidas durante o planejamento são:

- Como usuário eu posso cadastrar receitas para que outras pessoas possam fazê-las.
- Como administrador eu posso liberar receitas cadastradas pelos usuários para que outros usuários possam prepará-las.
- Como visitante eu posso ver receitas cadastradas para prepará-las.
- Como visitante eu posso procurar receitas cadastradas para prepará-las.
- Como usuário eu posso comentar receitas para compartilhar minha experiência com outras pessoas.
- Como visitante eu posso ver os comentários nas receitas para me ajudar a decidir se faço ou não a receita.
- Como usuário eu posso avaliar receitas para expressar o quanto gostei da mesma.
- Como visitante eu posso ver as avaliações das receitas para me ajudar a decidir qual receita fazer.
- Como visitante eu posso ver receitas por categoria para facilitar minha busca.
- Como administrador eu posso cadastrar outro administrador para gerenciar receitas e administradores.
- Como administrador eu posso bloquear receitas de usuários para não exibir conteúdo inadequado ou ilegal.
- Como administrador eu posso bloquear usuários para que não comentem ou cadastrem receitas.
- Como administrador eu posso excluir comentários para remover conteúdo inadequado ou ilegal.

- Como visitante eu posso ver receitas possíveis de fazer com um conjunto de ingredientes especificados para aproveitar o que já tenho.

4.3 BANCO DE DADOS

O SABRE usa o PostgreSQL para gerenciar o banco de dados, contando com os recursos do Ruby on Rails para acessar e manipular os registros.

A Figura 10 [1] é uma representação do banco de dados considerando a visão durante o planejamento do sistema que ignora as especificidades e limitações do *framework* que gerencia os dados. A versão atual em linguagem SQL pode ser vista em db/structure.sql no repositório do servidor.

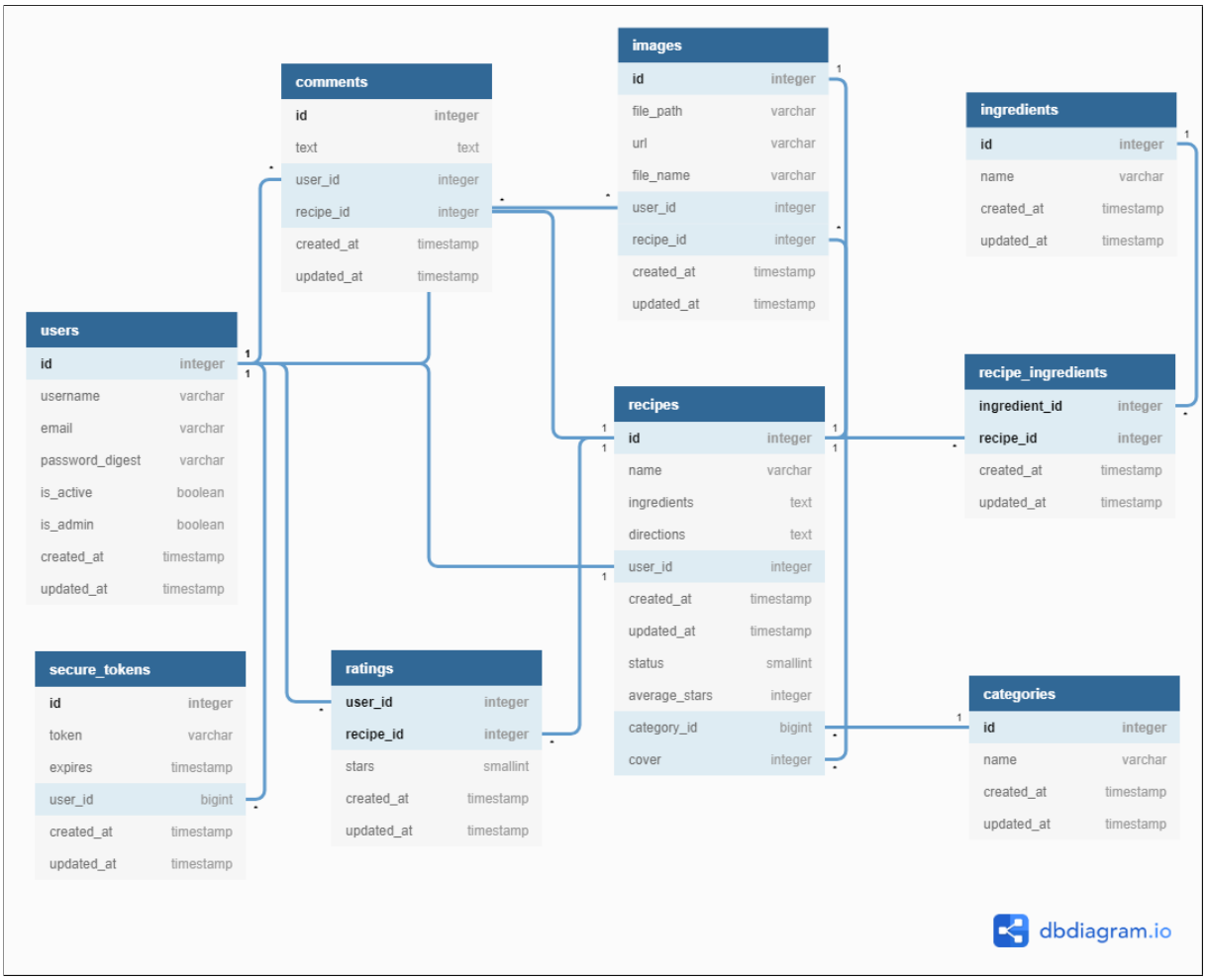

Figura 10 – Diagrama do Banco de Dados

[1] Este modelo foi desenvolvido com dbdiagram.io e está disponível publicamente em <https://dbdiagram.io/d/5be4b302cf7cf50014f2a984>.

4.4 TELAS DO CLIENTE WEB

Esta seção apresenta algumas telas do cliente SABRE.

A página inicial (Figura 11) contém uma barra com caixa de pesquisa, uma área com as categorias e listagem de receitas.

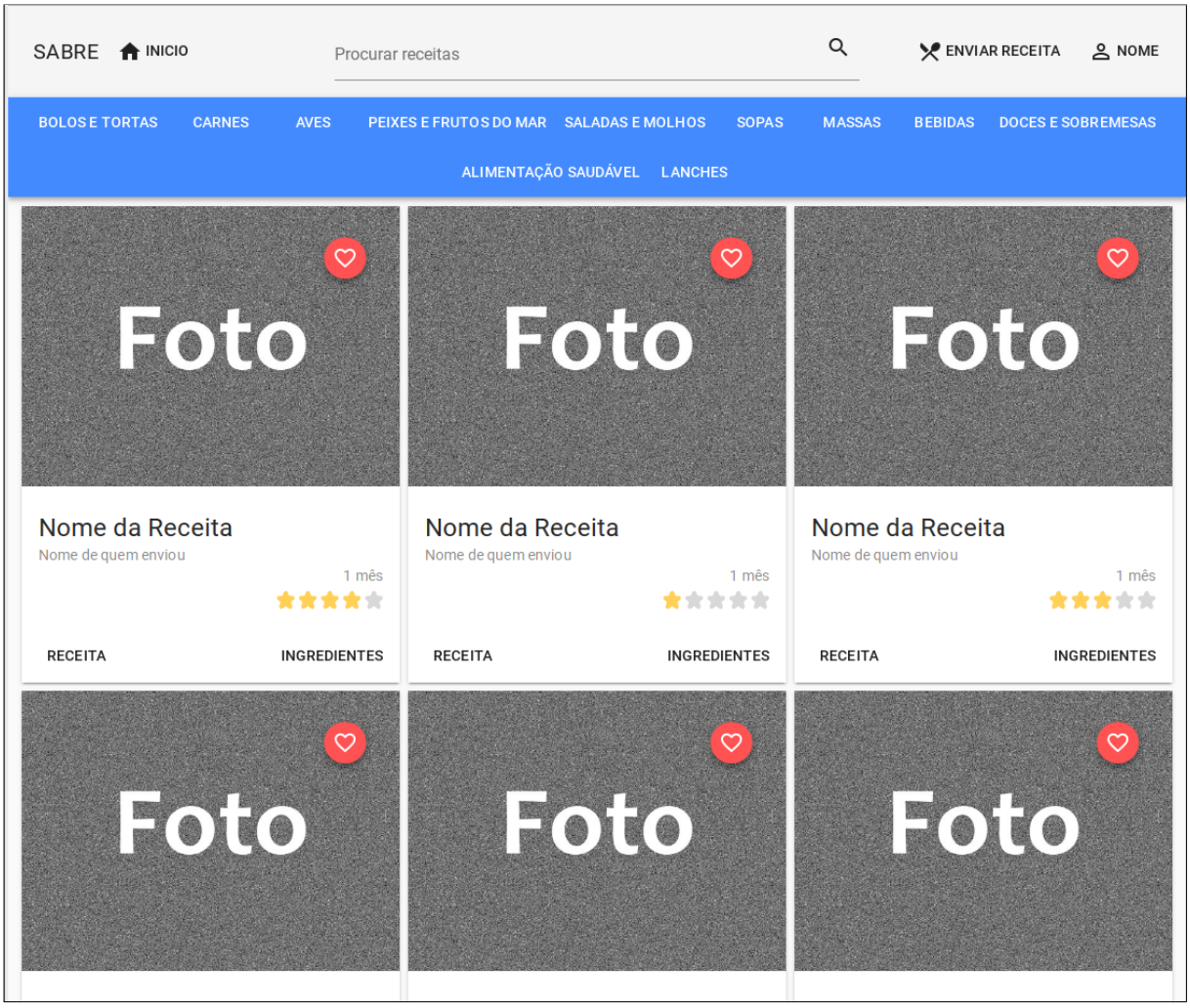

Figura 11 – Página inicial

Cada receita na página inicial (Figura 12) exibe o nome da receita, nome de quem enviou, foto, média de avaliações, botão para adicionar aos favoritos, botão para ver a receita completa e botão para visualizar os ingredientes sem precisar abrir a receita.

A Figura 13 exibe a caixa de pesquisa em três estados: não ativa; ativa; ativa com itens separados por vírgula. Quando ativa aparece a sugestão para separar os ingredientes com vírgula. Se uma vírgula é detectada aparece a opção para procurar por receitas que possam ser feitas apenas com aqueles ingredientes.

A página de receitas de usuário (Figura 14) exibe as receitas usando a mesma aparência familiar da página inicial. A página de favoritos é similar, mas exibe as receitas que foram marcadas como favoritas.

Figura 12 – Receita com pré-visualização de ingredientes

Um usuário administrativo pode acessar todas as receitas, podendo filtrar por suas receitas e receitas pendentes de aprovação (Figura 15).

Outras telas podem ser vistas no Apêndice A.

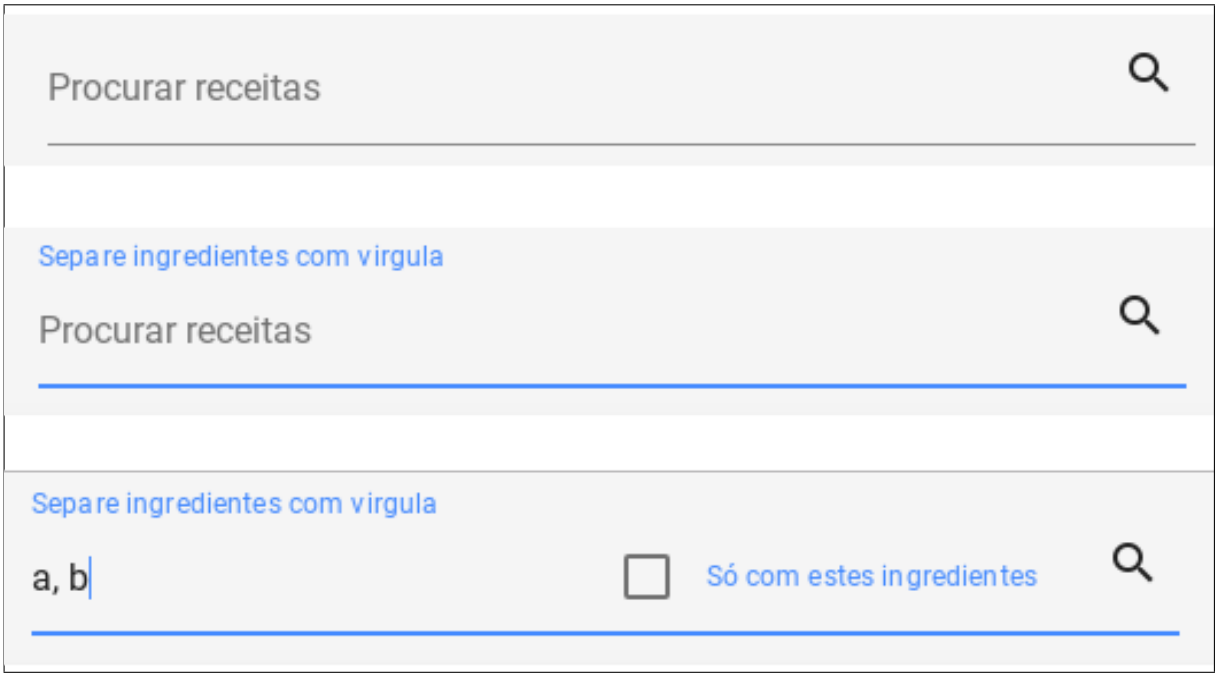

Figura 13 – Caixa de pesquisa

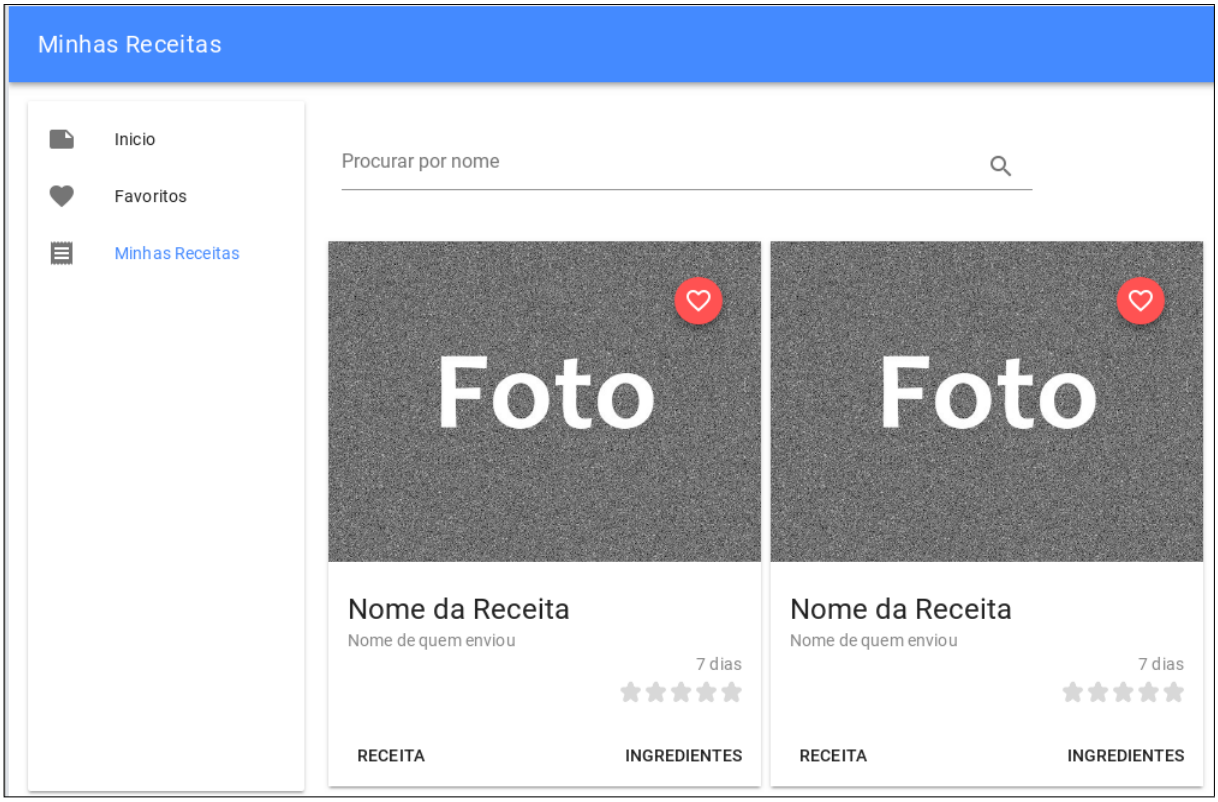

Figura 14 – Receitas do usuário

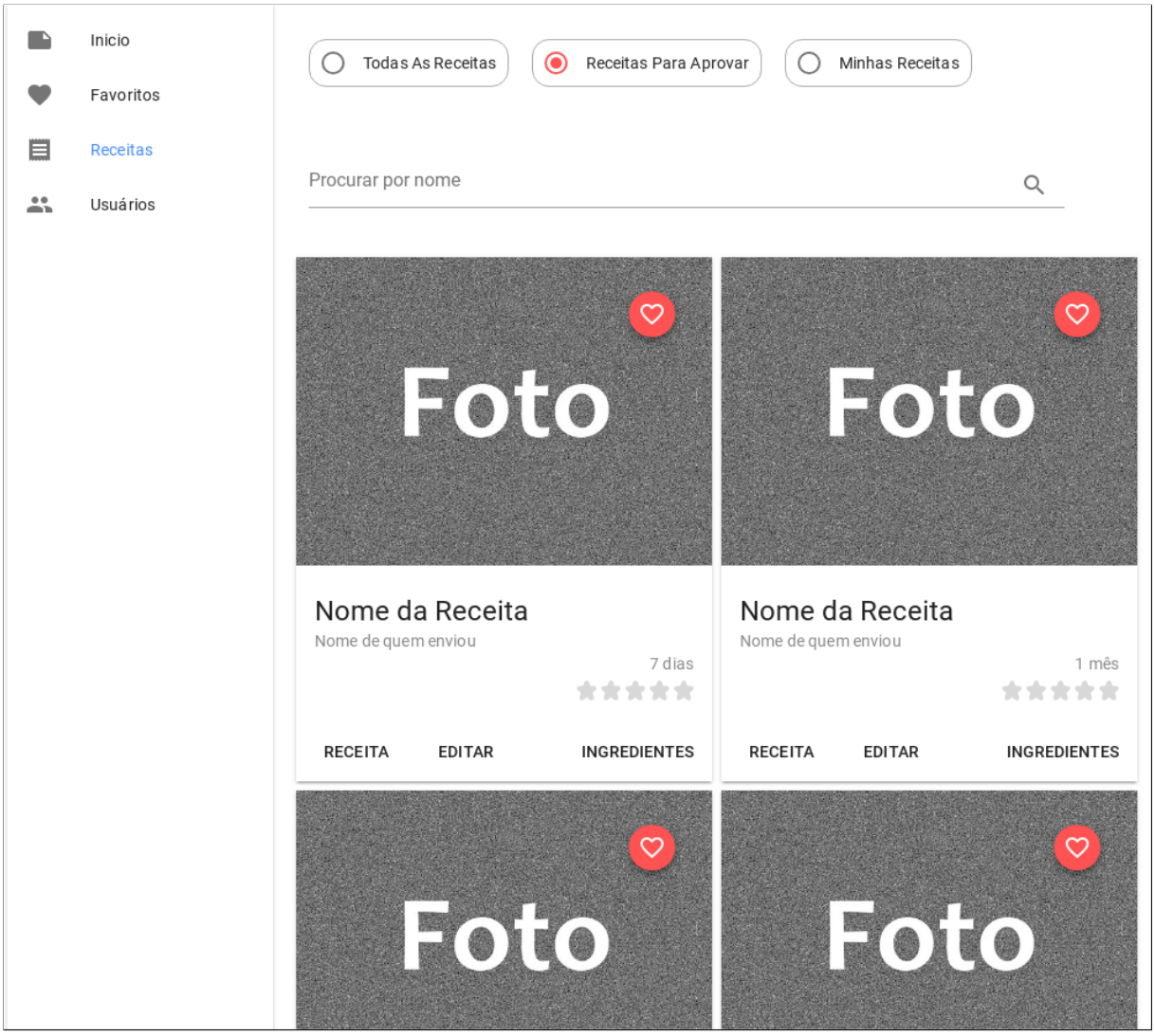

Figura 15 – Receitas para aprovar (usuário administrativo)

4.5 PLANEJAMENTO DO TRABALHO

O Quadro 1 contém as atividades relacionadas ao TCC e as datas aproximadas de início e conclusão das mesmas.

Atividades	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**
1. Revisão dos apontamentos da banca	X										
2. Revisão bibliográfica	X	X									
3. Levantamento de requisitos	X	X									
4. Redação do projeto de TCC	X	X	X								
5. Desenvolvimento do sistema		X	X	X	X	X					
6. Defesa do projeto de TCC				X							
7. Escrita da Monografia de TCC					X	X	X	X			
8. Elaboração da apresentação final							X	X	X	X	
9. Defesa final do TCC											X

Quadro 1 – Cronograma de Atividades.

5 DESENVOLVIMENTO DO SISTEMA

O desenvolvimento teve início em 25 de setembro de 2018. Foram estimadas 135 horas para transformar os requisitos em um produto de software. Com uma carga de trabalho de oito horas por semana/*sprint* e sem alterações dos requisitos, a conclusão do projeto ocorreria no final de fevereiro de 2019. Esta seção contém um resumo das *Sprints* concluídas até o término deste documento e um gráfico do progresso (gráfico *burndown*).

5.1 SPRINTS

Cada *Sprint* teve duração de uma semana, destinando-se oito horas semanais para o projeto, sendo seis de desenvolvimento e duas para estudo e imprevistos.

5.1.1 Sprint 1: Registro e Autenticação de Usuário

Iniciada em 25 de setembro. Foi definido o tempo de oito horas destinadas para desenvolvimento em cada *Sprint* e selecionada a *User Story* "Como usuário eu posso cadastrar receitas para que outras pessoas possam fazê-las" para desenvolver. Foi criado o projeto do servidor em *Ruby on Rails* e desenvolvidos cadastro de usuário, sistema de geração de *token* para identificar o usuário, cadastro de receitas (nome, ingredientes e modo de preparo) e testes seguindo o modelo TDD no servidor. Também foi criado o projeto cliente *web* usando Vue.js com formulários de cadastro e *login*. Não foi possível criar o formulário de cadastro de receitas e os testes no lado do cliente por falta de conhecimento das tecnologias utilizadas. Um *token* é uma sequência aleatória de caracteres gerada sempre que o login e senha é enviado ao servidor, ele fica salvo no banco de dados e cada vez que o cliente acessa uma área restrita tem que passar o *token* no cabeçalho da requisição - identificando e validando o usuário.

5.1.2 Sprint 2: Cadastro de Receitas

Iniciada em 2 de outubro de 2018. Foi ajustado o tempo de oito para seis horas de desenvolvimento, deixando duas horas para estudos e imprevistos. Essa *Sprint* também foi reservada para a conclusão da *Sprint* anterior e calcular o tempo necessário para desenvolver cada *User Story*. Foram feitas pequenas correções no servidor e desenvolvido o formulário de cadastro de receitas no cliente.

5.1.3 Sprint 3: Edição de Receitas

Iniciada em 9 de outubro de 2018. Selecionada a *User Story* "Como administrador eu posso liberar receitas cadastradas pelos usuários para que outros usuários possam prepará-las". No servidor foi adicionada a opção de atualizar receitas pelo usuário que tem status

administrativo. No cliente *web* foi desenvolvida uma *dashboard*, com listagem e edição de receitas acessíveis apenas para usuários administrativos. Nesta etapa foi percebido um crescente débito técnico pela falta de testes no cliente *web* que terá que ser resolvido posteriormente.

5.1.4 Sprint 4: Listagem de Receitas e Busca Simples

Iniciada em 16 de outubro de 2018. Selecionadas as *User Stories* "Como visitante eu posso ver receitas cadastradas para prepará-las" e "Como visitante eu posso procurar receitas cadastradas para prepará-las". Nesta etapa foi feita a listagem de receitas, função de busca por termo (que considera o nome e os ingredientes) e exibição de receita individualmente, tanto no servidor quanto no cliente *web*.

5.1.5 Sprint 5: Comentários em Receitas

Iniciada em 23 de outubro de 2018. Selecionadas as *User Stories* "Como usuário eu posso comentar receitas para compartilhar minha experiência com outras pessoas" e "Como visitante eu posso ver os comentários nas receitas para me ajudar a decidir se faço ou não a receita". Foram desenvolvidas no servidor e cliente *web* as funcionalidades de comentar e ver comentários nas receitas.

5.1.6 Sprint 6: Sistema de Avaliação de Receitas

Iniciada em 30 de outubro de 2018. Selecionadas as *User Stories* "Como usuário eu posso avaliar receitas para expressar o quanto gostei da mesma" e "Como visitante eu posso ver as avaliações das receitas para me ajudar a decidir qual receita fazer". Foi desenvolvido um sistema de avaliação de receitas com escala de um a cinco. O servidor recalcula a média de avaliações a cada avaliação e o cliente *web* as exibe com ícones de estrelas.

5.1.7 Sprint 7: Categorias

Iniciada em 6 de novembro de 2018. Selecionada a *User Story* "Como visitante eu posso ver receitas por categoria para facilitar minha busca". Foi desenvolvida listagem de categorias no servidor (adicionadas manualmente) e associada às receitas. No cliente *web* foi adicionada categoria no cadastro de receitas e menu de categorias nas áreas públicas. Também foi dedicado um tempo extra para produzir telas para este documento (Seção 4.4).

5.1.8 Sprint 8: Usuários Administrativos

Iniciada em 13 de novembro de 2018. Selecionada a *User Story* "Como administrador eu posso cadastrar outro administrador para gerenciar receitas e administradores". Foram desenvolvidas no cliente e servidor a listagem, cadastro e edição de usuários por usuários administradores. Também foi dedicado um tempo extra para desenvolver este documento.

5.1.9 Sprint 9 e 10: Período Inativo

Nenhuma atividade foi realizada neste período, 13 e 20 de novembro. Esta inatividade resultou em trabalho acumulado, que necessitariam mais tempo dedicado ao projeto ou alteração do prazo final.

5.1.10 Sprint 11: Bloqueio de Usuários e Receitas

Iniciada em 4 de dezembro de 2018. Selecionadas as *User Stories* "Como administrador eu posso bloquear usuários para que não comentem ou cadastrem receitas" e "Como administrador eu posso bloquear receitas de usuários para não exibir conteúdo inadequado ou ilegal". Foi adicionado um campo no modelo do usuário para identificar se ele está ativo. No formulário de edição de usuários pelo administrador foi adicionada a opção de bloquear e liberar o usuário. Tal opção também foi disponibilizada direto na listagem de usuários. A página de "receitas para aprovar" foi atualizada para ter opções de exibir as receitas do usuário, receitas para aprovar e todas as receitas (de todos os usuários), facilitando encontrar qualquer receita para edição e bloqueio. Na página pública de receita individual foram adicionados botões para editar a receita e o dono da receita (se o usuário que acessar for um administrador).

5.1.11 Sprint 12: Favoritos e Exclusão de Comentários

Iniciada em 11 de dezembro de 2018. Selecionadas as *User Stories* "Como usuário eu posso adicionar receitas aos favoritos para poder ver depois" e "Como administrador eu posso excluir comentários para remover conteúdo inadequado". Foi desenvolvido um sistema para adicionar e remover receitas nos favoritos. É possível adicionar receitas pela página individual de cada receita ou nas listagens (página inicial, categorias e resultado de busca). Os favoritos são listados na área privada do usuário. Também foi adicionado uma opção para remover comentários (apenas por administradores). Para fins de auditoria, os comentários não são removidos do banco de dados, apenas tem atributos para registrar quando e por quem foi removido.

5.1.12 Sprint 13: Imagem Principal da Receitas

Iniciada em 18 de dezembro de 2018. Selecionada a *User Story* "Como usuário eu posso cadastrar receitas com foto para que outras pessoas possam ver o resultado final". Foi desenvolvido um sistema de envio de imagens durante o cadastro e edição das receitas. Foram estimadas 4 horas (5 pontos) para o desenvolvimento mas foram necessárias 12 horas para concluir esta etapa. Também foram integrados os serviços *Travis-ci* e *Code Climate* ao projeto do servidor.

5.1.13 Sprint 14: Refinamento da Busca Simples

Iniciada em 25 de dezembro de 2018. Nenhuma *User Story* foi selecionada. Semana dedicada ao refinamento da busca simples, onde foram adicionadas as extensões *Unaccent*, *Fuzzystrmatch* e *Trigram* ao PostgreSQL, além de *cache* dos campos "nome" e "ingredientes" para agilizar as pesquisas usando o método *Full Text Search*. No servidor foram adicionadas as ferramentas Bundler-Audit, Reek, Bullet e Brakeman para auxiliar o desenvolvedor, apontando falhas previsíveis e melhorias. O Travis-CI foi configurado para fazer a implantação no *Heroku* automaticamente para cada atualização no repositório GIT. No cliente *web* foi adicionado o *Prettier*, *Code Climate* e *Travis-CI*, que também faz a implantação automaticamente.

5.1.14 Sprint 15: Busca Avançada com Full Text Search

Iniciada em 1 de janeiro de 2019. Nenhuma *User Story* foi selecionada. Foram feitos alguns testes com pg_search, usando 627 receitas e com termos de busca que deveriam retornar 26 receitas. Nos testes foi percebido que apenas o parâmetro *threshold* da extensão *Trigram* alterava significativamente o resultado. Também foi percebido que as opções para reduzir o total de respostas inválidas resultava em uma diminuição de respostas válidas, não exibindo todas as receitas possíveis de fazer com os ingredientes informados. A Tabela 1 contém alguns resultados.

Tabela 1 – Resultado dos testes com Full Text Search.

Threshold	Receitas Válidas	Receitas Inválidas	Total
0.040	26	597	623
0.230	26	445	471
0.232	25	443	468
0.256	25	413	438
0.257	24	413	437
0.282	24	375	399
0.283	22	375	397
0.300	22	351	373
0.320	21	321	342
0.380	20	240	260
0.400	19	221	240
0.420	18	195	213
0.820	5	0	5

5.1.15 Sprint 16: Sem atividade

Período sem atividade.

5.1.16 Sprint 17: Filtrando Receitas por Ingredientes 1

Iniciada em 15 de janeiro de 2019. Selecionada a *User Story* "Como visitante eu posso ver receitas possíveis de fazer com um conjunto de ingredientes especificados para aproveitar o que já tenho". Foi desenvolvido um algoritmo que resgata todas as receitas do banco de dados e em seguida filtra de acordo com os parâmetros do usuário. Na filtragem primeiro o número de identificação de cada ingrediente é encontrado, em seguida é percorrido a lista de todas as receitas e verificado se cada ingrediente da receita está na lista de ingredientes informada (pelo número de identificação).

5.1.17 Sprint 18: Filtrando Receitas por Ingredientes 2

Iniciada em 22 de janeiro de 2019. Foi desenvolvido outro algoritmo que delegou essa responsabilidade para o PostgreSQL. Ele executa os seguintes passos:

1. Procura número de identificação (doravante abreviado para *id*) de todos os ingredientes informados;
2. Usando a tabela intermediaria entre "receita" e "ingrediente", que contém o *id* da receita e do ingrediente, seleciona o *id* de todas as receitas que tenha correspondência com algum *id* na lista de ingredientes informada;
3. Com o resultado anterior e trabalhando na mesma tabela, resgata o *id* de todas as receitas que estão na lista gerada no passo anterior e com ingredientes que não estão na lista de ingredientes. Com isso temos duas listas de *id* de receitas, uma com válidas e inválidas e outra apenas com inválidas;
4. A última consulta é realizada na tabela de receitas, filtrando por *id* que se encontre na primeira lista mas não tenha correspondente na segunda.

Um exemplo em código SQL pode ser visto em Algoritmo 1.

Foi realizado um teste de carga em ambos os algoritmos e o resultado pode ser visto na Seção 5.4

5.1.18 Sprint 19: Docker e Google Cloud Platform

Após a conclusão das tarefas programadas foi feito uma *sprint* extra para gerar as imagens Docker com todas as dependências do sistema e realizar a implantação na Google Cloud Platform e as devidas configurações para Integração Contínua (ver Seção 5.3).

5.2 GRÁFICO BURNDOWN

A Figura 16 contém o gráfico *burndown*, com a linha vermelha representando a distribuição ideal dos "pontos de desenvolvimento" estimados no período previsto e a linha azul representando os pontos desenvolvidos pelo aluno. Em um cenário ideal as duas linhas seguem juntas ou a azul fica abaixo da vermelha. O gráfico não exibe a *sprint* extra.

Algoritmo 1: Código de exemplo em SQL

```
-- Supondo que os ids de ingredientes sejam: 1, 2 e 3

SELECT *
FROM recipes
WHERE id IN (
        SELECT "recipe_id"
        FROM "ingredients_recipes"
        WHERE "ingredient_id" IN (1, 2, 3)
        GROUP BY "recipe_id"
)
AND id NOT IN (
        SELECT "recipe_id"
        FROM "ingredients_recipes"
        WHERE "ingredient_id"
        NOT IN (1, 2, 3)
        AND "recipe_id"
        IN (
                SELECT "recipe_id"
                FROM "ingredients_recipes"
                WHERE "ingredient_id" IN (1, 2, 3)
                GROUP BY "recipe_id"
        )
        GROUP BY "recipe_id"
);
```

5.3 IMPLANTAÇÃO E INTEGRAÇÃO CONTÍNUA

Para a implantação foi criado um *cluster* Kubernetes na Google Cloud Platform (Figura 17), inicialmente com duas máquinas virtuais, também chamadas de "nós" neste contexto. O Kubernetes se encarrega de distribuir os artefatos entre os "nós" de maneira transparente. Para este projeto foram usados os seguintes artefatos:

- *Ingress*. Funciona como servidor de *proxy* e *API Gateway*. Sua função é receber as requisições dos usuários e repassá-las para o cliente *web* ou servidor.
- *Deployment*. É responsável por criar instâncias do cliente *web*, servidor e PostgreSQL. É configurado com um determinado número de instâncias e garante que tenha no mínimo o valor informado. Há uma configuração para o servidor, uma para o cliente e outra para o PostgreSQL.
- *Service*. É usado para permitir acesso via rede aos servidores, ele permite que o *Ingress* faça chamadas ao cliente e servidor, além de liberar o acesso do servidor ao PostgreSQL.
- *Persistent Volume Claim*. Todos os dados em uma instância Kubernetes são voláteis e deixam de existir em atualizações ou reinicializações. Este artefato requisita uma unidade de persistência externa para persistir os dados. É usado para salvar o banco de dados do

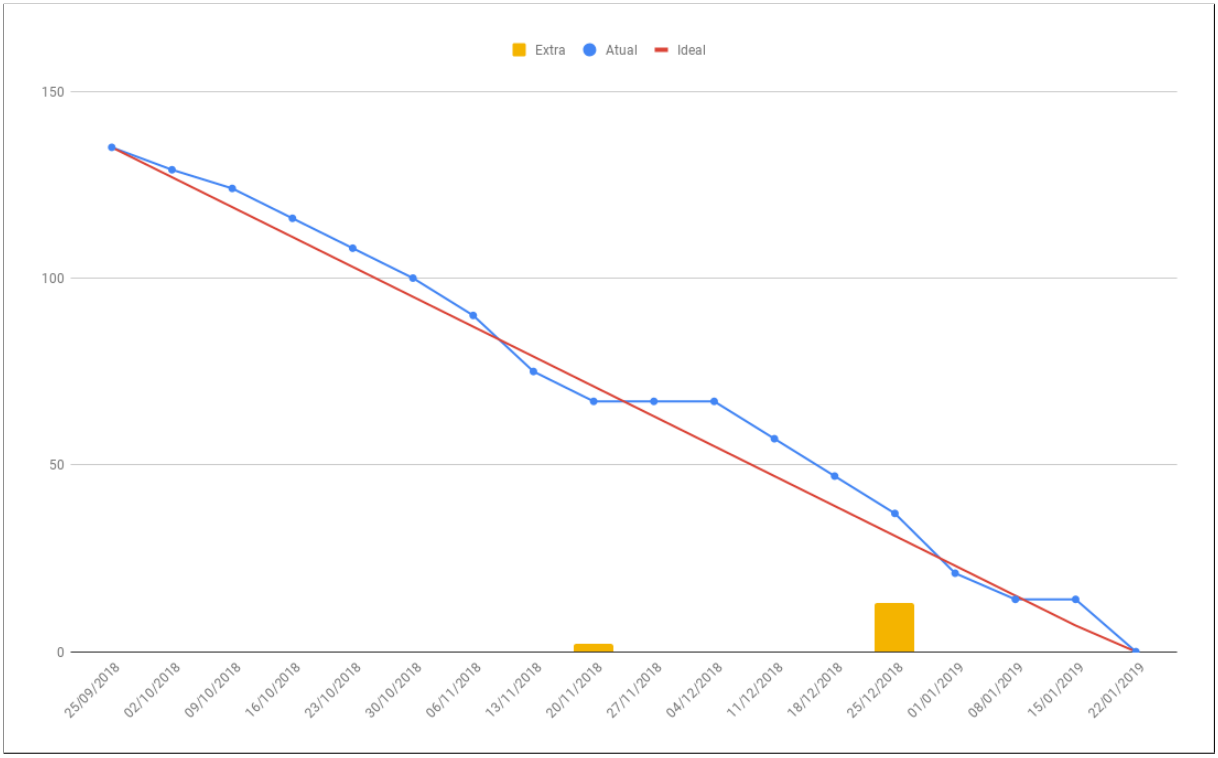

Figura 16 – Gráfico Burndown com as 18 Sprints concluídas

PostgreSQL.

- *Secrets*. É um objeto para armazenar senhas e informações sensíveis. Ele mantém a senha do banco de dados e chaves de acesso aos serviços Google Cloud Platform.

Para facilitar as atualizações do servidor e cliente *web* o Travis-CI foi configurado de forma a automatizar todo o trabalho de testes e implantação, em um processo chamado Integração Contínua. A Figura 18 exibe uma simplificação do ciclo.

Essa configuração ajuda o desenvolvedor a focar apenas no processo de desenvolvimento. Também facilita o escalonamento, pois com alguns cliques é possível aumentar a quantidade de máquinas virtuais/nós, e uma ou duas linhas de comando são suficientes para incrementar a quantidade de servidores. Usar o *Ingress* como ponto de entrada também facilita a migração de um servidor monolítico para um em microsserviços - para qualquer microsserviço adicionado basta configurar o *Ingress* para direcionar as rotas necessárias para este novo serviço.

5.4 TESTES DE DESEMPENHO

Teste de desempenho é uma parte essencial no desenvolvimento de grandes *web sites* públicos. Um dos testes de desempenho mais comum é o teste de carga, que mensura o tempo de resposta do servidor usando um software que executa requisições sintéticas no servidor (VERONA, 2016).

Para o servidor SABRE foi selecionado o aplicativo jMeter para realizar os testes

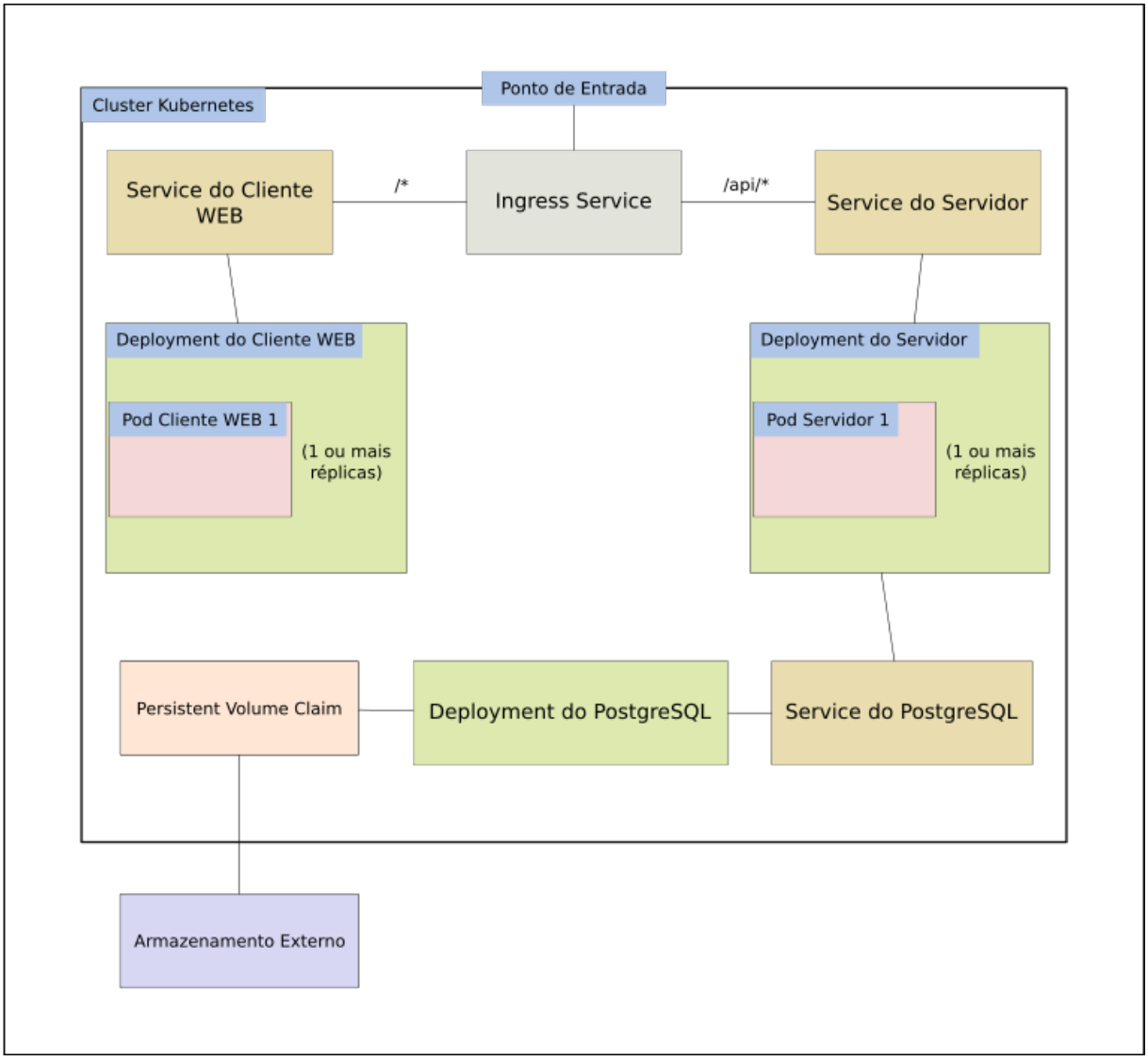

Figura 17 – Estrutura do SABRE no Kubernetes

de carga. Nele foi criado um plano de testes que inicialmente deve mensurar o tempo de resposta de listagem das receitas mais recentes, do resultado da busca simples e da busca por ingredientes. Junto com o plano de testes há uma configuração do Docker e Docker Compose para iniciar o servidor com as configurações similares ao modo de produção, mas com dados falsos e imutáveis - para cada teste ser realizado sempre no mesmo conjunto de receitas e ingredientes. Em todos os testes foi usado o mesmo conjunto de 13 ingredientes e 8.191 receitas - resultado da combinação de todos os 13 ingredientes em conjuntos de 1 a 13 sem repetição. A quantidade de ingredientes e receitas foi escolhida para popular o banco de dados da máquina de teste em poucos minutos (entre dois e três minutos).

Os resultados foram obtidos utilizando um computador com Linux Manjaro 64 bits (especificações em Quadro 2).

Para testar a listagem de receitas recentes foram usadas 5 *threads* (simulando 5

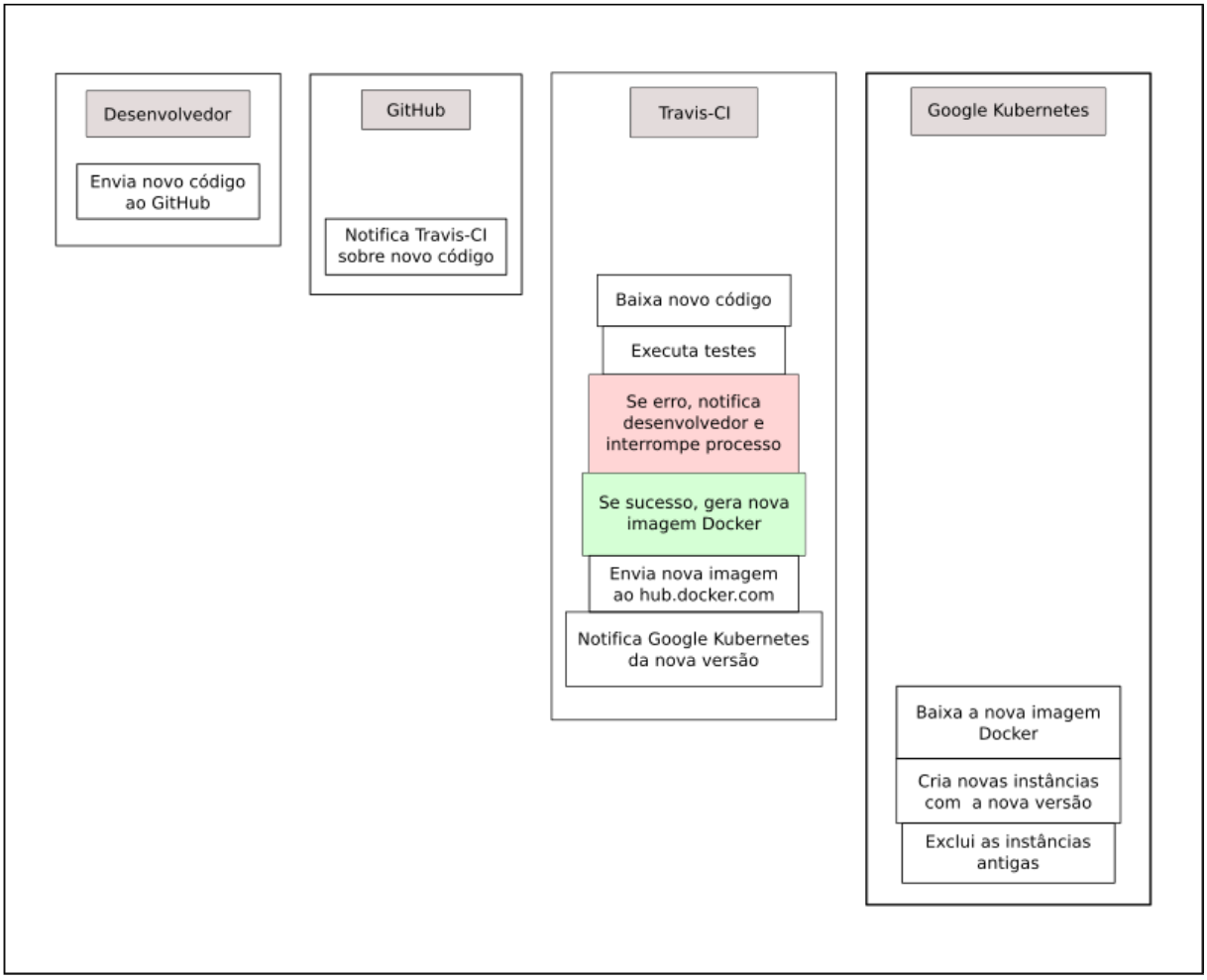

Figura 18 – Integração Contínua

Processador	Intel i7-6700K @4.00GHz
Memoria	16GB @2400Mhz
Unidade de armazenamento	Ssd SATA Crucial MX300 275gb
Docker	18.09.0-ce
Ruby	2.5.3
PostgreSQL	9.6
jMeter	5.1.1 r1853635
Kernel	4.19.13-1-MANJARO

Quadro 2 – Especificações do Computador de Testes.

usuários) realizando 1.000 requisições. O teste durou 1 minuto e 19 segundos para executar as 5.000 requisições, não obteve erros do servidor e teve um tempo médio de resposta de 140 milissegundos. O resultado obtido pode ser visto em Figura 19.

No teste de busca simples também foram usadas 5 *threads* realizando 1.000 requisições. Para cada requisição foi usado um ingrediente de uma lista de 13 ingredientes usados para popular o banco de dados, garantindo que cada requisição tenha algum retorno (pois a busca

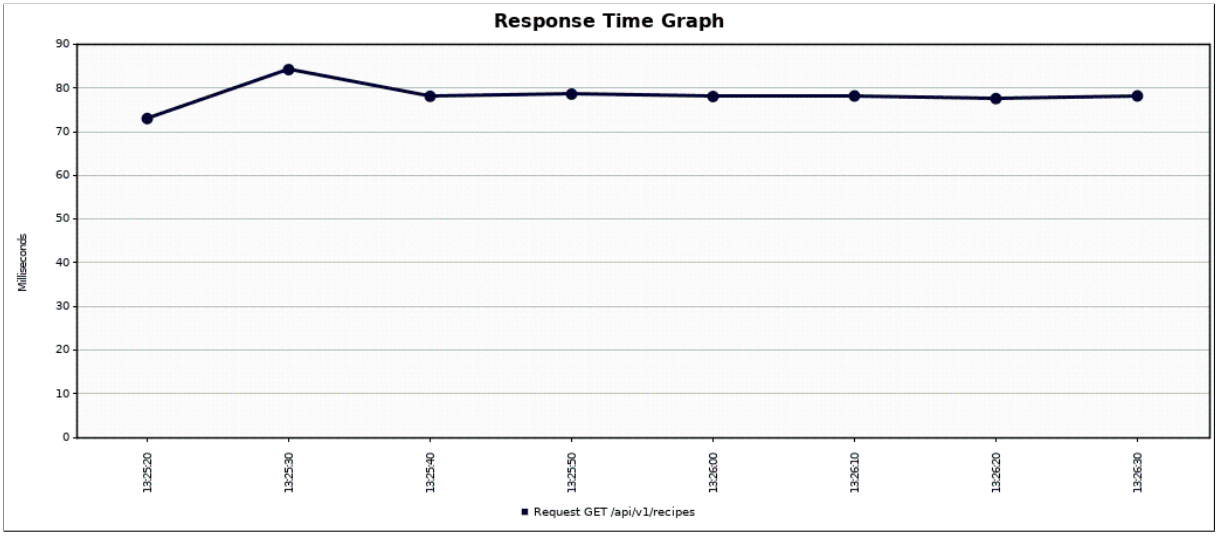

Figura 19 – Listagem de Receitas

simples leva em consideração o nome e os ingredientes da receita). O teste levou 6 minutos e 49 segundos para efetuar 5 mil requisições, com tempo de resposta variando de 310 até 640 milissegundos e não obteve erros. O resultado pode ser visto em Figura 20.

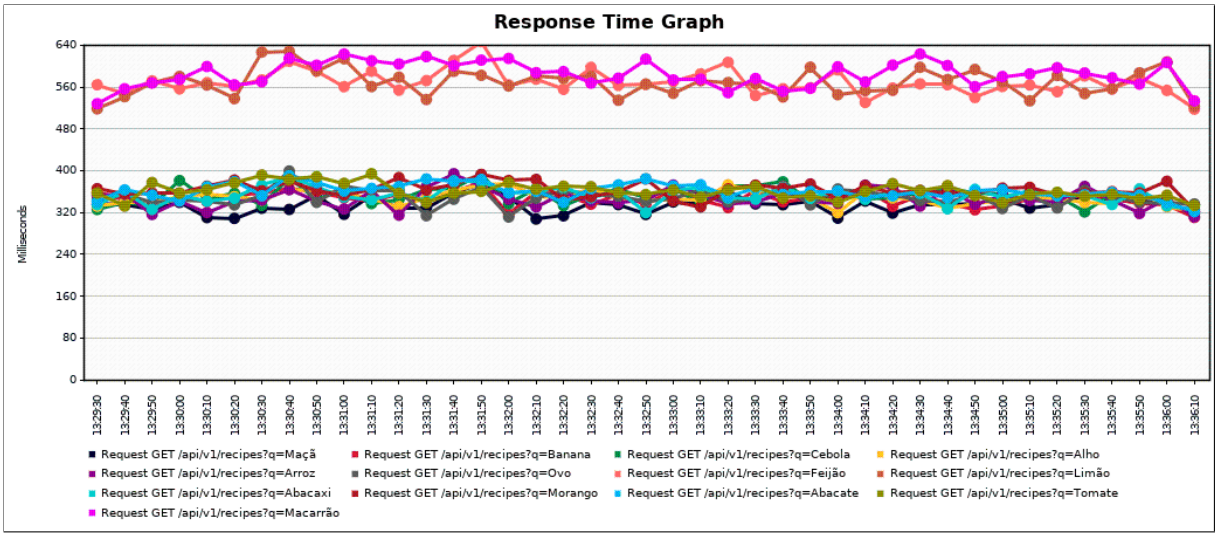

Figura 20 – Busca Simples

O teste de busca por ingredientes usando o algoritmo que relaciona todas as receitas na camada de aplicação levou 8 horas, 37 minutos e 19 segundos para efetuar 3.420 requisições. O teste deveria realizar 5.000 requisições, mas foi cancelado devido a grande demora na execução (mais de oito horas). Foi usado uma lista com cinco termos de busca, contendo de dois a quatro ingredientes. Pelo gráfico Figura 21 é observável que as pesquisas levavam entre 40 e 60 segundos, tempo proibitivo para usar em produção.

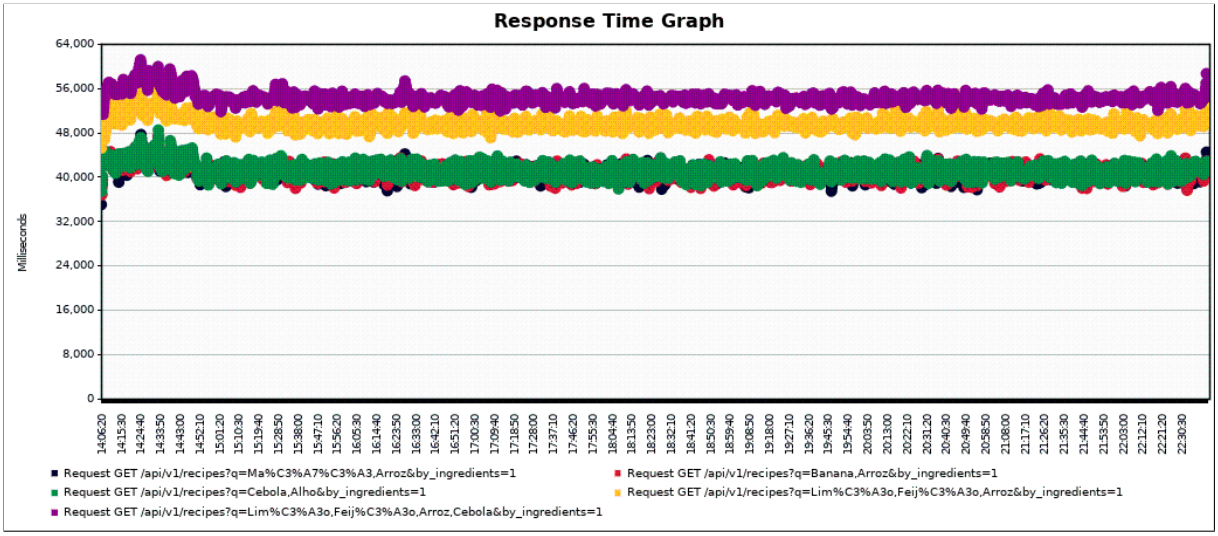

Figura 21 – Busca por Ingredientes na Camada de Aplicação

Um segundo teste foi feito na busca por ingredientes, desta vez testando o segundo algoritmo, que delega para o gerenciador de banco de dados fazer o relacionamento. O teste durou 18 minutos e 31 segundos para realizar as 5 mil requisições, usando os mesmos ingredientes da pesquisa anterior, com um tempo de resposta entre 90 e 130 milissegundos. Não houveram erros no servidor. Embora tenha um tempo de resposta menor que a busca simples, ele demorou mais para realizar todas as requisições. Este fato curioso precisa ser investigado no futuro. O resultado pode ser visto na Figura 22.

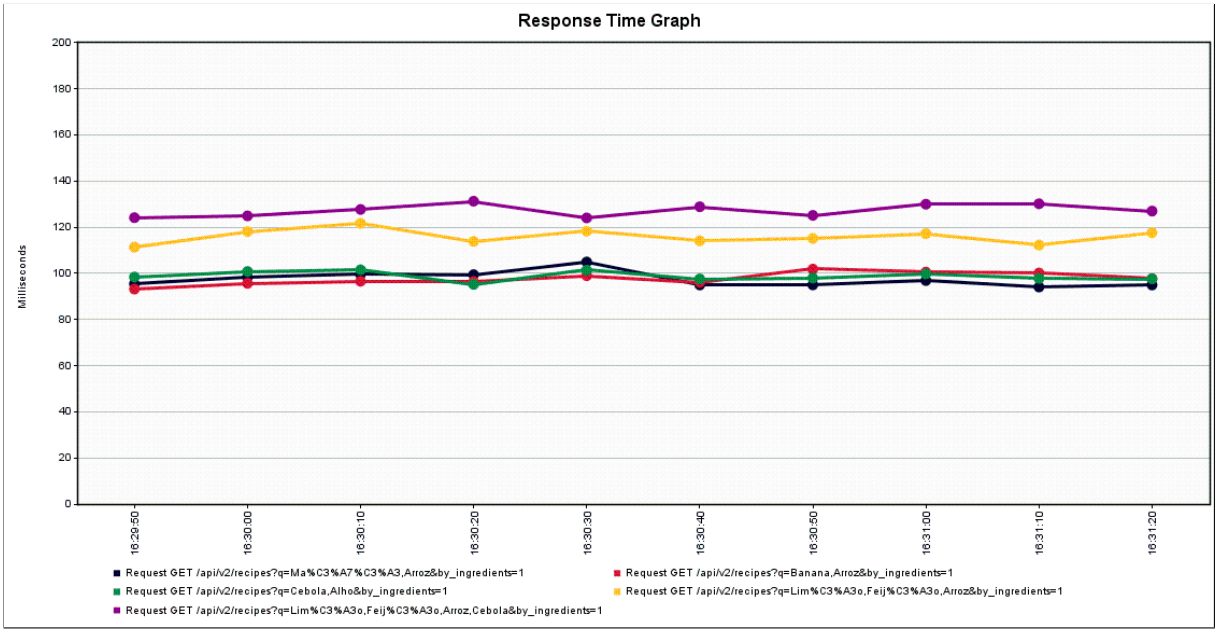

Figura 22 – Busca por Ingredientes na Camada de Banco de Dados

O último teste foi realizado simulando cinco pessoas executando simultaneamente os

mesmos testes anteriores (com exceção da busca por ingredientes na camada de aplicação). O teste durou 22 minutos e 43 segundos para realizar 15 mil requisições sem erros, com tempo de resposta entre 150 e 850 milissegundos. O Resultado pode ser visto na Figura 23.

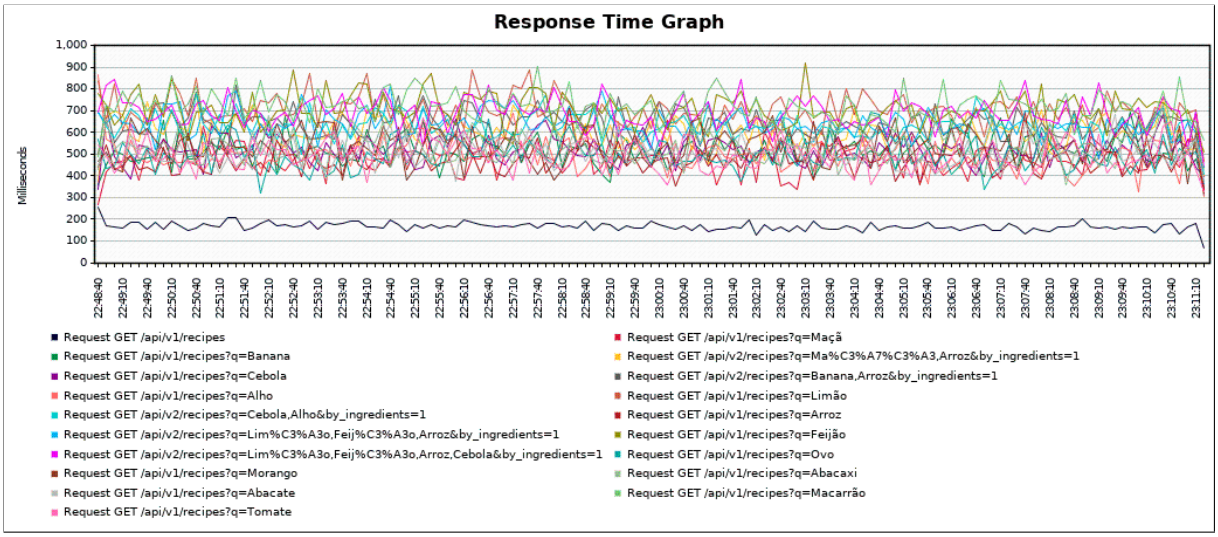

Figura 23 – Listagem, Busca Simples e Busca por Ingredientes

Estes testes cobrem apenas a listagem e busca de receitas. Para o futuro é necessário testar o desempenho de todas as possíveis interações com o servidor. Estes testes devem ser realizados a cada interação no processo de Integração Contínua, para que seja possível verificar o impacto que cada alteração terá no desempenho.

Também é necessário realizar os testes em diversas máquinas virtuais oferecidas no Google Cloud, para ter dados concretos de como a aplicação funcionará em cada configuração, facilitando a escolha do melhor custo-beneficio ou se é necessário migrar para outro serviço de nuvem.

É importante salientar que a máquina onde os testes formam realizadas não era dedicada a testes, tendo diversos processos do próprio sistema sendo executado em segundo plano que podem alterar o resultado. O PostgreSQL e jMeter também estavam funcionando na mesma máquina do servidor, disputando os mesmos recursos e também podendo alterar o resultado. A escolha de 5 *threads*/usuários foi feita para combinar com os 5 *workers*/processos da configuração padrão do servidor, no futuro é importante verificar com outras combinações.

6 CONSIDERAÇÕES FINAIS

Este trabalho levou em consideração o problema de desperdício de alimentos e suas consequências, propondo o desenvolvimento do SABRE. Uma ferramenta que pode auxiliar o consumidor a utilizar os ingredientes que já possui em casa, reduzindo desperdícios. Também apresenta o método de desenvolvimento, tecnologia utilizada e os resultados obtidos.

Vale salientar que o maior desafio para este projeto foi definir o algoritmo que procurará as receitas, já que é crucial para o desenvolvimento e tem o potencial de aumentar consideravelmente o processamento nos servidores, portanto deve ser analisado minuciosamente. Durante o desenvolvimento foram experimentadas algumas possibilidades, mas há margem para melhorias. Para o futuro outras possibilidades devem ser analisadas, como delegar para o cliente esse relacionamento ou desenvolver um sistema exclusivamente para tratar desse relacionamento.

Embora todas as funcionalidades propostas neste documento tenham sido implementadas, antes de liberar para o público devem ser realizados testes de usabilidade para validar o *design* do cliente *web* - a funcionalidade existe, mas as pessoas saberão como usar? Também é necessário adicionar mais testes no cliente e testes de carga nas diversas rotas do servidor. Também é interessante testar o servidor com implementações diferentes do Ruby 2.5.3, podendo avaliar o desempenho com jRuby, Rubinous e Ruby 2.6.

Referências

AGILEALLIANCE. **What is Test Driven Development (TDD)?** 2015. Disponível em: <https://www.agilealliance.org/glossary/tdd/>. Acesso em: 26 de agosto de 2018. Citado na página 14.

AGILEMODELING. User Stories: An Agile Introduction. 2013. Disponível em: <http://www.agilemodeling.com/artifacts/userStory.htm>. Citado na página 14.

AUDIT bundler. **bundler-audit**. 2019. Disponível em: <https://github.com/rubysec/bundler-audit>. Citado na página 10.

AXIOS. **Axios**. 2018. Disponível em: <https://github.com/axios/axios/>. Citado na página 11.

BRAKEMAN. **Brakeman**. 2019. Disponível em: <https://github.com/presidentbeef/brakeman>. Citado na página 9.

BULLET. **Bullet**. 2019. Disponível em: <https://github.com/flyerhzm/bullet>. Citado na página 9.

CYBERCOOK. **CyberCook**. 2018. Disponível em: <https://cybercook.uol.com.br/>. Acesso em: 19 de agosto de 2018. Citado na página 4.

DOCKER. **Overview of Docker Compose**. 2019. Disponível em: <https://docs.docker.com/compose/overview/>. Citado na página 11.

DOCKER. **What is a Container**. 2019. Disponível em: <https://www.docker.com/resources/what-container>. Citado na página 11.

ESLINT. **About - ESLint**. 2018. Disponível em: <https://eslint.org/docs/about/>. Citado na página 11.

FAO. **Global food losses and food waste**. Rome: [s.n.], 2011. 37 p. Disponível em: <http://www.fao.org/docrep/014/mb060e/mb060e.pdf>. Acesso em: 19 de agosto de 2018. Citado na página 1.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. Tese (PhD in Computer Science) — UNIVERSITY OF CALIFORNIA, 2000. Citado na página 8.

FOWLER, M. The New Methodology. **martinfowler.com**, 2005. Disponível em: <https://martinfowler.com/articles/newMethodology.html>. Citado na página 13.

G1. Pão mais antigo do mundo tem receita de 14 mil anos revelada por arqueólogos. 2018. Disponível em: <https://g1.globo.com/olha-que-legal/noticia/pao-mais-antigo-do-mundo-tem-receita-de-14-mil-anos-revelada-por-arqueologos.ghtml>. Acesso em: 19 de agosto de 2018. Citado na página 1.

GIT. **Git**. 2018. Disponível em: <https://git-scm.com/>. Citado na página 11.

GITHUB. **The world's leading software development platform · GitHub**. 2018. Disponível em: <https://github.com/>. Citado na página 11.

GOOGLE. **Google Cloud**. 2019. Disponível em: <https://cloud.google.com/>. Citado na página 12.

GSHOW. **Receitas Gshow**. 2018. Disponível em: <https://gshow.globo.com/receitas-gshow/>. Acesso em: 19 de agosto de 2018. Citado na página 5.

GUARD. **Guard**. 2019. Disponível em: <https://github.com/guard/guard>. Citado na página 10.

JMETER, A. **Apache JMeter**. 2019. Disponível em: <https://jmeter.apache.org/>. Citado na página 10.

JSON.ORG. **JSON**. 2018. Disponível em: <www.json.org/>. Citado na página 8.

KARMA. **Karma - How It Works**. 2018. Disponível em: <https://karma-runner.github.io/3.0/intro/how-it-works.html/>. Citado na página 11.

KUBERNETES. **Kubernetes**. 2019. Disponível em: <https://kubernetes.io/>. Citado na página 12.

LIPINSKI brian et al. Reducing food loss and waste. 2013. Disponível em: <http://www.wri.org/publication/reducing-food-loss-and-waste>. Acesso em: 19 de agosto de 2018. Citado na página 1.

MOZILLA. **About JavaScript**. 2018. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript/>. Citado na página 10.

MYFRIDGEFOOD. **MyFridgeFood**. 2019. Disponível em: <http://myfridgefood.com>. Acesso em: 2 de abril de 2019. Citado na página 7.

MYRECIPES. **MyRecipes**. 2019. Disponível em: <https://www.myrecipes.com/ingredients>. Acesso em: 2 de abril de 2019. Citado na página 6.

PALMIRINHA, V. **Vovó Palmirinha**. 2018. Disponível em: <https://www.vovopalmirinha.com.br/>. Acesso em: 19 de agosto de 2018. Citado na página 4.

PGSEARCH. **pg_search**. 2018. Disponível em: <https://github.com/Casecommons/pg_search/>. Citado na página 9.

POSTGRESQL. **PostgreSQL: About**. 2018. Disponível em: <https://www.postgresql.org/about/>. Citado na página 8.

RAILSGUIDES. **Getting Started with Rails**. 2018. Disponível em: <https://guides.rubyonrails.org/getting_started.html#creating-articles/>. Citado na página 9.

REEK. **Reek**. 2019. Disponível em: <https://github.com/troessner/reek>. Citado na página 9.

RSPEC. **RSpec**. 2018. Disponível em: <http://rspec.info/>. Citado na página 9.

RUBOCOP. **RuboCop**. 2018. Disponível em: <https://github.com/rubocop-hq/rubocop/>. Citado na página 9.

RUBY. **About Ruby**. 2018. Disponível em: <https://www.ruby-lang.org/en/about/>. Citado na página 9.

SCHWABER, K.; SUTHERLAND, J. The scrum guide. 2017. Disponível em: <https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf>. Acesso em: 26 de agosto de 2018. Citado na página 13.

SUPERCOOK. **SuperCook**. 2019. Disponível em: <https://www.supercook.com/#/recipes>. Acesso em: 2 de abril de 2019. Citado na página 6.

TRACEROUTE. **Traceroute**. 2019. Disponível em: <https://github.com/amatsuda/traceroute>. Citado na página 10.

TRAVIS-CI. **Travis CI - Test and Deploy with Confidence**. 2019. Disponível em: <https://travis-ci.org/>. Citado na página 12.

TRELLO. **Trello**. 2018. Disponível em: <https://trello.com/>. Citado na página 12.

TUDOGOSTOSO. **TudoGostoso**. 2018. Disponível em: <https://www.tudogostoso.com.br>. Acesso em: 19 de agosto de 2018. Citado na página 3.

VERONA, J. **Practical DevOps**. 1. ed. Birmingham: Packt Publishing, 2016. Citado na página 27.

VUE. **Introduction: What is Vue.js?** 2018. Disponível em: <https://vuejs.org/v2/guide/>. Citado na página 10.

W3C. **HTML & CSS**. 2018. Disponível em: <https://www.w3.org/standards/webdesign/htmlcss/>. Citado na página 10.

WRI. What's Food Loss and Waste Got to Do with Climate Change? A Lot, Actually. 2015. Disponível em: <https://www.wri.org/blog/2015/12/whats-food-loss-and-waste-got-do-climate-change-lot-actually>. Acesso em: 19 de agosto de 2018. Citado na página 1.

Apêndices

APÊNDICE A – Protótipos de Componentes Visuais

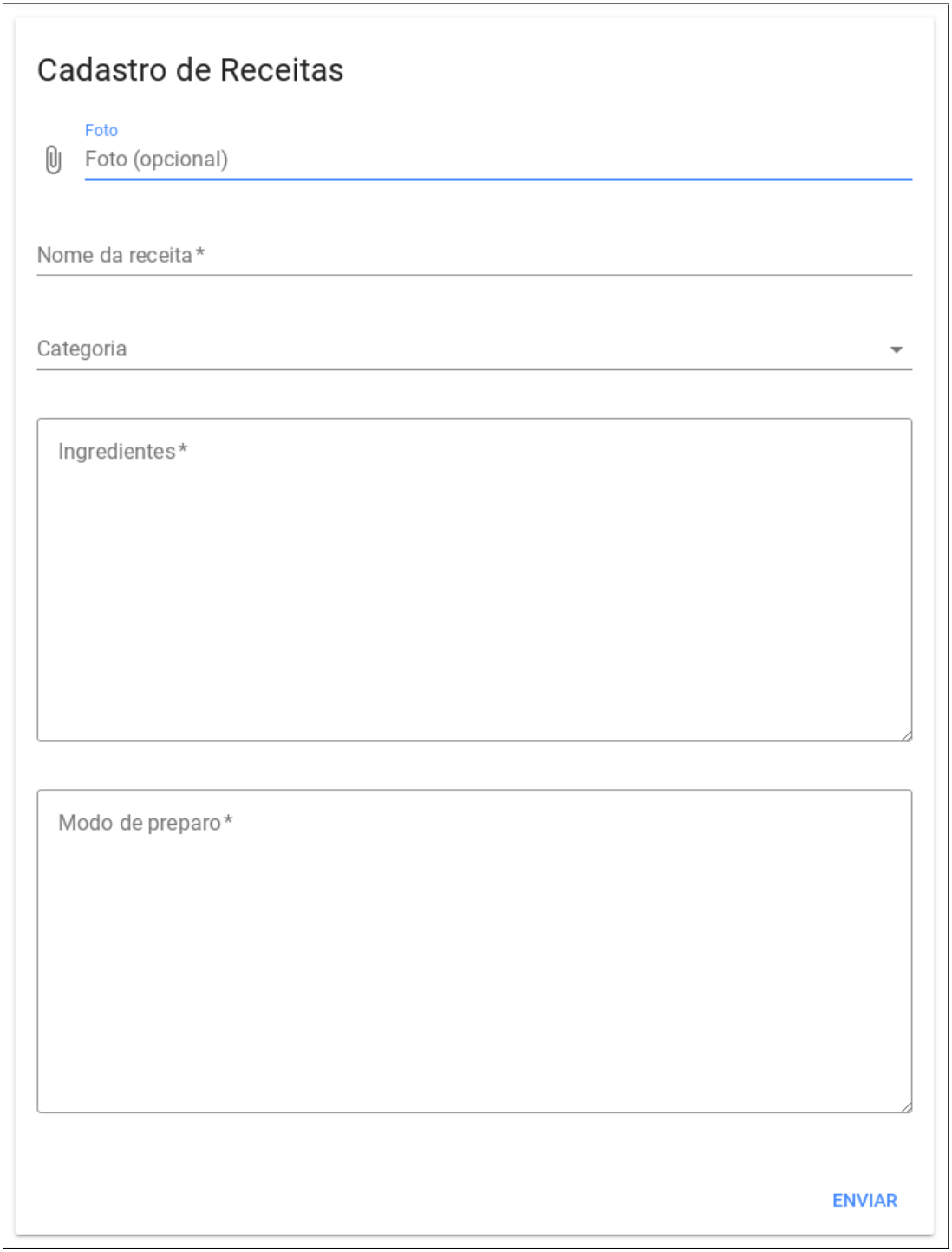

Figura 24 – Formulário de envio de receitas

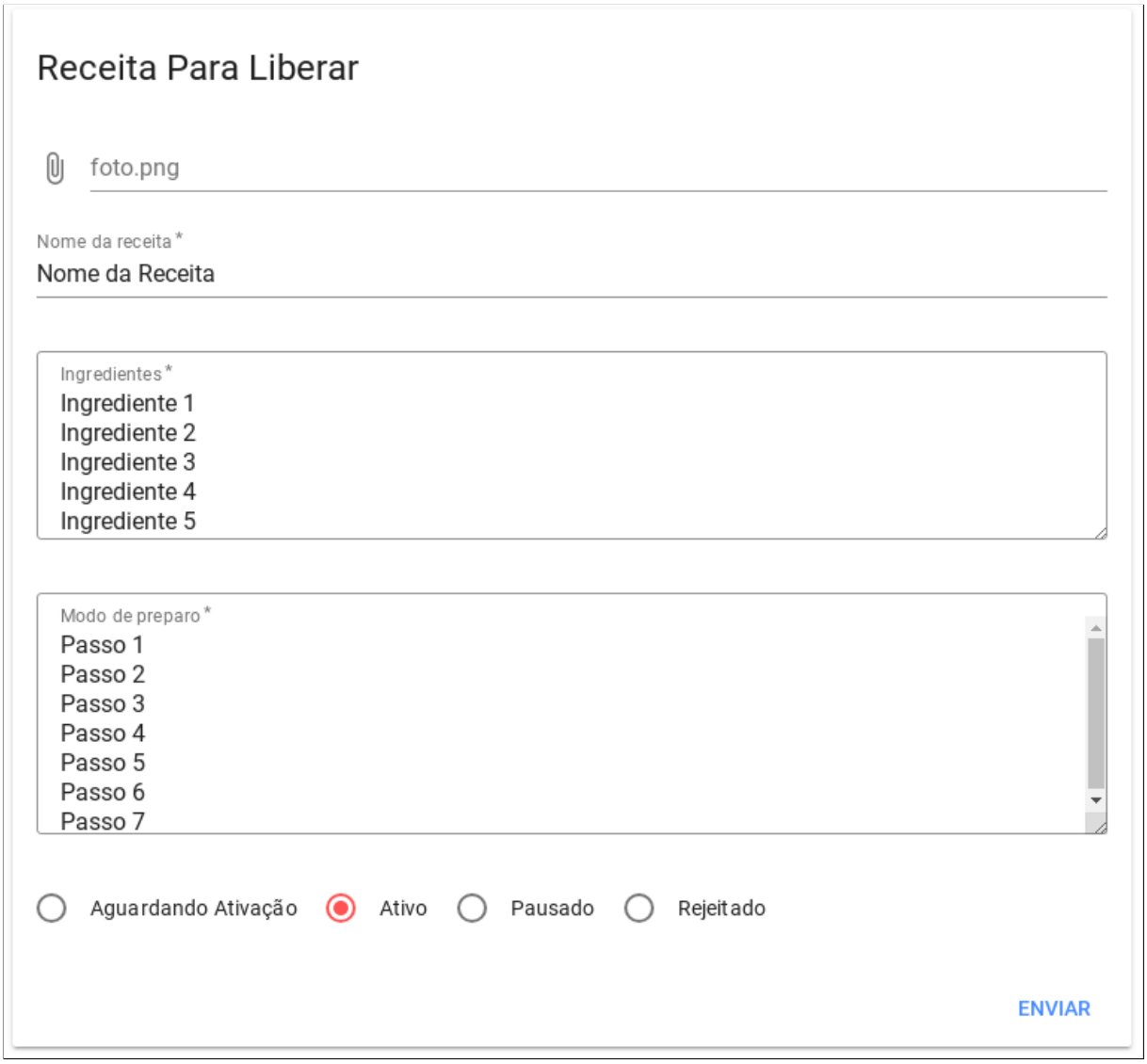

Figura 25 – Editar receita para aprovar (usuário administrativo)

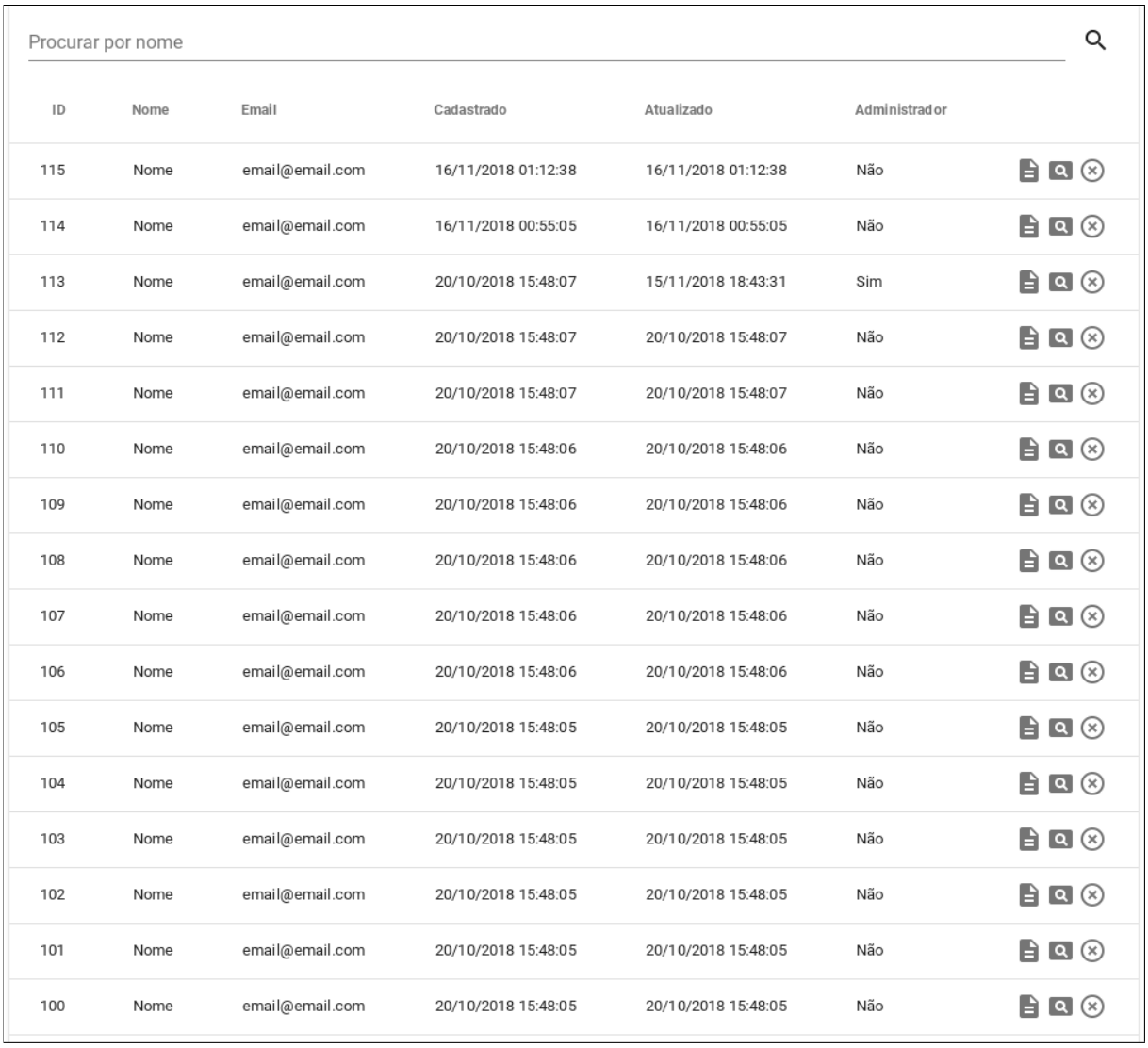

Figura 26 – Listagem de usuários (usuário administrativo)

Figura 27 – Formulário de cadastro

Figura 28 – Formulário de login

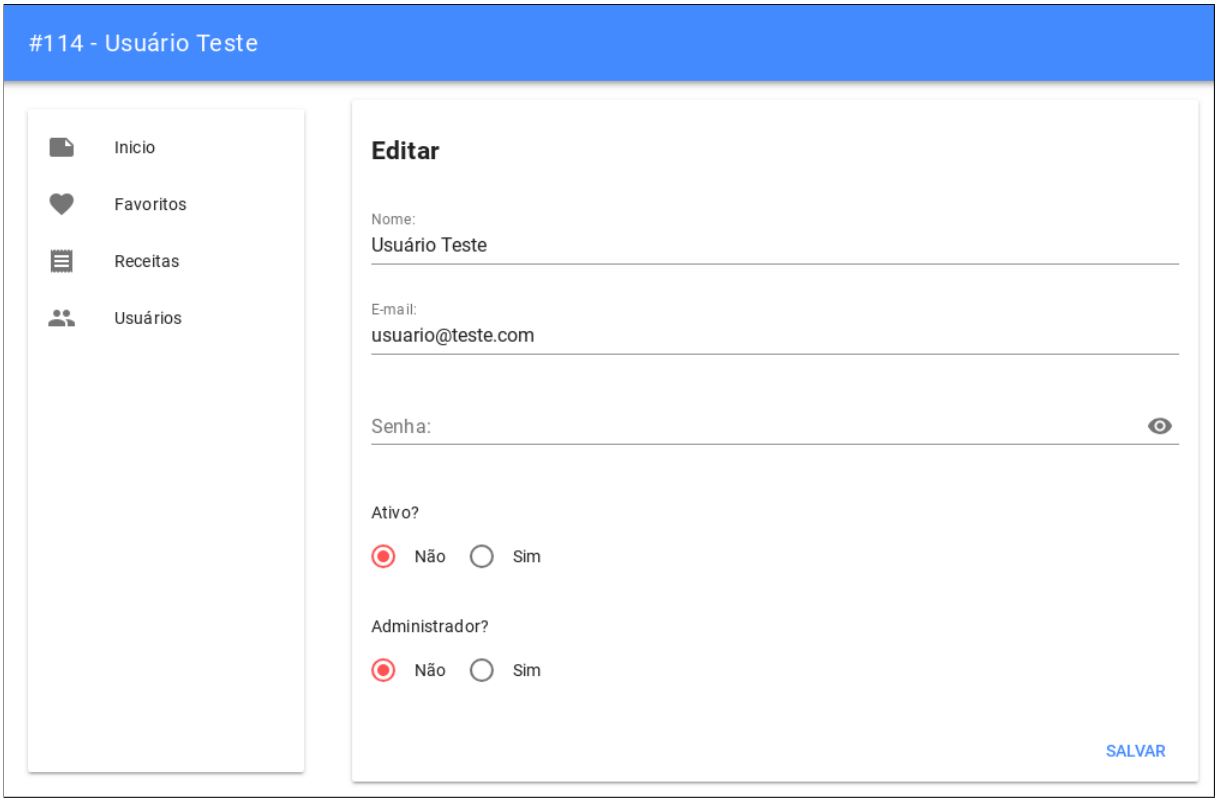

Figura 29 – Formulário de edição de usuário (usuário administrativo)