UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GIUSEPPE GUERRA MONGIOVI

CHEFASSISTANT:
UMA FERRAMENTA DE AUXÍLIO À CULINÁRIA

CAMPINA GRANDE - PB

2019

GIUSEPPE GUERRA MONGIOVI

CHEFASSISTANT:
UMA FERRAMENTA DE AUXÍLIO À CULINÁRIA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Professora Dra. Melina Mongiovi Cunha Lima Sabino.

CAMPINA GRANDE - PB

2019

M743c Mongiovi, Giuseppe Guerra.
ChefAssistant : uma ferramenta de auxílio à culinária. / Giuseppe Guerra Mongiovi. – 2019.

10 f.

Orientadora: Profa. Dra. Melina Mongiovi Cunha Lima Sabino.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Culinária - aplicativo. 2. Ferramenta online - culinária. 3. Desenvolvimento de software. 4. Algoritmo de recomendação. I. Sabino, Melina Mongiovi Cunha Lima. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

GIUSEPPE GUERRA MONGIOVI

CHEFASSISTANT:
UMA FERRAMENTA DE AUXÍLIO À CULINÁRIA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professora Dra. Melina Mongiovi Cunha Lima Sabino
Orientadora – UASC/CEEI/UFCG

Professor Dr. Wilkerson de Lucena Andrade
Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni
Examinador – UASC/CEEI/UFCG

Trabalho aprovado em: 25 de novembro 2019.

CAMPINA GRANDE - PB

ChefAssistant: uma ferramenta de auxílio à culinária

Trabalho de Conclusão de Curso

Giuseppe Guerra Mongiovi
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
giuseppe.mongiovi@ccc.ufcg.edu.br

RESUMO

Uma dúvida muito comum no dia a dia das pessoas é qual receita preparar com os alimentos disponíveis. Ao enfrentar essa dúvida, ao invés de procurar alternativas, muitos acabam optando por algo repetitivo ou não têm ideia do que preparar. Com o propósito de resolver esse problema, o objetivo desta proposta é desenvolver o ChefAssistant: uma ferramenta online que sugere receitas com base nas preferências do usuário e nos alimentos selecionados pelo mesmo. O uso dessa ferramenta resultará em refeições mais diversificadas e na expansão do conhecimento culinário.

1. INTRODUÇÃO

É de conhecimento geral que comer muitas refeições fora de casa não é a melhor escolha. Muitas vezes sai caro e os alimentos escolhidos tendem a não ser tão bons para saúde. Também tem o fato de que os ingredientes presentes nas refeições e o modo de preparo são freqüentemente desconhecidos.

Um indivíduo é mais propenso a comer mais em restaurantes do que quando ele prepara sua própria refeição. Muitos restaurantes oferecem brindes ou promoções irresistíveis, além de grandes pratos e acompanhamentos que têm porções muito grandes. Isso confunde o cérebro e o corpo, fazendo você pensar que ainda está consumindo uma porção, porque é isso que lhe foi servido, mas a realidade é que o tamanho das porções são muito maiores do que deveriam ser. Um estudo publicado no *European Journal of Clinical Nutrition* [1] afirmou que os indivíduos que saem para comer acabam consumindo em média 200 calorias a mais do que aqueles que comem em casa.

Os alimentos nos restaurantes, particularmente em *fast foods*, tendem a ter muito mais gorduras saturadas e trans do que as refeições que você pode preparar. De acordo com a *Everyday Health* [2], os especialistas dizem que para melhorar a sua saúde, você deve tentar diminuir a quantidade de gorduras que você consome na dieta, em vez do próprio colesterol, já que uma dieta rica em gordura eleva os níveis de colesterol.

Com isso, para poupar dinheiro e preservar a saúde, é necessário aumentar a quantidade de refeições preparadas em casa. Diante de uma diversidade ou até mesmo da ausência de alguns ingredientes na cozinha, surgem dúvidas sobre o que preparar para as refeições diárias. Essas dúvidas acabam surgindo principalmente para o público com restrições alimentares, tais como os veganos e vegetarianos. A partir disso, a descoberta de novas receitas se torna essencial para expandir os conhecimentos culinários e até mesmo para melhorar a alimentação do dia a dia.

Já existe ferramentas similares ao que deseja-se implementar, como por exemplo o Supercook [3]. Porém, diferentemente do ChefAssistant, não é possível adicionar receitas à lista de ignorados, uma funcionalidade que ajuda bastante a melhorar o sistema de recomendações para um usuário que utiliza a ferramenta frequentemente.

2. SOLUÇÃO

O ChefAssistant é uma ferramenta *web single page* criada com o intuito de recomendar receitas com base nos ingredientes informados pelo usuário. É possível selecionar filtros para restringir o escopo da busca para indivíduos com restrições alimentares como veganos, vegetarianos, intolerantes a lactose e entre outros.

A ferramenta foi desenvolvida em inglês pelo fato de que não encontrou-se uma base de dados em português tão completa como a que escolheu-se utilizar. A base de dados foi escolhida tanto pela riqueza de informações, como instruções detalhadas de preparo e dez possíveis filtros, quanto pelo seu tamanho, que consiste em aproximadamente 24 mil receitas e 240 mil ingredientes.

2.1 Funcionalidades

Ao acessar a página web da ferramenta, o usuário poderá escolher um ingrediente principal, vários ingredientes secundários e filtros (Figura 5). Com base no que o usuário preenche, uma lista de receitas é exibida para o mesmo (Figura 6).

Assim como diversas ferramentas, o ChefAssistant possui a opção de cadastro de usuário, que é de suma importância visto que essa funcionalidade abre portas para outras duas que são extremamente

úteis. A primeira tem como função salvar receitas na lista de favoritos e dessa forma poder criar seu próprio livro virtual de receitas. Já a segunda tem a finalidade de melhorar as recomendações, visto que com ela é possível adicionar receitas a uma lista de ignorados.

2.2 Arquitetura

A arquitetura de uma aplicação web define as interações entre aplicativos, middlewares e bancos de dados para garantir que vários aplicativos possam trabalhar em conjunto. Os principais atores da arquitetura utilizada são o *back-end* (servidor) e *front-end* (cliente). O servidor é responsável por receber, processar e devolver requisições, assim como manipular o banco de dados que tem como funções estruturar, criar, remover e editar dados. Já o cliente é responsável por por exibir uma interface visual para que o usuário possa interagir e, dessa forma, utilizar as funcionalidades disponíveis na aplicação.

2.2.1 Tecnologias do back-end

O servidor foi implementado utilizando Node.js juntamente com o *framework* Express. Essa tecnologia foi escolhida por diversos motivos, como, por exemplo, o fato de ser de código aberto, fácil de configurar e por ser na linguagem javascript, tornando mais agradável o desenvolvimento em conjunto com o cliente, que também utiliza javascript.

Para atender a necessidade de persistência de dados, utilizou-se MongoDB, um banco de dados orientado a documentos de plataforma múltipla e de código aberto, um tipo de banco de dados NoSQL. Como um banco de dados NoSQL, o MongoDB evita a estrutura baseada em tabela do banco de dados relacional para adaptar documentos semelhantes a JSON que possuem esquemas dinâmicos que ele chama de BSON.

2.2.2 Estrutura do back-end

Similar a grande parte das aplicações REST feitas em Node.js, o *back-end* foi estruturado em tal forma que dividiu-se em cinco diretórios principais, sendo esses *controllers, helpers, models, routes* e *workers* como pode ser visto na Figura 1.

Os *controllers* são responsáveis por receber as entradas da rota e invocar as ações apropriadas a serem executadas. Cada *controller* possui mais de uma rota e comunica-se com *models* e/ou outros *controllers*, realizando que ações distintas de leitura e/ou escrita.

Os *routes* controlam quais solicitações cada *controller* recebe e, dessa forma, definem os *Endpoints* utilizados pelo cliente. Os *Routes* na API REST são representados por URIs e são todos únicos. Os parâmetros podem ser passados como *Query Parameter* ou como *Path Parameter* e os dados podem ser passados através do *body* de alguns tipos de requisições, como POST, PATCH e PUT.

São os *models* que definem a estrutura dos objetos a serem manipulados e armazenados pelo banco de dados, que neste projeto são *User, Recipe* e *Ingredient*. Eles também definem os tipos de cada atributo e se eles são obrigatórios ou não na criação do objeto (Figura 2).

Figura 1 - Organização estrutural dos diretórios

```
const ingredientSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    name: {
        type: String,
        required: true
    },
    label: {
        type: String,
        required: true
    },
    recipeId: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'Recipe'
    },
})
```

Figura 2 - Modelo de um ingrediente

Já os *helpers* servem para modularizar melhor o código e, em alguns casos, evitar repetição de código. Foram criados três *helpers*, sendo o primeiro responsável por realizar a conexão com o banco de dados através da biblioteca Mongoose. Já o segundo é responsável por, assim como seu nome diz, tratar os erros. Por

fim, o último é responsável por definir, através de expressões regex, quais rotas são públicas e, portanto, não necessitam de autenticação (Figura 3).

```
function jwt() {
    const secret = config.secret;
    return expressJwt({ secret, isRevoked }).unless({
        path: [
            // public routes that don't require authentication
            /^\/users\/authenticate/,
            /^\/users\/register/,
            /^\/ingredients\/similarIngredients/,
            /^\/ingredients\/.*/,
            /^\/recipes\/.*/,
            /^\/recipes\/recommendedRecipes/,
        ]
    });
}
```

Figura 3 - Função que define as rotas públicas

Apesar do Node.js ser de thread único, é possível, por meio de *worker threads*, gerar múltiplas threads que executam em paralelo para delegar trabalho da *thread* principal e, consequentemente, mantê-la livre para processar novas solicitações mais rapidamente.

2.2.3 *Autenticação*

Utilizou-se JWT (*Json Web Tokens*) para autenticação de usuários e proteção de determinadas rotas. O JWT define uma maneira compacta e independente de transmitir informações com segurança entre cliente e servidor através de um objeto JSON. Essas informações podem ser verificadas e confiáveis porque são assinadas digitalmente. Cada token possui um tempo de validade e a informação de identificação do usuário a quem pertence.

O fluxo funciona da seguinte forma: primeiramente o usuário, através do cliente, insere seu nome de usuário e senha; em seguida o *back-end* recebe e valida as informações e, caso estejam corretas, gera um token referente ao usuário com um prazo de validade; por último, o token é enviado para o cliente que o armazena para utilizá-lo no campo *Authorization* do *Header* de futuras chamadas referentes a rotas protegidas.

2.2.4 *Tecnologias do front-end*

O cliente foi implementado utilizando React, uma biblioteca JavaScript para criar interfaces de usuário. É mantido pelo Facebook e por uma comunidade de desenvolvedores e empresas individuais. Essa tecnologia foi escolhida pelos seguintes motivos: maior domínio devido a experiências prévias em outros projetos já desenvolvidos; fornece uma melhor modularização do código por possuir uma estrutura baseada em componentes reusáveis, cada um com sua própria lógica interna; possui um *Virtual DOM* que é mantido em memória para acelerar a renderização.

Grandes aplicativos possuem cada vez mais estados, sendo vários deles provavelmente mais complexos, portanto gerenciá-los fica cada vez mais complicado à medida que o aplicativo cresce. Com isso, considerando um possível crescimento futuro do sistema, utilizou-se Redux, uma biblioteca JavaScript para gerenciar estados de uma aplicação. A mesma é fortemente conectada com React pois seu criador foi Dan Abramov que trabalha no Facebook, empresa que inventou React.

2.2.5 *Estrutura do front-end*

Como pode ser visto na Figura 4, o cliente foi dividido em duas maiores categorias. A primeira é o *Store*, contendo *Actions* e *Reducers* para gerenciar os estados. Os *reducers* são funções puras que recebem um estado e ação e retornam um novo estado. Já os *Actions* são responsáveis tanto por estabelecerem uma conexão com o servidor, definindo os *Endpoints, Headers, Body* e tipo de chamada, quanto por tratar a resposta retornada pelo servidor.

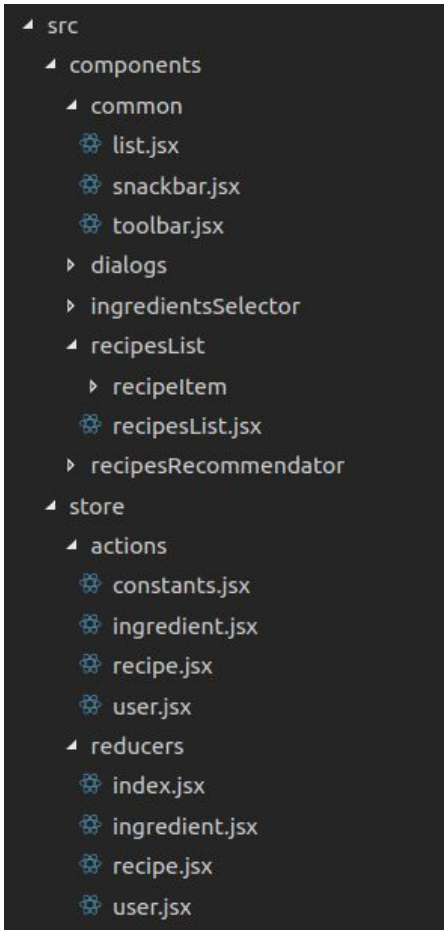

Figura 4 - Organização estrutural dos diretórios

Na segunda categoria estão todos os componentes. Na pasta common encontram-se os componentes que são utilizados diversas vezes por outros componentes como *Toolbar, Snackbar* e

List. Todos os dialogs, como por exemplo o de login e registro, estão presentes em uma única pasta nomeada *dialogs*. Há também um componente responśavel por permitir que o usuário preencha informações como ingredientes, ingrediente principal e filtros, para em seguida obter a recomendação de receitas (Figura 5) e outro por exibir uma lista de receita (Figura 6). Por último, tem-se o componente principal que é o recomendador de receitas, contendo dele o componente responsável por (Junção da Figura 5 com a Figura 6).

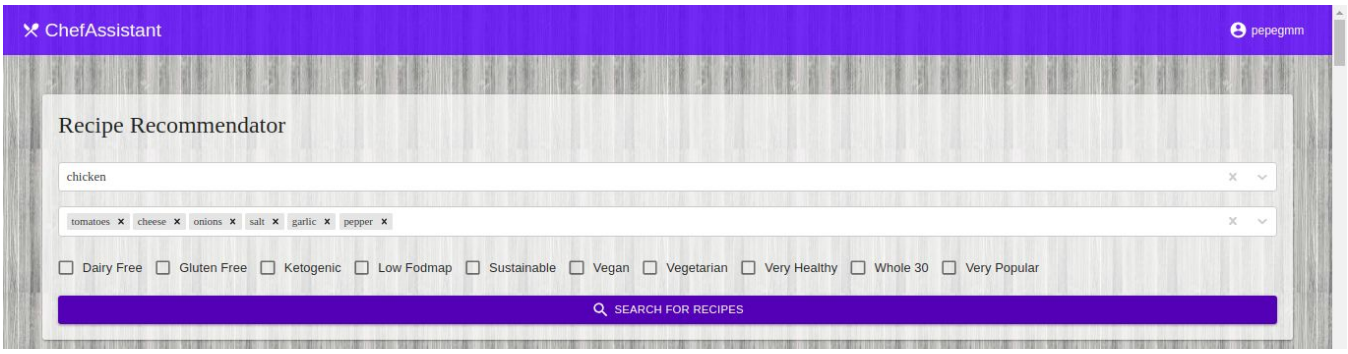

Figura 5 - Componente responśavel pela pesquisa de receitas

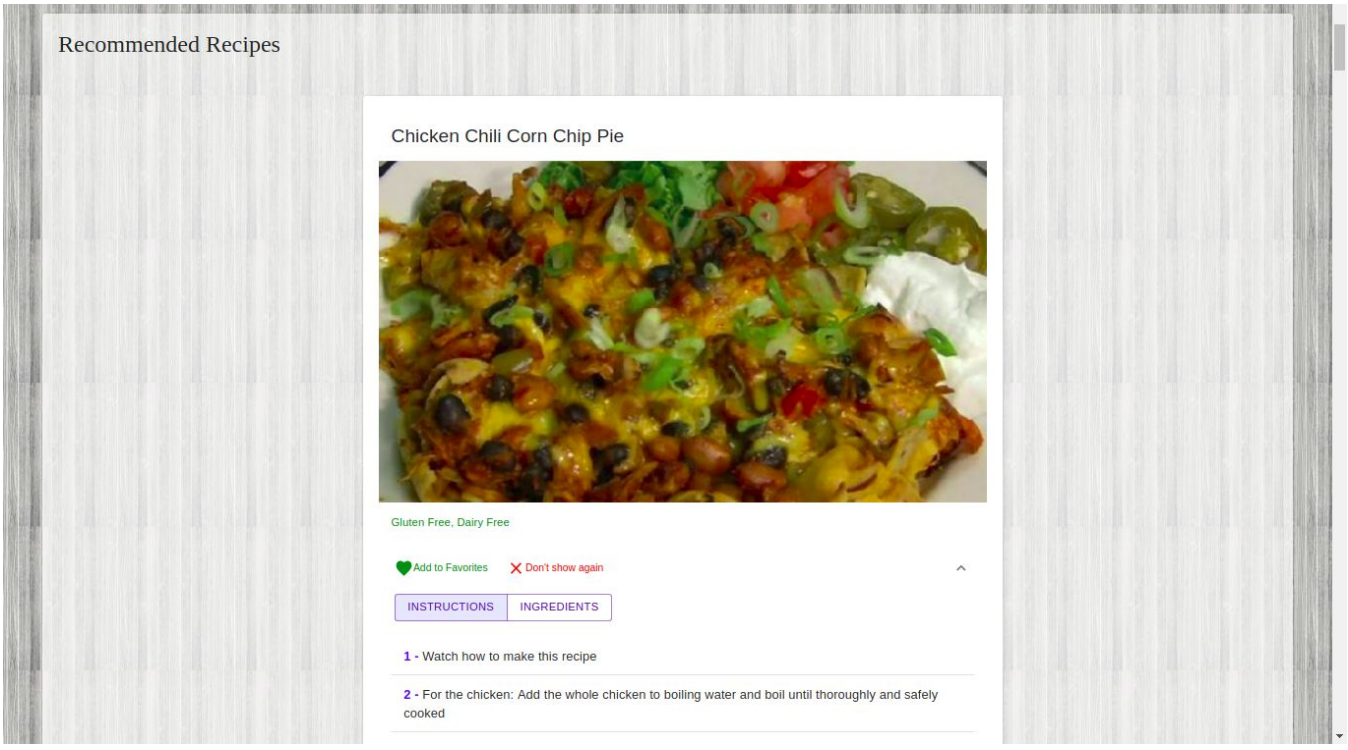

Figura 6 - Componente responsável por exibir a lista de receitas recomendadas

2.2.6 Algoritmo de recomendação

Primeiramente é calculado a similaridade entre cada um dos ingredientes escolhidos pelo usuário com todos os ingredientes distintos presentes na base de dados. Essa similaridade é calculada por meio da Distância de Levenshtein [4], um algoritmo que determina a distância de edição entre dois strings (duas sequências de caracteres) através do número mínimo de operações necessárias para transformar um string no outro. Essas operações podem ser uma inserção, deleção ou substituição de um carácter e

cada uma possui um valor de custo. Em seguida, as distâncias são ordenadas de forma crescente e ocorre uma filtragem por uma distância mínima definida juntamente com um número máximo de ingredientes.

Após obter os ingredientes mais similares a cada um dos que foram selecionados pelo usuário, é feita uma intersecção entre eles por meio do identificador de receita de cada um. Em seguida, os identificadores de receita são ordenados pelo número de intersecções e uma porção inicial com mais intersecções é filtrada. Por fim, essa porção filtrada é reordenada de tal forma que as receitas contendo o ingrediente principal sejam exibidas antes do restante e as 20 primeiras receitas são, por fim, o resultado.

3. SISTEMA EM USO

Nesta seção serão descritas a metodologia e resultados do sistema em uso.

3.1 Metodologia

Existem diversas maneiras de medir a satisfação do usuário em relação a um produto de software, dentre elas a mais comum é por meio de questionários como QUIS (*Questionnaire for User Interaction Satisfaction*), SUMI (*Software Usability Measurement Inventory*), PSSUQ (*Post-Study System Usability Questionnaire)*, e SUS *(Software Usability Scale*). O método escolhido para avaliar a satisfação do usuário foi por meio do PSSUQ, um instrumento de pesquisa desenvolvido pela IBM para avaliar a satisfação do usuário quanto à usabilidade.

Esse questionário foi respondido por 8 estudantes do curso Ciência da Computação da Universidade Federal da Paraíba e consiste em 19 itens destinados a abordar as cinco características de usabilidade do sistema: conclusão rápida do trabalho; facilidade de aprendizado; documentação e informações on-line de alta qualidade; adequação funcional; aquisição rápida do sistema. Cada item possui uma escala de 7 pontos (onde 1=Concordo fortemente e 7=Discordo fortemente).

3.2 Resultado

Foram obtidas quatro sub-escalas derivadas da satisfação do usuário (Tabela 1): satisfação geral (itens 1 a 19); utilidade do sistema (itens 1 a 8); qualidade das informações (itens 9 a 15); qualidade da interface (itens 16 a 19) [5]. Os resultados podem ser visto com mais detalhes através do gráfico de barras da Figura 7 e pode-se perceber que a média dos itens foram bem similares e balanceadas, mostrando que a usabilidade do sistema é de fato satisfatória.

O questionário possui os seguintes itens:

1) No geral, estou satisfeito com a facilidade de uso do sistema.
2) Foi simples usar o sistema.
3) Eu pude efetivamente completar as tarefas e os cenários usando o sistema.
4) Eu consegui completar as tarefas e os cenários rapidamente usando o sistema.
5) Eu consegui completar eficientemente as tarefas e os cenários usando o sistema.
6) Eu me senti confortável usando o sistema.
7) Foi fácil aprender a usar o sistema.
8) Acredito que eu poderia me tornar mais produtivo usando o sistema.
9) O sistema mostrou mensagens de erros que mostraram claramente como resolver os problemas.
10) Quando cometi um erro usando o sistema, consegui reperá-lo fácil e rapidamente.
11) A informação (por exemplo, ajuda online, mensagens na tela ou documentação) fornecida pelo sistema era clara.
12) Foi fácil encontrar a informação que precisei.
13) A informação fornecida pelo sistema era fácil de entender.
14) A informação foi efetiva ao me ajudar a completar meu trabalho.
15) A organização da informação do sistema era clara.
16) A interface do sistema é amigável.
17) Eu gostei de usar a interface do sistema.
18) O sistema possui todas as funções que eu esperava.
19) No geral, estou satisfeito com o sistema.

Figura 7 - Gráfico de barras horizontal que exibe a média de cada uma das 19 questões presentes no questionário PSSUQ

Tabela 1 - Sub-escalas derivadas da satisfação do usuário

Fatores de usabilidade	Pontuação média
Utilidade do sistema	1.493
Satisfação geral	1.546
Qualidade das informações	1.482
Qualidade da interface	1.406

Com base nos resultados, é possível perceber que o sistema necessita melhorar o ponto levantado pelo item 4, que tem relação com tempo necessário para realizar as tarefas. Isso pode ser feito através de algumas otimizações no algoritmo de recomendação ou até mesmo uma escalabilidade vertical ou horizontal dos servidores.

4. EXPERIÊNCIA

Desenvolver mais uma aplicação web por completo foi uma experiência que com certeza agregou conhecimento e ajudou a aperfeiçoar mais as habilidades de programação.

4.1 Processo de desenvolvimento

O processo de desenvolvimento baseou-se no CVDS (Ciclo de Vida do Desenvolvimento de Software) e foi realizado de forma incremental. Primeiramente foi feito um diagnóstico onde o problema foi abrangido e detalhado para garantir que o *software* atenda todas as necessidades. Com base no diagnóstico foi feito um levantamento e análise de requisitos, onde elaborou-se alternativas de solução e *user stories* foram definidas, analisadas e por fim validadas. Baseando-se nas experiências obtidas em outros projetos já desenvolvidos e nas *user stories* definidas, escolheu-se quais tecnologias seriam utilizadas. Em seguida foi construído um protótipo da página principal, como pode ser visto nas Figuras 8 e 9, para auxiliar no desenvolvimento da interface do usuário.

Após a prototipagem iniciou-se a implementação, revezando entre *back-end* e *front-end*, construindo o *software* de forma incremental, como já citado. Testes manuais foram realizados à medida que uma nova funcionalidade fosse implementada ou aprimorada. Por fim, ao finalizar a implementação e assegurar que as funcionalidades estavam de acordo com o esperado, foi feito o *deploy* do *front-end* e o *back-end* foi hospedado em uma máquina virtual do google para diminuir o tempo de processamento necessário para recomendações de receitas.

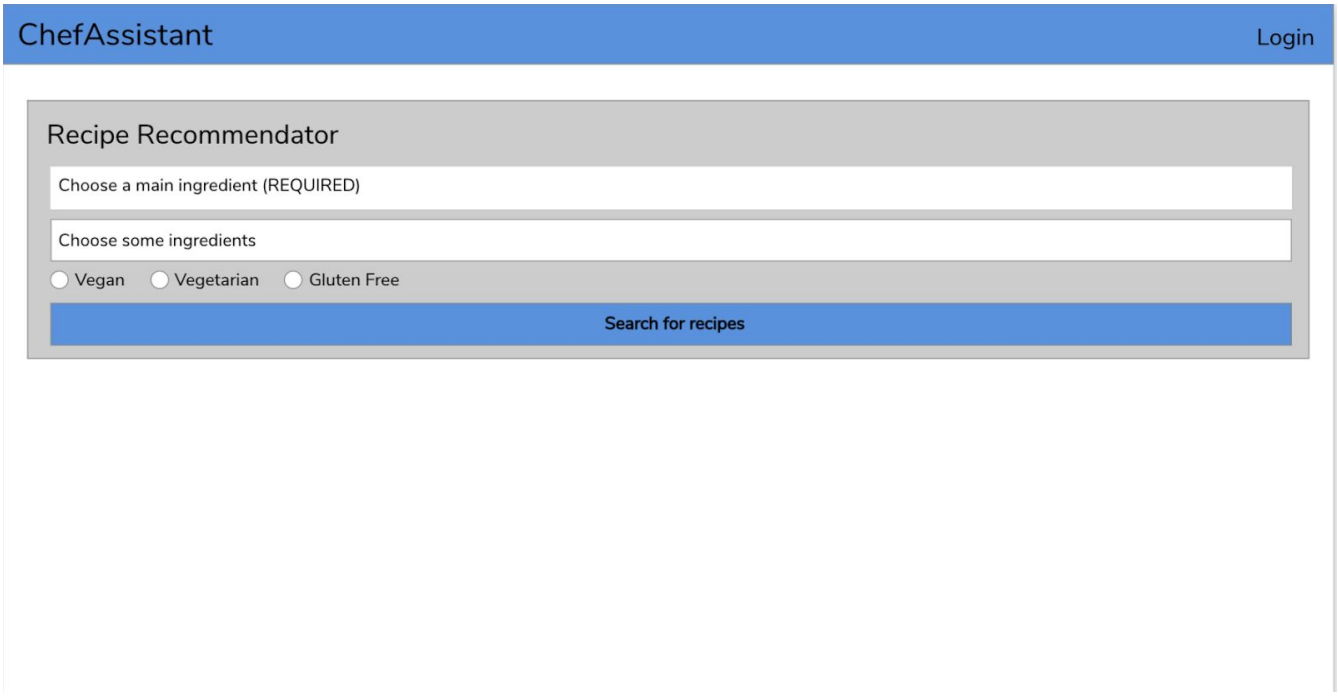

Figura 8 - Seção de filtragem exibida ao entrar no site

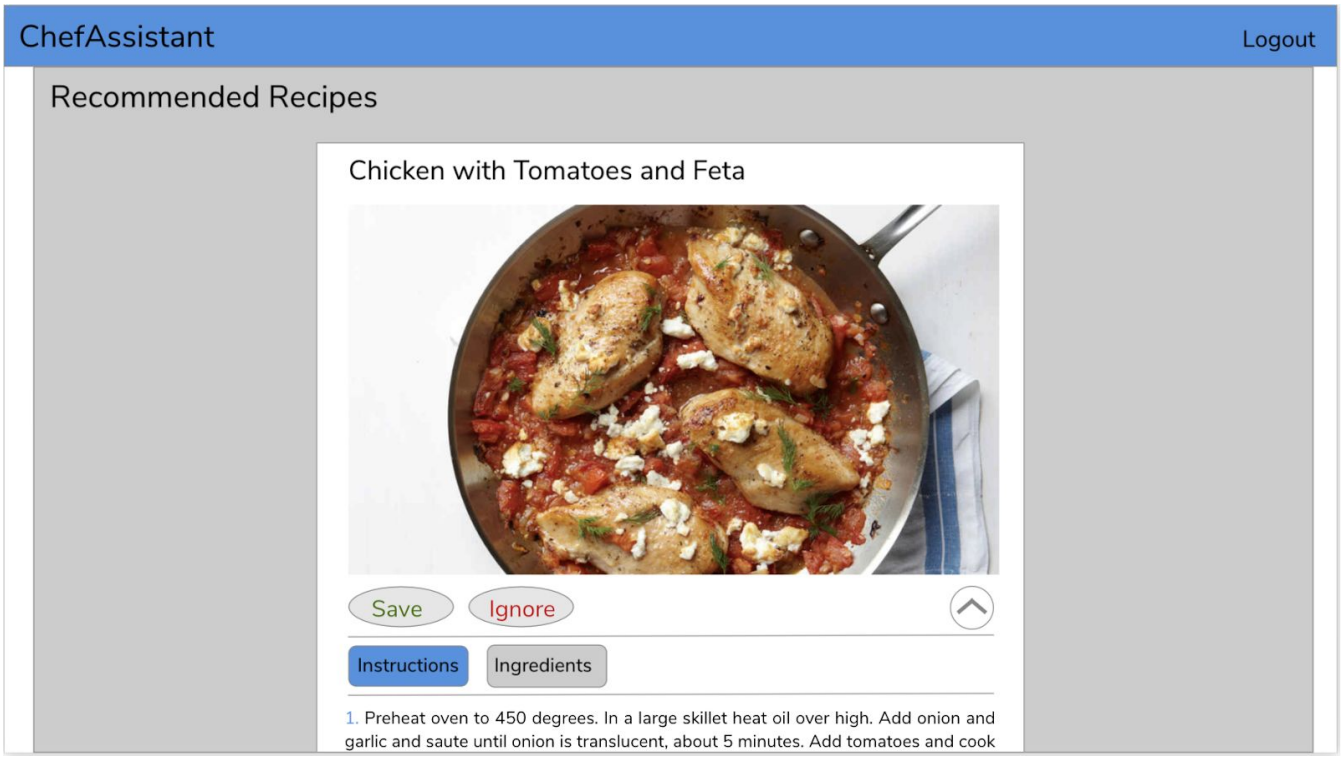

Figura 9 - Seção de receitas recomendadas exibida após filtragem

4.2 Desafios

A princípio foi preciso achar uma base de dados com informações suficientes para implementar um algoritmo que gerasse resultados satisfatórios. Procurava-se uma base de dados que, além de ter instruções detalhadas de como preparar a receita juntamente com a lista de ingredientes necessários, também tivesse categorias relacionadas a restrições alimentares.

Como a base de dados tinha uma boa quantidade de dados e o processamento não era simples, foi necessário deixar o servidor rodando em uma *Virtual Machine* do Google para acelerar o algoritmo de recomendações.

4.3 Trabalhos futuros

O ChefAssistant visa implementar novas funcionalidades para aprimorar tanto o algoritmo de recomendações, quanto a interface do usuário. Alguns exemplos dessas novas funcionalidades desejadas são: ingredientes agregados por categorias, como vegetais e frutas; a ampliação e melhoria na base de dados; tornar a interface mais sofisticada e amigável para o usuário.

5. REFERÊNCIAS

[1] What regularly dining out does to your body. Disponível em https://go.nature.com/2oHxBFj.

[2] Healthy dining out with high cholesterol. Disponível em https://www.everydayhealth.com/high-cholesterol/living-with/dining-out-with-high-cholesterol/.

[3] Supercook. Site de Recomendações de receitas. Disponível em https://www.supercook.com.

[4] Distância Levenshtein. Disponível em https://pt.wikipedia.org/wiki/Dist%C3%A2ncia_Levenshtein.

[5] Engenharia de Usabilidade Aplicada no Desenvolvimento de um Sistema Interativo para Soldagem Robotizada. Disponível em http://www.scielo.br/pdf/si/v17n4/a03v17n4.pdf.