UNIVERSIDADE FEDERAL DE SANTA CATARINA - CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

APLICAÇÃO DE METABUSCA DE RECEITAS CULINÁRIAS

Daniela Heckler Mello

Florianópolis
2022

Universidade Federal de Santa Catarina
Departamento de Informática e Estatística

APLICAÇÃO DE METABUSCA DE RECEITAS CULINÁRIAS

Trabalho de Conclusão de Curso de Graduação em Ciências da Computação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Ciências da Computação.

Autor: Daniela Heckler Mello

Orientador: Professor Raul Sidnei Wazlawick

Florianópolis
2022

RESUMO

O ato de cozinhar é considerado por muitos autores como uma atividade que define os seres humanos. Na era contemporânea, as evoluções tecnológicas obtidas nas últimas décadas facilitaram muito o ato de cozinhar. A criação de diversos equipamentos permitiu a evolução da culinária de forma geral. No entanto, nada revolucionou mais o compartilhamento de receitas culinárias como a internet. Com a utilização da web, indivíduos de qualquer lugar do mundo podem publicar e compartilhar receitas, desde as que foram passando entre gerações nas famílias, até mesmo receitas inovadoras e originais. Esse compartilhamento facilitado trouxe diversas vantagens para as pessoas, como a difusão de diversas culturas entre a população. Contudo, a busca do usuário é muitas vezes dificultada por diversos fatores, como o excesso de informações, formatação inadequada do conteúdo, com excesso de propaganda e falta de clareza, causando uma má experiência por parte dos usuários. Nesse contexto, o seguinte trabalho tem como objetivo desenvolver um sistema de metabusca de receitas culinárias, capaz de realizar uma pesquisa simultânea em diversos sites, extrair os dados e formatá-los. Então, elas são exibidas em uma aplicação mobile android, com um padrão único, utilizando práticas de *User Experience* (UX) e *User Interface* (UI), sem propagandas ou poluição visual. Com isso, espera-se contribuir com o compartilhamento de receitas e com o ato de cozinhar, melhorando a experiência do usuário ao utilizar a busca de receitas culinárias.

Palavras-chave: Cozinhar, Receitas, Metabusca, Android, *Mobile, Scraping*.

Sumário

1. INTRODUÇÃO

Muitos autores defendem que a ideia de cozinhar é uma atividade que define os seres humanos. O escritor escocês James Boswell, em 1773, definiu o Homo sapiens como “o animal que cozinha”, ao observar que “nenhum animal é um cozinheiro”. Mais tarde, em 1825, o gastrônomo francês Jean Anthelme Brillat-Savarin lançou o livro A fisiologia do gosto, onde afirmou que cozinhar transformou os humanos, contribuindo muito para o avanço da civilização. Um pouco depois, em 1964, Lévi-Strauss registrou em seu livro *O cru e o cozido* que várias das culturas no mundo percebem no ato de cozinhar a atividade que diferencia animais e pessoas (POLLAN, 2013).

Atualmente, tem ocorrido uma série de evoluções tecnológicas que colaboram com o ato de cozinhar. Primeiramente, foram desenvolvidos diversos equipamentos que facilitaram a culinária como um todo (BELLIS, 2019). No entanto, a internet pode ser considerada um fator disruptor nesse quesito. Através do uso da internet, pessoas do mundo todo publicam receitas, podendo ser desde receitas de família, compartilhadas por gerações, ou até mesmo as originais, recém inventadas. O website de receitas culinárias cookpad, por exemplo, é o mais acessado da categoria no mundo todo, recebendo quase 120 milhões de acessos apenas em setembro de 2022 (SIMILARWEB, 2022).

A publicação das receitas culinárias é realizada em inúmeros websites ou aplicativos específicos para esse fim, os tornando grandes repositórios com grande quantidade de acessos. No Brasil, conforme o website SimilarWeb[1], o site de receitas mais acessado é o site TudoGostoso, também disponível em aplicativo, com 35,5 milhões de acessos em setembro de 2022.

Em diversos desses repositórios, qualquer usuário tem a possibilidade de publicar e compartilhar as receitas com qualquer outra pessoa. Essa facilidade de propagar as informações trouxe muitas vantagens para os indivíduos, como a difusão de cultura culinária de cada população para pessoas do mundo todo (CUYLENBURG, 2021).

No entanto, a facilidade de compartilhamento trouxe não apenas benefícios, como alguns problemas que atrapalham a busca do usuário. O excesso de

[1] https://www.similarweb.com/pt/website/tudogostoso.com.br/#overview

informações obtidas ao realizar a pesquisa em sites de busca muitas vezes confunde o usuário, fazendo com que o mesmo tenha que consultar diversas fontes até encontrar a receita desejada. Além disso, o excesso de propagandas e a falta de clareza na exibição de conteúdo causam uma má experiência por parte dos usuários[2].

Tendo conhecimento dos problemas que afetam o usuário ao realizar a busca por receitas, o presente trabalho consiste em desenvolver um sistema de metabusca de receitas culinárias. O metabuscador é capaz de realizar uma pesquisa em site de receitas culinárias, e fazer a extração dos dados encontrados nos sites buscados. Após essa etapa, o sistema formata todos esses dados, os exibindo em um padrão único, sem quaisquer propagandas ou poluição visual.

Com a finalidade de proporcionar a melhor experiência para o usuário na utilização do sistema, são utilizadas práticas de User Experience (UX) e User Interface (UI) no desenvolvimento do design da aplicação. Por fim, o sistema foi disponibilizado em uma aplicação mobile android, que realiza a integração da metabusca de receitas culinárias e do design da aplicação.

[2] Fonte: Pesquisa de Elicitação de Requisitos, apresentada no capítulo 6 deste trabalho.

2. OBJETIVOS

2.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver uma solução para os problemas encontrados hoje na busca de receitas culinárias, como a necessidade de visitar diversos sites, excesso de propagandas, além da falta de clareza na exibição do conteúdo, que causam uma má experiência por parte dos usuários. Para isso, foi desenvolvida uma aplicação de metabusca de receitas, que realiza uma pesquisa em website, formata os dados encontrados e os apresenta de forma limpa e clara para o usuário, sem poluição visual e propagandas.

2.2 Objetivos específicos

Os objetivos específicos do trabalho são os seguintes:

1. Realizar busca simultânea em diversos sites de receitas culinárias.
2. Realizar a extração dos dados encontrados nos sites buscados.
3. Formatar todos os dados encontrados em um sistema único, para poder exibir para os usuários.
4. Realizar a integração de todos os dados formatados com uma aplicação mobile.
5. Exibir o conteúdo de forma clara e objetiva, utilizando conceitos de User Experience (UX) e User Interface (UI) para melhor experiência por parte do usuário.

3. METODOLOGIA

A metodologia utilizada neste trabalho consiste nos seguintes passos:

1. Fundamentação teórica, através de estudo e análise de conceitos fundamentais ao desenvolvimento do projeto, buscando o entendimento de sistemas de metabusca, aplicações mobile, e conhecimento sobre UX e UI, visando um embasamento teórico adequado que propicie a implementação da aplicação.
2. Realização de uma pesquisa de trabalhos correlatos, visando identificar soluções similares à proposta pelo atual trabalho, e estudar suas características positivas e negativas.
3. Desenvolvimento de um sistema que realize a metabusca em sites de receitas culinárias, extraindo e formatando os dados encontrados.
4. Implementação de uma aplicação mobile que seja capaz de receber e exibir os dados extraídos pelo sistema de metabusca.
5. Aplicar conceitos de UX e UI na aplicação mobile, com o objetivo de ter uma aplicação de fácil utilização por parte do usuário.
6. Realização de testes com usuários, para avaliar a usabilidade da aplicação, além de verificar se o sistema de metabusca está realizando a extração e coleta dos dados conforme o esperado, e se a aplicação mobile está exibindo os dados de forma correta.

4. FUNDAMENTAÇÃO TEÓRICA

4.1 Metabusca

Metabusca é o processo de consultar diversos motores de busca de forma simultânea. Dessa forma, é possível definir um metabuscador como um sistema que realiza a metabusca, possibilitando que usuários da internet realizem a pesquisa a vários websites simultaneamente, através de apenas uma consulta. Assim, uma aplicação de metabusca realiza uma padronização dos dados encontrados para o usuário, facilitando a pesquisa (MENG, 2019).

Os metabuscadores realizam a busca a outros mecanismos de busca, e utilizam a base de índices própria desses mecanismos. Logo, os sistemas de metabusca não possuem uma base de índices própria para armazenar os documentos indexados, não necessitando então de uma implementação de um sistema de indexação de documentos. Assim, com a utilização de uma base de indexação mais ampla, é possível obter uma maior abrangência de resultados encontrados, permitindo que o usuário alcance um melhor retorno para sua busca (OLIVEIRA, A. L.; OLIVEIRA, F. L.; FAGUNDES, 2004)

Para realizar uma metabusca, a consulta de um usuário é enviada para múltiplos motores de busca. Então, quando os resultados retornados pelos motores de busca são recebidos pelo sistema de metabusca, eles são posicionados em uma única lista, e esta é então apresentada para o usuário. (MENG, 2019).

Alguns dos principais desafios na implementação de um sistema de metabusca incluem como enviar as consultas de usuários para outros mecanismos de pesquisa, identificar os resultados das páginas retornadas e padronizar a apresentação de dados obtidos através de consultas a diferentes fontes de pesquisa. Motores de metabusca mais sofisticados também realizam a seleção de fontes de pesquisa, ou seja, identificam os motores de busca mais apropriados e enviam uma consulta a apenas esses websites, filtrando assim resultados indesejados. (HOWE; DREILINGER, 1997).

4.2 Web scraping

O web scraping é o processo de coleta de dados de websites, e posterior armazenamento das informações coletadas em um sistema de arquivos ou banco de dados, para posterior recuperação ou análise. A captura de dados por web scraping pode ser feita manualmente ou com a utilização de um robô ou um web crawler, utilizando protocolo HTTP ou através de um navegador. (KAUSAR; DHAKA; SINGH, 2013).

É possível dividir o processo de coleta de dados da internet em dois passos a serem realizados em sequência, sendo o primeiro deles a obtenção de recursos da web e o segundo a extração das informações desejadas dos dados adquiridos. A captação dos dados pode ser realizada através de uma requisição a uma URL, que retorna as informações necessárias. Essas informações podem ser em formato JSON, XML, ou em formato de mídia, como vídeos ou imagens, por exemplo.

O processo de coleta de dados, realizado através de uma requisição HTTP, na linguagem de programação Python pode utilizar uma biblioteca como Urllib ou Selenium, por exemplo. A biblioteca Urllib utiliza de diversas funções, como cookies e autenticação, para lidar com as requisições. Diferentemente dela, a biblioteca Selenium simula um navegador para automatizar o processo de utilização da internet por um usuário (ZHAO, 2017).

Já o processo de formatação e extração das informações coletadas do código HTML pode ser realizado em Python através da utilização da biblioteca BautifulSoup, por exemplo. Os dados extraídos podem então ser armazenados e exportados em diferentes formatos, como XML, CSV ou JSON, por exemplo (BUDKEWICZ, 2018).

As informações obtidas através do processo de web scraping podem ser utilizadas para diferentes aplicações, como monitoramento de preços de diferentes serviços ou produtos, monitoramento de clima, monitoramento de ações, ou até mesmo opiniões do público sobre grupos políticos (PPLWARE, 2020).

4.3 Bibliotecas para scraping

Dentre as ferramentas existentes para a realização de scraping, é possível destacar três delas: Requests, Beautiful Soup e Selenium. A captura dos dados é o primeiro passo para o processo de web scraping. A biblioteca Requests possibilita a captura dos dados a partir da realização de uma requisição HTML para o servidor de uma página web.

As principais vantagens da biblioteca Request são a simplicidade e facilidade de utilização e suporte a HTTP, o que a torna uma das bibliotecas Python mais consumidas, gerando cerca de 233 milhões de downloads por mês. Já as principais desvantagens de sua utilização são que retorna apenas o conteúdo estático de uma página, não permitindo a formatação sem a utilização de outra ferramenta. (SHARMA, 2020).

Outra biblioteca Python de grande importância para a realização de web scraping é a Beautiful Soup, criada por Leonard Richardson. O funcionamento do Beautiful Soup consiste primeiramente em realizar uma leitura e análise do código de páginas HTML informado e, a partir dele, gerar um objeto formatado em uma estrutura de dados encadeada. Com esse objeto, é possível realizar a extração dos dados desejados (CRUMMY, 2022).

Por fim, Selenium pode ser definida como um conjunto de ferramentas de código multiplataforma, com diversas aplicações, como testes automatizados e extração de dados, por exemplo, e suporte a diferentes linguagens de programação, dentre elas C#, Java e Python.

Diferentemente do Beautiful Soup, que realiza uma análise no código HTML informado, o Selenium simula a execução de um navegador, como Chrome ou Firefox, por exemplo, como se um usuário estivesse utilizando o sistema. Com isso, ele consegue acessar e capturar os dados exibidos na tela (TREINAWEB, 2022).

4.4 Aplicações móveis (*mobile app*)

Uma aplicação móvel, ou simplesmente a abreviação *app*, é uma aplicação de software implementada para ser executada em um dispositivo móvel, como um smartphone por exemplo. As aplicações móveis podem ser classificadas em três tipos: aplicações web, híbridas e nativas.

Aplicações web podem ser escritas em linguagem própria para web, e normalmente são executadas em um navegador. Algumas das linguagens mais utilizadas para aplicações web são HTML5 e CSS.

As aplicações híbridas, também conhecidas como multiplataforma, são aquelas que podem ser executadas em diferentes sistemas operacionais, como por exemplo Android e iOS. Dentre as principais linguagens de programação utilizadas para implementar aplicações híbridas, as mais conhecidas são React Native e Flutter.

As aplicações nativas são aquelas que são desenvolvidas especificamente para um sistema operacional, sendo estes os mais comuns Android e iOS. Dentre as principais linguagens de programação utilizadas na implementação de apps nativos, temos swift para o iOS, e Java e Kotlin, para o sistema Android (AMAZON, 2022).

Para o desenvolvimento deste trabalho, foi realizada a implementação do sistema de metabusca de receitas culinárias em uma aplicação Android, por ser o sistema mais utilizado no mundo. De acordo com a empresa alemã de pesquisa de dados Statista, usuários de Android representaram 85,1% de todos os usuários de smartphones em 2017, e é estimado que em 2023 irão representar 87,4% dos usuários.

4.5 User Interface (UI) e User Experience (UX)

Uma Interface de Usuário, conhecida como UI, refere-se à interação entre um sistema e um usuário através de comandos ou inserção de dados. As interfaces de usuário são utilizadas em sistemas como computadores, dispositivos móveis e jogos, por exemplo.

O termo User Experience, ou Experiência do Usuário, refere-se à experiência geral que um usuário possui enquanto interage com um sistema, podendo sentir diferentes percepções, emoções, sentimentos, e através deles expressar diferentes reações e comportamentos (JOO, 2017).

Dessa forma, um sistema com uma boa implementação de UI e UX deve ser capaz de facilitar a compreensão por parte dos usuários, e proporcionar uma boa experiência de utilização. Para isso, existem alguns conceitos fundamentais a serem seguidos, como:

- Hierarquia visual: Os componentes a serem exibidos na tela devem estar em uma hierarquia, ou seja, posicionados de forma a prender a atenção do usuário nos itens adequados. Dessa forma, o usuário vai ser capaz de encontrar as informações desejadas de forma mais rápida e eficaz.
- Acessibilidade: É a capacidade de acessar um sistema por todos os usuários, independente das limitações. Por exemplo, as aplicações devem possuir fontes e ícones de tamanho adequado, evitando fontes com dificuldade de legibilidade e posicionamento incorreto dos ícones, por exemplo.
- Consistência visual: Todos os itens visuais de uma aplicação devem seguir um padrão, tanto no quesito fonte, tamanhos, ícones, entre outros. A utilização de itens sem consistência e padronização podem afetar também a hierarquia visual, causando uma dificuldade de compreensão do sistema por parte do usuário.
- Consistência de ação: Usuários inerentemente transferem conhecimentos passados a novos contextos enquanto exploram partes da aplicação. Dessa forma, ações consistentes fazem com que a aplicação seja utilizada sem a necessidade de realizar qualquer esforço, tornando-se natural para o usuário. Para isso, é necessário que as ações do usuário sejam implementadas com

uma padronização na navegação, nos cliques e na disposição das telas apresentadas ao usuário, por exemplo (PEREA; GINER, 2017).

4.6 Aspectos Legais da Aplicação

Durante a pesquisa e desenvolvimento deste trabalho, foram considerados os aspectos legais da aplicação, ou seja, se as receitas a serem capturadas poderiam ter a proteção de direitos autorais, estando assim impossibilitadas de serem replicadas em outras aplicações. Segundo a Lei nº 9610/98 - Lei de Direitos Autorais[3], não há qualquer menção a receitas gastronômicas na descrição da lei. Mais ainda, os primeiros parágrafos do Art. 8 mencionam como obras *não* protegidas “as idéias, procedimentos normativos, sistemas, métodos, projetos […] esquemas, planos ou regras para realizar atos mentais, jogos ou negócios”. Depende da interpretação, mas é possível que receitas culinárias sejam classificadas em algum destes tipos.

Além disso, ao considerar os possíveis sites para realizar o scraping de receitas gastronômicas, foram procurados os Termos de Uso, a fim de garantir que não iria ser feita a captura de receitas em sites que tivesse explícito em seu Termo de Uso a não permissão de reprodução de seu conteúdo. Dos sites verificados, nenhum proibia explicitamente o uso das informações ali reproduzidas.

[3] Lei 9.610/98. Disponível em: https://www.planalto.gov.br/ccivil_03/leis/l9610.htm

5. APLICAÇÕES EQUIPARÁVEIS

Para a realização deste trabalho, foram pesquisadas algumas aplicações equiparáveis ou correlatas à presente monografia, com o objetivo de verificar a relevância da aplicação proposta, e analisar as características dos aplicativos, tanto positivas quanto negativas.

Como critério para a seleção, primeiramente foi considerada uma aplicação de receitas, pela similaridade do tema com o presente trabalho. Com base nisso, foi selecionado o aplicativo Tudo Gostoso, por ser um dos mais famosos no Brasil, com mais de 10 milhões de downloads na loja do Google[4].

Além disso, foi considerada também uma aplicação que utiliza um sistema de metabusca, também pela similaridade com o trabalho proposto. Para isso, foi escolhido o aplicativo Trivago, por sua grande visibilidade e sucesso na Play Store, possuindo mais de 50 milhões de downloads.

5.1 Tudo gostoso

É um aplicativo de receitas muito utilizado, possuindo mais de 10 milhões de downloads na Play Store. A principal semelhança dele com a aplicação proposta é o tema, sendo voltado para usuários que desejam encontrar receitas culinárias.

A principal qualidade do aplicativo é que ele possui uma ampla quantidade de receitas, muitas delas publicadas por usuários. Além disso, permite a avaliação das receitas por parte dos usuários, classificando com nota e comentários.

Dentre os principais defeitos, é possível destacar a quantidade excessiva de propagandas. Ao acessar uma receita qualquer, aparece uma propaganda que preenche a tela toda, fazendo o usuário ter que fechar o anúncio para poder visualizar a receita. Além disso, apresenta também propagandas entre as seções da receita, obrigando o usuário a deslizar muito a tela para buscar as informações desejadas, o que atrapalha a experiência de uso. Na figura 1 é possível ver alguns dos anúncios exibidos pelo aplicativo.

[4] Fonte: Play Store

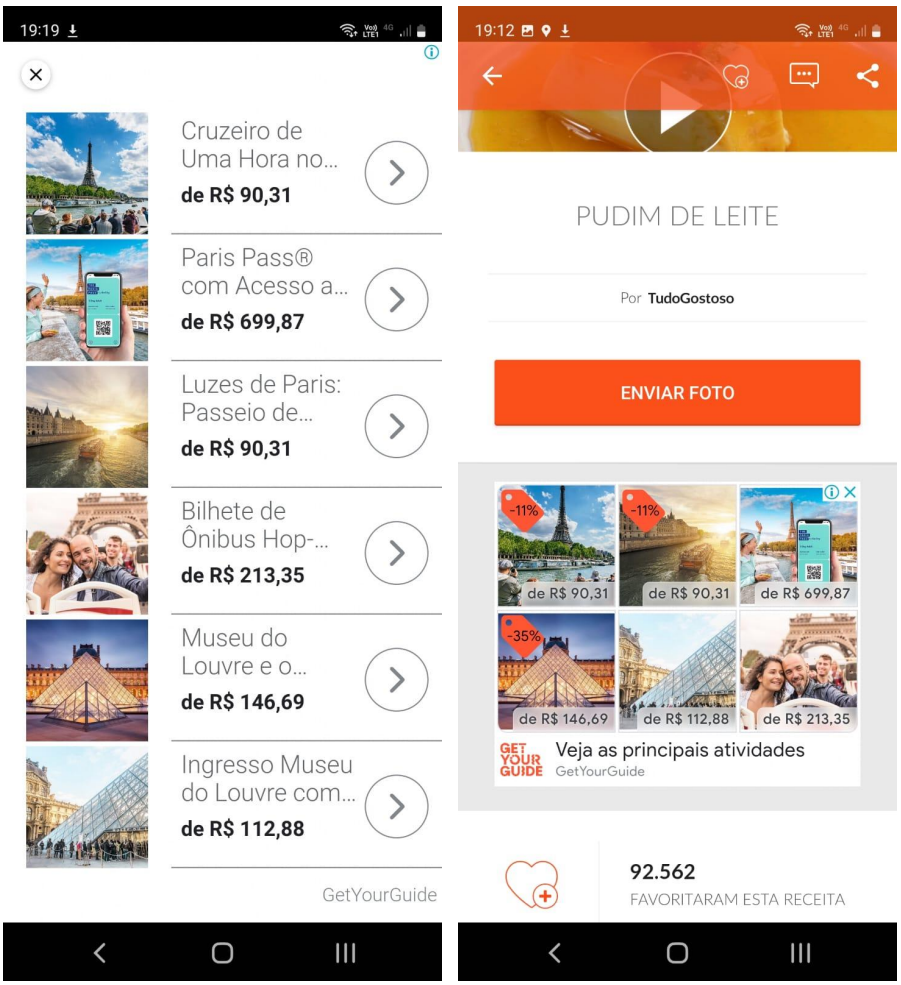


Figura 1. Excesso de anúncios atrapalham a utilização da aplicação.

5.2 Panelinha

O Panelinha é um site de busca de receitas muito utilizado no Brasil, disponível em https://www.panelinha.com.br. Ele está em 8º lugar no Brasil dentre os mais acessados da categoria e recebeu 4,7 milhões de acessos em setembro de 2022, de acordo com o site Similarweb[5].

A principal qualidade do website é não ter um excesso de propagandas, e informações expostas de forma clara. A busca e exibição das receitas e dos resultados são fáceis de navegar.

Dentre os problemas encontrados no Panelinha, os mais relevantes foram algumas informações desnecessárias nas receitas, e algumas propagandas de produtos culinários para venda, além de ser publicado apenas em website, não possuindo versão em aplicativo.

[5] Fonte: https://www.similarweb.com/pt/website/panelinha.com.br/#overview

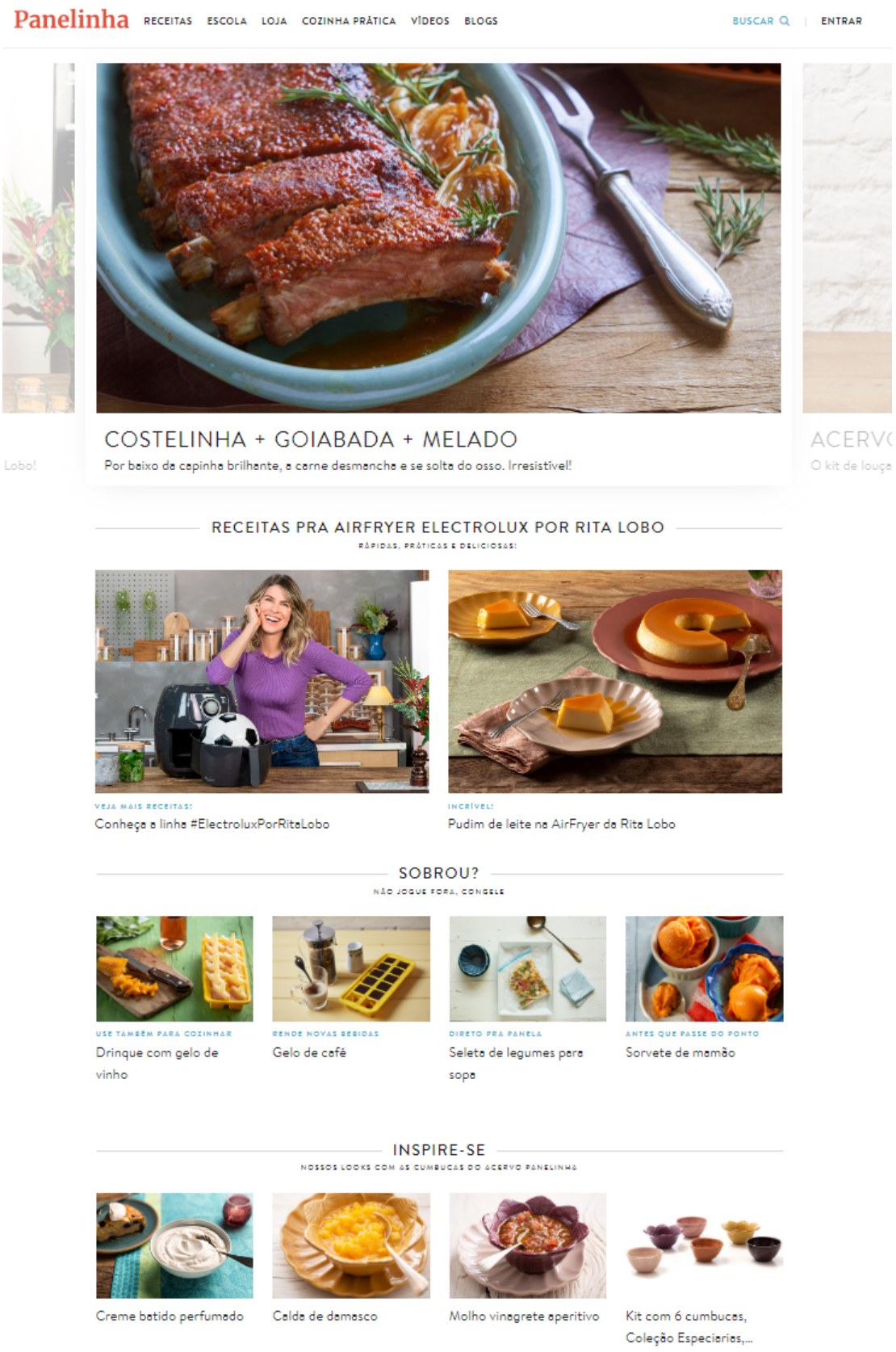


Figura 2. Home do website panelinha.

A Figura 2 mostra como é visualmente a home do website Panelinha. Apesar de ter algumas sugestões de receitas, percebe-se que o website possui conteúdo que talvez não seja o foco para usuários que desejam encontrar receitas de forma mais direta. Esse fato pode ser observado com a exibição de algumas propagandas, como o Kit com 6 cumbucas, por exemplo, e a linha de airfryer Electrolux.

Já a página de busca de receitas é mais direta na exibição de conteúdo. Ao buscar por pudim, por exemplo, como é possível ver na figura 3, o resultado é exibido apenas em receitas.

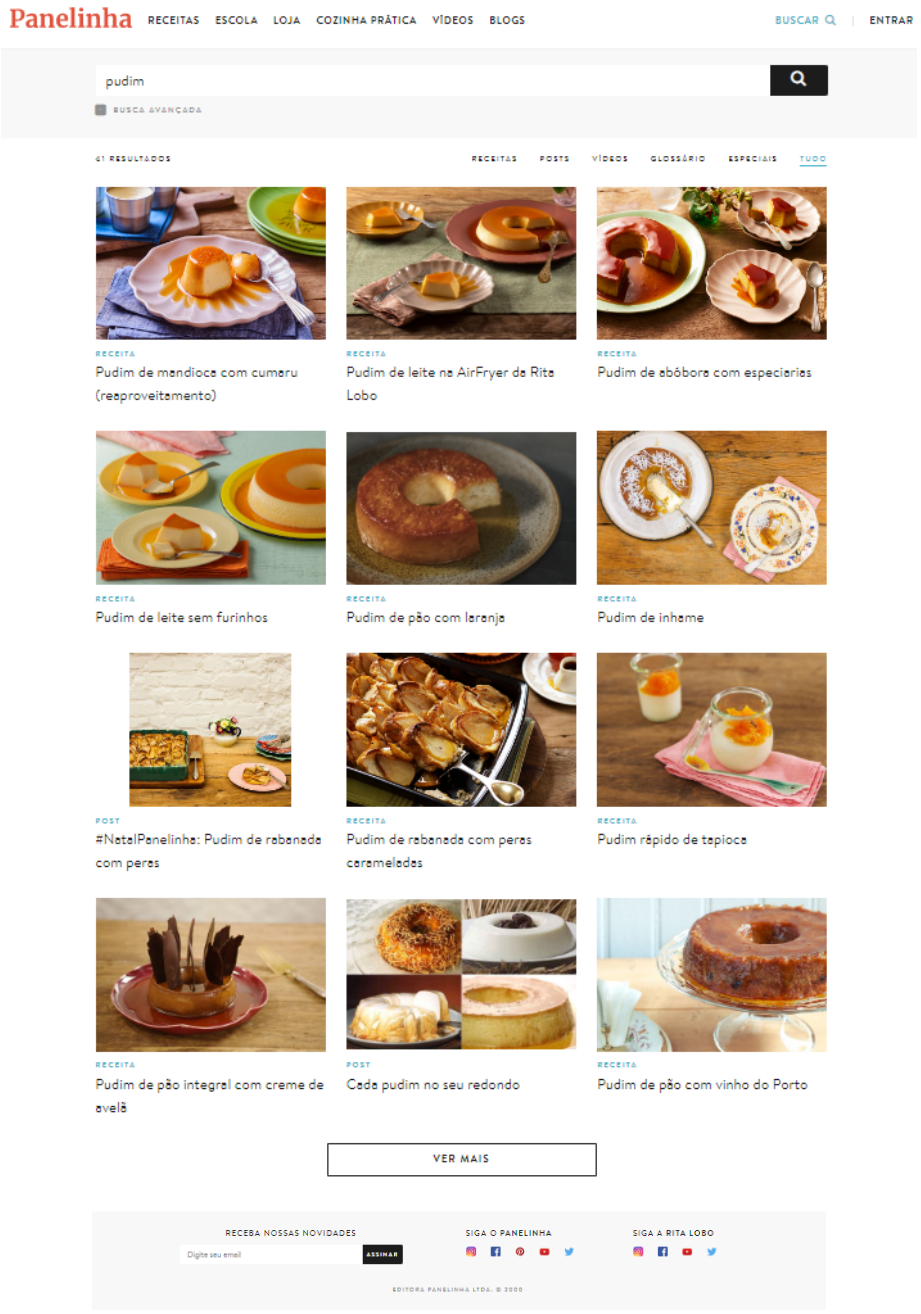


Figura 3. Página de busca de receita por palavra-chave.

Na página de uma receita específica, é possível observar que algumas delas possuem propaganda de itens para venda, como percebe-se na figura 4. Além disso, há um texto introdutório, e sugestão de cardápio, que podem desviar a atenção do usuário.

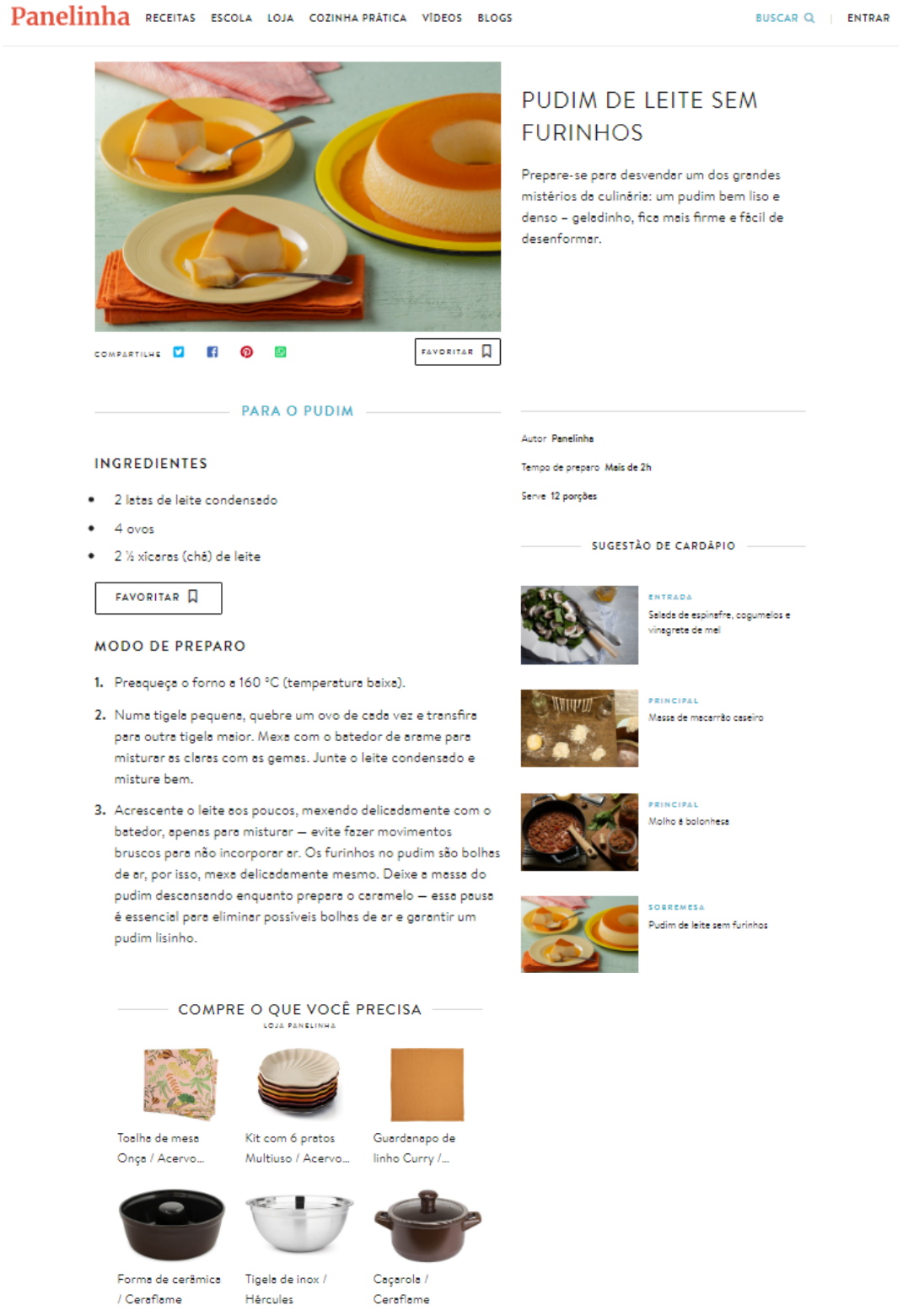


Figura 4. Parte da página de uma receita, com propaganda de produtos para venda.

5.3 Trivago

O Trivago é um aplicativo de busca de hotéis, com mais de 50 milhões de downloads na Play Store. A principal semelhança do Trivago com a aplicação proposta é o fato de possuir a mesma forma de busca de informações. Assim como o presente trabalho está propondo, o Trivago é uma aplicação que utiliza metabusca para obter os dados.

Quando o usuário abre a aplicação, ele precisa inserir qual é o destino desejado, e o período que irá se hospedar no local. Então, o sistema realiza uma busca em websites de hospedagem, enviando as informações de destino e período e, a partir disso, apresenta para o usuário o resultado da busca de forma padronizada. A partir desse momento, o usuário pode clicar em uma oferta desejada, e visualizar detalhes da hospedagem. Para realizar a reserva da acomodação, o usuário precisa clicar em ‘Ver oferta’, que irá encaminhar para o site original.

Como principais qualidades do aplicativo Trivago, é possível destacar a facilidade de utilização e a ausência de propagandas. Além disso, disponibiliza diversas informações para os usuários, como detalhes das acomodações, e comparação de ofertas. As opções de filtros e ordenação também facilitam bastante a busca por parte dos usuários.

Na imagem abaixo é possível observar a página inicial do aplicativo, onde é possível realizar a busca inserindo o destino, o período e o número de hóspedes e quartos desejados. Além disso, é possível ver uma breve descrição, onde é exibido alguns dos sites de reserva que o sistema do Trivago utiliza na metabusca.

Além disso, na figura 5 é possível visualizar a disposição do resultado da busca, e a exibição de alguns dos detalhes do anúncio.

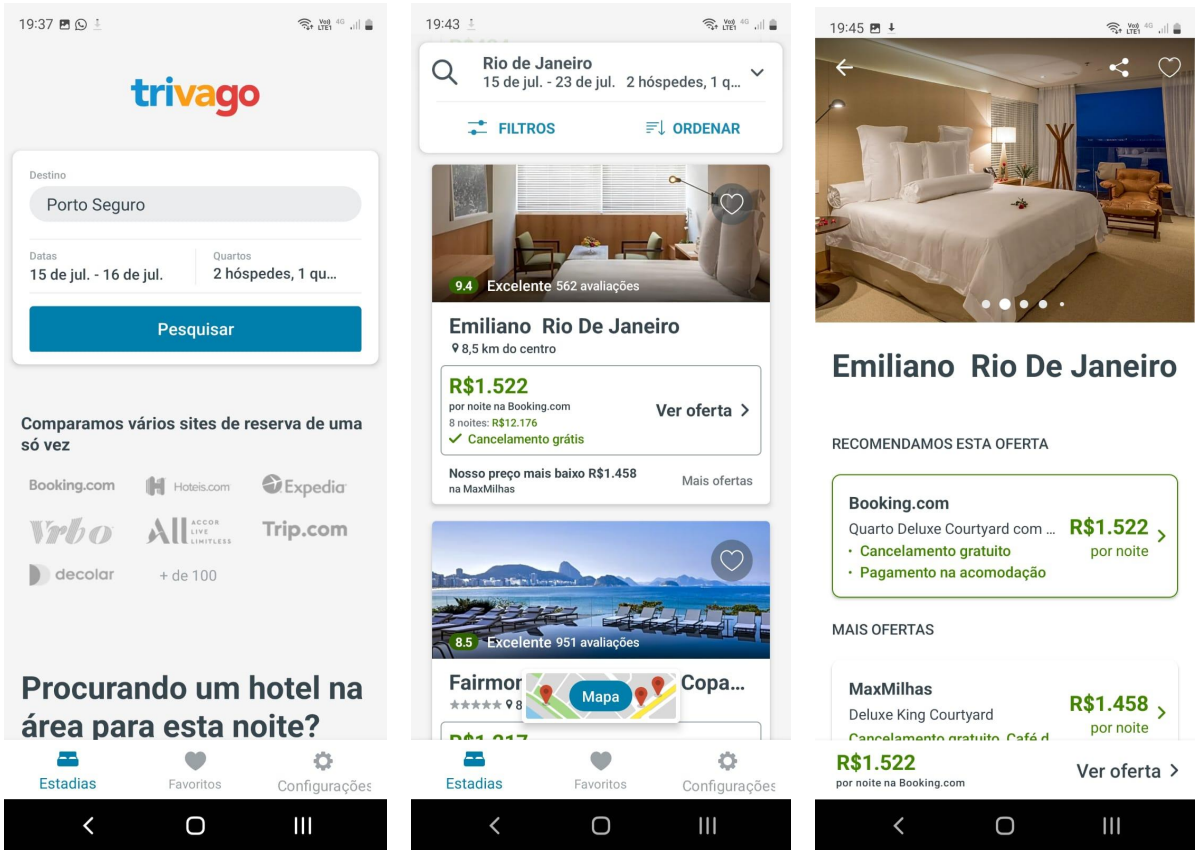


Figura 5. Layout do aplicativo Trivago

6. ELICITAÇÃO DE REQUISITOS

Com a finalidade de identificar alguns requisitos para o desenvolvimento da aplicação foi realizada uma pesquisa e enviada para cerca de 350 pessoas. O público alvo abordado foi uma população entre 18 e 65 anos de idade, gênero e profissões diversas. O critério para essa seleção de público foi abordar a maior diversidade possível, visto que a proposta não busca um gênero, idade ou profissão específica. Dessa forma, a pesquisa foi enviada para diferentes grupos, através de redes sociais, como facebook e whatsapp. Ao final da pesquisa, foram obtidos 229 participantes, gerando uma taxa de resposta de 0,65. Assim, através de algumas perguntas, foi possível obter quais os maiores problemas existentes em aplicativos de receitas.

Conforme a Figura 6, 77,7% dos entrevistados têm o hábito de cozinhar, e 75,3% deles cozinham mais de 1 vezes na semana.

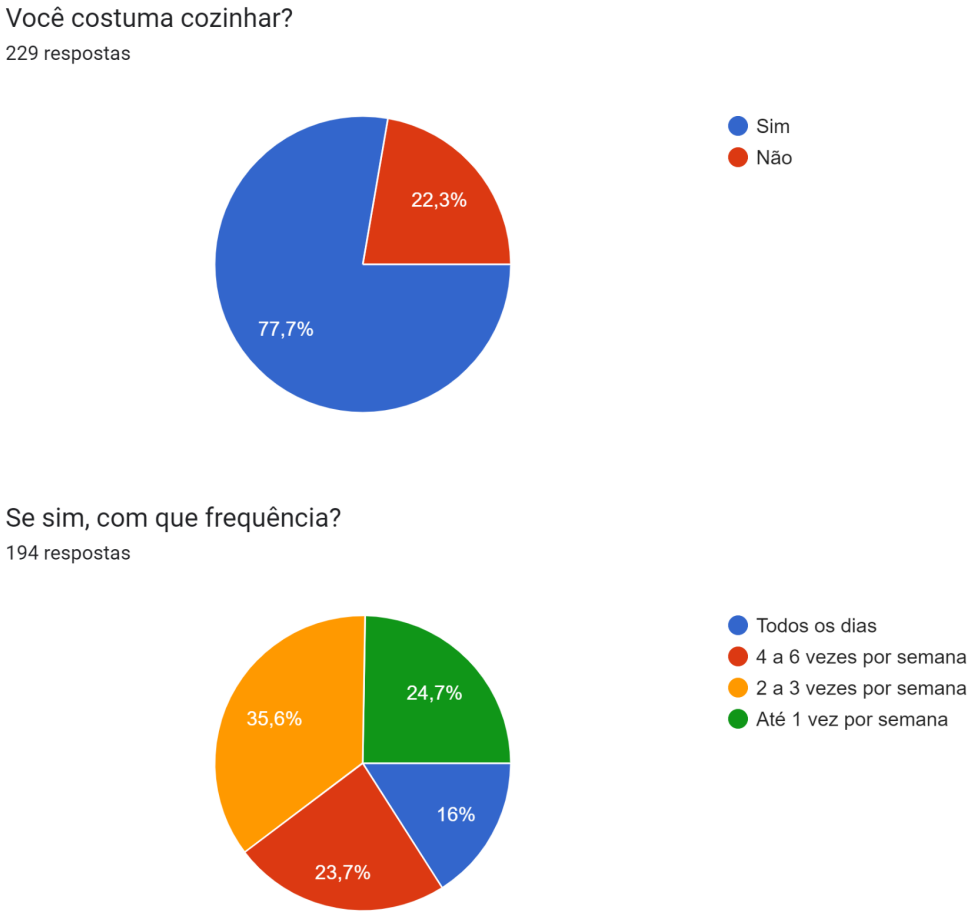


Figura 6. Perguntas sobre o hábito de cozinhar

Além disso, conforme é possível observar na Figura 7, 76,8% dos entrevistados acessam sites ou aplicativos de receitas culinárias. Quando perguntados sobre quais os maiores problemas ao acessar sites e aplicativos de receitas, 26,4% dos entrevistados afirmaram ser propagandas, 10,9% reclamaram da falta de clareza na exibição do conteúdo, 21,4% afirmaram que um grande desafio é ter que buscar diversos sites e aplicativos até achar uma receita boa, enquanto 22,7% declararam que sofrem por todos esses problemas.

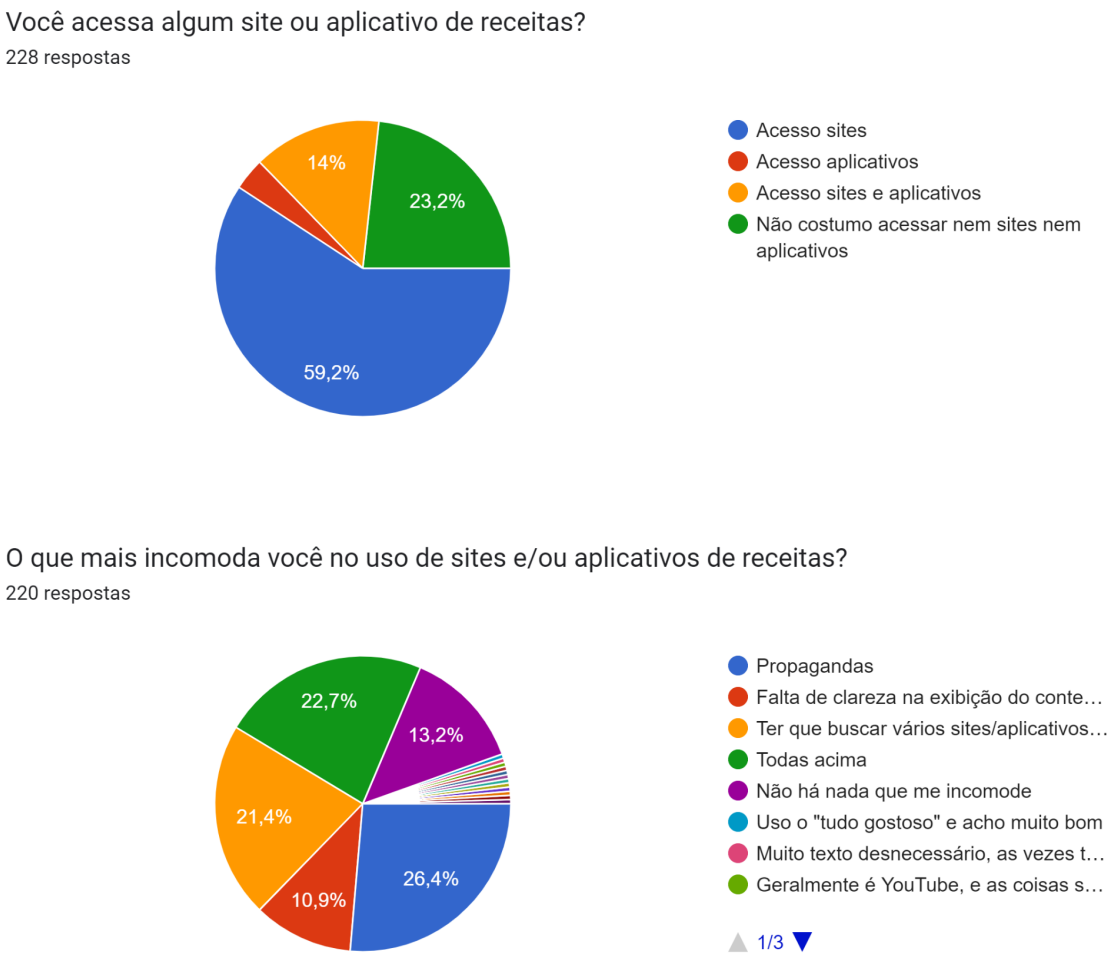


Figura 7. Perguntas sobre a utilização de aplicativos e sites de receitas

Por fim, foi realizada uma breve descrição de como seria a aplicação proposta neste trabalho, e perguntado se teria interesse em utilizar tal aplicativo. Conforme é possível ver na Figura 8, 61,8% afirmaram que usariam, e 32,9% dos entrevistados alegaram que também utilizariam.

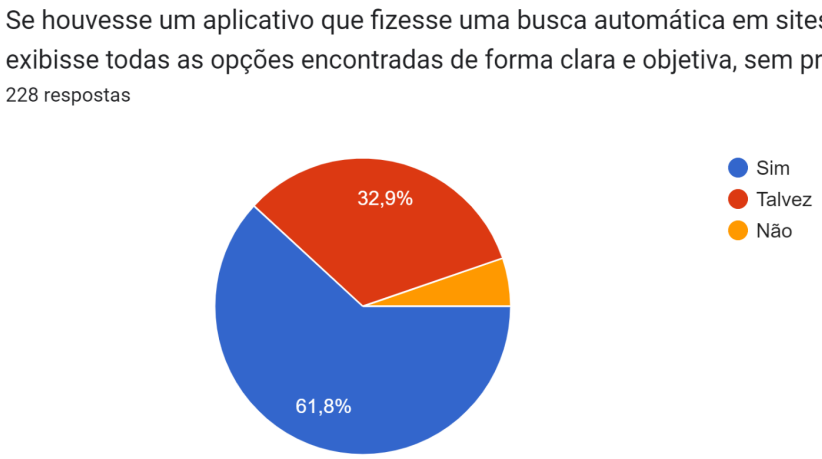


Figura 8. Utilização da aplicação proposta

Dessa forma, a partir das respostas ao questionário e das demais pesquisas realizadas referente à busca de receitas culinárias, foram determinados os requisitos funcionais e não funcionais necessários para o desenvolvimento do presente trabalho.

6.1 Requisitos funcionais

O primeiro requisito funcional é que o software seja capaz de realizar busca em sites de receitas. Essa busca deverá retornar uma lista de receitas encontradas, que possam ser acessadas pelos usuários.

Então, a aplicação deve realizar a formatação dos dados capturados, para exibir eles em uma única formatação, removendo propagandas e informações não essenciais, permitindo melhor entendimento por parte dos usuários. Assim, o segundo requisito deve ser a formatação dos dados encontrados.

Por fim, o aplicativo deve ser capaz de permitir que, dentre a lista de receitas encontradas a partir de uma palavra chave, o usuário possa acessar uma receita, e visualizar as informações necessárias para sua realização (como ingredientes e

modo de preparo), também formatadas em um único padrão. Então, o terceiro e último requisito funcional é que o software seja capaz de realizar a busca e exibição de uma receita escolhida pelo usuário a partir da palavra chave inserida inicialmente. Assim, o conjunto de requisitos funcionais do projeto fica definido como:

RF01 - O software deve ser capaz de realizar busca em sites de receitas.

RF02 - O software deve realizar a formatação dos dados coletados.

RF03 - O software deve ser capaz de realizar busca e exibição de uma receita escolhida pelo usuário.

6.2 Requisitos não funcionais

Para o desenvolvimento da aplicação, foi escolhido como primeiro requisito não funcional a utilização de linguagem de programação Kotlin, por permitir o desenvolvimento da aplicação para dispositivos android.

Como segundo requisito não funcional, para a raspagem dos dados dos websites, foi escolhida a linguagem de programação Python, por possuir bibliotecas que permitem a implementação de forma mais facilitada, como BeautifulSoup e Selenium.

O terceiro requisito não funcional é a disponibilização de um serviço, com endpoints que permitam que o aplicativo realize o envio da palavra chave inserida pelo usuário, e receba os dados formatados como retorno. Esse serviço também deverá receber qual a receita que o usuário quer acessar, e retornar ela formatada, para que a aplicação possa exibi-la.

RNF01 - O aplicativo deve ser desenvolvido com Kotlin.

RNF02 - O software de metabusca deve ser desenvolvido com Python.

RNF03 - O software deve ser disponibilizado em um serviço.

7. DESENVOLVIMENTO

Nessa seção será descrito o processo de desenvolvimento da aplicação proposta realizado até o momento. Para isso, serão abordadas algumas das bibliotecas necessárias, as linguagens de programação escolhidas, e o código implementado.

7.1 Linguagem de programação

Para realizar a implementação do código que irá realizar o processo de metabusca, foi escolhida a linguagem de programação Python[6], por ter uma familiaridade prévia, e por esta possuir algumas bibliotecas fundamentais para o processo de coleta dos dados.

Para a implementação do frontend do aplicativo foi escolhida a linguagem de programação Kotlin[7], por ser a recomendada pelo Google, e também por haver conhecimento prévio.

7.2 Arquitetura e Padrão de Projeto

Com a finalidade de desenvolver o proposto projeto, foi necessária a implementação de um serviço web, para que o aplicativo realize requisições, obtendo os dados necessários. Esse serviço foi implementado utilizando o framework Flask, por ser um dos mais utilizados no desenvolvimento de APIs com Python, e pela facilidade de implementação.

Para isso, foram definidos dois endpoints. O primeiro deles, */search*, que é responsável por retornar uma lista de receitas obtidas através da busca com uma palavra-chave. Por exemplo, quando o usuário buscar no aplicativo por “pudim”, este deverá realizar uma requisição do tipo GET, enviando a palavra ‘pudim’, e retornando uma lista de resultados encontrados para essa palavra chave. O segundo endpoint, */recipe*, irá ser consultado quando o usuário clicar em uma das receitas retornadas ao realizar a busca inicial. Esse endpoint deve passar uma

[6] https://www.python.org/
[7] https://kotlinlang.org/

identificação para saber qual das receitas deve ser consultada, e retornar todos os dados da receita em formato .json.

Figura 9. Arquitetura da comunicação entre frontend e backend

Para o desenvolvimento da arquitetura do aplicativo, foi definida a implementação em três camadas:

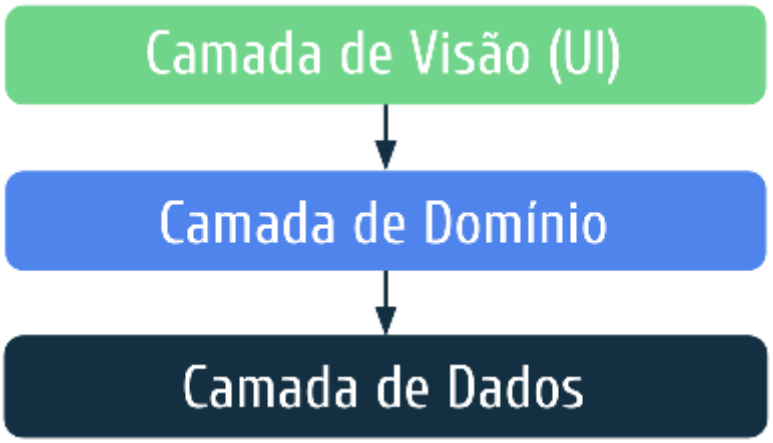


Figura 10. Arquitetura do aplicativo mobile. Fonte: https://developer.android.com/topic/architecture

- Camada de Visão, responsável pela exibição dos dados da aplicação na tela.
- Camada de domínio, responsável por encapsular a lógica de negócios, simplificando-a e realizando a comunicação entre a interface gráfica (camada de visão) e as camadas de dados.
- Camada de dados, que possui a lógica de negócios, como os repositórios, e é responsável por expor os dados ao restante do app, abstraindo fontes de dados.

O padrão de projeto escolhido para desenvolver o aplicativo foi o MVVM[8] (Model View View Model). Para isso, as camadas do código foram separadas em três partes:

- Model: Responsável pela abstração dos dados. Model e ViewModel realizam a obtenção e salvam os dados.
- View: Camada responsável por informar o ViewModel sobre as ações realizadas pelo usuário. A View tem visibilidade do ViewModel, e não possui nenhum tipo de lógica de aplicativo.
- ViewModel: Realiza os fluxos de dados necessários para a camada de Visualização (View), conectando o Modelo e a View.

Figura 11: Arquitetura MVVM da aplicação.

Fonte: https://www.devmedia.com.br/entendendo-o-pattern-model-view-viewmodel-mvvm/18411

7.3 Estrutura de dados

Para o desenvolvimento desta aplicação, foi necessário inicialmente definir a estrutura de dados na qual serão inseridos os dados referentes às receitas. Primeiramente, foi definida a classe SearchResult, que irá armazenar a resposta de uma busca do usuário, que é uma lista de receitas, apenas com informações reduzidas. Conforme figura 12, os dados necessários são o *author* (site origem do resultado da busca), a URL da imagem e o título da receita, o índice da receita na lista (utilizado para posterior ordenação de resultados), o slug (trecho da URL de uma receita específica), e o título da receita.

[8] https://www.digitalocean.com/community/tutorials/android-mvvm-design-pattern

```python
class SearchResult:
    def __init__(self, author, image_url, index, slug, title):
        self.author = author
        self.image_url = image_url
        self.index = index
        self.slug = slug
        self.title = title
```

Figura 12. Classe SearchResult

Também foi implementada a classe Steps, que é responsável por armazenar os dados de cada preparo da receita. Para cada preparo, há um título, uma lista de ingredientes, e uma lista de modo de preparo.

```python
class Steps:
    def __init__(self, title, ingredients, preparation):
        self.title = title
        self.ingredients = ingredients
        self.preparation = preparation
```

Figura 13. Classe Steps

Além disso, foi definida a classe que possui os dados da receita, com o nome de Recipe. Conforme é possível observar na Figura 14, os dados necessários para a receita são o título da receita, a URL da imagem, o slug, o nome do site de origem da receita, a quantidade de porções, o tempo de preparo e os passos necessários para realizar a receita.

```python
class Recipe:
    def __init__(self, title, image_url, slug, author, serves,
prepare_time, steps):
        self.title = title
        self.image_url = image_url
        self.slug = slug
        self.author = author
        self.serves = serves
        self.prepare_time = prepare_time
        self.steps = steps
```

Figura 14. Classe Recipe

7.4 Extração de dados de website

No início do desenvolvimento da aplicação, foi escolhido o site Panelinha[9] para realizar o processo de metabusca. A escolha do site Panelinha foi realizada por ser um site com uma grande variedade de receitas e não possuir ainda uma versão em aplicativo. Posteriormente, em trabalhos futuros, serão escolhidos outros websites, para poder ter uma base maior de receitas.

Para realizar as buscas de receitas, primeiramente foi necessária a criação de dois endpoints. O primeiro, */search*, que acessa o método do_search(), recebe a palavra chave buscada pelo usuário, e envia ela para o método get_search_result(), que irá realizar a busca de uma lista de resultados encontrados a partir da palavra chave consultada. O segundo endpoint, */recipe*, acessa o método open_recipe(), recebe o nome da receita clicada pelo usuário e o website da onde ele irá buscar os dados da receita, e envia para o método get_recipe(), que realiza a extração de todos os dados de uma receita, e a retorna para o aplicativo.

```
from flask import Flask, request
from scrapingPanelinha import *
from policy import *

app = Flask(__name__)

@app.route('/search')
def do_search():
    searched_word = request.args.get('word')
    return get_search_result(searched_word)

@app.route('/recipe')
def open_recipe():
    slug = request.args.get('slug')
    author = request.args.get('author')
    return get_recipe(slug, author)

if __name__ == '__main__':
    app.run()
```

Figura 15. endpoints para a busca de uma lista de resultados de uma busca e de uma receita

[9] https://www.panelinha.com.br/

Para realizar a busca de alguma receita, primeiramente foi necessário implementar o método get_search_result(), que recebe como parâmetro a string word, que representa a palavra chave da receita que o usuário pesquisou no aplicativo. Então, word é enviada para o método get_search_from_panelinha, sendo que o resultado da busca é armazenado em result_from_panelinha.

```
def get_search_result(word):

    result_from_panelinha = get_search_from_panelinha(word)
    all_lists_combined = order_results_equally(result_from_panelinha)

    return all_lists_combined
```

Figura 16. Implementação do método get_search_result()

Posteriormente, result_from_panelinha é enviado para order_results_equally, para ordenar com outros resultados, no caso de outras implementações de busca além do panelinha. Conforme é possível observar na Figura 17, o método order_results_equally() recebe uma concatenação de todas as listas de resultados obtidos de websites, e ordena eles pelo índice de cada elemento de cada lista que foi concatenada. Por exemplo, se houvesse um *result_from_other_site*, então:

all_lists_combined =

order_results_equally(result_from_panelinha+result_from_other_site)

Dessa forma, a lista final é ordenada de forma igualitária entre as fontes.

```
def order_results_equally(all_lists_combined):
    sorted_list = sorted(all_lists_combined, key = itemgetter(2))
    list_of_search_result = []
    for i in range(len(sorted_list)):
        search_item = SearchResult(
            author = sorted_list[i][0],
            image_url = sorted_list[i][1],
            index = sorted_list[i][2],
            slug = sorted_list[i][3],
            title = sorted_list[i][4]
        )
        list_of_search_result.append(search_item.__dict__)
    return list_of_search_result
```

Figura 17. Implementação do método order_results_equally()

A função get_search_from_panelinha() é responsável por, a partir da palavra chave recebida, concatenar com a URL do serviço do Panelinha, e buscar a lista completa de resultados que retornam a partir daquela busca inicial.

Para isso, foi utilizada a biblioteca *Requests*, que realiza a requisição a uma URL do site. Conforme é possível observar na figura 18, o retorno dessa requisição é um arquivo convertido para .json, que possui a lista de resultados da consulta do usuário. Com isso, é utilizado um laço *for* para capturar apenas os dados necessários e adicioná-los em uma lista, que será retornada para o aplicativo.

```
def get_search_from_panelinha(word):

    search_url =
"https://panelinha-api-server-prod.herokuapp.com/v1/search?pageSize=100
0&title="+word
    search_request = requests.get(search_url).json()

    search_list = []
    results = search_request['data']['results']
    for i in range(len(results)):
        author = "panelinha"
        image_url = results[i]['imageUrl']
        index = i
        slug = results[i]['slug']
        title = results[i]['title']
        recipe_found = [author, image_url, index, slug, title]

        if results[i]['imageFolder'] == 'receita':
            search_list.append(recipe_found)

    return search_list
```

Figura 18. Requisição e retorno de uma lista de resultados a partir de uma busca por palavra chave

A partir de então, foi iniciado o processo de captura dos dados das receitas. Para obter esses dados, foi implementada a função get_recipe(), que recebe como parâmetro a string slug, que representa o nome da receita clicada pelo usuário, e a string author, que representa o website de origem da receita selecionada.

Então, em recipe é armazenado um objeto Recipe com dados inicialmente vazios. Conforme demonstrado na figura 19, a partir da string author é possível saber qual a origem da receita escolhida e, então, poder executar o método correto conforme a fonte.

```
def get_recipe(slug, author):

    recipe = Recipe("","","","","","","")
    if author == "panelinha":
        recipe = get_recipe_from_panelinha(slug)

    return recipe.__dict__
```

Figura 19. Implementação do método get_recipe()

Com a finalidade de obter os dados das receitas do site Panelinha, foi implementado o método get_recipe_from_panelinha(), demonstrado na figura 20.

O resultado da requisição utilizada para capturar os dados da receita utilizava os ingredientes com ids, ao invés da descrição dos nomes, o que impossibilitou obter a lista de ingredientes através da requisição. Por causa dessa particularidade, foi utilizada a biblioteca BeautifulSoup, para realizar o web scraping do site.

A biblioteca BeautifulSoup, também do Python, realiza o scraping do site, ou seja, captura tudo o que é exibido na tela, através de código HTML. A partir disso, foram capturados os dados, filtrando pela classe na qual são exibidos os ingredientes, e armazenados em uma lista de listas de ingredientes, uma para cada preparo.

```
def get_recipe_from_panelinha(slug):
    url_bs = "https://www.panelinha.com.br/receita/"+slug
    data = requests.get(url_bs)
    page_bs = BeautifulSoup(data.content, 'html.parser')

    base = page_bs.find_all('div', attrs={'class': 'col-xs-12 col-sm-6
col-md-7'})
    ingredients_and_instructions = base[1].findAll('div',
class_='editor ng-star-inserted')
    ingredients_list = []

    for i in range(0, len(ingredients_and_instructions), 3):
```

```
        ingredients_only =
ingredients_and_instructions[i].findAll('li',
class_='ng-star-inserted')
        ingredients_in_intruction = []
        for ing in ingredients_only:
            ingredients_in_intruction.append(ing.text)
        ingredients_list.append(ingredients_in_intruction)
```

Figura 20. Captura da lista de ingredientes de uma receita escolhida

Com isso, para finalizar o processo de captura dos dados de uma receita escolhida pelo usuário. Conforme é possível observar na Figura 21, o resultado dessa requisição é também um arquivo convertido para .json, onde são capturados os valores necessários, e criada a lista de steps, que representa os processos. Após a criação de cada Step é adicionado o título, a lista de ingredientes, e as instruções de cada preparo, e posteriormente adicionado à stepList. Na sequência, é realizada a criação do objeto Recipe, e armazenado nele todas as informações referente à receita pesquisada.

```
recipe_url =
"https://panelinha-api-server-prod.herokuapp.com/v1/receita/"+slug
    recipe_request = requests.get(recipe_url).json()
    result = recipe_request['data']
    content = result['content']
    recipe_steps = content['recipeSteps']
    steps_updated = []
    for i in range(len(recipe_steps)):
        if 'ingredients' in recipe_steps[i]:
            steps_updated.append(recipe_steps[i])

    steps_list = []
    for i in range(len(steps_updated)):
        step_preparation = steps_updated[i]['body']
        formatted_step_preparation = BeautifulSoup(step_preparation,
'html.parser').getText()
        steps = Steps(
            steps_updated[i]['title'],
            ingredients_list[i],
            formatted_step_preparation
        )
```

```
        steps_list.append(steps.__dict__)

    recipe = Recipe(
        result['title'],
        result['imageUrl'],
        result['slug'],
        content['author'],
        content['serves'],
        content['prepareTime'],
        steps_list
    )

    return recipe
```

Figura 21. Armazenamento de todos os dados de uma receita

Após gerar o objeto Recipe, o sistema irá retornar esses dados para a aplicação android, responsável por ler essas informações já formatadas, e exibir em um layout adequado para o usuário.

Após a definição das estruturas de dados, e da implementação do código de extração de dados, foi definido um padrão de .json a ser seguido para padronizar as buscas a outros sites a serem realizados o scraping. O método get_search_result() deve seguir o padrão conforme demonstrado na figura 22.


```
[
  {
    "author": "panelinha",
    "image_url": "https://cdn.panelinha.com.br/receita/1657291945626-pudim-de
      -mandioca%20com%20cumaru.jpg",
    "index": 0,
    "slug": "pudim-de-mandioca-com-cumaru",
    "title": "Pudim de mandioca com cumaru (reaproveitamento)"
  },
  {
    "author": "panelinha",
    "image_url": "https://cdn.panelinha.com.br/receita/1667481994336-pudim
      .jpg",
    "index": 1,
    "slug": "pudim-na-airfryer",
    "title": "Pudim de leite na AirFryer da Rita Lobo"
  }
]
```


Figura 22. json representando a padronização de get_search_result()

Além disso, também foi definido que o método get_recipe() deve seguir um padrão, conforme é possível observar na figura 23.


```
{
  "author": "Panelinha",
  "image_url": "https://cdn.panelinha.com.br/receita/1657291945626-pudim-de
    -mandioca%20com%20cumaru.jpg",
  "prepare_time": "Mais de 2h",
  "serves": "Mais de 6 porções",
  "slug": "pudim-de-mandioca-com-cumaru",
  "steps": [
    {
      "ingredients": [
        "1 xícara (chá) de açúcar",
        "½ xícara (chá) de água fervente"
      ],
      "preparation": "\nSepare 8 forminhas lisas com capacidade para 150 ml.
        Leve uma chaleira com 1 litro de água ao fogo médio – ela vai ser
        usada para fazer a calda e para assar o pudim em banho-maria.\n",
      "title": "PARA A CALDA"
    },
    {
      "ingredients": [
        "1½ xícara (chá) de mandioca cozida, em temperatura ambiente",
        "3 ovos",
        "1 xícara (chá) de leite",
        "¾ de xícara (chá) de açúcar",
        "cumaru ralado na hora a gosto"
      ],
      "preparation": "\nPreaqueça o forno a 160 ºC (temperatura baixa).",
      "title": "PARA O PUDIM"
    }
  ],
  "title": "Pudim de mandioca com cumaru (reaproveitamento)"
}
```


Figura 23. json representando a padronização de get_recipe()

7.5 Implementação da aplicação mobile

Para a implementação da aplicação mobile, foi definida a classe AppViewModel, que é responsável por realizar a comunicação entre os dados e a interface gráfica da aplicação.

Conforme é possível observar na Figura 24, são definidos searchResult e recipeResult para lidar com o resultado das requisições de getSearchResult() e getRecipe().

```kotlin
@HiltViewModel
class AppViewModel @Inject constructor(
    private val recipesUseCase: RecipesUseCase,
) : ViewModel(), CustomCoroutineScope {

    override val fragmentExceptionHandler =
        CoroutineExceptionHandler { _, throwable ->
            _searchResult.postValue(SearchResult.Failure(throwable))
            _recipeResult.postValue(RecipeResult.Failure(throwable))
        }

    private val _searchResult = MutableLiveData<SearchResult>()
    val searchResult: LiveData<SearchResult> get() = _searchResult

    private val _searchResponse = MutableLiveData<List<Search>>()
    val searchResponse: LiveData<List<Search>> get() = _searchResponse

    private val _recipeResult = MutableLiveData<RecipeResult>()
    val recipeResult: LiveData<RecipeResult> get() = _recipeResult

    private val _recipeResponse = MutableLiveData<Recipe>()
    val recipeResponse: LiveData<Recipe> get() = _recipeResponse

    fun requestSearch(searchedWord: String) {
        viewModelScope.safeLaunch { this: CoroutineScope
            _searchResult.postValue(SearchResult.Loading)
            val searchInfo = recipesUseCase.getSearchResult(searchedWord)
            _searchResponse.postValue(searchInfo)
            _searchResult.postValue(SearchResult.Completed)
        }
    }

    fun buildRecipe(searchedRecipe: String, author: String) {
        viewModelScope.safeLaunch { this: CoroutineScope
            _recipeResult.postValue(RecipeResult.Loading)
            val recipeInfo = recipesUseCase.getRecipe(searchedRecipe, author)
            _recipeResponse.postValue(recipeInfo)
            _recipeResult.postValue(RecipeResult.Completed)
        }
    }
}
```

Figura 24. Implementação da classe AppViewModel.

As requisições são realizadas de forma assíncrona e, conforme o resultado da requisição, são definidas diferentes ações, como demonstrado na Figura 25.

```
private fun AppViewModel.setupSearchResponseObserver() {
    searchResult.observe(viewLifecycleOwner) {  it: SearchResult!
        when(it) {
            is SearchResult.Completed -> {
                searchResponse.value?.let { response ->
                    setTitle(wordSearched)
                    hideLoading()
                    binding.searchRecycler.setHasFixedSize(false)
                    binding.searchRecycler.adapter = SearchAdapter(
                        response, itemSearchClickListener()
                    )
                }
            }
            is SearchResult.Loading -> {
                showLoading()
            }
            is SearchResult.Failure -> {
                hideLoading()
            }
        }
    }
}
```

Figura 25. Implementação de ações conforme estado de SearchResult.

Conforme demonstrado na Figura 26, para realizar a exibição da lista de resultados da busca, foi definida uma RecyclerView, que permite melhor gerenciamento de uma quantidade de itens encontrados.

```
class SearchAdapter(
    private val searchList: List<Search>,
    private val searchClickListener: danihmello.tcc.recipes.ItemSearchClickListener
) : RecyclerView.Adapter<SearchViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): SearchViewHolder {
        val layoutInflater = LayoutInflater.from(parent.context)
        return SearchViewHolder(
            SearchItemBinding.inflate(layoutInflater, parent,  attachToParent: false),
            searchClickListener
        )
    }

    override fun onBindViewHolder(holder: SearchViewHolder, position: Int) {
        holder.bind(searchList[position])
    }

    override fun getItemCount(): Int = searchList.size
}
```

Figura 26. Implementação de RecyclerView para lista de resultados da busca.

7.6 Layout, User Interface e User Experience

Para a implementação da parte visual da aplicação, foram definidas três telas principais. A primeira delas é a tela inicial, onde se encontra o nome do aplicativo, e um campo para que o usuário possa buscar o nome da receita que deseja encontrar. Para essa tela houve a preocupação de deixar todo o campo de busca com área de toque ativa, para cumprir requisitos de acessibilidade, ou seja, para que usuários com deficiência visual tenham maior facilidade de utilizar o aplicativo. Além disso, o nome do app - MixReceitas - aparece em destaque, com a finalidade de manter a hierarquia visual.

Figura 27. Tela inicial do aplicativo

A segunda tela consiste em uma lista com todos os resultados encontrados para a receita buscada. Além disso, há um cabeçalho, onde o usuário pode ver o nome do item que ele buscou. Cada item encontrado possui uma imagem, que

preenche todo o espaço do item, com a finalidade de facilitar a visualização dos resultados. Os nomes das receitas são exibidos sobre a imagem, em frente a um degradê preto-transparente, para facilitar a leitura.

Figura 28. Tela de resultados encontrados

Por fim, a tela da receita é exibida quando o usuário clica em uma receita que deseja consultar. Para essa tela optou-se por exibir a imagem da receita no topo da tela, juntamente com o título, mantendo a disposição da tela de resultados de pesquisa. Em casos que a receita tenha mais de um preparo, cada um deles é exibido separadamente, em tópicos, para maior compreensão de hierarquia e da receita em si. Por exemplo, na figura 29, é possível ver uma receita que possui 2 preparos: a calda e o pudim. Dessa forma, eles estão divididos, e cada um deles possui ingredientes e modo de preparo também demonstrados separadamente.

Figura 29. Tela de exibição da receita

8. PESQUISA DE RESULTADOS

Após o desenvolvimento do aplicativo, foi realizada uma Pesquisa de Resultados com a finalidade de avaliar o resultado obtido, e enviada para cerca de 350 pessoas. O público alvo foi uma população entre 18 e 65 anos de idade, com gênero e profissões diversas. O aplicativo foi disponibilizado na loja da Google (Play Store), com o nome de MixReceitas, para facilitar o download e análise pelos participantes. Ao final da pesquisa foram obtidos 19 participantes, gerando uma taxa de resposta de 0,054.

Inicialmente, foram realizadas algumas perguntas com o objetivo de identificar o público, questionando idade, gênero e profissão. Essas perguntas foram realizadas para identificar alguma particularidade que pudesse ocorrer. As figuras 30, 31 e 32 demonstram o resultado desses questionamentos.

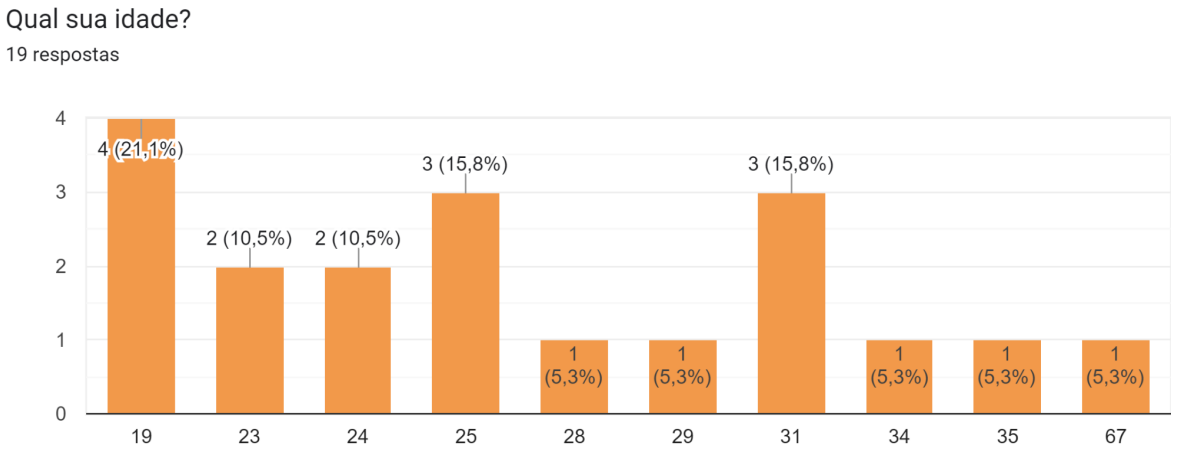


Figura 30: Idades do público que respondeu a pesquisa.

Profissão	**Número de respostas**
Arquiteta	1
Dentista	1
Desenvolvedor	11
Estudante	5
Quality Assurance	1

Figura 31: Profissões do público que respondeu a pesquisa.

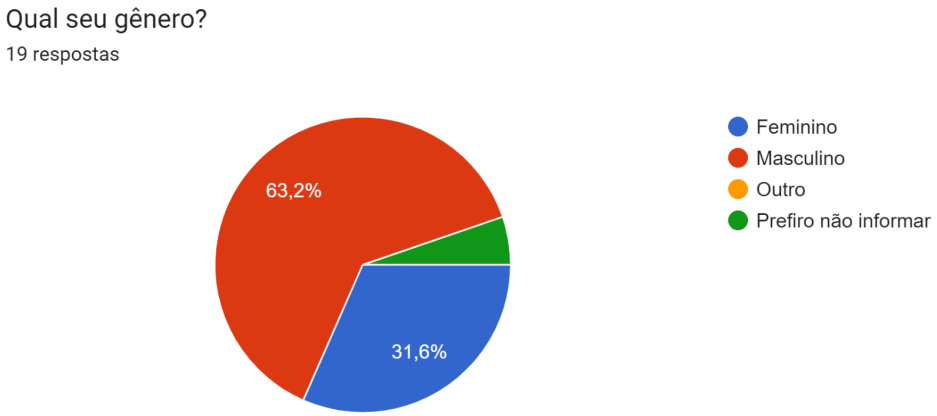


Figura 32: Gêneros do público que respondeu a pesquisa.

Como é possível observar, apesar do público que foi enviada a pesquisa ter sido diversificado, houve uma maior concentração de respostas entre 18 e 35 anos de idade, sendo 63,2% das respostas de pessoas do gênero masculino. Além disso, mesmo tendo enviado para cerca de 350 pessoas, foram obtidas apenas 19 respostas, ao contrário da pesquisa de Elicitação de Requisitos, que obteve 229 respostas. Acredita-se que esse fato seja decorrente da necessidade de realizar o download de um aplicativo, o que desencoraja o engajamento na pesquisa, por ser mais trabalhoso, por medo de baixar um aplicativo novo, ou mesmo por falta de espaço de armazenamento no dispositivo. Também tem o fato de apenas usuários de Android terem a possibilidade de responder a pesquisa de Resultados.

Com o objetivo de analisar o resultado da implementação do aplicativo, foram realizadas algumas perguntas referente a usabilidade do software. Conforme é possível observar nas figuras 33 e 34, todos os usuários conseguiram encontrar alguma receita. Apesar disso, alguns deles tiveram dificuldade em encontrar algumas receitas específicas.

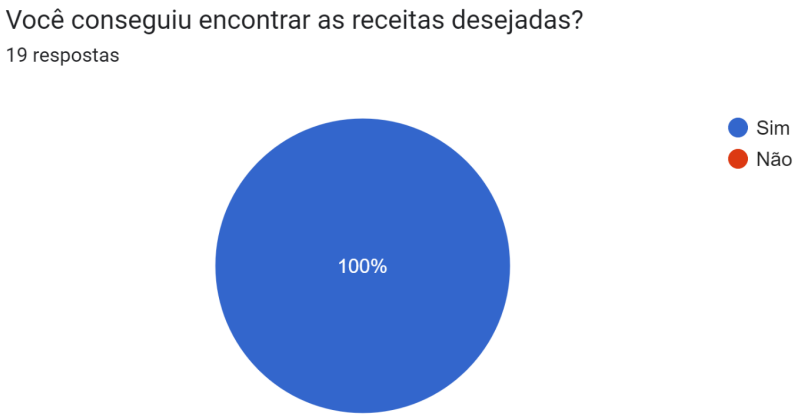


Figura 33: Usuários que encontraram as receitas

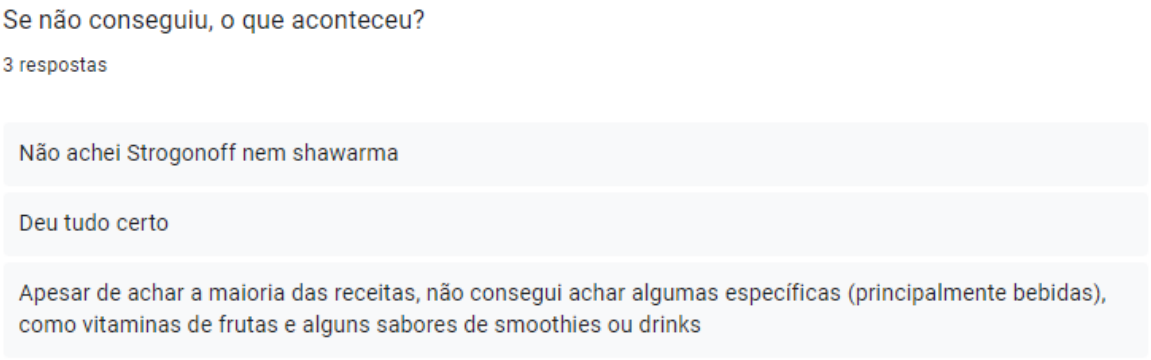


Figura 34: Usuários que obtiveram alguns problemas na busca.

Também foram realizadas algumas perguntas referente a facilidade de utilização da aplicação. As figuras 35, 36 e 37 demonstram os resultados obtidos.

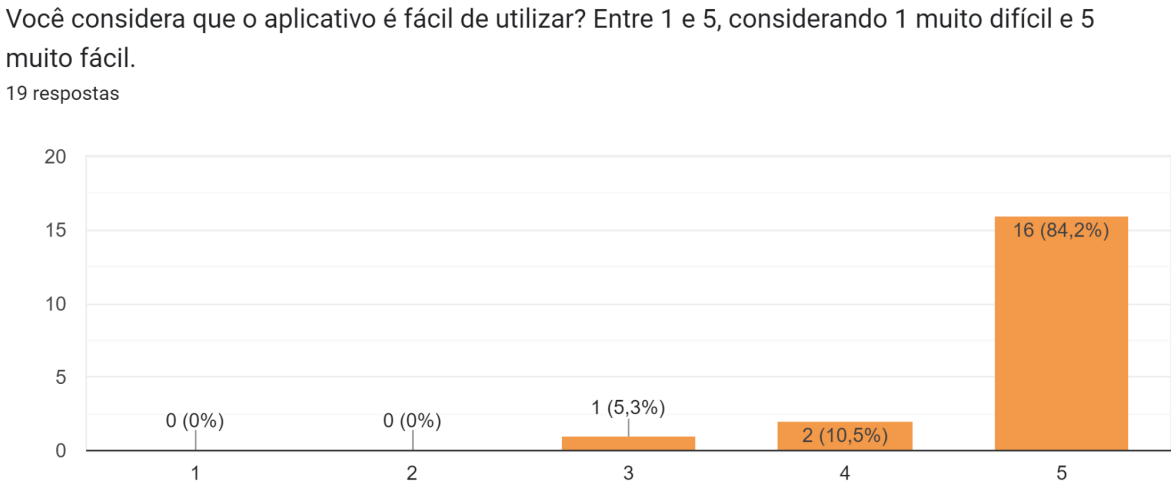


Figura 35: Análise da facilidade de uso do aplicativo.

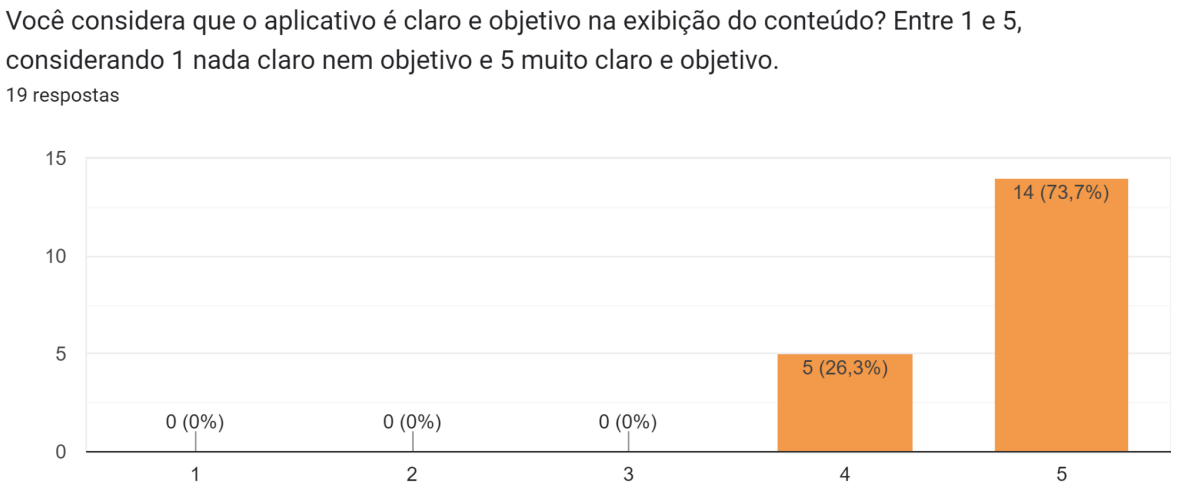


Figura 36: Análise da clareza de exibição do conteúdo.

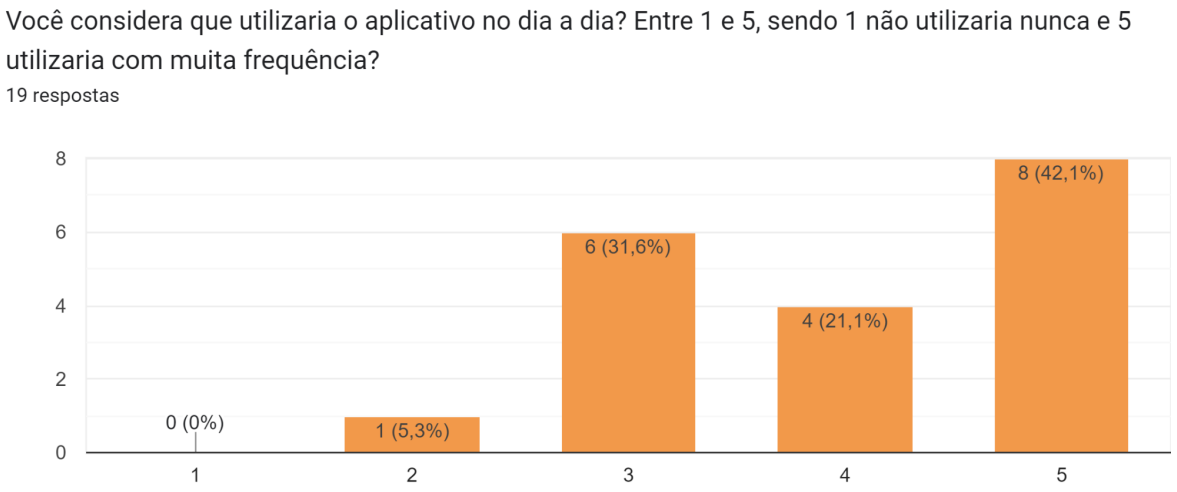


Figura 37: Análise da frequência de probabilidade de uso do aplicativo.

Com isso, é possível observar que a maioria dos usuários achou o aplicativo muito fácil de utilizar e com muita clareza na exibição do conteúdo, atingindo então os objetivos do presente trabalho. Quanto à frequência de utilização do aplicativo, 94,7% dos usuários responderam que o utilizariam com alguma frequência relevante.

Por fim, foi questionado aos usuários se após a utilização da aplicação eles tinham alguma sugestão de melhoria. A figura 38 demonstra as sugestões relatadas por eles.

Você teria alguma sugestão de melhoria no aplicativo?

13 respostas

O pesquisar por certas categorias de comida o aplicativo geralmente sempre mostra sugestões de pratos nem sempre totalmente relacionados a categoria; como ao pesquisar por pizza, onde uma das receitas sugeridas é scone

Um passo da receita era "assista o passo a passo em vídeo" porém não tem como ver o vídeo

Mais receitas

Excelente app

seria interessante informar a fonte daquela receita específica, se foi de um site, Internet, programa etc.

aparecer as receitas disponíveis

Iniciar o app com algumas sugestões na Home, as vezes o usuário não tá com nenhuma receita em mente e vai nas sugestões.

Favoritar e compartilhar é fundamental para esse tipo de aplicativo

Poderia aparecer mais variedades de receita para alguns pratos. Além disso, poderiam ter filtros (entrada, prato principal, sobremesa, salgado, doce, lanche) para quem não tem um prato específico em mente poder navegar e escolher algo para cozinhar

Poderia ter uma lista de receitas mais buscadas ou com mais resultados

Algumas pesquisas trazem poucos resultados (tipo "brigadeirão"), mas não fiz nenhuma pesquisa que não trouxesse ao menos um resulto o que foi positivo. Aumentar a base de pesquisa aumentaria o numero de respostas eu presumo.

Muito bacana o app Dani, ficou show! Como sugestão de melhoria seria legal o app exibir uma mensagem de que não encontrou nenhum resultado pra determinada receita, ou exibir algo próximo do que o usuário digitou, por exemplo, procurei por cheesecake, mas na primeira tentativa escrevi errado (cheescake), dessa forma não tive nenhum resultado. Em uma segunda tentativa, com a palavra escrita corretamente, o app exibiu os resultados.

Apesar de ser um aplicativos simples e objetivo, acredito que seria interessante uma opção de login para que fosse possível favoritar as receitas preferidas direto na conta do usuário e não depender de salvar essas receitas apenas no dispositivo. Só se fosse o caso de ter a opção de fazer um backup e importar ele de volta pro app quando instala-lo novamente. Outra questão que observei, é que ao pesquisar "bolo de cenoura" por exemplo, os resultados mais abaixo começam a fugir um pouco da pesquisa, indo para pratos salgados e diferentes como "lasanha bolonhesa". Mas no geral o app está bem bonito, intuitivo e de fácil manuseio.

Figura 38: Sugestões de melhoria do aplicativo

9. CONCLUSÃO

Neste trabalho foi desenvolvido um software que realiza o scraping de receitas em um website e o exibe em um aplicativo android, sem propagandas, de forma clara e objetiva. Como parte do trabalho foi realizada uma fundamentação teórica em relação a conceitos necessários para o entendimento de web scraping, além de um estudo de trabalhos correlatos, visando melhor entender aplicativos similares ao proposto nessa proposta.

Após a realização da aplicação proposta, foi possível realizar o scraping de um site de receitas, o Panelinha. Assim, o software consegue acessar o website, e extrair todos os resultados encontrados a partir de uma busca de uma receita específica. O acesso e extração dos dados do Panelinha foi realizado com o uso da biblioteca Requests e Beautiful Soup, ambas do Python. Após a extração dos dados, eles são formatados em um sistema padronizado, no formato .json. Então, são enviados e exibidos em uma aplicação mobile android para o usuário, sem propagandas.

Para a exibição do conteúdo, foram utilizados conceitos de User Experience (UX) e User Interface (UI), como padronização de cores, fontes, tamanhos, além de preocupação com acessibilidade e hierarquia. A preocupação com a acessibilidade se deu, principalmente, pelo tamanho do campo de busca, tamanho de fontes, e contraste de cores. Já os conceitos de hierarquia foram aplicados principalmente na página de receitas, setorizando os passos, ingredientes e modo de preparo.

Após a implementação completa da aplicação, ela foi disponibilizada na loja do Google (Play store), para realizar a Pesquisa de Resultados. O desenvolvimento da aplicação foi realizado apenas com objetivo acadêmico, não havendo intenção de lançar como produto. Dessa forma, a disponibilização do aplicativo foi apenas para finalidade de pesquisa com usuários, e por isso, já foi removida da loja, não sendo mais disponível para novos usuários. Após a realização da Pesquisa de resultados, com as análises feitas pelos usuários participantes, foi possível obter algumas sugestões de melhorias para a aplicação, que foram utilizadas para elencar os trabalhos futuros. Essas melhorias são possíveis tanto na parte de scraping quanto na parte de mobile (front-end).

Primeiramente, na parte de back-end, uma grande melhoria seria realizar a captura de dados de outros sites, além do Panelinha. Essa implementação iria sanar um dos problemas identificados na pesquisa de resultados, como a ausência de algumas receitas específicas. Além disso, outra melhoria seria a implementação de um banco de dados que permita o cadastro de usuários, o que possibilitaria que as receitas preferidas deles fossem salvas.

Já na parte de front-end, seria ideal a possibilidade de compartilhar externamente as receitas encontradas. Para isso, seria necessário gerar um deep link, ou seja, um link que direciona um usuário para uma página específica do aplicativo. Além disso, a implementação de filtros é uma boa chance de melhorar a aplicação, possibilitando que o usuário selecione as receitas por categorias, como pratos principais, doces, lanches, etc, ou mesmo por ingredientes. Por fim, a exibição de sugestões de receitas logo na página inicial também é de grande utilidade para a aplicação.

10. REFERÊNCIAS

BELLIS, Mary. 2019. **The History of Kitchen Appliance Inventions.** Acesso em 11/10/2022. Disponível em: https://www.thoughtco.com/history-of-kitchen-appliance-inventions-1992036

BUDKEWICZ, Matheus. **Como fazer webscraping com Python e Beautiful Soup.** 2018. Acesso em 05/09/2022. Disponível em https://medium.com/horadecodar/como-fazer-webscraping-com-python-e-beautiful-soup-28a65eee2efd

CRUMMY. **Beautiful Soup Documentation**. Acessado em 10/08/2022. Disponível em https://www.crummy.com/software/BeautifulSoup/bs4/doc/

CUYLENBURG, Jessica van. **World Cuisine: How to Easily Find Recipes From Other Cultures.** 2021. Disponível em: https://www.nerdynaut.com/world-cuisine-how-to-easily-find-recipes-from-other-cultures Acesso em: 23/10/2022

Design Consistency Guide: Best Practices for UI and UX Designers. Acesso em 10/06/2022. Disponível em: https://www.uxpin.com/studio/blog/guide-design-consistency-best-practices-ui-ux-designers/

FERREIRA, W., BENEVENUTO, F., SILVA, A.P., MERSCHMANN, L. **Comer, Comentar e Compartilhar: Análise de uma Rede de Ingredientes e Receitas.** Simpósio Brasileiro de Sistemas Colaborativos (SBSC). Manaus, Brasil. Outubro, 2013.

HOWE, A., DREILINGER, D. **SavySearch: A Metasearch Engine That Learns Which Search Engines to Query.** AI Magazine, Volume 18 Number 2. 1997.

JOO, Heonsik. **A Study on Understanding of UI and UX, and Understanding of Design According to User Interface Change.** International Journal of Applied Engneering Research ISSN. Volume 12, Number 20, 2017.

KAUSAR, M., DHAKA, V., SINGH, S., **Web Crawler: A Review.** International Journal of Computer Applications. Fevereiro, 2013.

KLINGER, A., LIMA, J. V, OLIVEIRA, J. P. M., **Visibilidade Web Baseada em Metabusca.** Cadernos de Informática. Volume 6, Número 1, 2011. Disponível em: https://www.seer.ufrgs.br/index.php/cadernosdeinformatica/article/view/v6n1p71-77/11739. Acesso em: 05/07/2022

MENG, W. **Metasearch Engines.** 2009. 10.1007/978-0-387-39940-9_217. Disponível em https://www.researchgate.net/publication/255394777_Metasearch_Engines.

OLIVEIRA, A. L., OLIVEIRA, F. L., FAGUNDES, F. **Estruturação e Implementação de um Sistema de Metabusca**. *In:* ENCOINFO - Congresso de Computação e Tecnologias da Informação, 6., 2004, Palmas - TO. Palmas - TO: CEULP/ULBRA, 2004. p. 71 - 80. ISSN e-ISSN: 2447-0767 versão *online*. Disponível em: https://ulbra-to.br/encoinfo/edicoes/2004/artigos/estruturacao-e-implementacao-de-um-sistema-de-metabusca-2/. Acesso em: 12/06/2022

PEREA, P., GINER, P. **UX Design for Mobile: Design apps that deliver impressive mobile experiences.** 1ªed. Birmingham: Packt Publishing Ltd, 2017.

POLLAN, Michael. **Cozinhar: Uma história natural da transformação.** 1ª ed.Rio de Janeiro: Intrínseca, 2013.

PPLWARE. **Webscraping - Saiba o que é e para que serve.** 2020. Acesso em 17/08/2022. Disponível em: https://pplware.sapo.pt/internet/web-scraping-saiba-o-que-e-e-para-que-serve/

PYPI. **Beautiful Soup**. Acessado em 10/10/2022. Disponível em https://pypi.org/project/beautifulsoup4/

Share of global smartphone shipments by operating system from 2014 to 2023. Acesso em 09/06/2022. Disponível em: https://www.statista.com/statistics/272307/market-share-forecast-for-smartphone-operating-systems/

SHARMA, Abhishek. **5 Popular Python Libraries to Perform Web Scraping.** 2020. Acessado em 02/08/2022. Disponível em: https://www.analyticsvidhya.com/blog/2020/04/5-popular-python-libraries-web-scraping/

SIMILARWEB. **Cookpad.com**. Acessado em 26/09/2022. Disponível em: https://www.similarweb.com/pt/website/cookpad.com/#overview

SIMILARWEB. **Tudogostoso.com**. Acessado em 28/09/2022. Disponível em: https://www.similarweb.com/pt/website/tudogostoso.com.br/#overview

TREINAWEB. **O que é Selenium?** Acessado em 12/08/2022. Disponível em https://www.treinaweb.com.br/blog/o-que-e-selenium

What is Mobile Application Development? Acesso em 05/06/2022. Disponível em: https://aws.amazon.com/pt/mobile/mobile-application-development/

WRANGHAM, Richard. **Pegando fogo: Por que cozinhar nos tornou humanos.** 1ª ed. Rio de Janeiro: Zahar, 2010.

Zhao, Bo. (2017). **Web Scraping.** 10.1007/978-3-319-32001-4_483-1. Acesso em 15/06/2022. Disponível em: https://www.researchgate.net/publication/317177787_Web_Scraping

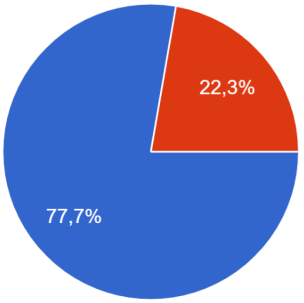
ANEXOS

ANEXO 1

Pesquisa para elicitação de requisitos

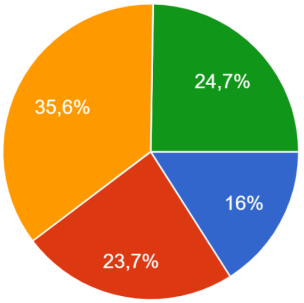
Você costuma cozinhar?

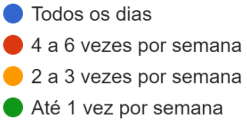
229 respostas



Se sim, com que frequência?

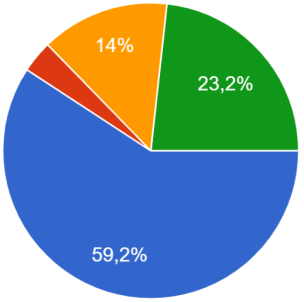
194 respostas

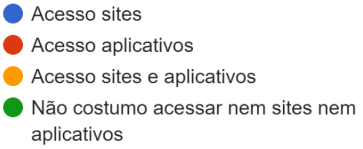




Você acessa algum site ou aplicativo de receitas?

228 respostas



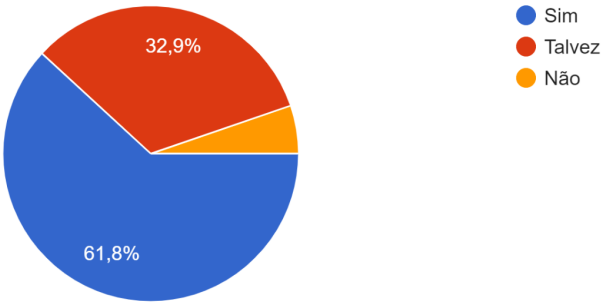


O que mais incomoda você no uso de sites e/ou aplicativos de receitas?

220 respostas

Se houvesse um aplicativo que fizesse uma busca automática em sites conhecidos de receitas, e exibisse todas as opções encontradas de forma clara e objetiva, sem propagandas, você utilizaria?

228 respostas

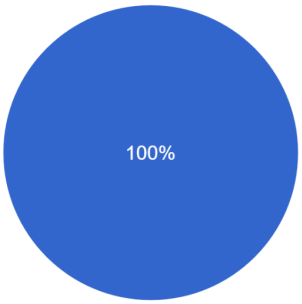


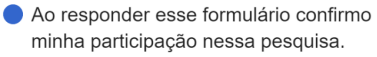
ANEXO 2

Pesquisa de Resultados

Olá! Obrigada por auxiliar respondendo esse formulário. Primeiramente, solicito que baixe o aplicativo MixReceitas na Google Play Store. O Link...tilize ele, buscando receitas de sua preferência.

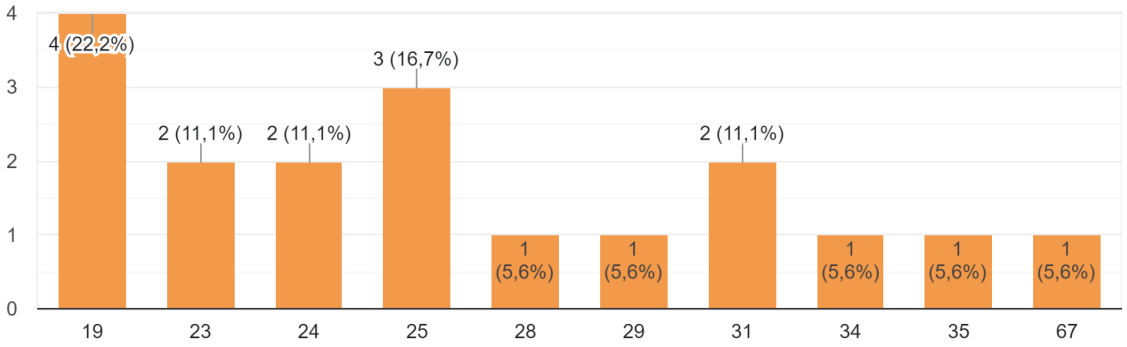
18 respostas





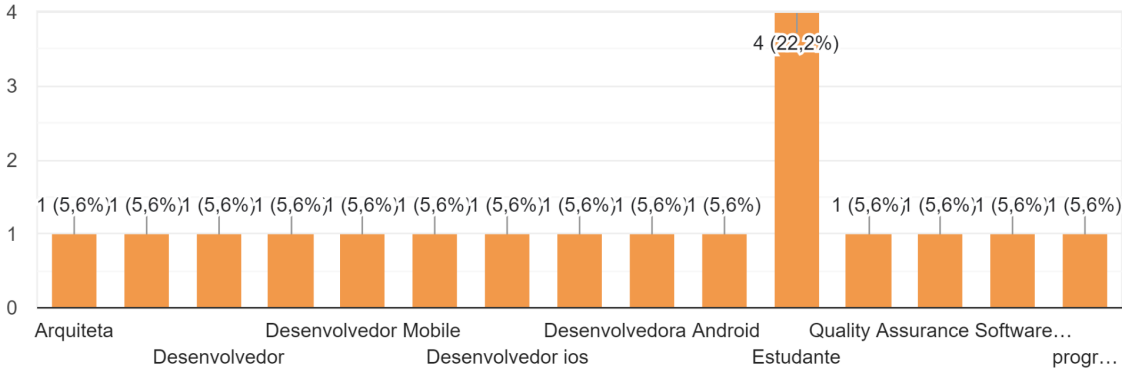
Qual sua idade?

18 respostas



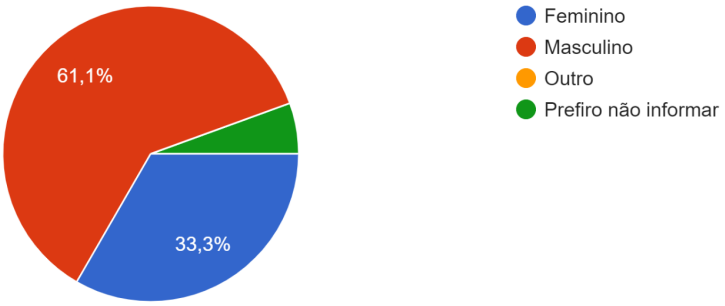
Qual sua profissão?

18 respostas



Qual seu gênero?

18 respostas



Você conseguiu encontrar as receitas desejadas?

18 respostas

Se não conseguiu, o que aconteceu?

3 respostas

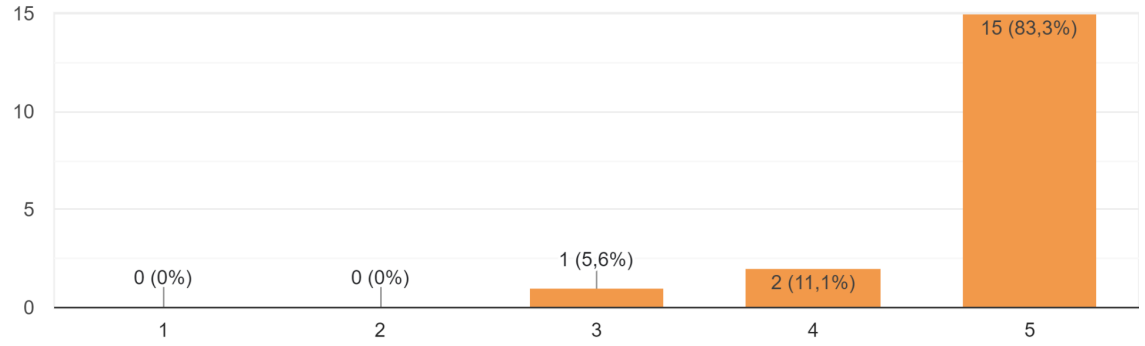
Não achei Strogonoff nem shawarma

Deu tudo certo

Apesar de achar a maioria das receitas, não consegui achar algumas específicas (principalmente bebidas), como vitaminas de frutas e alguns sabores de smoothies ou drinks

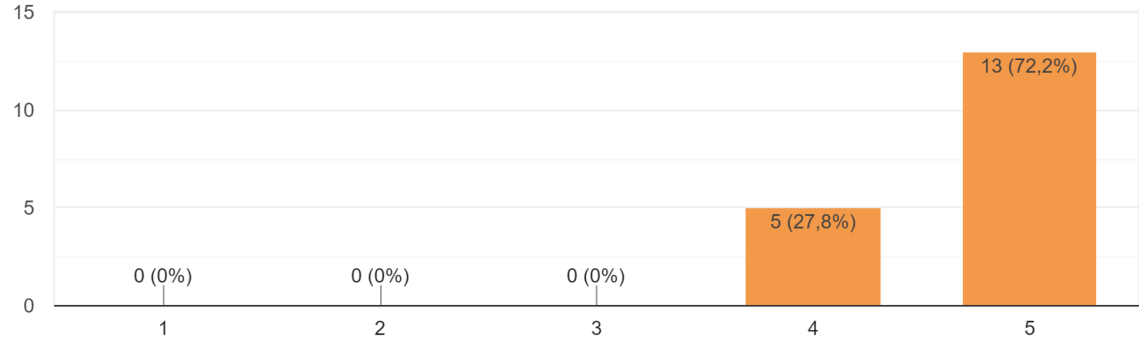
Você considera que o aplicativo é fácil de utilizar? Entre 1 e 5, considerando 1 muito difícil e 5 muito fácil.

18 respostas



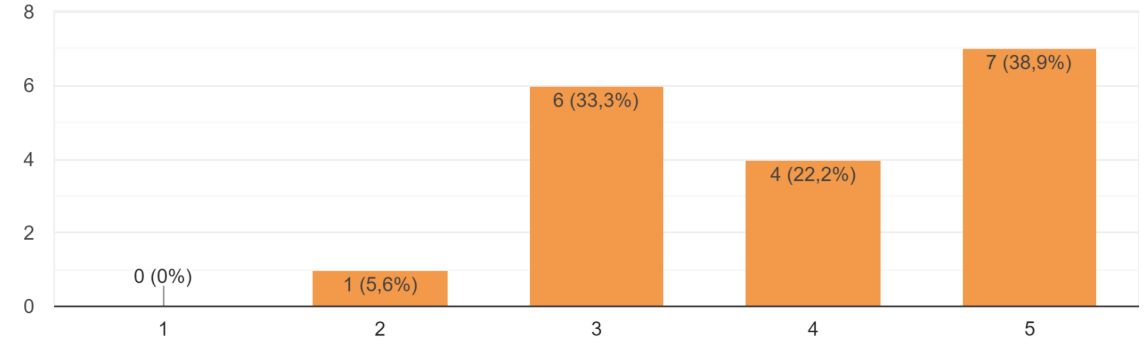
Você considera que o aplicativo é claro e objetivo na exibição do conteúdo? Entre 1 e 5, considerando 1 nada claro nem objetivo e 5 muito claro e objetivo.

18 respostas



Você considera que utilizaria o aplicativo no dia a dia? Entre 1 e 5, sendo 1 não utilizaria nunca e 5 utilizaria com muita frequência?

18 respostas



Você teria alguma sugestão de melhoria no aplicativo?

12 respostas

O pesquisar por certas categorias de comida o aplicativo geralmente sempre mostra sugestões de pratos nem sempre totalmente relacionados a categoria; como ao pesquisar por pizza, onde uma das receitas sugeridas é scone

Um passo da receita era "assista o passo a passo em vídeo" porém não tem como ver o vídeo

Mais receitas

Excelente app

seria interessante informar a fonte daquela receita específica, se foi de um site, Internet, programa etc.

aparecer as receitas disponíveis

Iniciar o app com algumas sugestões na Home, as vezes o usuário não tá com nenhuma receita em mente e vai nas sugestões.

Favoritar e compartilhar é fundamental para esse tipo de aplicativo

Poderia aparecer mais variedades de receita para alguns pratos. Além disso, poderiam ter filtros (entrada, prato principal, sobremesa, salgado, doce, lanche) para quem não tem um prato específico em mente poder navegar e escolher algo para cozinhar

Poderia ter uma lista de receitas mais buscadas ou com mais resultados

Algumas pesquisas trazem poucos resultados (tipo "brigadeirão"), mas não fiz nenhuma pesquisa que não trouxesse ao menos um resulto o que foi positivo. Aumentar a base de pesquisa aumentaria o numero de respostas eu presumo.

Muito bacana o app Dani, ficou show! Como sugestão de melhoria seria legal o app exibir uma mensagem de que não encontrou nenhum resultado pra determinada receita, ou exibir algo próximo do que o usuário digitou, por exemplo, procurei por cheesecake, mas na primeira tentativa escrevi errado (cheescake), dessa forma não tive nenhum resultado. Em uma segunda tentativa, com a palavra escrita corretamente, o app exibiu os resultados.

ANEXO 3

Código implementado

A implementação completa do scraping de receitas encontra-se no seguinte endereço: https://github.com/danihmello06/scraping

Para executar a aplicação localmente, execute o arquivo server.py. Se quiser verificar o funcionamento do endpoint de busca de receitas, acesse o endereço http://localhost:5000/search?word=pudim, se quiser simular uma busca de pudim, por exemplo. Para funcionar corretamente, é necessário ter instalado o Python 3, além das bibliotecas utilizadas, como flask, beautiful soup e requests.

Para realizar a busca dos dados formatados de uma receita específica, é necessário copiar o slug de uma das receitas retornada na busca, e utilizar como parâmetro no outro endpoint: http://localhost:5000/recipe?slug=pudim-de-mandioca-com-cumaru&author=panelinha

A implementação completa do aplicativo android encontra-se no seguinte repositório: https://github.com/danihmello06/recipes

Para executar a aplicação localmente, é necessária uma IDE com emulador, como o Android Studio, ou executar em um dispositivo físico, como um smartphone android.

ANEXO 4

Artigo no formato SBC

Culinary Recipe Metasearch Application

Daniela H. Mello[1], Raul S. Wazlawick[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

danihmello06@gmail.com, raul.wazlawick@ufsc.br

Abstract. *This article presents the development of a mobile application for meta search of culinary recipes. To this do so, web scraping concepts are used, standardizing the information obtained from a culinary recipe website in .json format. The software was successfully developed and was able to provide an Android application capable of performing searches in culinary recipe sources and returning to the user the recipes formatted in a clear and objective way, without advertisements and unnecessary information. However, there are still improvements to be made in future work.*

Resumo. *Este artigo traz o desenvolvimento de uma aplicação mobile de meta busca de receitas culinárias. Para isso são utilizados conceitos de web scraping, padronizando as informações obtidas em website de receitas culinárias em formato .json. O software foi desenvolvido com sucesso e foi capaz de disponibilizar um aplicativo Android capaz de realizar buscas em fontes de receitas culinárias e retornar para o usuário as receitas formatadas de forma clara e objetiva, sem propagandas e informações desnecessárias. No entanto, ainda possuem melhorias a serem realizadas em trabalhos futuros.*

1. Introdução

Atualmente, tem ocorrido uma série de evoluções tecnológicas que colaboram com o ato de cozinhar. Primeiramente, foram desenvolvidos diversos equipamentos que facilitaram a culinária como um todo (BELLIS, 2019). No entanto, a internet pode ser considerada um fator disruptor nesse quesito. Através do uso da internet, pessoas do mundo todo publicam receitas, podendo ser desde receitas de família, compartilhadas por gerações, ou até mesmo as originais, recém inventadas. O website de receitas culinárias cookpad, por exemplo, é o mais acessado da categoria no mundo todo, recebendo quase 120 milhões de acessos apenas em setembro de 2022 (SIMILARWEB, 2022).

A publicação das receitas culinárias é realizada em inúmeros websites ou aplicativos específicos para esse fim, os tornando grandes repositórios com grande quantidade de acessos. Em diversos desses repositórios, qualquer usuário tem a possibilidade de publicar e compartilhar as receitas com qualquer outra pessoa. Essa facilidade de propagar as informações trouxe muitas vantagens para os indivíduos, como a difusão de cultura culinária de cada população para pessoas do mundo todo (CUYLENBURG, 2021).

No entanto, a facilidade de compartilhamento trouxe não apenas benefícios, como alguns problemas que atrapalham a busca do usuário. O excesso de informações obtidas ao realizar a pesquisa em sites de busca muitas vezes confunde o usuário, fazendo com que o mesmo tenha que consultar diversas fontes até encontrar a receita desejada. Além disso, o excesso de propagandas e a falta de clareza na exibição de conteúdo causam uma má experiência por parte dos usuários.

Tendo conhecimento dos problemas que afetam o usuário ao realizar a busca por receitas, o presente trabalho consiste em desenvolver um sistema de metabusca de receitas culinárias. O metabuscador é capaz de realizar uma pesquisa em site de receitas culinárias, e fazer a extração dos dados encontrados nos sites buscados. Após essa etapa, o sistema formata todos esses dados, os exibindo em um padrão único, sem quaisquer propagandas ou poluição visual, utilizando práticas de User Experience (UX) e User Interface (UI) no desenvolvimento do design da aplicação. Por fim, o sistema foi disponibilizado em uma aplicação mobile android, que realiza a integração da metabusca de receitas culinárias e do design da aplicação.

2. Conceitos básicos

2.1 Metabusca

Metabusca é o processo de consultar diversos motores de busca de forma simultânea. Dessa forma, é possível definir um metabuscador como um sistema que realiza a metabusca, possibilitando que usuários da internet realizem a pesquisa a vários websites simultaneamente, através de apenas uma consulta. Assim, uma aplicação de metabusca realiza uma padronização dos dados encontrados para o usuário, facilitando a pesquisa realizada.

Para realizar uma metabusca, a consulta de um usuário é enviada para múltiplos motores de busca. Então, quando os resultados retornados pelos motores de busca são recebidos pelo sistema de metabusca, eles são posicionados em uma única lista, e esta é então apresentada para o usuário (MENG et al, 2019).

Alguns dos principais desafios na implementação de um sistema de metabusca incluem como enviar as consultas de usuários para outros mecanismos de pesquisa, identificar os resultados das páginas retornadas e padronizar a apresentação de dados obtidos através de consultas a diferentes fontes de pesquisa. Motores de metabusca mais sofisticados também realizam a seleção de fontes de pesquisa, ou seja, identificam os motores de busca mais apropriados e enviam uma consulta a apenas esses websites, filtrando assim resultados indesejados. (HOWE; DREILINGER, 1997).

2.2 Web scraping

O web scraping é o processo de coleta de dados de websites, e posterior armazenamento das informações coletadas em um sistema de arquivos ou banco de dados, para posterior recuperação ou análise. A captura de dados por web scraping pode ser feita manualmente ou com a utilização de um robô ou um web crawler, utilizando protocolo HTTP ou através de um navegador. (KAUSAR; DHAKA; SINGH, 2013).

O processo de coleta de dados, realizado através de uma requisição HTTP, na linguagem de programação Python pode utilizar uma biblioteca como Urllib ou Selenium, por exemplo. A biblioteca Urllib utiliza de diversas funções, como cookies e autenticação, para lidar com as requisições. Diferentemente dela, a biblioteca Selenium simula um navegador para automatizar o processo de utilização da internet por um usuário (ZHAO, 2017).

Já o processo de formatação e extração das informações coletadas do código HTML pode ser realizado em Python através da utilização da biblioteca BautifulSoup,

por exemplo. Os dados extraídos podem então ser armazenados e exportados em diferentes formatos de arquivo, como XML, CSV ou JSON, por exemplo (BUDKEWICZ, 2018).

3. Desenvolvimento

3.1 Arquitetura e Padrão de Projeto

Com a finalidade de desenvolver o proposto projeto, foi necessária a implementação de um serviço web, para que o aplicativo realize requisições, obtendo os dados necessários. Esse serviço foi implementado utilizando o framework Flask, por ser um dos mais utilizados no desenvolvimento de APIs com Python, e pela facilidade de implementação.

Para isso, foram definidos dois endpoints. O primeiro deles, */search*, que é responsável por retornar uma lista de receitas obtidas através da busca com uma palavra-chave. Por exemplo, quando o usuário buscar no aplicativo por "pudim", este deverá realizar uma requisição do tipo GET, enviando a palavra 'pudim', e retornando uma lista de resultados encontrados para essa palavra chave. O segundo endpoint, */recipe*, irá ser consultado quando o usuário clicar em uma das receitas retornadas ao realizar a busca inicial. Esse endpoint deve passar uma identificação para saber qual das receitas deve ser consultada, e retornar todos os dados da receita em formato .json.

Figura 1. Arquitetura da comunicação entre frontend e backend

Para o desenvolvimento da arquitetura do aplicativo, foi definida a implementação em três camadas: Camada de Visão, responsável pela exibição dos dados da aplicação na tela; Camada de domínio, responsável por encapsular a lógica de negócios, simplificando-a e realizando a comunicação entre a interface gráfica (camada de visão) e as camadas de dados; Camada de dados, que possui a lógica de negócios, como os repositórios, e é responsável por expor os dados ao restante do app, abstraindo fontes de dados.

O padrão de projeto escolhido para desenvolver o aplicativo foi o MVVM[10] (Model View View Model). Para isso, as camadas do código foram separadas em três partes: Model, responsável pela abstração dos dados; View, responsável por informar o ViewModel sobre as ações realizadas pelo usuário. ViewModel, que realiza os fluxos de dados necessários para a camada de Visualização (View), conectando o Modelo e View.

[10] https://www.digitalocean.com/community/tutorials/android-mvvm-design-pattern

3.2 Estrutura de Dados

Para o desenvolvimento desta aplicação, foi necessário definir a estrutura de dados na qual serão inseridos os dados referentes às receitas. Primeiramente, foi definida a classe SearchResult, que irá armazenar a resposta de uma busca do usuário, que é uma lista de receitas, apenas com informações reduzidas. Conforme figura 2, os dados necessários são o *author* (site origem do resultado da busca), a URL da imagem e o título da receita, o índice da receita na lista (utilizado para posterior ordenação de resultados), o slug (trecho da URL de uma receita específica), e o título da receita.

```
class SearchResult:
    def __init__(self, author, image_url, index, slug, title):
        self.author = author
        self.image_url = image_url
        self.index = index
        self.slug = slug
        self.title = title
```

Figura 2. Classe SearchResult

Também foi implementada a classe Steps, que é responsável por armazenar os dados de cada preparo da receita. Para cada preparo, há um título, uma lista de ingredientes, e uma lista de modo de preparo.

```
class Steps:
    def __init__(self, title, ingredients, preparation):
        self.title = title
        self.ingredients = ingredients
        self.preparation = preparation
```

Figura 3. Classe Steps

Além disso, foi definida a classe que possui os dados da receita, com o nome de Recipe. Conforme é possível observar na Figura 4, os dados necessários para a receita são o título da receita, a URL da imagem, o slug, o nome do site de origem da receita, a quantidade de porções, o tempo de preparo e os passos necessários para realizar a receita.

```
class Recipe:
    def __init__(self, title, image_url, slug, author, serves,
prepare_time, steps):
        self.title = title
        self.image_url = image_url
        self.slug = slug
        self.author = author
        self.serves = serves
```

```
        self.prepare_time = prepare_time
        self.steps = steps
```

Figura 4. Classe Recipe

3.3 Extração de dados de website

No início do desenvolvimento da aplicação, foi escolhido o site Panelinha para realizar o processo de metabusca. A escolha do site Panelinha foi realizada por ser um site com uma grande variedade de receitas e não possuir ainda uma versão em aplicativo. Posteriormente, em trabalhos futuros, serão escolhidos outros websites, para poder ter uma base maior de receitas.

Para realizar as buscas de receitas, primeiramente foi necessária a criação de dois endpoints. O primeiro endpoint, /search acessa o método get_search_result(), que recebe a palavra chave buscada pelo usuário, e realiza a busca de uma lista de resultados encontrados a partir da palavra chave consultada.

O segundo endpoint, /recipe, acessa um método get_recipe(), que recebe o nome da receita clicada pelo usuário e o website da onde ele irá buscar os dados da receita, e o envia para uma função responsável por realizar a extração de todos os dados de uma receita, e a retorna para o aplicativo.

```
from flask import Flask, request
from scrapingPanelinha import *
from policy import *

app = Flask(__name__)

@app.route('/search')
def do_search():
    searched_word = request.args.get('word')
    return get_search_result(searched_word)

@app.route('/recipe')
def open_recipe():
    slug = request.args.get('slug')
    author = request.args.get('author')
    return get_recipe(slug, author)

if __name__ == '__main__':
    app.run()
```

Figura 5. Arquivo server.py, com os endpoints implementados

```
def get_search_result(word):
    result_from_panelinha = get_search_from_panelinha(word)
    all_lists_combined =
order_results_equally(result_from_panelinha)
```

```
    return all_lists_combined
```

Figura 6. Implementação do método get_search_result().

```
def get_recipe(slug, author):
    recipe = Recipe("","","","","","","")
    if author == "panelinha":
        recipe = get_recipe_from_panelinha(slug)
    return recipe.__dict__
```

Figura 7. Implementação do método get_recipe()

```
def get_search_from_panelinha(word):
    search_url =
"https://panelinha-api-server-prod.herokuapp.com/v1/search?pageSize
=1000&title="+word
    search_request = requests.get(search_url).json()
    search_list = []
    results = search_request['data']['results']
    for i in range(len(results)):
        author = "panelinha"
        image_url = results[i]['imageUrl']
        index = i
        slug = results[i]['slug']
        title = results[i]['title']
        recipe_found = [author, image_url, index, slug, title]
        if results[i]['imageFolder'] == 'receita':
            search_list.append(recipe_found)
    return search_list
```

Figura 8. Implementação do método get_search_from_panelinha()

```
def order_results_equally(all_lists_combined):
    sorted_list = sorted(all_lists_combined, key = itemgetter(2))
    list_of_search_result = []
    for i in range(len(sorted_list)):
        search_item = SearchResult(
            author = sorted_list[i][0],
            image_url = sorted_list[i][1],
            index = sorted_list[i][2],
            slug = sorted_list[i][3],
            title = sorted_list[i][4]
        )
```

```
        list_of_search_result.append(search_item.__dict__)
    return list_of_search_result
```

Figura 9. Implementação do método order_results_equally()

```
def get_recipe_from_panelinha(slug):
    url_bs = "https://www.panelinha.com.br/receita/"+slug
    data = requests.get(url_bs)
    page_bs = BeautifulSoup(data.content, 'html.parser')

    base = page_bs.find_all('div', attrs={'class': 'col-xs-12
col-sm-6 col-md-7'})
    ingredients_and_instructions = base[1].findAll('div',
class_='editor ng-star-inserted')
    ingredients_list = []

    for i in range(0, len(ingredients_and_instructions), 3):
        ingredients_only =
ingredients_and_instructions[i].findAll('li',
class_='ng-star-inserted')
        ingredients_in_intruction = []
        for ing in ingredients_only:
            ingredients_in_intruction.append(ing.text)
        ingredients_list.append(ingredients_in_intruction)

    recipe_url =
"https://panelinha-api-server-prod.herokuapp.com/v1/receita/"+slug
    recipe_request = requests.get(recipe_url).json()
    result = recipe_request['data']
    content = result['content']
    recipe_steps = content['recipeSteps']
    steps_updated = []
    for i in range(len(recipe_steps)):
        if 'ingredients' in recipe_steps[i]:
            steps_updated.append(recipe_steps[i])

    steps_list = []
    for i in range(len(steps_updated)):
        step_preparation = steps_updated[i]['body']
        formatted_step_preparation =
BeautifulSoup(step_preparation, 'html.parser').getText()
        steps = Steps(
            steps_updated[i]['title'],
```

```
            ingredients_list[i],
            formatted_step_preparation
        )
        steps_list.append(steps.__dict__)

    recipe = Recipe(
        result['title'],
        result['imageUrl'],
        result['slug'],
        content['author'],
        content['serves'],
        content['prepareTime'],
        steps_list
    )

    return recipe
```

Figura 10. Implementação do método get_recipe_from_panelinha()

3.4 Padronização dos Dados

Após a definição das estruturas de dados, e da implementação do código de extração de dados, foi definido um padrão de .json a ser seguido para padronizar as buscas a outros sites a serem realizados o scraping. O método que retorna a lista de resultados encontrados a partir de uma palavra chave inserida pelo usuário deve seguir o padrão conforme demonstrado na figura 11.


```
[
  {
    "author": "panelinha",
    "image_url": "https://cdn.panelinha.com.br/receita/1657291945626-pudim-de
      -mandioca%20com%20cumaru.jpg",
    "index": 0,
    "slug": "pudim-de-mandioca-com-cumaru",
    "title": "Pudim de mandioca com cumaru (reaproveitamento)"
  },
  {
    "author": "panelinha",
    "image_url": "https://cdn.panelinha.com.br/receita/1667481994336-pudim
      .jpg",
    "index": 1,
    "slug": "pudim-na-airfryer",
    "title": "Pudim de leite na AirFryer da Rita Lobo"
  }
]
```


Figura 11. json representando a padronização da lista de resultados.

Além disso, também foi definido que o método de retorno de uma receita específica deve seguir um padrão, conforme é possível observar na figura 12.


```
{
  "author": "Panelinha",
  "image_url": "https://cdn.panelinha.com.br/receita/1657291945626-pudim-de
    -mandioca%20com%20cumaru.jpg",
  "prepare_time": "Mais de 2h",
  "serves": "Mais de 6 porções",
  "slug": "pudim-de-mandioca-com-cumaru",
  "steps": [
    {
      "ingredients": [
        "1 xícara (chá) de açúcar",
        "½ xícara (chá) de água fervente"
      ],
      "preparation": "\nSepare 8 forminhas lisas com capacidade para 150 ml.
        Leve uma chaleira com 1 litro de água ao fogo médio – ela vai ser
        usada para fazer a calda e para assar o pudim em banho-maria.\n",
      "title": "PARA A CALDA"
    },
    {
      "ingredients": [
        "1½ xícara (chá) de mandioca cozida, em temperatura ambiente",
        "3 ovos",
        "1 xícara (chá) de leite",
        "¾ de xícara (chá) de açúcar",
        "cumaru ralado na hora a gosto"
      ],
      "preparation": "\nPreaqueça o forno a 160 ºC (temperatura baixa).",
      "title": "PARA O PUDIM"
    }
  ],
  "title": "Pudim de mandioca com cumaru (reaproveitamento)"
}
```


Figura 12. json representando a padronização dos dados de uma receita.

3.5 Layout, User Interface e User Experience

Para a implementação da parte visual da aplicação, foram definidas três telas principais. A primeira delas é a tela inicial, onde se encontra o nome do aplicativo, e um campo para que o usuário possa buscar o nome da receita que deseja encontrar. Para essa tela houve a preocupação de deixar todo o campo de busca com área de toque ativa, para cumprir requisitos de acessibilidade, ou seja, para que usuários com deficiência visual tenham maior facilidade de utilizar o aplicativo. Além disso, o nome do app - MixReceitas - aparece em destaque, com a finalidade de manter a hierarquia visual.

A segunda tela consiste em uma lista com todos os resultados encontrados para a receita buscada. Além disso, há um cabeçalho, onde o usuário pode ver o nome do item que ele buscou. Cada item encontrado possui uma imagem, que preenche todo o espaço do item, com a finalidade de facilitar a visualização dos resultados. Os nomes das receitas são exibidos sobre a imagem, em frente a um degradê preto-transparente, para facilitar a leitura.

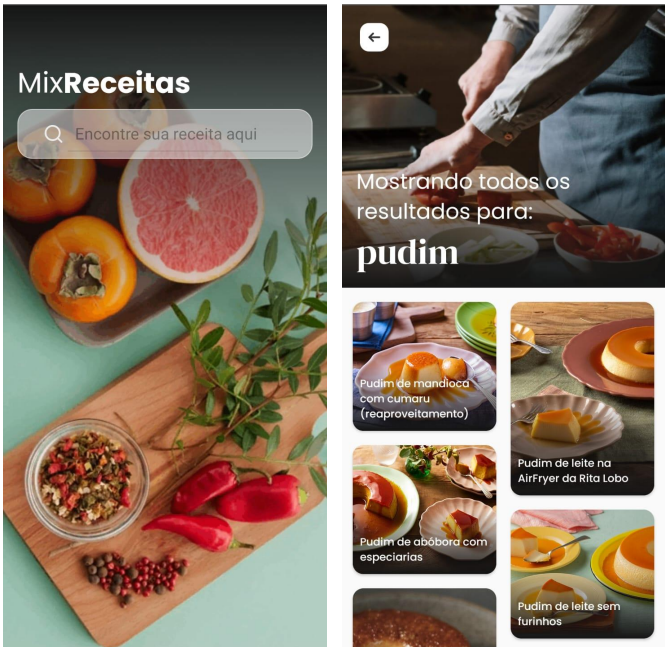


Figura 13. Tela inicial e tela de resultados do aplicativo.

Por fim, a tela da receita é exibida quando o usuário clica em uma receita que deseja consultar. Para essa tela optou-se por exibir a imagem da receita no topo da tela, juntamente com o título, mantendo a disposição da tela de resultados de pesquisa. Em casos que a receita tenha mais de um preparo, cada um deles é exibido separadamente, em tópicos, para maior compreensão de hierarquia e da receita em si, conforme é possível observar na figura 14.

Figura 14. Tela de exibição da receita.

4. Resultados

Após o desenvolvimento do aplicativo, foi realizada uma Pesquisa de Resultados com a finalidade de avaliar o resultado obtido, e enviada para cerca de 350 pessoas. O público alvo foi uma população entre 18 e 65 anos de idade, com gênero e profissões diversas. O aplicativo foi disponibilizado na loja da Google (Play Store), com o nome de MixReceitas, para facilitar o download e análise pelos participantes. Ao final da pesquisa foram obtidos 19 participantes, gerando uma taxa de resposta de 0,054.

A maioria dos usuários achou o aplicativo muito fácil de utilizar e com muita clareza na exibição do conteúdo, atingindo então os objetivos do presente trabalho. Quanto à frequência de utilização do aplicativo, 94,7% dos usuários responderam que o utilizariam com alguma frequência relevante.

Por fim, foi questionado aos usuários se após a utilização da aplicação eles tinham alguma sugestão de melhoria. Com as respostas, foi possível definir itens a serem melhorados em trabalhos futuros, como a implementação do web scraping em outros sites além do Panelinha e a disponibilização de um banco de dados para que seja possível o cadastro de usuários e eles poderem salvar receitas favoritas. Outra melhoria sugerida foi a possibilidade de compartilhar as receitas, realizar filtros por categorias ou alimentos. Por fim, a exibição de sugestões de receitas logo na página inicial também é de grande utilidade para a aplicação.

5. Conclusão

Após a realização da aplicação proposta, foi possível realizar o scraping de um site de receitas, o Panelinha. Assim, o software consegue acessar o website, e extrair todos os resultados encontrados a partir de uma busca de uma receita específica. Após a extração dos dados, eles são formatados em um sistema padronizado, no formato .json. Então, são enviados e exibidos em uma aplicação mobile android para o usuário, sem propagandas.

Dessa forma, é possível afirmar que a aplicação foi desenvolvida com sucesso, apesar de ter algumas melhorias a serem realizadas, como o scraping em outras fontes, possibilidade de cadastro de usuário e favoritar e compartilhar receitas e aplicar filtros, por exemplo.

Referências

BELLIS, Mary. 2019. **The History of Kitchen Appliance Inventions.** Acesso em 11/10/2022. Disponível em: https://www.thoughtco.com/history-of-kitchen-appliance-inventions-1992036

BUDKEWICZ, Matheus. **Como fazer webscraping com Python e Beautiful Soup.** 2018. Acesso em 05/09/2022. Disponível em https://medium.com/horadecodar/como-fazer-webscraping-com-python-e-beautiful-soup-28a65eee2efd

CUYLENBURG, Jessica van. **World Cuisine: How to Easily Find Recipes From Other Cultures.** 2021. Disponível em:

https://www.nerdynaut.com/world-cuisine-how-to-easily-find-recipes-from-other-cultures Acesso em: 23/10/2022

HOWE, A., DREILINGER, D. **SavySearch: A Metasearch Engine That Learns Which Search Engines to Query.** AI Magazine, Volume 18 Number 2. 1997.

KAUSAR, M., DHAKA, V., SINGH, S., **Web Crawler: A Review.** International Journal of Computer Applications. Fevereiro, 2013.

MENG, W. **Metasearch Engines.** 2009. 10.1007/978-0-387-39940-9_217. Disponível em https://www.researchgate.net/publication/255394777_Metasearch_Engines.

SIMILARWEB. **Cookpad.com**. Acessado em 26/09/2022. Disponível em: https://www.similarweb.com/pt/website/cookpad.com/#overview

Zhao, Bo. (2017). **Web Scraping.** 10.1007/978-3-319-32001-4_483-1. Acesso em 15/06/2022. Disponível em: https://www.researchgate.net/publication/317177787_Web_Scraping