UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

DEPARTAMENTO DE COMPUTAÇÃO

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

JOSUÉ NICHOLSON GIRÃO SILVA

DESENVOLVIMENTO DE UM APLICATIVO MOBILE PARA AVALIAÇÃO DA QUALIDADE DA PROTEÍNA DAS REFEIÇÕES DO R.U.

QUIXADÁ

2021

JOSUÉ NICHOLSON GIRÃO SILVA

DESENVOLVIMENTO DE UM APLICATIVO MOBILE PARA AVALIAÇÃO DA QUALIDADE DA PROTEÍNA DAS REFEIÇÕES DO R.U.

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Dr. Jefferson de Carvalho Silva

QUIXADÁ

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58d Silva, Josué Nicholson Girão.
Desenvolvimento de um aplicativo mobile para avaliação da qualidade da proteína das refeições do R.U. / Josué Nicholson Girão Silva. – 2021.
52 f.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2021.
Orientação: Prof. Jefferson de Carvalho Silva.

1. Aplicativos móveis. 2. Software-Desenvolvimento. 3. Votação. I. Título.

CDD 005.1

JOSUÉ NICHOLSON GIRÃO SILVA

DESENVOLVIMENTO DE UM APLICATIVO MOBILE PARA AVALIAÇÃO DA QUALIDADE DA PROTEÍNA DAS REFEIÇÕES DO R.U.

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em: _____/_____/_____

BANCA EXAMINADORA

Prof. Dr. Jefferson de Carvalho Silva (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Wladimir Araujo Tavares
Universidade Federal do Ceará (UFC)

Prof. Dr. Victor Aguiar Evangelista de Farias
Universidade Federal do Ceará (UFC)

Aos meus pais, que em nenhum momento deixaram de me apoiar e fizeram tudo que foi preciso para que eu chegasse até aqui. Sem eles, certamente eu não seria ninguém.

AGRADECIMENTOS

Agradecer é essencial para que reconheçamos o papel de outras pessoas na nossa vida. Este momento é único, e eu não poderia deixar de registrar todo o meu sentimento de gratidão àqueles que se fizeram presentes e me apoiaram por toda essa caminhada até aqui.

Primeiro, ao senhor do universo, por ter feito sua contribuição ao fornecer-me a segurança de sua mão enquanto não permitia que coisas ruins me derrubassem.

Meus agradecimentos à minha família, que sempre me auxiliou. À minha mãe, que nunca me deixou passar qualquer tipo de necessidade, seja ela física ou afetiva, e enfrentou inúmeros obstáculos por mim. Ao Meu pai, que mesmo nos momentos mais difíceis, acreditou em mim e apostou tudo o que tinha para me dar suporte. Ao meu irmão, que foi meu parceiro e amigo nesses meus 21 anos, e também à todo o resto da família.

Ao professor Dr. Jefferson de Carvalho Silva, por ter me orientado, se disposto à me ajudar nessa fase final da graduação, por ter dado dicas e conselhos sobre o trabalho final, e ter sido compreensivo durante esse tempo. Também à professora Dra. Antonia Diana Braga Nogueira, coordenadora do curso e considerada por muitos alunos uma mãe, por ter facilitado esse árduo caminho, fazendo tudo que estava ao seu alcance para me auxiliar.

Aos meus amigos Gleuberson, Gabriel e Castilho, que dividiram não só apartamento, mas as dores, as alegrias, os choros e os sorrisos durante todo esse longo percurso. Também não posso deixar de citar Mateus, Ávila e Lara, que mesmo após minha saída de Morada Nova, não permitiram que a distância pudesse nos afastar, nem tampouco permitiram que eu pensasse em desistir.

E é claro, à minha família que eu escolhi. Rafael, Clebson, Wilton, Victor e Pedro, grandes protagonistas dessa história, que estiveram ao meu lado desde o início do curso, me ajudaram nas disciplinas mais difíceis, apesar da minha chatice me acolheram e me concederam a honra de ser considerado amigo. Sem vocês, não teria dado certo não, will.

"Os acessórios se perdem à medida em que as grandezas se aproximam do infinito."

(Arthur Araruna)

RESUMO

Aplicações móveis vem sendo cada vez mais utilizadas como solução para problemas cotidianos devido a crescente demanda por soluções que exigem mobilidade, segurança e praticidade. Com isso, muitos vendedores de produtos e prestadores de serviços têm migrado para as plataformas móveis à medida em que seus usuários buscam por formas de comprar produtos ou usufruir de tais serviços em qualquer lugar, a qualquer momento e de forma fácil e rápida, como é o exemplo do bancos, lojas e restaurantes. Dentre os mais diversos serviços que já passaram ou estão passando por este processo de migração, podemos também citar os relacionados à votação de maneira geral, que tem criado novos conceitos como o *e-voting* (votação eletrônica) como resultado deste processo de evolução. O Restaurante Universitário (R.U) da UFC - Campus Quixadá utiliza um sistema de avaliação de refeições através de uma urna física, na qual as pessoas que se alimentam podem expressar sua opinião sobre a qualidade da refeição. A pandemia do coronavírus (Covid-19) trouxe à tona um problema de educação relacionada a higiene pessoal que por poucas pessoas antes era tratado com seriedade. Tocar em objetos os quais muitas pessoas têm contato diariamente se tornou um risco iminente à saúde. Assim surgiu o desafio de continuar promovendo votações no R.U., adequando-se os novos hábitos higiênicos das pessoas. Este trabalho propõe o desenvolvimento de um aplicativo móvel como alternativa para a utilização da urna física do R.U.

Palavras-chave: Aplicações móveis. Software-Desenvolvimento. Votação.

ABSTRACT

Mobile applications is increasingly being used as a solution for daily problems due to the increasing demand for solutions which require mobility, security and practicality. Because of this, many product sellers and services providers have been migrating to mobile platforms as their users have been seeking for a way to buy products or enjoy those services anywhere, anytime and in a easy and a fast way, as example of banks, shops and restaurants. Among the most diverse services that already passed or has been passing through this migration process, we can also cite the general voting related ones, which have been creating new concepts like e-voting as result of this evolution process. The UFC's (Federal University of Ceará) University Restaurant (R.U) - Campus Quixadá uses a food assessment system through a physic ballot box, which people who ate can express their opinion about the quality of the meal. The coronavirus (Covid-19) pandemic brought up personal hygienic related problems that were treated with seriousness by a few people before. Touching objects which many people has daily contact became a imminent risk for health. That is how the continue promoting voting in R.U challenge arose, adapting to the new people hygienic habits. This work proposes a mobile application development as an alternative for physic ballot box using in R.U.

Keywords: Mobile Applications. Software Development. Voting

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Segundo o relatório anual da GSMA[1], ao fim do ano de 2019 existiam 5,2 bilhões de conexões via *smartphones*, o que é uma quantidade expressiva uma vez que mostra que mais de dois terços da população mundial está conectada através de um aparelho móvel. Com um grande número de pessoas utilizando plataformas móveis, as organizações tem cada vez mais se interessado por desenvolver soluções para estas plataformas.

Também ao fim de do 2019, o mundo foi surpreendido por um novo vírus da família coronavírus, que com seu alto poder de transmissão, rapidamente se espalhou por todas as partes do mundo, causando enormes prejuízos à saúde pública, à economia, e às relações sociais. Por causa disto, a população mundial precisou se adequar a uma nova forma de se comunicar, de trabalhar e de estudar, além de adquirir hábitos recomendados pela OMS (Organização Mundial de Saúde)[2] para diminuir a transmissão do vírus e a disseminação de doenças infecciosas.

Dentre várias medidas individuais para atingir tal objetivo, a OMS pontua a higienização das mãos, limpeza dos ambientes e etiqueta respiratória, que se trata do comportamento correto ao espirrar e tossir. Tocar em objetos os quais muitas pessoas diariamente entram em contato se tornou um risco iminente à saúde, e muitas pessoas passaram a preferir não tocá-los por precaução. Exemplos de objetos que recebem muitos toques diariamente por várias pessoas diferentes são, principalmente, àqueles que oferecem algum tipo de interação entre o usuário e o sistema, tais como máquinas de crédito e débito, painéis de elevadores, telas sensíveis a toque, etc.

Na Universidade Federal do Ceará (UFC), campus Quixadá, há um Restaurante Universitário (RU), no qual muitos alunos, professores, e demais funcionários podem se alimentar e, após isto, realizar uma avaliação da qualidade da proteína que foi consumida. Essa avaliação funciona como um processo de votação, no qual há uma urna física onde cada pessoa deposita uma cédula de papel, classificando a proteína como: boa, regular ou ruim.

Além desta urna que usa cédulas de papel, Silva (2019) desenvolveu uma urna eletrônica para substituir a citada anteriormente, automatizando o processo de contabilização dos votos, melhorando a experiência dos votantes e ocasionando economia de papel. No entanto, este projeto não englobou uma segunda parte da avaliação, que é opcional e se trata de um comentário escrito numa cédula e depositado em uma segunda urna. Além disto, como discutido

[1] https://www.gsma.com/mobileeconomy/

[2] https://www.who.int/publications/i/item/responding-to-community-spread-of-covid-19

anteriormente, ambas as urnas citadas não seriam adequadas no cenário onde não é recomendado o contato direto com objetos compartilhados. Neste contexto, é proveitoso que as duas partes de avaliação seja unificadas, além de garantir o atendimento aos requisitos do cenário mencionado anteriormente.

Mediante a estas circunstâncias, este trabalho se propõe a desenvolver e validar um aplicativo para *smartphones* que poderá ser utilizado por aqueles que utilizam o serviço do R.U, para que possa ser realizado o ato de submeter a avaliação da proteína, atendendo às novas necessidades relacionadas aos novos hábitos de higiene e evitando que não somente a transmissão do coronavirus ocorra, mas também de outros tipos de vírus e bactérias. Além das pessoas que usufruem dos serviços do R.U, este trabalho busca dar suporte para possíveis novos requisitos por parte da equipe de administração do restaurante.

2 OBJETIVOS

O objetivo deste trabalho é propôr uma solução para *smartphones*, que tem como meta permitir a apuração de avaliações da qualidade da comida do R.U de forma a evitar o contato direto com a urna física.

2.1 Objetivos específicos

1. Disponibilizar um aplicativo *mobile* para votação da qualidade da proteína, que seja integrado com o sistema existente;
2. Validar a proposta junto à nutricionista;
3. Validar junto aos alunos;

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados e discutidos os trabalhos relacionados, que são produto de uma revisão bibliográfica utilizando o motor de busca Google Scholar, e bases virtuais amplamente conhecidas e conceituadas como a IEEE Xplore, Springer e ACM Digital Library . Serão discutidas as similaridades e diferenças destes trabalhos para o proposto.

Kielt *et al.* (2017) realizou um estudo para desenvolver um aplicativo para *smartphones* que foi usado em aulas de física para uma turma de ensino médio para implementar uma abordagem de ensino chamada *Peer Instruction*. Este aplicativo era responsável por colher respostas dos alunos à determinadas questões que compunham testes sobre variados assuntos da disciplina, enviá-las através da rede sem fio para o computador pessoal do professor, assim facilitando o processo de interpretação do professor sobre quais pontos específicos de um determinado assunto os alunos haviam tido maior dificuldade de aprendizado.

O presente estudo também se trata de um sistema de votação que será desenvolvido para *smartphones* e possuirá uma arquitetura de comunicação similar utilizando a rede local do R.U. Porém além de possuir um objetivo final diferente, vale ressaltar que a proposta almeja explorar o contexto das aplicações híbridas para que a maior parte dos usuários seja abrangida. Além disso, outra diferença é que o sistema que será resultado desta pesquisa possui o requisito de anonimidade, ou seja, o usuário poderá votar sem que seja possível identificá-lo através de seu voto.

Silva (2019) é o principal trabalho relacionado à esta pesquisa, uma vez que esta proposta se apresenta como uma extensão para atender novas necessidades. O trabalho mencionado se empenhou em construir uma urna eletrônica utilizando um Arduíno, que se comunica com um servidor para enviar os votos coletados, que por sua vez é responsável por armazenar e fornecer estes dados para um sistema web, disponibilizado para o serviço de nutrição do campus, que consegue ter a visualização dos votos.

A necessidade de promover votações sem ter contato direto com a urna física surgiu devido à pandemia e, portanto, o estado atual do sistema não possui suporte para esta alternativa, pois não foi projetado para tal. O principal diferencial que nosso trabalho busca explorar é atender a essa nova necessidade, reaproveitando a arquitetura e a implementação do *software* existente para servir como base para a elaboração e desenvolvimento da proposta.

Outro diferencial importante pode ser destacado é que a urna eletrônica desenvolvida não abrange as opiniões subjetivas. Na urna de cédulas que é atualmente utilizada, além de poder

classificar a qualidade da proteína consumida, o votante também tem a opção de escrever um comentário em uma segunda cédula de papel e depositá-la em outra urna. A aplicação proposta pelo presente trabalho se propõe a captar também esse tipo de avaliação, assim servindo como alternativa para as duas urnas.

É importante salientar que, uma vez que Silva (2019) elicitou requisitos, projetou a arquitetura do servidor e modelou o banco de dados, este trabalho se apoiou nesta documentação e estrutura para propôr uma evolução do sistema, acrescentando um novo tipo de cliente do servidor, que é o aplicativo. Portanto, os documentos relacionados a estas três atividades anteriormente citadas podem ser encontradas no texto do trabalho mencionado.

Um aplicativo *mobile* de votação eletrônica foi proposto por Kalaiyarasi *et al.* (2020) para *smartphones* com sistema operacional Android. Este se apresenta como uma implementação genérica de *e-voting* em relação a seu campo de aplicação, ou seja, não foi definido um subdomínio específico onde ele poderia ser utilizado. O sistema possui autenticação através de e-mail e senha e usou o algoritmo de criptografia AES256 para encriptação dos votos e OTP (do inglês, *One-time Password*) para legitimar o ato de votar. Além disso, possui um módulo de administrador, onde o responsável pode cadastrar novos candidatos para a votação e visualizar os resultados da eleição também.

É possível destacar como similaridade o intuito de incorporar a plataforma móvel como meio para promover votações eletrônicas, desta forma atendendo a requisitos de mobilidade e usabilidade. Dentre as principais diferenças que o presente estudo possui em relação ao anteriormente citado, é possível mencionar que este se propõe a atender um subdomínio específico que é o de votações para avaliar qualidade dos alimentos e a preocupação em desenvolver um aplicativo que possa ser utilizado tanto no sistema Android quanto no iOS. Além disso, o aplicativo que este estudo propõe não possui o módulo de administrador, pois este já está implementado como um sistema WEB, e é utilizado apenas pela administração do R.U.

Quadro 1 – Comparativo entre os trabalhos relacionados e o trabalho proposto

Trabalhos	**Implementa e-voting**	**Considera novos hábitos higiênicos**	**Possui domínio específico**	**Permite voto subjetivo**	**É aplicação *mobile* híbrida**	**Finalidade da aplicação**
Trabalho Proposto	Sim	Sim	Sim	Sim	Sim	Incentivar a permanência de novos hábitos de higiene.
Kielt *et al.* (2017)	Sim	Sim	Sim	Não	Não	Melhorar qualidade do aprendizado de aulas de física
Silva (2019)	Sim	Não	Sim	Não	Não	Evitar disperdício de papel e proporcionar melhor experiência de usuário.
Kalaiyarasi *et al.* (2020)	Sim	Sim	Não	Não	Não	Desenvolver um aplicativo de votação de propósito geral para dispositivos móveis.

Fonte: Elaborado pelo autor.

4 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são discutidos conceitos importantes sob os quais este trabalho se baseia para elaborar e estruturar a proposta de solução. A seção 4.1 irá tratar sobre as etapas comuns de desenvolvimento de *software*, e qual a importância de cada uma delas. A seção 4.2 aborda mais especificamente sobre a construção de aplicações híbridas para a plataforma de *smartphones*, uma vez que se espera que a solução seja disponibilizada para os sistemas operacionais *Android* e *IOS*. Subconceitos relacionados ao *e-voting* serão tratados na seção 4.3, onde requisitos específicos deste tipo de sistema que serão levados em consideração para o desenvolvimento da solução serão apresentados.

4.1 Processo de desenvolvimento de *software*

Segundo Sommerville (2011), existem várias formas de se desenvolver *softwares* e, para cada domínio, há um processo de *software* diferente, ou seja, uma abordagem metódica adequada para cada um. Isto porque cada contexto possui suas peculiaridades e portanto, apresentam necessidades diferentes que precisam ser atendidas. Entretanto, existem quatro grandes atividades essenciais que servem para todos os processos. O conceito de cada uma destas será utilizado para fundamentar os desenvolvimento do aplicativo. São eles:

4.1.1 Especificação de software

De maneira geral é nesta atividade que são documentados os requisitos, tais como os funcionais e não-funcionais, do sistema a ser construído. O objetivo é registra-los, enfatizando tanto a importância destes para o ambiente de negócio em que o *software* irá ser implantando, que é de interesse dos *stakeholders*, quanto o detalhamento técnico necessário para que os desenvolvedores implementem o sistema. A documentação é muito importante para que se possa fazer a validação do *software*, que será discutido na seção 4.1.3. Sommerville (2011) separa ainda quatro subetapas desta atividade, são elas:

1. Estudo de viabilidade. Define se é possível construir o *software*, respeitando restrições técnicas e financeiras, e se é vantajoso para a organização em termos de contribuição para os objetivos de negócio da empresa .
2. Elicitação e análise de requisitos. É nesta subatividade onde serão coletados, das mais diversas formas, os requisitos e o será analisado todo o contexto onde o

software irá ser inserido .

3. Especificação de requisitos. É o processo em que o resultado da subatividade anterior é utilizado para a elaboração de um documento, onde os requisitos de usuário e de sistema podem ser especificados .
4. Validação de requisitos. Esta subetapa se encarrega de averiguar algumas propriedades dos requisitos especificados, como completude, consistência e coerência com a realidade, e consequentemente identificar possíveis erros no documento gerado na fase anterior .

4.1.2 Desenvolvimento de software

Também conhecida como projeto e implementação de *software*, essa atividade consiste em desenhar uma estrutura geral do *software*, definindo soluções através de arquiteturas e detalhando módulos e componentes. A codificação do *software* também está inserida nesta fase, porém é importante ressaltar que geralmente, o sistema passa por várias iterações desta atividade até que atenda as especificações. Assim como na atividade de especificações, Sommerville (2011) identificou quatro subatividades comuns que compõe o desenvolvimento de *software*, e são apresentadas a seguir:

1. Projeto de arquitetura. Subatividade na qual é definido o arcabouço geral do projeto e onde os módulos, sua configuração e como se relacionam entre si é especificado.
2. Projeto de interface. Diz respeito a atividade de definir interfaces para facilitar a integração entre componentes, de maneira que os componentes não precisem saber como os outros estão implementados mas somente como se comunicar com eles.
3. Projeto de componente. Nesta etapa, o que se almeja é definir as respectivas funcionalidades e responsabilidades para cada componente.
4. Projeto de banco de dados. É nesta atividade que a estrutura de dados e as relações entre essas estruturas são especificadas para serem representadas no banco de dados.

4.1.3 Validação de software

Segundo Myers *et al.* (2004), podemos considerar um erro de *software* "quando o sistema não faz o que o usuário esperar que ele faça". Esta atividade é responsável por assegurar que o *software* que está sendo construído se comporta da forma como o usuário espera, atende suas necessidades, e é capaz de lidar com diferentes situações e entradas (SOMMERVILLE, 2011). Esta atividade também é referenciada como Verificação e Validação.

Os tipos de testes podem ser classificados em: teste de função, teste de sistema, de instalação e de aceitação (MYERS *et al.*, 2004). Neste trabalho, será dada maior ênfase a execução e documentação de testes de integração, de sistema. Os testes de integração podem ser feitos durante a fase de desenvolvimento, de maneira que verifique uma integração que é apenas parte de uma funcionalidade.

Os testes de sistema possuem foco maior na integração entre todos os componentes que compõe uma atividade por completo. Estes buscam encontrar falhas na comunicação entre elementos do sistema que foram desenvolvidos de maneira independente Sommerville (2011). Não é necessário ter conhecimento sobre o código das aplicações para executá-lo, portanto, ele pode ser considerado um teste de caixa preta.

4.1.4 Evolução de software

Depois da entrega e implantação do sistema, a evolução e manutenção do *software* é uma atividade que tende a ocorrer para que o mesmo continue atendendo as novas necessidades que surgem com o tempo. Geralmente, o maior custo que se tem no ciclo de vida do *software* é na evolução, isso porque enquanto o *software* estiver em uso sempre haverão melhorias e correções a serem feitas, e isso pode custar caro a longo prazo (SOMMERVILLE, 2011).

4.2 Aplicações móveis híbridas

A necessidade de se construir aplicativos para *smartphones* que possuem arquiteturas de sistemas operacionais diferentes e portanto, possuem diretrizes diferentes de gerência e execução de programas, se deu através da crescente dependência das pessoas ao uso dos celulares inteligentes (HUI *et al.*, 2013). É possível destacar "o desempenho da aplicação, as implementação das funcionalidades e a experiência do usuário como os principais desafios ao se desenvolver uma aplicação para dispositivos móveis"(BOSNIC *et al.*, 2016).

O conceito de aplicações híbridas vem sendo utilizado por grandes empresas para abranger essa necessidade de reúso de código para mais de um sistema operacional. Inicialmente, os primeiros *frameworks* para desenvolvimento deste tipo de aplicação utilizavam HTML, CSS e JavaScript, que são tecnologias muito conhecidas no ambiente de desenvolvimento para WEB (MALAVOLTA *et al.*, 2015). No entanto, atualmente existem *frameworks* que não se baseiam nessas três tecnologias, como o é o exemplo do Flutter[1], que possui uma forma própria de implementar as interfaces de usuário através dos *widgets*, utilizando a linguagem Dart para construção da lógica da aplicação.

O Flutter foi escolhido como a ferramenta para o desenvolvimento da solução principalmente devido à limitações técnicas para utilização de outras ferramentas. De acordo com Biørn-Hansen *et al.* (2020), que realizaram um estudo de comparação empírica entre diferentes ferramentas de desenvolvimento de aplicações multiplataforma, o flutter se destaca como a melhor ferramenta no quesito uso de RAM Computada. Em uma outra comparação especificamente entre o Flutter e o React Native, em plataformas iOS, o Flutter é melhor em uso de CPU e levemente melhor no uso de GPU de maneira geral (FENTAW, 2020).

Este estudo não busca fazer uma análise aprofundada sobre como aplicações híbridas funcionam internamente e como a comunicação com diferentes tipos de sistemas operacionais é feita, porém é importante apresentar de maneira resumida a ferramenta que será utilizada. Na figura 1 podemos observar as três camadas que compõem o Flutter, discutiremos um pouco sobre cada uma delas.

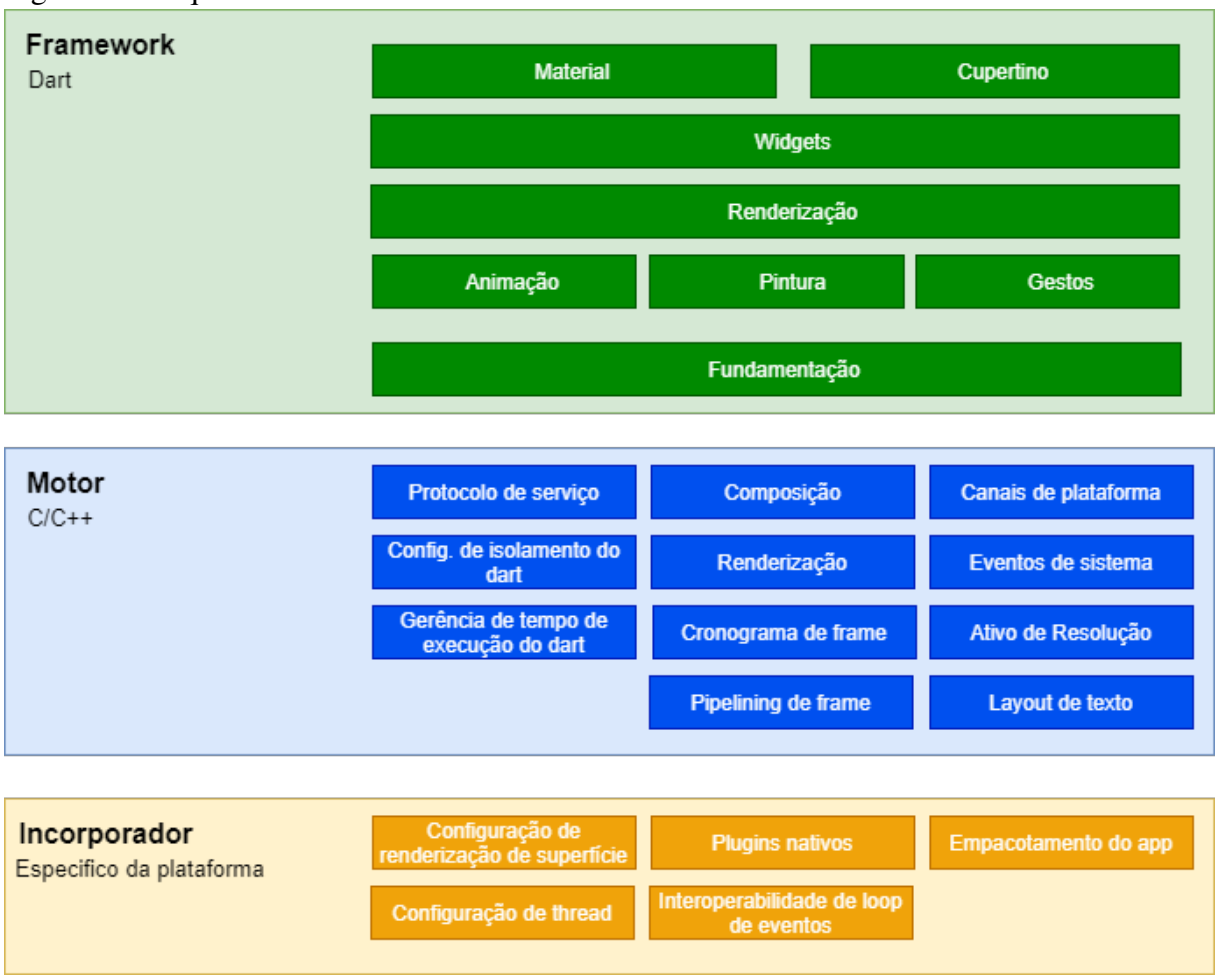
4.2.1 O framework

Sendo a camada mais superior, é responsável por prover ao desenvolvedor um conjunto de bibliotecas de *layout*, plataforma e de fundamentação, todas compostas por várias camadas.

- A fundamentação é responsável por fornecer serviços de animação, pintura e controle de gestos do usuário através de abstrações.
- A camada de renderização provê uma forma de construir *layouts* utilizando a abstração de uma árvore de objetos, que são os *widgets*. Essa árvore é atualizada de maneira reativa às mudanças que ocorrem através da manipulação dinâmica dos objetos que a compõe.

[1] https://flutter.dev/
[2] https://flutter.dev/docs/resources/architectural-overview

Figura 1 – Arquitetura de camadas do Flutter



Fonte: Adaptado de Flutter Docs[2]

- Os *widgets* são abstrações que permitem a implementação dos *layouts* que serão renderizados. Eles podem ser combinados, formando novos *widgets* que podem ser reutilizados, lembrando a lógica de componentização. Como citado anteriormente, essa camada é onde o modelo de programação reativa é introduzido.
- O Material e o Cupertino são bibliotecas utilizadas para implementar *widgets* relacionados aos padrões de design do Material ou iOS.

4.2.2 O motor

Escrito em sua grande parte na linguagem C++, o motor é responsável por rasterizar (processo de conversão de uma imagem vetorial em uma imagem raster) telas compostas sempre que um novo *frame* precisa ser exibido. É nesta camada onde a implementação de baixo nível do núcleo do Flutter se encontra, incluindo os gráficos, *layout* de texto, entrada e saída de rede e arquivos, suporte à acessibilidade, a *runtime* do Dart e o conjunto de ferramentas para compilação.

A forma como o *framework* se comunica e acessa as interfaces oferecidas pelo motor é utilizando o *dart:ui*, uma biblioteca que expõe todos as classes de baixo nível anteriormente citadas.

4.2.3 O incorporador

O incorporador é a camada responsável por fornecer pontos de entrada e as formas de acesso aos serviços do sistema operacional subjacente como renderização de interfaces, acessibilidade e entradas/saídas. Escritos na linguagem adequada para a plataforma, para cada sistema operacional existe um incorporador, permitindo que o código Flutter possa ser integrado a uma aplicação existente como um módulo ou como uma aplicação inteira. Através dele, as aplicações são empacotadas de maneira que para o sistema operacional não haja diferenças em comparação com aplicações nativas.

4.3 *E-voting*

O ato de votar geralmente é associado a uma imprescindível atividade que serve de sustento para o sistema democrático: as eleições. Por isso é necessário que o processo seja seguro do ponto de vista de integridade dos votos, uma vez que estes definem os rumos da sociedade. Muito embora a tecnologia da informação, que vem sendo cada vez mais inserida em todos os contexto da vida humana, esteja nos guiando para realizarmos atividades de maneira cada vez mais eficiente, a votação eletrônica é um caso muito específico onde a aceitação e aplicação da mesma têm sido um processo lento, tendo em vista a criticidade do seu domínio (WANG *et al.*, 2017).

É possível perceber o quão difícil é implementar a lógica relacionada a votação quando observamos os requisitos não-funcionais relacionados à votação, tais como: autenticação, privacidade e auditabilidade. Um votante precisa se autenticar para votar, deseja que seu voto seja contabilizado e que haja uma forma de comprovar isto sem que seu voto seja revelado, tal como é possível em uma cabine de votação comum (GIBSON *et al.*, 2016).

Por sua notória vantagem sobre o sistema de votação tradicional utilizando cédulas de papel, a votação eletrônica tem sido cada vez mais alvo de estudos e pesquisas (KALAIYARASI *et al.*, 2020). Ainda segundo o autor, à medida que *e-voting* vem ganhando espaço, as suas formas de implementação também tem chamado atenção, como é o exemplo das aplicações móveis. O autor também destaca que, apesar de possuir requisitos que são difíceis de serem

atendidos simultaneamente, o processo de votação através de cédulas de papel também não é em sua totalidade confiável, uma vez que possui falhas e alguns casos de fracasso.

Wang *et al.* (2017) realizou uma revisão de literatura sobre o estado da arte do e-voting, e dentre outros aspectos comuns a sistemas deste tipo, definiu um conjunto de requisitos específicos que geralmente são considerados essenciais. Dentre estes, há cinco que serão levados em consideração no desenvolvimento deste projeto, são eles:

- Não Reusabilidade. Este requisito exige que um votante não deve poder votar mais de uma vez, o que previne a ataques que podem fraudar uma votação.
- Usabilidade. O sistema deve atender as diretrizes para oferecer uma boa interação e experiência do usuário. Isto implica em prover uma interface não ambígua, simples e eficiente para que o usuário não cometa erros irreversíveis durante o processo de votação.
- Mobilidade. É um requisito classificado como adicional, no qual plataformas móveis são utilizadas para realizar a votação. No caso do nosso projeto, este requisito será satisfeito uma vez que o resultado deste projeto será um aplicativo *mobile*.
- Privacidade. Somente o votante pode saber qual a opção que ele escolheu, garantindo anonimidade do voto.
- Corretude. O sistema deve ser capaz de contar corretamente os votos, para que um resultado consistente seja apurado.

4.4 Avaliação Heurística

A avaliação heurística é um método de validação de usabilidade que se baseia nas 10 heurísticas de NielsenNielsen (1994), a seguir serão apresentadas cada uma delas:

1. Visibilidade do status do sistema. O sistema deve sempre manter os usuários informados sobre o que está acontecendo através de *feedback* adequado e em tempo aceitável;
2. Correspondência entre o sistema e o mundo real. O sistema deve utilizar palavras, expressões e conceitos que são conhecidos pelos usuários, ao invés de utilizar termos intrínsecos ao sistema ou jargão dos desenvolvedores;
3. Controle e liberdade do usuário. Os usuários frequentemente realizam ações equivocadas no sistema e precisam que uma "saída de emergência" seja nitidamente apontada para evadir do estado indesejado sem ter de percorrer um longo processo. A interface deve permitir ao usuário desfazer e refazer suas ações;

4. Consistência e padronização. Os usuários não devem gastar tempo tentando imaginar se palavras, situações ou ações diferentes significam a mesma coisa. O *designer* deve seguir as convenções da plataforma ou do ambiente computacional;
5. Prevenção de erros. Melhor do que uma boa mensagem de erro é um projeto que considere as ações previsíveis do usuário e evite que um equívoco ocorra, caso seja possível;
6. Reconhecimento em vez de recordação. O *designer* deve tornar os objetos, as ações e opções visíveis. O usuário não deve ter de se lembrar qual é a razão da existência um elemento de interface cujo símbolo não é reconhecido diretamente; nem deve carregar a responsabilidade de recordar de informação de uma parte da aplicação quando estiver em outra fase do fluxo para execução de uma ação, ou realizando outra atividade dentro do sistema. Sempre que necessário, as orientações sobre o uso do *software* devem estar visíveis ou facilmente acessíveis;
7. Flexibilidade e eficiência de uso. Aceleradores — imperceptíveis aos usuários novatos — podem tornar a interação do usuário mais rápida e eficiente, permitindo que o sistema consiga servir igualmente bem os usuários experientes e inexperientes. Exemplos de aceleradores são botões de comando em barras de ferramentas ou teclas de atalho para acionar itens de menu ou botões de comando. Além disso, o *designer* pode oferecer mecanismos para os usuários customizarem ações frequentes;
8. Estética e *design* minimalista. A interface não deve conter informação que seja irrelevante ou raramente necessária. Cada unidade extra de informação em uma interface reduz sua visibilidade relativa, pois compete com as demais unidades de informação pela atenção do usuário;
9. Ajude os usuários a reconhecer, diagnosticar e se recuperar de erros. As mensagens de erro devem ser expressas em linguagem simples, indicar precisamente o problema e sugerir uma solução de forma construtiva;
10. Ajuda e documentação. Embora seja melhor que um sistema possa ser utilizado sem documentação, é necessário oferecer ajuda e documentação de alta qualidade. Tais informações devem ser facilmente encontradas, focadas na tarefa do usuário, enumerar passos concretos a serem realizados e não ser muito extensas.

Além das heurísticas, Nielsen e Mack (1994) definiram também uma escala de

severidade para as violações dessas. Essa escala foi escolhida para ser utilizada pelos avaliadores por ser de fácil interpretação sobre seus significados. Fazem parte da escala os seguintes níveis:

1. Somente um problema cosmético: Não precisa ser consertado a menos que haja tempo sobrando no projeto;
2. Pequeno problema de usabilidade: Deve ser dada prioridade baixa para o conserto deste problema;
3. Grave problema de usabilidade: Importante consertar, então deve ser dada prioridade alta;
4. Catástrofe de usabilidade: Imprescindível consertar antes de o produto ser lançado.

Tanto as heurísticas quanto a escala de gravidade serão utilizadas para realizar a avaliação prevista no passo metodológico descrito na seção 5.4.

5 METODOLOGIA

Neste capítulo é definido a sequência de passos a ser executada de forma que os objetivos, geral e específicos, sejam atingidos.

5.1 Propor melhorias para o sistema já existente

É importante que possíveis falhas e inconsistências no sistema sejam listados, além de melhorias, para que eventuais correções na lógica de funcionamento sejam feitas. Dentre os tipos de falhas, estarão inclusas somente aquelas relacionadas às regras de negócio do sistema. É esperado um relatório de pontos de melhoria e falhas encontradas, para que se tenha uma visão geral do estado da aplicação.

5.2 Implementar as melhorias na *API*

A *API* (do inglês, *Application Programming Interface*) atualmente é consumida por dois sistemas: a urna eletrônica física (Arduíno) e o *website* utilizado pela administração do RU. Tomando como base a arquitetura existente, se fará necessária a implementação de novos *endpoints*, que são pontos de acesso que os clientes da *API*, exclusivos para o aplicativo. Além disso, correção de bugs e melhorias na lógica relacionada à regras de negócio já implementadas serão contempladas, a fim de obter como resultado um código mais eficiente e correto.

5.3 Desenvolver o aplicativo

De acordo com a arquitetura definida, o desenvolvimento é a etapa essencial, uma vez que o seu produto é o próprio aplicativo que será usado pelas pessoas. A tecnologia escolhida para desenvolver o aplicativo é o *framework* Flutter, lançada no ano de 2017, criado e mantido pela Google, e que usa a linguagem de programação Dart. O ambiente de desenvolvimento integrado Android Studio é a ferramenta que será usada para construir e testar as funcionalidades do aplicativo. Além disto, o Figma será a ferramenta utilizada para concepção dos *layouts* das telas de ambas as versões do aplicativo. Espera-se que nesta fase sejam considerados os padrões de projeto e boas práticas de programação, que são base para o código ter bom desempenho, boa legibilidade e seja de fácil manutenção.

Ao fim desta etapa, é necessário que haja uma versão inicial do aplicativo pronta para

ser utilizada tanto para celulares com o sistema Android quanto IOS, e que seja disponibilizada nas lojas oficiais: Google Play Store e Apple Store.

5.4 Realizar avaliação heurística para captar melhorias de usabilidade e sugestões gerais

Após a implementação, deverá ser feito um experimento conduzido com alunos dos cursos da área de tecnologia da informação, que possuam previamente conhecimentos sobre interface humano-computador. Antes avaliarem o aplicativo, os alunos receberão um treinamento para conhecer as 10 heurísticas de usabilidade de Nielsen (1994), bem como a escala de gravidade de violação das heurísticas.

O objetivo ao final desta avaliação é captar os principais pontos de melhoria de usabilidade, que passaram despercebidos na fase de concepção da interação entre o usuário e o sistema. A avaliação será conduzida através de uma reunião online, na qual os avaliadores poderão registrar simultaneamente as suas considerações em um formulário digital. Espera-se coletar dados suficientes para um relatório de melhorias.

5.5 Realizar entrevista semiestruturada com a nutricionista

Esta etapa compreende a realização de uma entrevista para que a nutricionista, que é responsável pelo repasse de informações coletadas nas avaliações para o setor responsável por fornecer o alimento das refeições. Essa entrevista tem como objetivo final validar a solução do ponto de vista dos benefícios que ela traz para o trabalho do grupo responsável pela administração do RU.

A entrevista do tipo semiestruturada prevê que um roteiro predefinido seja utilizado, mas não necessariamente que somente os tópicos ou perguntas previstos sejam estritamente discutidos. É possível que durante o curso da entrevista, pontos específicos surjam em decorrência da discussão dos grandes tópicos tratados.

Espera-se que a entrevista seja gravada e documentada, com ciência de ambas as partes, a fim de que seja gerada uma ata, contendo as principais informações que possam contribuir para o enriquecimento da validação. O documento gerado irá traduzir não só a opinião da equipe de administração, que é um dos usuários finais do sistema, sobre o aplicativo, como também possíveis sugestões de melhoria e novas funcionalidades.

5.6 Implementar as melhorias

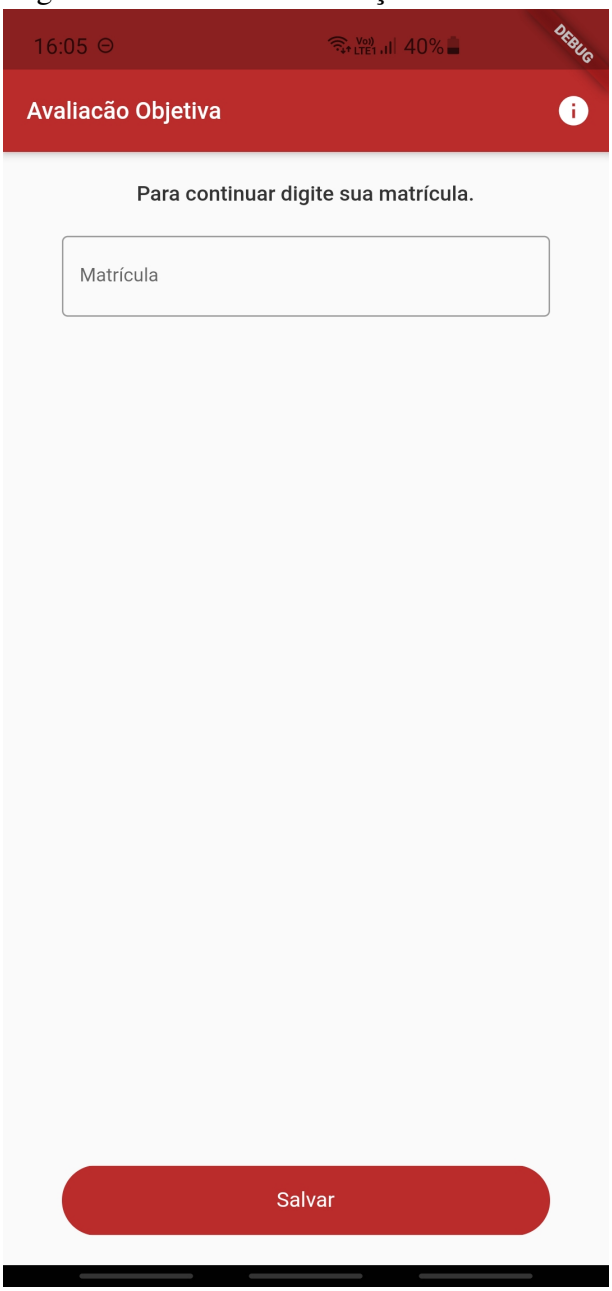
Nesta fase, serão implementadas as correções de *bugs* e melhorias apontadas no relatório. A intenção é fornecer uma segunda versão do aplicativo que seja mais adequada às necessidades dos usuários finais e que seja melhor do ponto de vista da usabilidade.

6 RESULTADOS

6.1 Primeira versão do aplicativo

Nesta seção, as telas da primeira versão do aplicativo para *Android* serão exibidas, será explicado como ocorre o fluxo de passos envolvidos no processo de envio das avaliações, e também será discutido como foi realizada a verificação e validação dos requisitos do sistema.

Figura 2 – Tela de identificação com matrícula

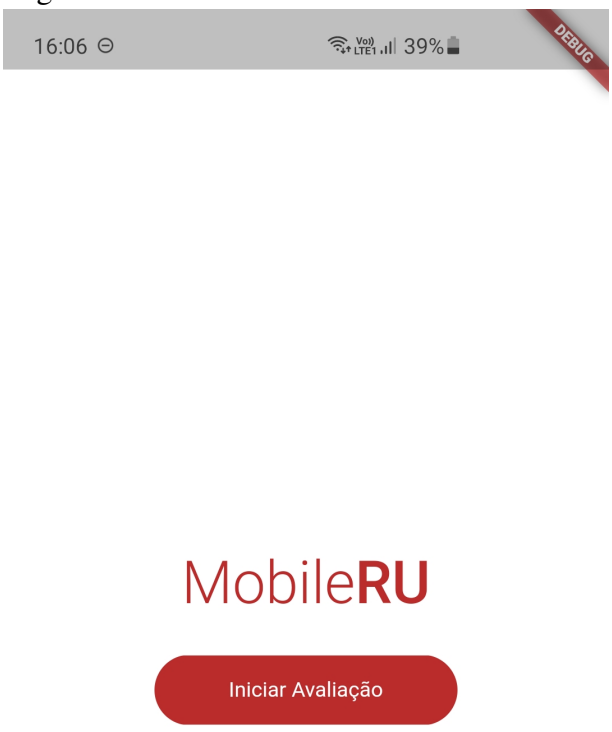


Fonte: O autor

No primeiro acesso, a primeira tela que o usuário irá encontrar é a de inserção de

matrícula, apresentada na figura 2. O número é uma entrada obrigatória para que o usuário se identifique, e assim o sistema seja capaz de validar, por exemplo, caso ele tente votar mais de uma vez. Ao clicar em salvar, o sistema salva esse dado em um local no armazenamento onde somente o próprio aplicativo pode acessar, e que será consultado sempre que for necessário realizar uma nova requisição. O dado persiste mesmo que o aplicativo seja encerrado, portanto, o usuário só precisa informá-lo uma única vez. Na barra superior, é possível encontrar um botão de informações, onde se encontram as explicações ao usuário sobre o uso e armazenamento da matrícula.

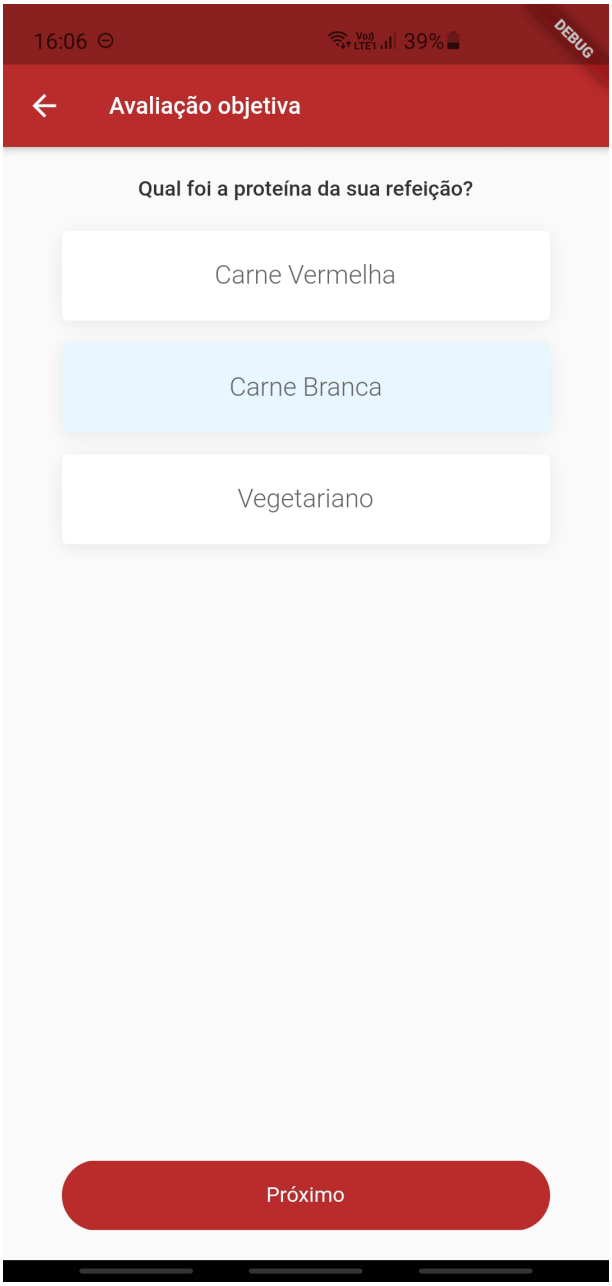
Figura 3 – Tela inicial



Fonte: O autor

Após a matrícula ser informada, o usuário é redirecionado para a tela inicial, tal como ilustrada na figura 3 a primeira tela exibida em posteriores utilizações do *software*. Nesta tela, o botão "Iniciar Avaliação"só será exibido após o servidor confirmar que está em horário de votação e que no horário da refeição atual este usuário ainda não votou. É também nesta tela onde será exibida a mensagem de justificativa quando o usuário não puder iniciar uma avaliação.

Figura 4 – Tela de alternativas de proteína

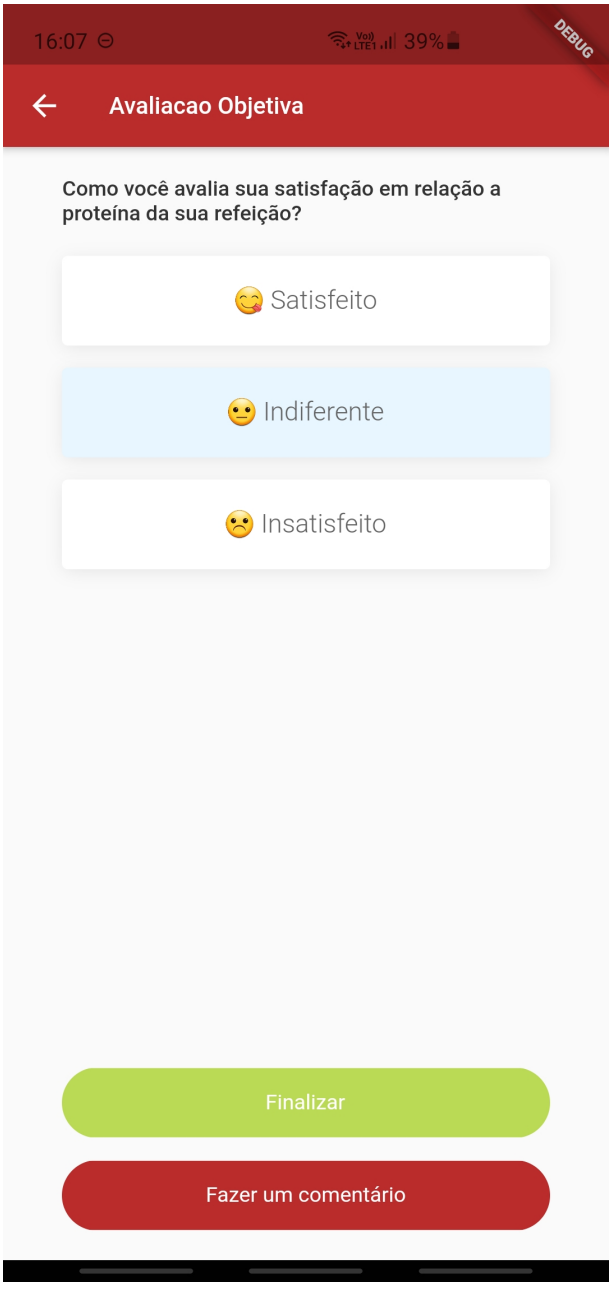


Fonte: O autor

Na figura 4 é possível visualizar a tela de alternativa de proteína, onde o usuário irá selecionar qual a opção correspondente à proteína que ele comeu na refeição. O botão

"Próximo"só irá surgir após ser selecionada alguma opção, sendo impossível avançar sem selecionar nenhuma alternativa.

Figura 5 – Tela de alternativas de satisfação

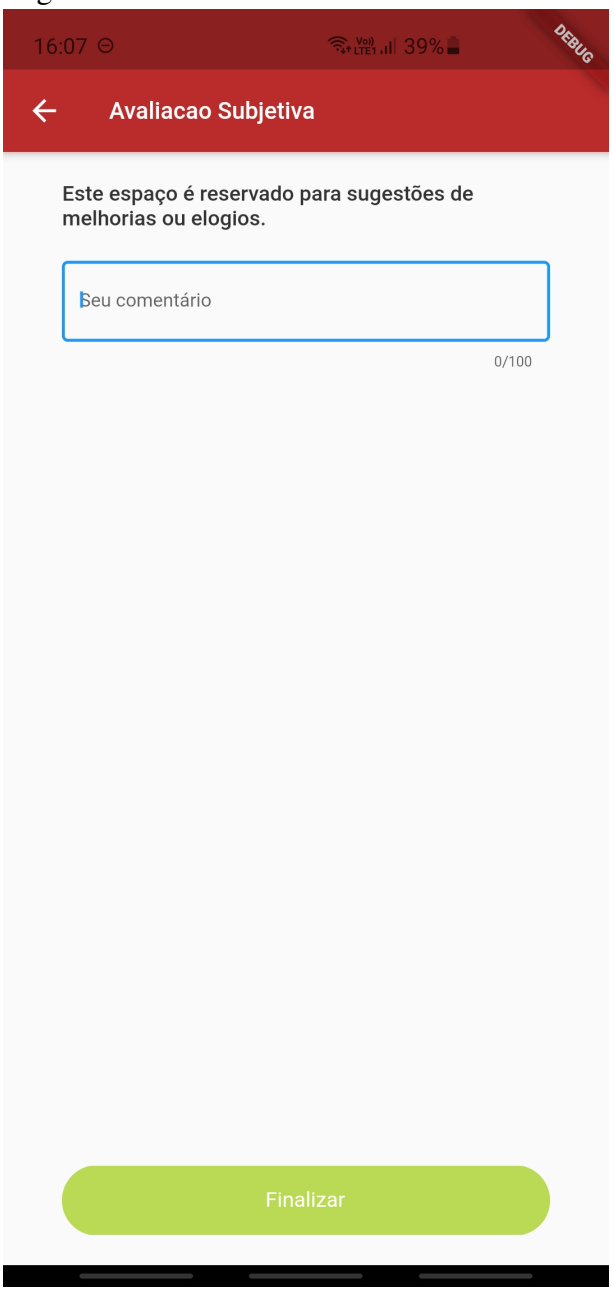


Fonte: O autor

Na tela seguinte, representada na figura 5, a pesquisa em relação a satisfação das pessoas é feita. É possível selecionar uma alternativa que represente sua satisfação. A partir de então, o usuário tem duas opções: finalizar a avaliação ou fazer um comentário.

É possível tecer um breve comentário na tela exibida pela figura 6, indicando alguma observação, elogio ou sugestão de melhoria em relação à proteína que ele comeu. Este comentário

Figura 6 – Tela de submissão de comentário



Fonte: O autor

é limitado a 100 caracteres e é salvo separadamente do voto, de modo que não é possível descobrir qual votante fez determinado comentário. Após finalizar a sua avaliação, é apresentada ao usuário uma tela de agradecimento pelo voto.

A figura 7 mostra a tela de agradecimento, que é exibida somente em caso de sucesso ao computar o voto. Neste passo do fluxo, somente uma ação pode ser feita, que é sair do aplicativo. Isto para evitar que o usuário tente voltar para as telas de alternativa para submeter outro voto, comprometendo a corretude da contagem de votos.

Figura 7 – Tela de agradecimento

Fonte: O autor

Os códigos do aplicativo[1] e da API[2] estão disponibilizados em repositórios públicos, para que possa ser analisado por qualquer um que tenha interesse em conhecer melhor sobre como foram implementadas as soluções para o atendimento dos requisitos ou em contribuir para a manutenção do sistema tenha acesso.

[1] https://github.com/JosueNicholson/mobile-ru
[2] https://github.com/JosueNicholson/api-ru

6.1.1 Verificação e validação

A verificação do bom funcionamento e tratamento de erro das funcionalidades foram feitas através de testes de sistema, os quais foram documentados em uma planilha de testes[3]. À medida em que essas funcionalidades foram sendo desenvolvidas, os testes relacionados a elas foram realizados e os *bugs* encontrados foram corrigidos. Ao fim da fase de desenvolvimento, a suíte de testes de sistema foi executada de maneira integral.

Em relação aos requisitos específicos do domínio de *e-voting*, é possível dizer que dentre os que foram selecionados para compôr a base da concepção do projeto, todos foram satisfeitos. Será apresentada a seguir a implementação de elementos que fazem parte da validação destes requisitos.

6.1.1.1 Não reusabilidade e privacidade

Para garantir a não reusabilidade, ou seja, que um votante não consiga realizar a votação mais de uma vez, foi preciso usar um identificador único, que não fosse um dado pessoal crítico do usuário. A opção escolhida foi a matrícula. Esse número é salvo somente em um local no armazenamento do *smartphone* de uso exclusivo do aplicativo, evitando que outros programas tenham acesso.

No banco de dados é salvo apenas um *hash*, que é uma representação de um dado após passar pelo processamento de uma função *hash*, que por sua vez trata de transformar o dado de entrada em um dado de comprimento fixo. A função escolhida para isso foi a MD5, pois sempre gera a mesma saída para a mesma entrada e não é possível chegar ao texto de entrada tendo como insumo apenas o texto de saída. Isso garantiu que a anonimidade do usuário fosse preservada e, ao mesmo tempo, deu ao sistema a capacidade de fazer comparações entre o *hash* da matrícula de alguém que deseja votar com os *hashes* existentes no banco de dados.

A figura 8 mostra como foi implementado o *endpoint* requisitado assim que o usuário inicializa o aplicativo, ou após inserir a matrícula. Nele estão contidas as validações para todos os cenários descobertos até a fase de testes. Essas validações tratam, por exemplo, casos como: o aplicativo é acessado fora do horário de refeição padrão (para o almoço, entre 11:00 e 14:00 horas e para a janta, entre 17:00 e 19:00) e quando é tentado realizar a avaliação mais de uma vez para a mesma refeição.

[3] https://bit.ly/3sbvxBC

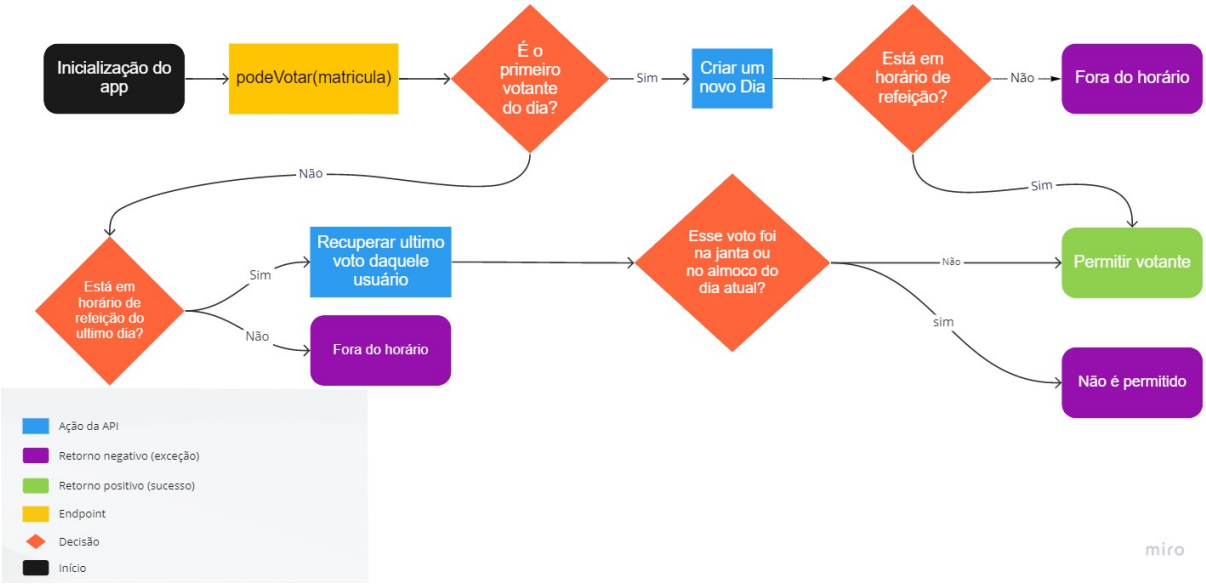
Figura 8 – Função para verificar se o usuário pode votar

```
43      @GetMapping("podeVotar/{matricula}")
44      public ResponseEntity<Object> podeVotar(@PathVariable String matricula) {
45          Object body = diaController.getUltimoDia().getBody();
46          Dia ultimoDia = body != null ? (Dia) body : null;
47          Calendar hoje = Calendar.getInstance(TimeZone.getTimeZone("GMT-03:00"));
48          if(ultimoDia == null || hoje.get(Calendar.DAY_OF_MONTH) != ultimoDia.getDia()) {
49              Dia novoDia = new Dia(hoje.get(Calendar.DAY_OF_MONTH), hoje.get(Calendar.MONTH)+1, hoje.get(Calendar.YEAR));
50              if(!novoDia.isAlmocoTime() && !novoDia.isJantaTime()) {
51                  return ResponseEntity.status(HttpStatus.PRECONDITION_FAILED).body("Não está em horário de refeição");
52              }
53              salvarNovoDia(novoDia);
54              return ResponseEntity.ok("");
55          }
56          else if(!ultimoDia.isAlmocoTime() && !ultimoDia.isJantaTime())
57              return ResponseEntity.status(HttpStatus.PRECONDITION_FAILED).body("Não está em horário de refeição");
58          String matriculaHash = "";
59          try {
60              matriculaHash = Hasher.getHash(matricula);
61          } catch (NoSuchAlgorithmException e) {
62              ResponseEntity.status(HttpStatus.BAD_REQUEST).body("algo deu errado com a matricula");
63          }
64          if(!votadosService.jaVotou(matriculaHash, ultimoDia)) {
65              return ResponseEntity.ok("");
66          }
67          return ResponseEntity.status(HttpStatus.CONFLICT).body("Voce nao pode votar");
68
69      }
70
```

Fonte: O autor

O fluxograma apresentado na figura 9 foi elaborado com a intenção de explicar para os leigos o design da primeira comunicação entre o aplicativo e o servidor, bem como o processamento interno feito no servidor para garantir o atendimento aos dois requisitos deste tópico através das verificações iniciais citadas anteriormente.

Figura 9 – fluxograma de verificação da capacidade do usuário de realizar uma avaliação



Fonte: O autor

É importante destacar alguns pontos sobre esse fluxo. A ação da API chamada "Criar um novo Dia"se refere ao comando de criação no banco de dados de um dado de formato definido no *model* que representa um dia. Esse comando se justifica sob a necessidade futura de já haver uma refeição criada no banco para um determinado dia, pois a requisição subsequente será para

atualizar essa refeição, computando um novo voto relacionado a ela.

Outro ponto é a verificação de horários em dois pontos divergentes no fluxo. A verificação após a criação de um novo dia é justificável pois, pensando na evolução do sistema, um dia específico pode ter horários de almoço ou janta diferentes dos habituais, logo cada dia poderia ter seu horário definido no momento da sua criação, necessitando de uma averiguação de horário diferente dos outros dias.

6.1.1.2 Usabilidade

Para validar a usabilidade do aplicativo, foi realizada uma avaliação heurística, tendo como base o planejamento apresentado na seção 5.4. A avaliação contou com a participação de 7 avaliadores, alunos do curso de Engenharia de Software, que já haviam feito a disciplina de IHC (Interface Humano-Computador) e portanto, possuíam conhecimentos básicos nesta área.

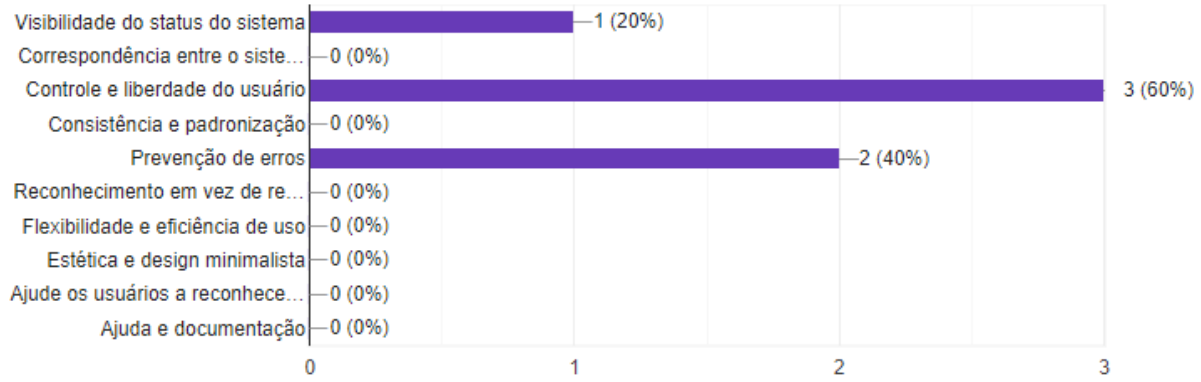
Após um breve treinamento para explicar sobre as heurísticas de Nielsen, assim como a escala de severidade definida por ele, foi apresentado como se dá o processo de avaliação. Foram selecionadas 5 atividades principais dentro do sistema para serem avaliadas, são elas: Salvar matrícula, escolher proteína, escolher alternativa de satisfação, fazer um comentário e concluir a avaliação. Para cada dessas atividades, foi requisitado que a cada avaliador que respondesse as seguintes questões:

- Quais heurísticas estão sendo quebradas?
- Qual o motivo?
- Qual é a severidade?
- Qual é a sugestão de resolução?

Era possível responder que mais de uma heurística estava sendo violada, desde que as três últimas perguntas acima fossem respondidas para cada violação. Estas respostas foram coletadas através de um formulário *online*.

Após a coleta de todas as respostas, um relatório foi documentado [4] contendo todas as informações referentes as avaliações. A seguir, discutiremos sobre as heurísticas que foram consideradas violadas pela maioria dos avaliadores para cada atividade.

Figura 10 – Quantidade de votos sobre heurísticas violadas na tela de salvar matrícula



Fonte: O autor

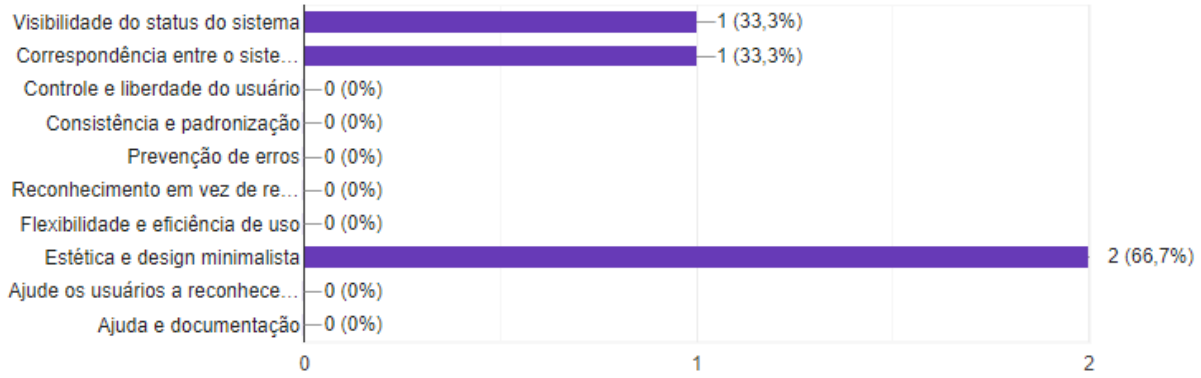
6.1.1.2.1 Atividade 1 - Salvar a matrícula

Na figura 10 é possível observar que a maior parte das heurísticas violadas na primeira atividade foram "Controle e liberdade do usuário"e "Prevenção de erros". 5 dos 7 participantes foram responsáveis por estas considerações. Para ambas, foi classificada a severidade 2 - Pequeno problema de usabilidade. O motivo relacionado à primeira é que a matrícula é um dado não modificável, e o usuário não tem a opção de edição caso coloque uma matrícula errada. Já o motivo atrelado a segunda é que não há validação quanto ao tamanho do número da matrícula.

Uma das soluções recomendadas para "Controle e liberdade do usuário"foi permitir a edição de matrícula até o momento antes de ser enviada a primeira avaliação do usuário. Isso permitiria que o usuário pudesse ter o controle de edição sem que entrasse em conflito de não reusabilidade, uma vez que caso a edição pudesse ser feita mesmo depois de enviar a avaliação, um usuário poderia fazê-la para enviar um segundo voto para a mesma refeição.

Para "Prevenção de erros"foi recomendada a inserção de uma validação de tamanho limite. Para isso é preciso somente pesquisar qual é o tamanho da maior matrícula dentre os perfis que se alimentam no restaurante. No entanto, validações em relação a existência de uma determinada matrícula só seriam possíveis através de uma integração com o banco de dados da UFC. Isso será citado como trabalho futuro na conclusão. Além disso, será inserido também um campo de confirmação de matrícula, para que seja reforçada a prevenção de erros de digitação ao inserir este dado imutável.

Figura 11 – Quantidade de votos sobre heurísticas violadas na tela de escolher a proteína



Fonte: O autor

6.1.1.2.2 Atividade 2 - Escolher a proteína

A figura 11 revela menos heurísticas violadas na segunda atividade em comparação com a primeira atividade, que a segunda que conteve mais problemas de usabilidade expostos. Os 4 apontamentos em relação a essa foram feitos por 3 dos 7 avaliadores. É possível analisar que "Estética e design minimalista"foi a heurística que obteve mais votos negativos.

Em relação a heurística citada anteriormente, o motivo que os avaliadores apontaram como causador da violação foi a cor de foco quando uma opção de proteína é selecionada é muito sutil. Isto foi considerado pelos dois avaliadores como um pequeno problema de usabilidade e, a solução apresentada por eles foi igual: escolher um tom de cor que dê mais contraste à alternativa selecionada em relação as demais.

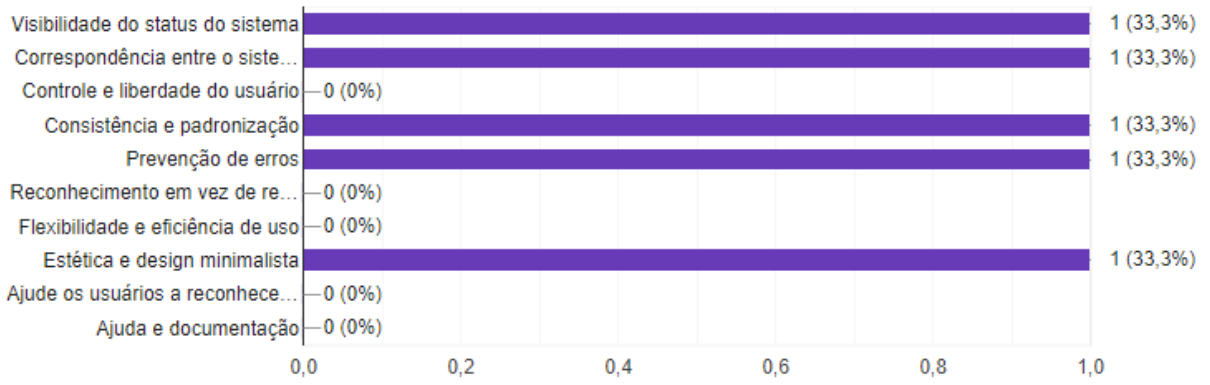
Em relação às demais heurísticas, a razão apresentada pelo avaliador que considerou "Correspondência entre o sistema e o mundo real"infringida com severidade nível 2, foi que o estilo de botão escolhido para as alternativas não informava que somente uma delas poderia ser escolhida. A solução proposta foi mudar este estilo para *radio*. A quebra da heurística "Visibilidade do status do sistema"será discutida de forma individual após a apresentação dos dados relacionados à quarta atividade.

6.1.1.2.3 Atividade 3 - Escolher alternativa de satisfação

Apenas 3 dos 7 avaliadores afirmaram que haviam heurísticas sendo infringidas na atividade 3, cujos dados são apresentados na figura 12. No entanto, esta atividade teve o maior número de infrações, juntamente com a atividade 1, porém os votos foram divididos igualmente, o que revela que ela possui mais problemas de usabilidade. Estes serão discutidos a seguir.

[4] https://bit.ly/2VCr9A8

Figura 12 – Quantidade de votos sobre heurísticas violadas na tela de escolher alternativa de satisfação



Fonte: O autor

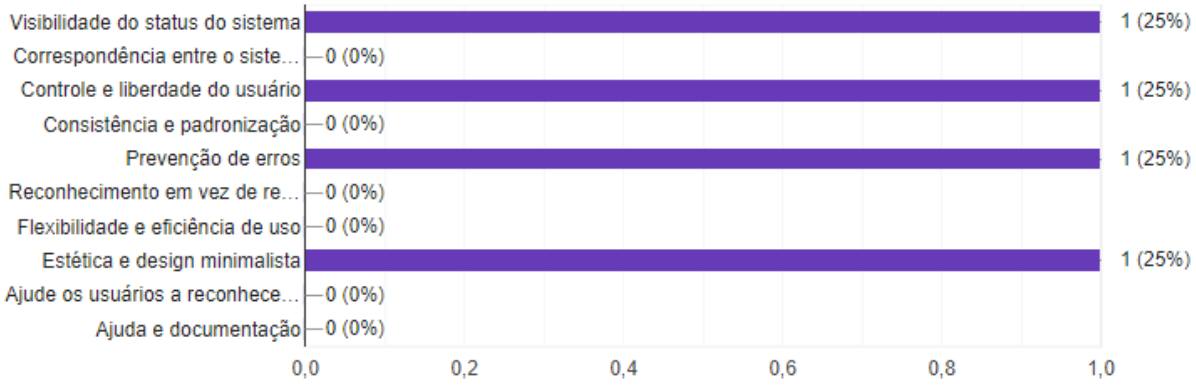
A heurística "Correspondência entre o sistema e o mundo real"foi apontada como violada pelo mesmo motivo, com a mesma severidade e recomendação da atividade anterior. Já o motivo relativo a quebra da "Consistência e padronização"foi a posição do botão "finalizar", e isto foi classificado como um pequeno problema de usabilidade. A recomendação feita foi alterar este posicionamento do botão para a parte inferior, além de mover o campo de comentário para abaixo das alternativas de satisfação.

A razão apresentada para a violação da "Prevenção de erros"foi a falta de uma confirmação de finalização caso o usuário deseje finalizar a avaliação, e foi considerado somente um problema cosmético. A recomendação foi apresentar ao usuário um resumo de sua avaliação para que ele faça a conferência de suas escolhas.

As causas apontada para a infração de "Estética e design minimalista"foram a cor do botão ao selecionar uma alternativa e erros de digitação. Essa infração foi classificada como pequeno problema de usabilidade e teve como sugestão de solução a mudança do tom de cor da alternativa selecionada e correção dos erros de digitação.

Por fim, houve uma heurística que o avaliador não marcou, por isso não é mostrado no gráfico gerado, porém as 3 perguntas relacionadas foram respondidas. Se trata da "Ajuda e documentação". A causa dessa infração se deve ao fato de os termos "satisfeito", "indiferente"e "insatisfeito"não serem fáceis de terem seus significados compreendidos, e essa violação foi considerada um grave problema de usabilidade. A recomendação foi adicionar uma breve descrição dos termos nas alternativas (e.g, "Satisfeito: adorei o sabor").

Figura 13 – Quantidade de votos sobre heurísticas violadas na tela de fazer um comentário



Fonte: O autor

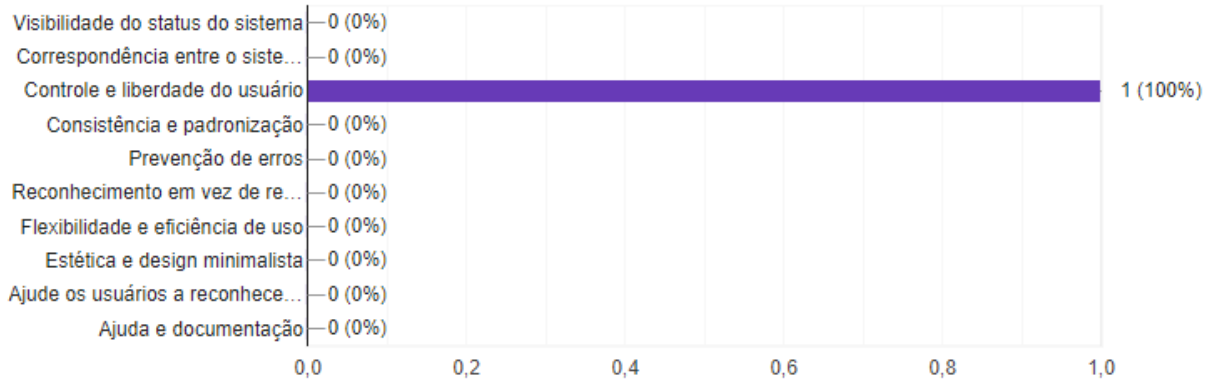
6.1.1.2.4 Atividade 4 - Fazer um comentário

Na quarta atividade, os votos exibidos na figura 13 evidenciam 4 problemas de usabilidade, dentre os quais "Controle e liberdade do usuário"teve a maior severidade, tendo sua causa classificada como um grave problema de usabilidade. O motivo apontado foi a obrigatoriedade de finalizar a avaliação sem a impossibilidade de, após escrever um comentário, voltar para as opções de proteína e satisfação sem perder o texto digitado. A recomendação foi adicionar um botão para salvar o texto.

Para a violação da "Prevenção de erros", foi atribuída a severidade nível 2, e sua justificativa e recomendação foram as mesmas relacionadas a violação desta na atividade anterior. Finalmente a infração da "Estética e design minimalista"também foi classificada com severidade nível 2, o motivo está na quantidade de linhas, ou seja, tamanho vertical do campo de comentário ser insuficiente para 100 caracteres. A solução apontada foi aumentar esse tamanho vertical de forma que não seja necessário rolar o campo para visualizar todo o comentário.

A quebra da heurística "Visibilidade e status do sistema"se repetiu nestas 4 primeiras atividades, e possui motivo, severidade e recomendação iguais. Se justifica na observação de que a barra no topo das telas não possui um título adequado, que informe para o usuário especificamente qual ponto no fluxo de votação ele se encontra. A severidade desta violação de nível grave e recomendação foi encontrar títulos para a barra superior que informem o *status* atual do processo de avaliação.

Figura 14 – Quantidade de votos sobre heurísticas violadas na tela de concluir a avaliação



Fonte: O autor

6.1.1.2.5 Atividade 5 - Concluir a avaliação

A quinta e última atividade, que tem seus votos representados na figura 14, foi aquela que obteve menor quantidade de problemas apontados, somente 1. A heurística relacionada a este apontamento foi "Controle e liberdade de usuário", que obteve a classificação "grave problema de usabilidade", e se fundamentou na impossibilidade de voltar para a primeira tela para iniciar uma segunda avaliação, que poderia ser de um segundo usuário que não possui *smartphone*. Recomendou-se adicionar o botão para voltar para a tela inicial, e fosse possível iniciar outra avaliação.

Infelizmente, a última recomendação entra em conflito com o requisito de não reusabilidade, e portanto, não pode ser a solução adotada para o problema, uma vez que não há como validar se pode estar havendo ou não a tentativa de fraude na votação. Pensando no uso malicioso, isso permitiria a adição de quantos votos o usuário quiser.

6.1.1.3 Corretude

Em relação à corretude, foi utilizada também a avaliação para validar este requisito. À medida em que os avaliadores realizavam todas as atividades, a consequência disto era um novo voto computado. Foi observado que ao fim de todas as avaliações, haviam 7 votos computados, comprovando que o sistema foi capaz de coletar corretamente os votos. A titulo de verificação, foi questionado à cada avaliador de maneira particular em qual proteína e qual satisfação ele havia votado, e as respostas coincidiram com as contagens.

6.1.1.4 Mobilidade

A Mobilidade é uma requisito específico opcional quando se trata de *e-voting*. É trivial validar e explicar como esse requisito foi atendido pois, devido a natureza da aplicação desenvolvida, já é possível afirmar que o sistema atende o atende, uma vez que se trata de um *software* para *smartphones*.

6.1.2 Validação com a nutricionista

Foi realizada uma entrevista semiestruturada com a nutricionista, a fim de que fosse validada a solução com o perfil de administrador, procurando descobrir se os requisitos foram cumpridos, se as necessidades foram atendidas e qual era a sua opinião sobre a interação com o aplicativo. O roteiro predefinido iniciou-se com uma apresentação dos objetivos da entrevista, citados anteriormente, seguindo para a apresentação tanto do *software* desenvolvido quanto da modificação feita na tela de resultados das votações no sistema web, utilizado somente pelo administrador.

Após isto, foram realizadas as seguintes perguntas:

- Quais os pontos principais que você destacaria como vantagens do aplicativo?
- Você destacaria algum ponto negativo em termos de funcionalidade ou requisitos não atingidos? Se sim, quais?
- Como você avalia a relevância do aplicativo de maneira geral? E na situação em que vivemos, com a possibilidade de volta gradual das aulas durante a pandemia, acha que ele pode ser ainda mais relevante?
- Na sua opinião, este aplicativo pode, de fato, facilitar de alguma maneira o trabalho das pessoas que trabalham na administração RU?
- Que sugestões de melhoria ou de novas funcionalidades você daria para o aplicativo?

Segundo a nutricionista, o principal destaque positivo do aplicativo é a resolução problema do de segurança sanitária causado pelo contato com a urna eletrônica física, que poderia impactar na quantidade de avaliações coletadas uma vez que as pessoas têm evitado tocar em objetos compartilhados. Isto já responde também a pergunta sobre a relevância do *software* nas circunstâncias da pandemia. Outro ponto de destaque foi o importante papel das notificações para lembrar as pessoas de realizarem a votação.

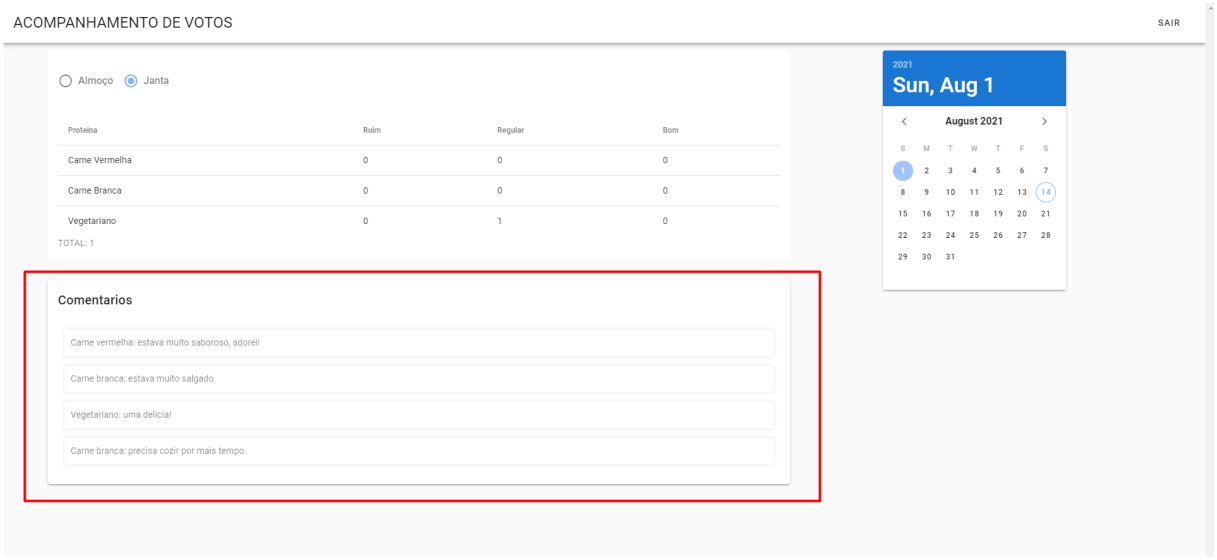
Foi respondido que, de maneira geral, o aplicativo possui ótima relevância por se

tratar de uma urna alternativa. No entanto foi salientado que devido à condição socioeconômica de uma parte dos estudantes, a principal desvantagem é que o sistema não abrange pessoas que não possuem *smartphone* ou que não possuem nele espaço suficiente para a instalação.

Com relação a facilitação do trabalho da administração, foi afirmado que sim, o aplicativo ajuda o trabalho pois automatiza a contagem dos votos e contribui para a economia de papel utilizado na urna física utilizada atualmente.

Por fim, houveram sugestões de melhoria para o aplicativo, tais como: Exibir o aviso de uso da matrícula quando o usuário clica no botão "Salvar"e ao salvar um comentário, sempre atrela-lo à proteína escolhida. Além disso, foi ressaltada a importância de atribuir um significado mais claro sobre as alternativas de satisfação, uma vez que o usuário está acostumado com outra nomenclatura. Essa sugestão coincidiu com uma recomendação de um dos avaliadores na avaliação heurística. No *website*, foi sugerida a funcionalidade de gerar relatório da refeição contendo data e hora de emissão, dados da refeição referida (votos e comentários), e dia da refeição.

Figura 15 – Tela de resultado de votação



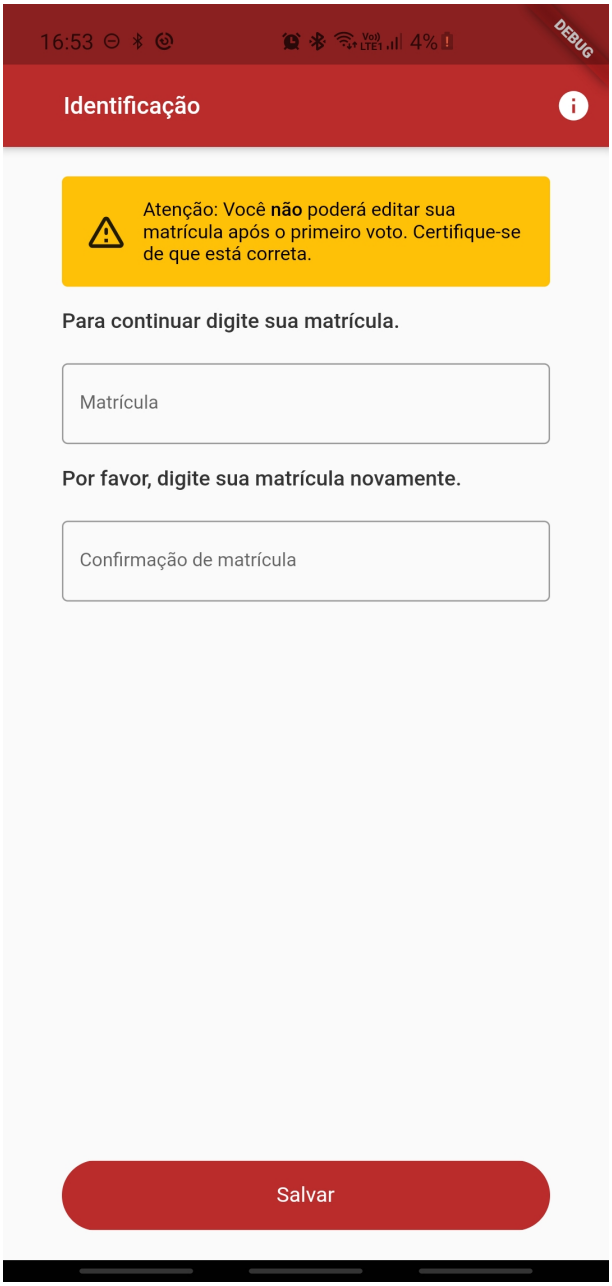
Fonte: O autor

A figura 15 destaca a seção de comentários que foi adicionada na tela de resultados. No início de cada comentário o prefixo correspondente ao tipo de proteína escolhida foi inserido. Esse prefixo é adicionado automaticamente pelo aplicativo antes de enviar a avaliação para a API, essa foi a solução encontrada para o problema da relação entre o comentário e o tipo da proteína que não havia sido considerado em tempo de desenvolvimento.

6.1.3 Segunda versão do aplicativo

Uma segunda versão foi desenvolvida, a fim de corrigir os problemas de usabilidade apontados pelo relatório da avaliação heurística e implementar sugestões de melhoria feitas pela nutricionista. Somente algumas telas sofreram alterações, elas são apresentadas e justificadas a seguir.

Figura 16 – Tela de identificação com matrícula



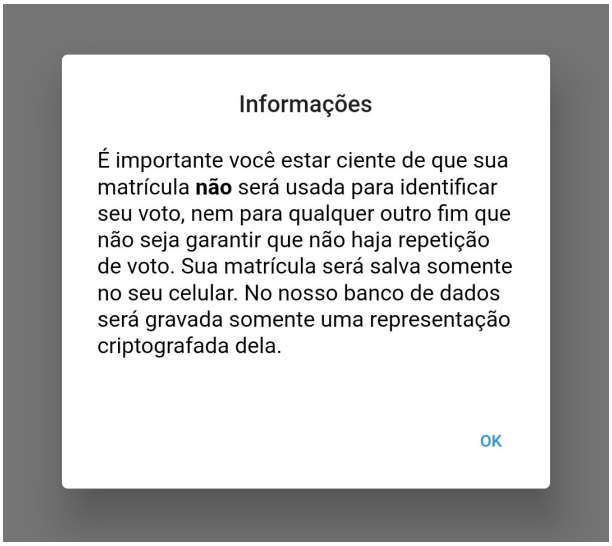
Fonte: O autor

A tela que a maioria dos avaliadores consideraram problemática foi a de inserção de matrícula. As mudanças em relação a ela são apresentadas na figura 16. Um campo de

confirmação foi adicionado, para tentar prevenir um possível erro de digitação. Além disso, foi inserido uma mensagem de alerta para que o usuário fique ciente da imutabilidade da matrícula e o título da barra do topo foi alterado para que traduzisse melhor em que fase do processo se encontra o usuário.

O *modal* mostrado na figura 17, que traz informações importante sobre o uso da matrícula, que antes só era exibido caso o usuário tocasse no ícone de informações na barra de topo da tela de identificação, agora também é exibido ao tocar no botão "Salvar"desta tela.

Figura 17 – *Modal* de informações sobre matrícula

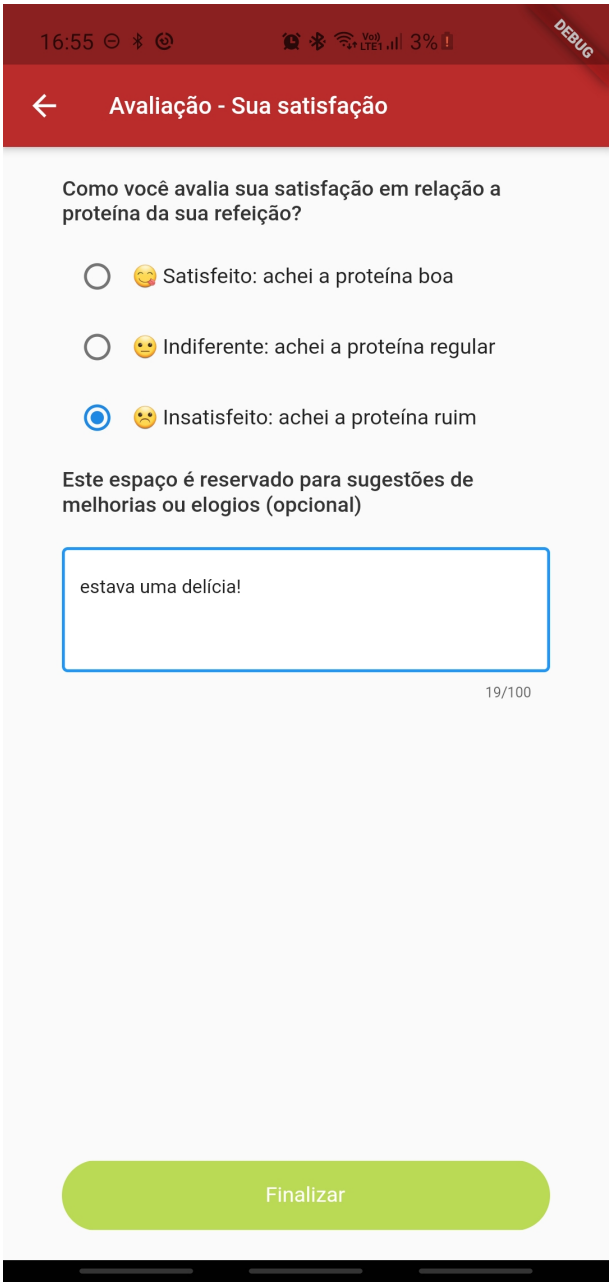


Fonte: O autor

A tela de alternativas de satisfação foi aquela que apresentou mais problemas apontados pelos avaliadores. Exibida a sua nova versão na figura 18, é possível observar que o comentário, que antes possuía uma tela exclusiva para ele, foi movido para esta. Também foi modificado o estilo dos botões de seleção da alternativa para atender ao padrão definido pelo guia do *Material UI* (a biblioteca de componentes da Google). As opções da tela de escolha da proteína também sofreram esta alteração.

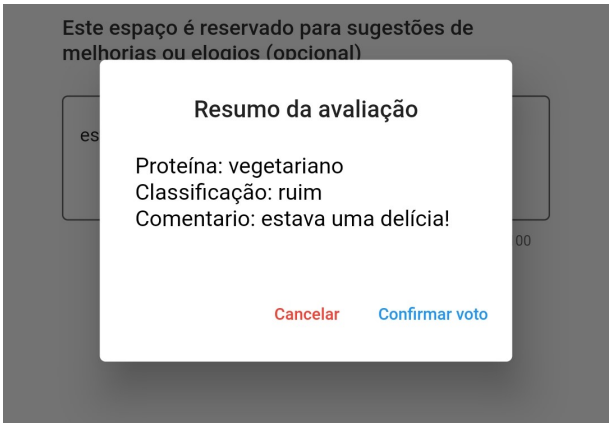
Além disto, é possível observar que o título na barra de topo foi modificado com a mesma justificativa apresentada para a tela de identificação. As opções agora apresentam uma breve descrição do que significa cada uma destas. Por fim, o tamanho vertical do campo de comentário foi aumentado de maneira que fique visível todo o comentário que o usuário digitou. Por fim, foi adicionado um *modal* de confirmação de voto, conforme mostra a figura 19, que apresenta um resumo da avaliação para que o usuário confira se aquele voto de fato expressa sua opinião.

Figura 18 – Tela de alternativas de satisfação



Fonte: O autor

Figura 19 – *Modal* de confirmação de voto



Fonte: O autor

7 CONCLUSÃO

7.1 Conclusão

Diante disso, através da validação da usabilidade e da entrevista semiestruturada com a nutricionista, este trabalho conclui a versão inicial do aplicativo que, integrado à estrutura do sistema, foi capaz de cumprir os objetivos deste trabalho e os requisitos previamente definidos pela nutricionista, contribuindo para a automatização de uma tarefa atualmente desgastante para a equipe de administração.

Além disso, é possível afirmar que o *software* possui um bom nível de usabilidade visto que dentre os problemas encontrados na primeira versão, que são naturais de qualquer sistema, nenhum foi classificado como uma catástrofe de usabilidade, ou seja, algo que impedisse a entrega do sistema para o usuário final. Para reforçar essa conclusão, a segunda versão seguiu recomendações dos avaliadores para resolver os problemas mais graves.

Apesar de ter sido um inserido como uma nova parte do sistema que computa os votos, contribuindo para a evolução do mesmo, é de suma importância considerar que do ponto de vista socioeconômico, o aplicativo pode não ser uma alternativa viável para uma parcela de estudantes que não possuem *smartphone* ou que possuem, porém com espaço de armazenamento insuficiente para instalar o *software*, mesmo considerando é pequeno o tamanho da memória utilizada.

Desenvolver este trabalho me trouxe grandes aprendizados no âmbito técnico pois me fez adquirir experiência ao encarar desafios que o processo de desenvolvimento de *software* faz emergir. Dentre as maiores dificuldades, destaco o aprendizado da tecnologia utilizada (Flutter) para desenvolver o aplicativo e o da ferramenta de prototipação das telas, mas consegui contorná-las através de pesquisas em fóruns de comunidades e portais de dúvidas e respostas.

7.2 Trabalhos futuros

O aplicativo ainda possui poucas funcionalidades, mesmo tentando resolver as principais dores dos usuários. Possíveis extensões podem incluir *features* como por exemplo ver cardápio do dia e da semana e espaço para enquetes. Além disso, para abranger um número maior de usuários, a interface utilizando a biblioteca de componentes do sistema iOS (Cupertino) precisa ser implementada.

Outro ponto a destacar é a integração com o SI3 (também conhecido com SIGAA - Sistema Integrado de Gestão de Atividades Acadêmicas) para verificação da existência da matrícula informada. Caso essa integração fosse feita, também seria possível, por exemplo, incluir a funcionalidade de ver saldo de *tickets*.

Além disso, a lógica de armazenamento do *hash* pode ser melhorada, através do tratamento de *hashes* duplicados, que foi uma verificação não abrangida pelo trabalho apresentado. Também pode ser re-estudado o uso do algoritmo MD5, uma vez que ele atende às necessidades atuais de segurança, porém existem algoritmos mais modernos e confiáveis que poderiam ser empregados, como o SSH1.

Para o sistema utilizado pela nutricionista, existem funcionalidades que podem ser implementadas, como, por exemplo a geração de relatórios das refeições em formato *pdf*. Ainda é possível pesquisar mais a fundo através de entrevistas e estudos de campo para entender sobre dores do usuário que não foram contempladas por este estudo.

REFERÊNCIAS

BIØRN-HANSEN, A.; RIEGER, C.; GRØNLI, T.-M.; MAJCHRZAK, T. A.; GHINEA, G. An empirical investigation of performance overhead in cross-platform mobile development frameworks. **Empirical Software Engineering**, Springer, [*S. l.*], v. 25, p. 2997–3040, 2020.

BOSNIC, S.; PAPP, I.; S, N. The development of hybrid mobile applications with apache cordova. In: **2016 24th Telecommunications Forum (TELFOR)**. [*S. l.*: *s. n.*], 2016. p. 1–4.

FENTAW, A. E. **Cross platform mobile application development: a comparison study of react native vs flutter**. [*S. l.*: *s. n.*], 2020.

GIBSON, J. P.; KRIMMER, R.; TEAGUE, V.; POMARES, J. A review of e-voting: the past, present and future. **Annals of Telecommunications**, Springer, [*S. l.*], v. 71, n. 7, p. 279–286, 2016.

HUI, N. M.; CHIENG, L. B.; TING, W. Y.; MOHAMED, H. H.; ARSHAD, M. R. H. M. Cross-platform mobile applications for android and ios. In: **6th Joint IFIP Wireless and Mobile Networking Conference (WMNC)**. [*S. l.*: *s. n.*], 2013. p. 1–4. ISBN 978-1-4673-5615-2.

KALAIYARASI, G.; BALAJI, K.; NARMADHA, T.; NAVEEN, V. E-voting system in smart phone using mobile application. In: **2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)**. [*S. l.*: *s. n.*], 2020. p. 1466–1469.

KIELT, E. D.; SILVA, S. d. C. R. d.; MIQUELIN, A. F. Implementação de um aplicativo para smartphones como sistema de votação em aulas de física com peer instruction. **Revista Brasileira de Ensino de Física**, scielo, [*S. l.*], v. 39, 00 2017. ISSN 1806-1117. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1806-11172017000400506&nrm=iso. Acesso em: 14 jun. 2021.

MALAVOLTA, I.; RUBERTO, S.; SORU, T.; TERRAGNI, V. Hybrid mobile apps in the google play store: An exploratory investigation. In: **2015 2nd ACM International Conference on Mobile Software Engineering and Systems**. [*S. l.*: *s. n.*], 2015. p. 56–59.

MYERS, G. J.; BADGETT, T.; THOMAS, T. M.; SANDLER, C. **The art of software testing**. [*S. l.*]: Wiley Online Library, 2004. v. 2.

NIELSEN, J. **Usability engineering**. [*S. l.*]: Morgan Kaufmann, 1994.

NIELSEN, J.; MACK, R. Heuristic evaluation. In: **Usability Inspection Methods**. [*S. l.*: *s. n.*], 1994. p. 22–62.

SILVA, I. B. **Desenvolvimento do sistema de avaliação da qualidade do alimento do RU da UFC em Quixadá**. 70 p. Monografia (Graduação em Engenharia de Software) – Universidade Federal do Ceará, Campus de Quixadá, Quixadá, 2019. Disponível em: http://www.repositorio.ufc.br/bitstream/riufc/49706/1/2019_tcc_ibsilva.pdf. Acesso em: 15 jan. 2021.

SOMMERVILLE, I. **Engenharia de software**. PEARSON BRASIL, 2011. ISBN 9788579361081. Disponível em: https://books.google.com.br/books?id=H4u5ygAACAAJ. Acesso em: 17 fev. 2021.

WANG, K.-H.; MONDAL, S. K.; CHAN, K.; XIE, X. A review of contemporary e-voting: Requirements, technology, systems and usability. **Data Science and Pattern Recognition**, [*S. l.*], v. 1, n. 1, p. 31–47, 2017.