UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO

FELIPE GUSMÃO ARAÚJO

DOCUMENTAÇÃO ÁGIL PARA DESENVOLVIMENTO DE UM APLICATIVO MÓVEL DE RESULTADOS MICROBIOLÓGICOS (MICAPP)

BELÉM
2020

FELIPE GUSMÃO ARAÚJO

DOCUMENTAÇÃO ÁGIL PARA DESENVOLVIMENTO DE UM APLICATIVO MÓVEL DE RESULTADOS MICROBIOLÓGICOS (MICAPP)

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação, do Campus Universitário de Belém, da Universidade Federal do Pará, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Dr(a). Danielle Costa C. Couto

BELÉM
2020

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

G982d Gusmão, Felipe.
DOCUMENTAÇÃO ÁGIL PARA DESENVOLVIMENTO DE UM APLICATIVO MÓVEL DE RESULTADOS MICROBIOLÓGICOS (MICAPP) / Felipe Gusmão. — 2020.
87 f. : il. color.

Orientador(a): Profª. Dra. Danielle Couto
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Pará, Instituto de Ciências Exatas e Naturais, Faculdade de Computação, Belém, 2020.

1. documentação de software. 2. metodologia ágil. 3. aplicativo móvel. I. Título.

CDD 004

FELIPE GUSMÃO ARAÚJO

DOCUMENTAÇÃO ÁGIL PARA DESENVOLVIMENTO DE UM APLICATIVO MÓVEL DE RESULTADOS MICROBIOLÓGICOS (MICAPP)

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação, do Campus Universitário de Belém, da Universidade Federal do Pará, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Data da aprovação: 30 / 11 / 2020

Conceito: EXCELENTE

BANCA EXAMINADORA

Drª Danielle Costa Carrara Couto
Campus Ananindeua \ UFPA

Drª Regiane Kawasaki
FACOMP – ICEN \ UFPA

Dr Vinícius de Abreu
FACOMP – ICEN \ UFPA

AGRADECIMENTOS

Gostaria de agradecer, primeiramente, a minha mãe, Maria de Jesus, e minha tia, Terezinha Gusmão, pelo suporte em todos os momentos dessa longa jornada. Foram quase cinco anos de muito estudo e empenho, os quais não seriam possíveis sem a ajuda de vocês.

Gostaria, também, de agradecer à minha orientadora, professora Dra. Danielle Couto, por todo suporte e paciência durante o desenvolvimento deste trabalho, e aos meus amigos que me ajudaram, seja me apoiando e auxiliando na formatação do TCC. Não citarei nomes, pois correria o risco de acabar esquecendo alguém, mas saibam que vocês fazem parte deste trabalho.

Por fim, gostaria de agradecer ao Gladson Carvalho, idealizador do MICAPP e aluno da MACPro, pela confiança no desenvolvimento deste trabalho e a Universidade Federal do Pará que, personificada nas figuras de professores e auxiliares administrativos, possibilitou a conquista desse título de bacharel.

A todos, o meu muito obrigado.

RESUMO

O desenvolvimento de software requer a documentação de cada etapa dentro do seu ciclo de vida. Apesar disso, poucos trabalhos científicos abordam o assunto, e algumas empresas de desenvolvimento tendem a usar seus próprios métodos de documentação. Por essa razão, este trabalho teve como objetivo principal encontrar as melhores práticas de documentação de software a partir de uma revisão de literatura, tendo como foco a metodologia ágil aplicada ao desenvolvimento móvel. O software documentado foi o Vigilante Microbiológico MICAPP, criado para auxiliar e acelerar o registro e a entrega de resultados de culturas de bactérias e antibiogramas para profissionais de saúde. Este aplicativo foi desenvolvido utilizando o *framework cross-platform* Ionic em conjunto com o SDK Firebase para operações de *backend*. A metodologia do trabalho foi exploratória qualitativa, buscando trabalhos científicos na área de documentação de software que seguissem a metodologia ágil. As ferramentas utilizadas para a documentação, bem como quais documentos gerar, foram definidos a partir da análise das boas práticas encontradas na revisão de literatura. Para documentação de requisitos e teste de aceitação de usuário foi usado o Google Docs; para o teste exploratório foi utilizado o Xmind e, por fim, o *Structurizr* foi utilizado na documentação de arquitetura. O teste de aceitação foi feito a fim de validar o aplicativo para entrega ao cliente, possibilitando a continuidade no processo de registro do software que já está em andamento pelo MACPro-UFPA. Este trabalho obteve sucesso na criação de uma documentação de software mais simples e direta para desenvolvimento ágil de um aplicativo móvel, além de ressaltar a relevância e importância do planejamento e documentação adequados a cada projeto de software.

Palavras-chave: documentação de software; metodologia ágil; aplicativo móvel.

ABSTRACT

The software development requires the documentation of each step inside its life cycle. Despite that, a few papers go through this subject, and some development companies tend to use their own documentation methods. For this reason, this work has as its main goal find the best practices for software documentation from a literature review, focusing on agile methodology applied to mobile development. The documented software was Vigilante Microbiológico MICAPP, created to auxiliate and speed up the processes of register and delivery of results of bacterial culture and antibiogram. This app was developed utilizing the cross-platform framework Ionic together with Firebase SDK for backend operations. This work methodology was exploratory qualitative, searching for scientific works in the area of software documentation that were based on agile methodology. The tools used for documenting, as well as which documents should be created, were defined based on an analysis of the good practices found on the literature review. For documenting the requirements and the user acceptance test it was used the Google Docs; for the exploratory test it was used the Xmind and, lastly, the Structurizr was utilized for documenting the architecture. The acceptance test was done to validate the app before delivering it to the client, making it possible to continue the process of software registration which is already in progress by MACPro-UFPA. This work has achieved success on the creation of a simple and direct software documentation, in addition to highlighting the relevance and importance of adequate planning and documentation for each project.

Key-word: software documentation; agile methodology; mobile app.

LISTA DE FIGURAS

LISTA DE QUADROS E TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MICAPP	Aplicativo Vigilante Microbiológico
TI	Tecnologia da informação
UFPA	Universidade Federal do Pará
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
RF	Requisitos funcionais
RNF	Requisitos não funcionais
UML	*Unified modeling language*
API	*Application programming interface*
UAT	*User acceptance test*
CRUD	*Create, read, update and delete*
MIC	*Minimum inhibitory concentration*
SDK	*Software Development Kit*

SUMÁRIO

1 INTRODUÇÃO

A utilização de aplicações, sobretudo móveis, para assistência à saúde tem se mostrado benéfica para diferentes profissionais da área. Estudos mostram que a utilização de tais ferramentas digitais diminui os riscos no tratamento de pacientes, agilizam o processo de diagnóstico e tratamento, e encurtam a distância física entre médico e paciente, e médico e centro de tratamento. A aplicação Vigilante Microbiológico (MICAPP), idealizada por Gladson Correa Carvalho, aluno do Programa de Pós-graduação em Análises Clínicas Profissional – MACPro na UFPA, busca trazer essas vantagens a área de exames laboratoriais, informatizando e agilizando os registros e resultados de antibiogramas e cultura de bactérias.

O desenvolvimento dessas aplicações de assistência à saúde, bem como de qualquer software, necessita da criação de uma série de documentos para registro de informações, seja do processo ou do sistema, que auxiliem no desenvolvimento e manutenção da aplicação. Estes documentos, muitas vezes, são negligenciados por desenvolvedores, por serem desorganizados e possuírem informações desnecessárias ou pouco relevantes, o que dificulta a localização e uso da documentação posteriormente.

Além disso, é difícil encontrar um consenso no que e como documentar. Muitas empresas utilizam seus próprios métodos e aplicações para gerar a documentação, sem realmente analisar quais documentos devem ser gerados e quais aplicações usar no processo. Como resultado, muitos documentos acabam inutilizados, comprometendo todas as etapas do ciclo de vida do software.

Tendo em vista os problemas citados, este trabalho visa reunir informações sobre documentação em metodologia ágil, a fim de identificar e aplicar as melhores práticas de documentação no aplicativo Vigilante Microbiológico. A documentação será feita a partir de ferramentas propostas, segundo uma pesquisa bibliográfica na área de documentação de software e metodologia ágil.

O objetivo geral deste trabalho é propor melhores práticas de documentação em metodologia ágil para o desenvolvimento de um aplicativo móvel, a fim de encontrar métodos e padrões para a escolha e criação de documentos objetivos e coerentes.

Objetivos específicos:

- propor ferramentas, para a documentação de software em metodologias ágeis, que sigam as boas práticas de mercado;

- aplicar estas ferramentas para criação da documentação do aplicativo Vigilante microbiológico (MICAPP), que foi desenvolvido sem a geração de documentação auxiliar;
- documentar e realizar testes de validação do MICAPP para auxiliar em posterior registro do software.

Este trabalho encontra-se subdividido em referencial teórico, metodologia, resultados e discussões e conclusão. O referencial teórico apresenta todos os trabalhos selecionados durante o levantamento bibliográfico, para servir de base na escolha das ferramentas e tipos de documentação. A metodologia detalha como foi feito o levantamento bibliográfico e mostra o fluxo de documentação do MICAPP, bem como as ferramentas escolhidas e a motivação para isso. Nos resultados e discussões são apresentados os documentos gerados do MICAPP a partir das ferramentas definidas na metodologia, e suas vantagens para compreensão do software como um todo. Por último, na conclusão, são discutidos todos os pontos positivos e negativos encontrados durante a realização deste trabalho, e, também, são apresentadas propostas para trabalhos futuros que complementam a pesquisa aqui apresentada.

2 REFERENCIAL TEÓRICO

2.1 Aplicativos Móveis para Assistência à Saúde

A tecnologia da informação (TI) proporciona uma maior agilidade e conectividade para o envio de informações de assistência à saúde. Tais características permitem que profissionais da saúde tomem decisões com mais rapidez, além de contribuir para um diagnóstico mais preciso e, de um modo geral, proporcionar uma maior eficácia na cura e vida do enfermo (BARRA et al., 2016) (MATSUDA et al., 2015).

Para Marques et al. (2017) existem riscos inerentes a atividades relacionadas à saúde. Por esse motivo, a busca por ferramentas digitais, em especial aplicativos móveis, que auxiliem e ajudem na realização de tais atividades, minimizando as chances de erro, é sempre encorajada aos profissionais desta área.

O distanciamento geográfico é outro aspecto afetado positivamente pelo uso da TI na área da saúde. A incorporação de aplicações móveis acessíveis por celular através de uma rede móvel, por exemplo, permite encurtar a distância entre o paciente e o profissional da saúde, assim como entre o profissional e o centro de tratamento (GUIMARÃES e GODOY, 2012).

Banos et al. (2015) e Peres e Marin (2012) evidenciam que existem muitos estudos que mostram que a implementação de tecnologias e aplicações móveis na área de saúde diminuem o risco a saúde dos pacientes, por gerarem e entregarem informações mais seguras, permitindo maior precisão nos resultados de exames e fornecendo melhor compreensão dos fatores que determinam o estado patológico do enfermo.

2.1.1 Desenvolvimento de Tecnologia Assistencial para informe rápido de resultados de microbiologia a Profissionais da Saúde (MICAPP)

Com base na necessidade e relevância da tecnologia da informação na área de saúde, evidenciadas no levantamento bibliográfico, Raimundo Gladson Corrêa Carvalho (2020), estudante de pós-graduação em análises clínicas, pela UFPA, iniciou o projeto de um aplicativo móvel que auxiliasse na análise e registro de antibiogramas, para uso laboratorial.

O desenvolvimento deste aplicativo, Vigilante Microbiológico (MICAPP), foi baseado em uma pesquisa metodológica que busca construir, validar e avaliar técnicas de pesquisa centradas no desenvolvimento de ferramentas tecnológicas que auxiliassem na assistência à saúde. (CARVALHO, 2020).

A pesquisa foi dividida em quatro etapas: Diagnóstico observacional; Revisão de literatura; Identificação dos temas geradores; Construção da tecnologia (CARVALHO, 2020).

O diagnóstico observacional foi feito sobre a rotina de comunicação dentro do laboratório Dr. Paulo Azevedo, situado no município de Belém. Nesta etapa, foram coletadas informações sobre a forma de repasse dos dados de cultura e antibiogramas gerados pelo laboratório, sob autorização da administração do mesmo. (CARVALHO, 2020).

O levantamento bibliográfico foi feito sobre artigos originais, revisões bibliográficas, sites e aplicativos móveis, voltados a monitoramento de infecções bacterianas, que estavam disponíveis na íntegra e publicados no período compreendido entre 2009 e 2019. A pesquisa foi feita nas bases de dados SciELO, Medline/PubMed e Portal de Periódicos da CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) e nos sites oficiais do Ministério da Saúde do Brasil. (CARVALHO, 2020).

Na terceira etapa, foram definidos os temas geradores para a criação do aplicativo: tempo de entrega de resultados de cultura; Tempo para o clínico intervir com tratamento no processo infeccioso; com base no diagnóstico observacional e levantamento bibliográfico.

Por fim, identificados os critérios técnicos e o público-alvo, foi iniciado o desenvolvimento da aplicação. Esta aplicação tem como objetivo ser uma ferramenta, WEB e móvel, de registro e controle de antibiogramas, facilitando a comunicação e agilizando a entrega de resultados aos pacientes. Foi desenvolvida pelo programador Amir Zahlan utilizando o *Framework Cross-platform* Ionic, versão 5.0.7, em integração com Firebase, versão 5.0.1, que fornece, através de APIs, serviços de autenticação do usuário, banco de dados e envio de notificações (CARVALHO, 2020).

A versão móvel é compatível com dispositivos que possuem sistema operacional Android, versão 8 ou superior. A versão WEB está disponível em <http://micapp.surge.sh/> e pode ser acessada por qualquer *browser* de internet padrão (CARVALHO, 2020).

2.2 Documentação de Software

Entende-se por documentação de software todo e qualquer artefato gerado durante o ciclo de vida do software. Para Garousi et al. (2015) a documentação é uma parte integral de qualquer processo de desenvolvimento de software. Seu objetivo é dar informações precisas sobre o sistema que está sendo desenvolvido a partir de diferentes pontos de abstração.

"Todos os documentos devem ajudar os engenheiros de software na compreensão de um dado sistema e a executar suas tarefas de forma mais efetiva e eficiente" (GAROUSI et al., 2015, p. 665).

Para Forward (2002), a documentação de software é mais do que simples descrições textuais do software; sua real importância está em estabelecer uma comunicação entre os membros de uma equipe.

O tipo e quantidade de documentação a ser desenvolvida em um projeto não é algo exato. Tal definição é feita analisando diversos fatores, como tamanho do projeto, tamanho da equipe, objetivo do produto, entre outras variáveis específicas de cada equipe.

Sommerville (2001) diz que é impossível prever um conjunto específico de documentação requerida para um dado projeto. O contrato com o cliente, o tipo do sistema sendo desenvolvido, a cultura e o tamanho do time de desenvolvimento, além da agenda de desenvolvimento, são aspectos que influenciam diretamente nessa decisão. Briand (2003) complementa esta ideia, dizendo que a decisão sobre quais documentos gerar é dependente do contexto, por tanto, não pode ser generalizada.

Na busca por referências bibliográficas que categorizam e diferenciam os tipos de documentação, poucos trabalhos foram encontrados. Arya (2019) também relata este problema em sua tese de mestrado, além de mostrar que autores como Sommerville (2001) e Garousi et al. (2015) não apresentam um consenso nesta categorização. Enquanto Sommerville (2001) subdivide os tipos de documentação em duas grandes classes: processo e produto, Garousi et al (2015) apresenta uma visão mais simples, subdividindo entre documentação técnica e manuais de usuário.

Para este trabalho, utilizamos a classificação dos tipos de documentação de software apresentada por Sommerville (2001), por possuir uma representação mais estruturada e relacionada com a função de cada documento. Nela, a documentação é subdividida em grupos, segundo sua origem e função dentro do ciclo de desenvolvimento do software. A seguir, é apresentada esta classificação, que se inicia na divisão em duas classes, segundo seu objetivo e uso:

- Documentação de processos: registra os processos de desenvolvimento e manutenção, como planos, agendas, documentos de qualidade de processo e organizacional e padrões de projeto.
- Documentação de produto: pode ser subdividido em documentação de sistema e documentação de usuário. Documentação de sistema descreve o produto do ponto de vista dos desenvolvedores (Ex: requisitos, arquitetura, testes); A documentação de usuário, analogamente, descreve o produto para fácil entendimento do cliente (Ex: manual de usuário).

2.2.1 Documentação de Processo

Segundo Sommerville (2001), este tipo de documentação é originado da necessidade de se visualizar todos os processos executados no desenvolvimento de um software. Devido a estes processos serem, principalmente, atividades cognitivas e não físicas, a única maneira de se obter visibilidade sobre tais atividades é através deste tipo de documentação.

A documentação de processo é subdividida e definida em cinco subclasses, como é mostrado a seguir (SOMMERVILLE, 2001):

- Planos, estimativas e agendas: São documentos utilizados no controle e predição dos processos de software. São utilizados pelos gerentes de projeto.
- Relatórios: Servem para o registro dos recursos disponíveis, bem como a maneira como estão sendo utilizados dentro do projeto.
- Padrões e normas: Definem como os processos devem ser implementados. Esses padrões podem seguir regras da organização, regras nacionais ou regras internacionais de normatização e padronização de software.
- Documentos de trabalho: São os principais documentos de comunicação técnica em um projeto. Neles ficam registradas informações de vários setores do processo de software: pensamentos dos engenheiros; versões temporárias de outras documentações do produto; descrição das estratégias de implementação; problemas identificados.
- Memorandos e e-mails: Aqui é registrada a comunicação diária entre os gerentes de projeto e os engenheiros de desenvolvimento.

A maioria dos documentos de processo é exclusiva de uma etapa específica de um processo. Como consequência, a maioria destes documentos se torna desatualizada e, em alguns casos, obsoleta. Mas, eles ainda devem ser mantidos, pois podem ser úteis na implementação de tarefas similares ou na manutenção do software (SOMMERVILLE, 2001).

2.2.2 Documentação do Produto

Sommerville (2001) diz que a documentação do produto tem como objetivo descrever o software. Comparada com a documentação de processo, que se torna desatualizada mais rapidamente, a documentação do produto tem uma vida útil maior. Essa documentação é composta por documentação de usuário e documentação de sistema, as quais dizem ao usuário como utilizar o software e descrevem o sistema e seus componentes, respectivamente.

2.2.2.1 Documentação de Sistema

A documentação de sistema fornece uma visão geral do software e ajuda engenheiros e *stakeholders* no entendimento da tecnologia empregada na sua produção. Normalmente, consiste em requisitos, arquitetura, código fonte, documentos de validação e documentos de manutenção (SOMMERVILLE, 2001). Vale lembrar que os documentos listados não são os únicos, sendo necessário a análise das características do projeto e da equipe para decidir pela adição ou remoção de itens a esta lista.

Abaixo são definidos cada um dos principais documentos de sistema (SOMMERVILLE, 2001):

- Requisitos: Nele se tem as informações sobre a funcionalidade do sistema. Geralmente, são estruturados em forma de sentenças contendo o ator, a ação e o objetivo. Além disso, podem ser classificados como funcionais (o que o sistema fará) e não-funcional (como o sistema fará).
- Arquitetura: Aqui é registrada toda a arquitetura do produto na forma de diagramas e em formato textual.
- Código fonte: O código fonte deve possuir comentários para explicar partes complexas do código. Quanto maior o uso de nomes significativos para variáveis e funções, de estilos estruturados de programação, entre outras boas práticas, menor a necessidade da adição de comentários ao código.
- Teste e validação: São atividades complementares e servem para garantir que a funcionalidade do software esteja implementada corretamente e conforme os requisitos.

2.2.2.2 Documentação de Usuário

A documentação de usuário, como o nome sugere, é voltada para os usuários do produto. Ela deve descrever as formas de uso e funcionalidades do produto final. No entanto, existe uma distinção nos tipos de usuário do software. Segundo Sommerville (2001), o criador da documentação de usuário deve estruturá-la fazendo distinção entre usuário final e administrador de sistema.

A documentação para o usuário final deve ser simples e precisa, evitando termos técnicos. Este tipo de usuário quer saber como o software pode ajudá-lo. Ele não tem interesse em detalhes técnicos ou administrativos (SOMMERVILLE, 2001).

Os administradores de sistema, por outro lado, são responsáveis pela gerência do software usado pelos usuários finais. Entre as atividades do administrador podem ser citadas: operar o sistema em um grande sistema de mainframe, atuar como gerente de rede, resolver problemas dos usuários finais (SOMMERVILLE, 2001).

2.3 Desenvolvimento Ágil

No desenvolvimento de software, a metodologia ágil é uma área que estuda o conjunto de comportamentos, processos, práticas e ferramentas para a criação e produção do software de forma célere e precisa, sem a perda da qualidade do mesmo.

Para o desenvolvimento e melhoria de tais metodologias, alguns valores chave foram estipulados por Fowler et al. (2001) no manifesto ágil. Este manifesto foi criado por 17 desenvolvedores de software e busca agrupar metodologias para o desenvolvimento mais rápido de um software. Abaixo estão os quatro valores deste manifesto:

- Indivíduos e interações são mais importantes que processos e ferramentas.
- Software em funcionamento é mais importante que documentação abrangente.
- Colaboração do cliente é mais importante que negociação de contrato.
- Reagir a mudanças é mais importante que seguir um plano.

Hazzan e Dubinsky (2014), a partir dos valores ágeis supracitados, reconhecem a importância dos processos e ferramentas, mas percebem que a interação de indivíduos é de maior importância. De maneira similar, uma documentação abrangente não necessariamente é ruim, mas o objetivo principal deve ser sempre o produto final em funcionamento.

Os mesmos autores também citam que os contratos de negócio podem providenciar condições limite para o trabalho dos membros da equipe, mas apenas a colaboração contínua com o cliente pode levar ao entendimento e entrega do que ele quer (HAZZAN e DUBINSKY, 2014).

Além disso, apesar do planejamento prévio de um projeto ser necessário para fornecer uma base inicial do que executar, é necessário ter cuidado para que o planejamento não o torne resistente a mudanças. O histórico de vários projetos bem sucedidos mostra que poucos, se algum, foi entregue seguindo o que foi planejado desde o início (FOWLER e HIGHSMITH, 2001).

Em resumo, segundo Fowler e Highsmith (2001), o movimento da metodologia ágil não é anti-metodologia. Ele dá suporte a modelagem e documentação, mas não para que estas sejam armazenadas e largadas sem propósito e uso durante o desenvolvimento de um

software. Assim como, para eles, o planejamento é, também, de suma importância, mas possui limitações dentro de um ambiente turbulento de criação.

Desde a sua criação, o manifesto ágil é tido como a base de um novo paradigma de pensamento sobre todo o ciclo de vida no desenvolvimento de software. Da elicitação de requisitos a manutenção, cada etapa tem sido estudada e reformulada por equipes de desenvolvimento ágil, sempre orientados pelos quatro valores do manifesto (WILLIAMS, 2012).

2.4 Trabalhos Correlatos

Para tentar entender melhor como otimizar a quantidade de documentação dentro de um projeto de software segundo os valores ágeis, Garousi et al. (2015), Voigt et al. (2016), Satish e Anand (2016), Tripathi e Goyal (2014) e Hadar et al. (2013) realizaram uma revisão de literatura sobre o assunto.

2.4.1 Usage and usefulness of technical software documentation: An industrial case study.

Garousi et al. (2015) realizaram um estudo de caso em uma empresa de desenvolvimento de software para analisar o uso e a utilidade da documentação de software. Foram feitas análises quantitativas e qualitativas de 55 documentos e de suas 1630 revisões e, ao final, foi realizada uma pesquisa com 25 membros da empresa sobre o assunto.

O estudo de caso mostrou que o nível de atualização e a precisão dos documentos tem impacto direto na sua utilidade. Documentos mais precisos e concisos facilitam a localização da informação, e que estar atualizado com a atual situação de desenvolvimento também é importante.

Por fim, os autores mostram que documentos extensos foram citados como um dos pontos mais negativos entre membros da equipe, diminuindo o uso e a utilidade da documentação e, por tanto, devem ser evitados.

2.4.2 A Study of Documentation in Agile Software Projects.

Além da revisão de literatura, Voigt et al. (2016) realizaram entrevistas com funcionários de duas empresas de desenvolvimento de software. Em sequência, foi feita observação participativa com análise das funções de trabalho em três empresas. E, por último, uma pesquisa online, da qual obtiveram 104 respostas completas.

A pesquisa desenvolvida foi baseada em um modelo de comportamento da informação, desenvolvido por Choo (1998). Este modelo subdivide a informação em necessidade por informação, busca por informação e uso de informação. Essas subdivisões são influenciadas por necessidade cognitiva e resposta afetiva, as quais não foram levadas em consideração devido a limitação no tamanho da publicação.

Como resultado da revisão, os autores afirmam que a literatura sobre documentação ágil é muito focada em aspectos específicos, sem propor uma solução geral. Além disso, existe uma boa base empírica nesta área, mas poucas delas foram publicadas.

Eles também concluíram, com base na análise das entrevistas, observações e questionários, que grandes volumes de documentação não necessariamente aumentam o nível de satisfação com a informação disponível em uma equipe de desenvolvimento. Assim como, a documentação de arquitetura precisa de atenção especial pois se mostrou uma área fundamental para o entendimento geral do projeto.

2.4.3 A Document Driven Approach for Agile Software Development.

Tripathi e Goyal (2014) definiram a identificação dos fatores de seleção de documentação e o mínimo necessário de documentos como objetivo da pesquisa.

Após uma extensa revisão bibliográfica, os autores listaram os fatores mais recorrentes e relevantes para a definição da documentação de um projeto, encontrados na revisão:

- Tamanho e tipo de produto: quanto maior a complexidade do produto maior a quantidade de documentação.
- Ambiente de implantação e desenvolvimento: Equipes separadas geograficamente necessitam de mais documentação para se manterem sincronizadas e atualizadas. Além disso, ambientes de implantação com alta escalabilidade e disponibilidade precisam de documentação mais específica para manutenção e controle.
- Experiência da equipe: Para equipes menos experientes é recomendado mais documentação, a fim de garantir um maior entendimento de atividades que são a mais facilmente compreendidas por equipes mais experientes.
- Requisitos do cliente: O cliente pode definir quais documentos de entrega são realmente necessários.
- Tipos e níveis de usuários: Os tipos de usuário podem requerer uma maior quantidade de documentação específica para estes.

2.4.4 Less is More: Architecture Documentation for Agile Development.

Hadar et al. (2013) fizeram uma pesquisa em uma empresa internacional de desenvolvimento de software com o objetivo de descobrir os principais problemas na documentação de arquitetura. Eles realizaram questionários com membros sêniores da equipe, analisaram documentos e observaram discussões e desafios da equipe de arquitetos.

Como resultado, os autores identificaram quatro fatores para a criação de documentação de arquitetura: entendimento, localização de informações relevantes e atualização. Para garantir entendimento e fácil localização de informação é preciso gerar um documento curto, focado e estruturado. Além do mais, tais documentos devem ser criados de forma a facilitar o seu processo de atualização.

2.4.5 Software Documentation Management Issues and Practices: A Survey.

Satish e Anand (2016) extraíram os principais problemas relacionados a documentação encontrados na literatura. Cada problema encontrado foi relacionado com uma prática proposta pelos autores para resolvê-los.

A partir dessa revisão bibliográfica, os autores chegaram à conclusão de que o tamanho do projeto, a distribuição geográfica e a experiência dos envolvidos são fatores determinantes para a definição do conteúdo da documentação. Para eles, projetos com pequenas equipes e sem separação geográfica podem ser realizados com o mínimo de documentação. Também citaram que, o uso de ferramentas na geração da documentação pode facilitar a navegação e busca por estes.

2.4.6 Comparação entre Trabalhos Correlatos

Decidir a quantidade de documentação necessária em um projeto de software ainda é um grande desafio enfrentado pelos seus desenvolvedores. Mesmo com anos de prática no desenvolvimento de software, a resposta para esse questionamento ainda parece difícil (GAROUSI et al., 2015).

Em metodologias ágeis, a máxima é “Software funcionando é mais importante que documentação abrangente” (FOWLER e HIGHSMITH, 2001, p. 2). Segundo Voigt et al. (2016), muitas vezes essa sentença é mal interpretada. Alguns agilistas chegam até a considerar a documentação como perda de tempo, achando que não contribui para o produto final.

Tripathi e Goyal (2014) dizem que a documentação é uma das práticas mais antigas no desenvolvimento de software e ainda mantém o seu papel fundamental. Além disso, a sua falta pode criar problemas a longo prazo em equipes ágeis.

Por outro lado, Tripathi e Goyal (2014), ressaltam que para projetos grandes e complexos a documentação tem papel ainda maior. Nestes casos, o uso de metodologias tradicionais acaba gerando muitos artefatos, fazendo com que os membros da equipe fiquem facilmente perdidos ao procurar por informação.

Hadar et al. (2013) complementam esta ideia dizendo que a documentação precisa dar suporte ao software e não o contrário. É preciso tirar toda informação desnecessária e focar em: Entendimento, localização de informações e confiança através de documentos atualizados. Portanto, encontrar um equilíbrio na quantidade de documentação sem interferir na compreensão e agilidade do projeto é essencial.

Neste contexto, este trabalho tenta extrair as melhores práticas encontradas na pesquisa bibliográfica e aplicar na documentação de um software. Todas as ferramentas usadas e documentos gerados foram definidos com base nas ideias defendidas nos artigos supracitados (Quadro 1).

Quadro 1 – Resumo dos Trabalhos Correlatos

Autores e Ano da Publicação	Título	Metodologia	Objetivos	Resultados
Garousi et al. (2015)	*Usage and usefulness of technical software documentation: An industrial case study*	Revisão de literatura e análise empírica qualitativa e quantitativa sobre um estudo de caso.	Identificar os principais fatores que afetam o uso e a utilidade dos documentos de software.	Objetividade e estado de atualização de um documento são os pontos que mais afetam o seu uso e utilidade.
Voigt et al. (2016)	*A Study of Documentation in Agile Software Projects.*	Revisão de literatura e metodologia de triangulação: entrevistas parcialmente estruturadas, observação e questionário online.	Relacionar o grau de satisfação pelo uso de um documento com o volume de informação presente nele.	Grandes volumes de documentação não aumentam o nível de satisfação durante o seu uso.
Tripathi e Goyal (2014)	*A Document Driven Approach for Agile Software Development.*	Revisão de literatura.	Identificar fatores importantes para escolha da documentação de um software.	Tamanho e tipo de produto; ambiente de implantação e desenvolvimento; Experiência da equipe; Requisitos do cliente; e Tipos e níveis de usuários são os principais

				fatores a levar em consideração para escolha da documentação.
Hadar et al. (2013)	*Less is More: Archtecture Documentation for Agile Development.*	Revisão de literatura e análise empírica quantitativa e qualitativa sobre um estudo de caso.	Identificar os principais fatores para a geração de documentos de arquitetura de software.	Entendimento, localização de informações relevantes e atualização são os fatores mais importantes para a geração da documentação de arquitetura de software.
Satish e Anand (2016)	*Software Documentation Management Issues and Pratices: A Survey.*	Revisão de literatura.	Resumir os problemas encontrados durante a documentação e propor soluções para estes.	O tamanho do projeto, a distribuição geográfica e a experiência dos envolvidos são fatores muito importantes na definição da documentação. O uso de ferramentas que facilitem a integração e a comunicação são propostas para minimizar esses problemas.

Fonte: Próprio autor.

3 METODOLOGIA

Neste trabalho foi aplicada uma metodologia exploratória qualitativa. Toda documentação foi feita com base em teste exploratório do aplicativo (app), análise de código fonte e teste de aceitação de usuário, onde o aprendizado sobre o software caminhou em paralelo com a documentação.

O desenvolvimento do trabalho foi feito sobre o app **Vigilante Microbiológico (MICAPP)**, idealizado por Gladson Corrêa, estudante da Pós-graduação em Análises Clínicas pela UFPA. Este aplicativo foi desenvolvido entre abril e setembro de 2019, e tem por objetivo ser uma plataforma móvel e web de registro e controle dos resultados de cultura e antibiograma de pacientes com infecções bacterianas, possuindo um sistema de notificações para alterações e criação de novos registros.

O aplicativo foi desenvolvido pelo programador Amir Zahlan, utilizando o *framework cross-platform* Ionic. Durante o desenvolvimento do app não foram feitas as documentações de software, portanto, todas as ferramentas, APIs e *frameworks* utilizadas neste processo foram descritas com base em testes exploratórios do software e do código fonte, em conjunção com o estudo das suas documentações.

Através do portal de periódicos da CAPES, GOOGLE Acadêmico e IEEE, foi realizada a busca, em 17 de abril de 2020, de documentos com fundamentação técnico-científica que permitissem identificar as metodologias e práticas mais atuais na área de documentação de software.

Como base de pesquisa para a documentação (Tabela 1), foi definida a metodologia ágil, visando descobrir as melhores práticas para escolha e geração de artefatos, sem a criação de documentação excessiva ou a perda da objetividade destes.

Tabela 1 - Número de resultados em cada site pôr termo de pesquisa

	Agile Methodology	***Soft. Documentation***	***Agile Documentation***	**TOTAL**
CAPES	957	3.516	28	4.501
Google Acadêmico	16.000	28.500	616	45.116
IEEE	292	477	6	775

Fonte: Próprio autor.

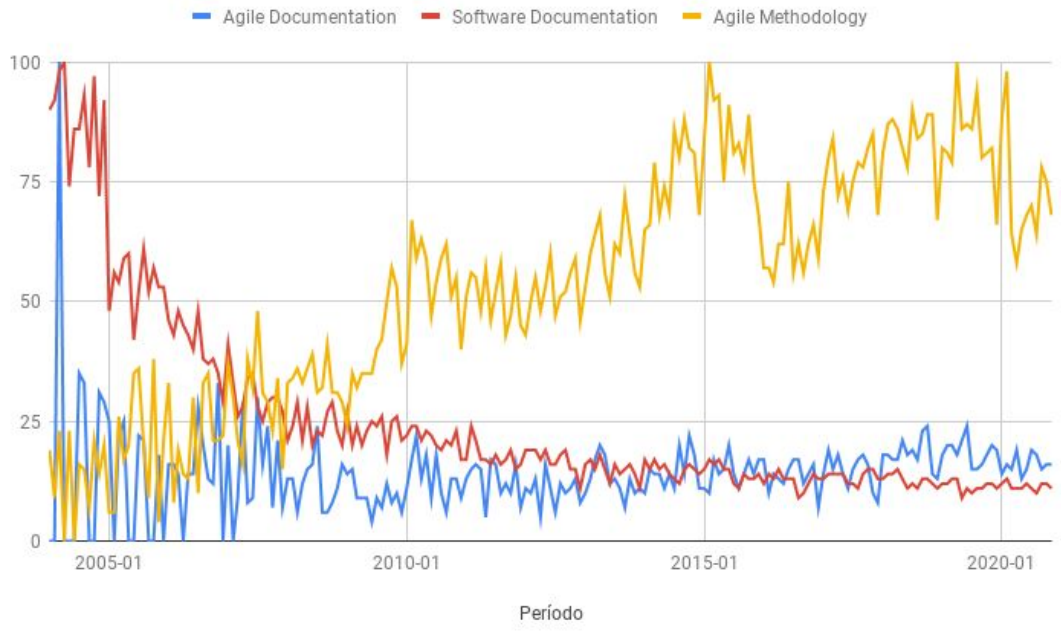
Os termos utilizados para pesquisa foram *"agile methodology"*, *"software documentation" e "agile documentation"*. O uso de termos em inglês foi proposital, visando

aumentar o número de resultados e enriquecer o levantamento com artigos, livros e trabalhos internacionais.

O número de publicações específicas para documentação ágil é bem inferior às publicações referentes a metodologia ágil e documentação de software. Além disso, muitos materiais encontrados têm mais de 8 anos de publicação. Desta forma, para manter a relevância do trabalho, os resultados foram selecionados com base na data de publicação e similaridade do assunto. Estudos com mais de 9 anos foram evitados, dando preferência sempre aos estudos mais atuais.

Foi realizada também uma pesquisa bibliométrica (Figura 1) dos termos utilizados na pesquisa bibliográfica. O objetivo foi verificar o número de buscas por esses termos ao longo dos anos, no período de 2004 até 2020. Como resultado, foi observado que a busca por "*agile methodology*" teve um aumento considerável entre 2006 e 2020, enquanto "*software documentation*" teve uma queda a partir de 2004 e "*agile documentation*" sempre manteve a quantidade de busca inferior aos demais. A partir disso, pudemos verificar que a metodologia ágil teve uma relação direta na diminuição da importância dada à documentação de software.

Figura 1 – Pesquisa Bibliométrica dos termos utilizados nos últimos 16 anos.



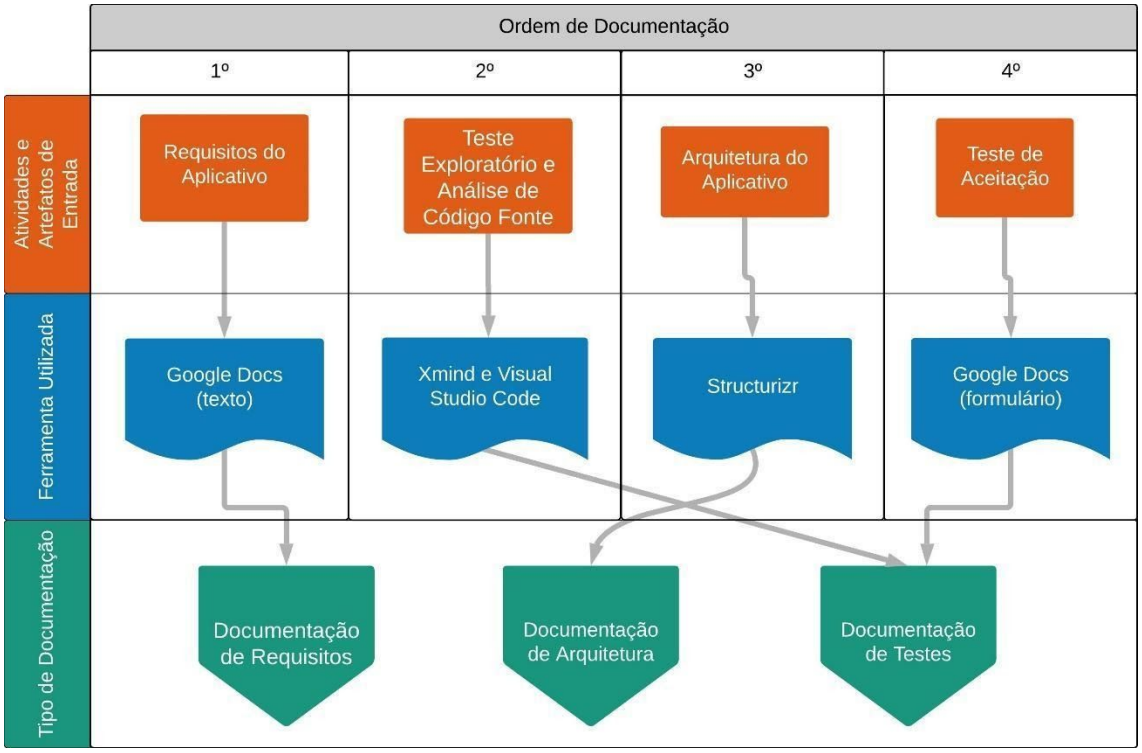
Fonte: Próprio autor.

A partir das pesquisas bibliográficas e bibliométricas, decidimos propor uma série de ferramentas para a realização da documentação do app Vigilante Microbiológico, seguindo as melhores práticas citadas pelos autores pesquisados, como: evitar documentação excessiva,

utilização de ferramentas que facilitem a documentação e facilidade de localização e atualização de informação.

Além disso, a seleção de quais documentos gerar foi baseada no tamanho da equipe de desenvolvimento e complexidade da aplicação, citados por Tripathi e Goyal (2014) e Satish e Anand (2016) como pontos fundamentais na tomada dessa decisão; além dos valores da metodologia ágil.

Figura 2 – Fluxo de documentação do aplicativo



Fonte: Próprio autor.

Acima (Figura 2) está apresentado o fluxo de documentação seguido no desenvolvimento deste trabalho.

Inicialmente, foi definido que não seria gerada documentação de processo, apenas de produto. Como a documentação está sendo feita após o desenvolvimento do aplicativo, não fui julgado necessário a criação ou registro de qualquer atividade administrativa do desenvolvimento, característica da documentação de processo mostrada por Sommerville (2001).

A seguir, foi feita a documentação de requisitos do aplicativo. Os requisitos são essenciais em qualquer desenvolvimento de software, sendo citados em todos os trabalhos correlatos como parte fundamental de qualquer documentação.

Esta etapa foi realizada a partir de conversas com o idealizador do app, Gladson Carvalho. Todos os requisitos, funcionais e não funcionais, foram registrados em um

documento de texto no Google Docs, onde é possível fazer modificações e compartilhamentos a partir de qualquer dispositivo, facilitando a edição, atualização e manutenção do documento.

O passo seguinte foi analisar o código fonte do programa, utilizando o editor de código fonte *Visual Studio Code*. Aqui, o objetivo foi entender melhor, a nível de código, o funcionamento do software e de suas funcionalidades, além de analisar o seu nível de complexidade. Além disso, um teste exploratório foi realizado paralelamente, buscando entender o funcionamento, a nível de usuário, do app. Este teste foi feito com auxílio da ferramenta Xmind, criado para organização de ideias, pensamentos e tarefas, em formato de mapas mentais.

Os testes exploratórios, em conjunto com a análise do código fonte e dos requisitos, forneceram as informações necessárias sobre complexidade e tamanho da aplicação. De posse disso, foi definido que também seriam documentados a arquitetura e os testes do aplicativo. Esta decisão foi tomada com base nas pesquisas feitas por Tripathi e Goyal (2014), Satish e Anand (2016) e Voigt et al. (2016), após se verificar que a aplicação possui uma complexidade entre média e baixa, e o tamanho da equipe e do aplicativo são pequenos, não sendo necessário documentar nada além de requisitos, arquitetura e testes.

É importante citar que, devido à falta de uma definição mais objetiva, nos trabalhos correlatos, de como classificar a complexidade e o tamanho da aplicação, isso foi feito com base no número de funcionalidades do app, bem como o número de APIs e tamanho do banco de dados.

A documentação da arquitetura do aplicativo foi gerada na sequência, utilizando a ferramenta Structurizr. Esta ferramenta foi escolhida seguindo a análise de Hadar et al. (2013), que diz ser necessário criar um documento curto, focado e estruturado a fim de facilitar a localização e atualização de informações. Por possuir uma estrutura interativa e de fácil atualização, subdividida em apenas quatros níveis de abstração, o Structurizr se mostrou eficaz nesta etapa.

A última documentação gerada foi a de teste de aceitação do usuário. Este teste foi realizado para avaliar se os requisitos do sistema estão sendo implementados ao software, assim como se o software já se encontrava pronto para entrega. A escolha pelo teste de aceitação ocorreu devido a esta documentação prover transparência, tanto para aos desenvolvedores quanto aos clientes, sobre o estágio de desenvolvimento da aplicação, além de fornecer informações importantes para a manutenção da mesma. Tais características citadas anteriormente são essenciais para a documentação de qualquer software, segundo Tripathi e Goyal (2014).

Esta etapa foi realizada com treze estudantes e profissionais da área de farmácia, entre 25 de setembro de 2020 e 2 de outubro de 2020, que assinaram um termo de consentimento concordando em participar do teste, como mostrado no Anexo A. Este teste foi baseado no livro de Hambling e Goethem (2013) e adaptado para garantir o máximo de agilidade e o mínimo de documentação.

O *link* do site, onde está hospedado o aplicativo web, foi enviado aos participantes, por e-mail, que deveriam realizar uma série de testes definidos em um documento de registro. Os testes definidos neste documento foram criados com base nos requisitos do sistema, e, ao fim de cada um, o resultado era relatado em um formulário criado no Google Docs, cujo *link* também foi enviado no e-mail, para posterior análise.

3.1 Ferramentas

Abaixo são descritas todas as ferramentas utilizadas durante o processo de documentação do aplicativo Vigilante Microbiológico.

3.1.1 Google Docs

O Google Docs é um pacote de aplicativos do Google dedicado a criação e edição de textos, planilhas, apresentações e formulários. Estes aplicativos funcionam de forma síncrona e assíncrona, portanto, podem ser acessados online, através de um *browser,* ou offline através de aplicativos de extensão instalados diretamente do Google. Outra característica é a possibilidade de uso compartilhado de um mesmo documento, ou seja, mais de uma pessoa pode editar o mesmo documento online.

O pacote do Google está disponível gratuitamente, sendo necessário pagamento apenas se o usuário quiser aumentar o tamanho do sistema de armazenamento em nuvem, Google Drive, onde são salvos os documentos do Google Docs.

Todas as informações sobre o Google Docs estão disponíveis em: <https://docs.google.com>.

3.1.2 Visual Studio Code, versão 1.48.2

O Visual Studio Code é um editor de código fonte, desenvolvido pela Microsoft. Ele é compatível com Windows, Mac e Linux, e possui suporte nativo das linguagens *JavaScript*, *TypeScript* e *Node,js*. Além do suporte nativo, é possível acessar um largo ecossistema de extensões para outras linguagens, como C, C++, Python, entre outras.

É possível acessar a documentação e fazer o download do editor através do site, disponível em: <https://code.visualstudio.com>.

3.1.3 Xmind 8 Pro

Xmind é uma ferramenta de mapeamento de ideias, criada pela empresa Xmind Ltd. Ela oferece uma série de *templates,* onde é possível registrar e estruturar ideias, pensamentos e atividades segundo uma lógica definida pelo usuário.

Este é um software pago, com diferentes planos de inscrição, inclusive o gratuito, usado neste trabalho.

Todas as informações referentes ao Xmind podem ser encontradas no site, disponível em: <https://www.xmind.net/>.

3.1.4 Struciturizr, versão 2120

O Structurizr é um conjunto de ferramentas para criação de diagramas arquiteturais de software, baseado no modelo C4. Seu criador, Simon Brown, foi o idealizador do modelo de visualização de software, C4, cujo princípio é gerar uma apresentação mais simples, interativa e compreensiva da arquitetura de software.

O acesso a ferramenta e a documentação sobre o modelo C4 estão disponíveis em: <https://c4model.com>.

4 RESULTADOS E DISCUSSÕES

Nos resultados apresentamos toda a documentação gerada para o aplicativo Vigilante Microbiológico. Esta documentação, bem como as ferramentas utilizadas, seguem o fluxo mostrado na figura 1, o qual foi planejado seguindo as boas práticas encontradas na pesquisa bibliográfica.

4.1 Requisitos

O primeiro documento gerado foi o de requisitos. Este tipo de documento serve para definir e registrar o que deve ser implementado no software a nível de requisitos funcionais e não-funcionais. Por requisitos funcionais, entendem-se todas as funcionalidades do software; são as ações que podem ser realizadas dentro de um programa. As formas como essas funcionalidades serão implementadas, assim como restrições de tempo, armazenamento, segurança, portabilidade, entre outras, são chamadas de requisitos não-funcionais.

Neste trabalho, mesmo com a documentação sendo feita após o desenvolvimento do software, os requisitos foram obtidos para uso na manutenção e para compreensão do aplicativo. A partir das informações registradas neste primeiro documento, foi possível compreender as funcionalidades do Vigilante Microbiológico, assim como permitiu a criação e realização de testes. Ademais, de posse dos requisitos, será mais fácil modificar e adicionar funcionalidades ao aplicativo no futuro, além de ser necessário para o registro do software, pela UFPA, que corre sob o número de processo: 23073.016756/2019-31.

A criação deste documento foi feita a partir de reuniões e troca de e-mails com o idealizador do aplicativo, Gladson Corrêa (Programa de Pós-graduação em Análises Clínicas Profissional - UFPA). As reuniões tiveram início no dia 1 de abril de 2020, quando realizamos uma reunião presencial, mas que, com o avanço da pandemia de COVID-19, passaram a ser realizadas online através de chamadas de voz e aplicativos de mensagens.

Ao todo, foram feitas 3 reuniões, com duração de 30 minutos cada. A primeira foi realizada em 3 de abril de 2020, a segunda em 17 de abril de 2020 e a última foi realizada dia 1 de maio de 2020, quando o documento de requisitos foi aprovado, por Gladson Corrêa.

Os requisitos foram estruturados em um documento de texto, editado no Google Docs, contendo: Visão Geral, Requisitos Funcionais e Requisitos Não-Funcionais. Na Visão Geral, é mostrado um resumo do objetivo e motivação para a criação do software. Em Requisitos Funcionais, cada funcionalidade recebeu um código único iniciado por RF (requisito funcional), seguido de uma numeração sequencial para facilitar a identificação. Ao lado do

código está o nome da funcionalidade, seguido, logo abaixo, pela descrição da mesma. De forma análoga, Requisitos Não-Funcionais possui código, nome e descrição, mudando apenas o início do código, que é iniciado por RNF (requisitos não-funcionais). O documento completo é mostrado no Apêndice A.

Optou-se por não adicionar nenhum diagrama aos requisitos, seja de sequência ou casos de uso, visando manter o documento simples e direto, deixando essas informações para a arquitetura.

4.2 Teste exploratório

O teste exploratório foi feito em 20 de junho de 2020, durando aproximadamente 4 horas, dando sequência a documentação do aplicativo. Este teste consiste em aprendizado e teste, paralelamente. Isto quer dizer que, à medida que você usa e descobre as funcionalidades do programa, você cria e executa casos de teste para avaliar se estão funcionando conforme o esperado.

Apesar de já possuir os requisitos do aplicativo, o meu entendimento do seu funcionamento na prática ainda era limitado. Junto a isso, existia a necessidade de testá-lo para descobrir possíveis *bugs* que tivessem passado despercebidos pelo desenvolvedor. Portanto, visando facilitar a realização destas duas tarefas, o teste exploratório foi escolhido, sendo realizado em conjunto com a análise do código fonte do aplicativo.

A etapa seguinte, foi a escolha de uma ferramenta para o registro do teste. Um documento de texto seria o suficiente para isso, porém poderia comprometer a busca por informações quando necessário. Por isso, foi escolhida uma ferramenta que permitisse a montagem de um mapa mental.

O mapa mental é uma técnica de organização de ideias que consiste em criar resumos com símbolos, cores, setas e frases, objetivando organizar e facilitar a associação de informações. Por ser um teste em que os casos de teste são criados no momento da testagem, o mapa mental fornece as estruturas perfeitas para a organização de tais casos.

Dentre as ferramentas pesquisadas, foi escolhido o Xmind 8 Pro, da Xmind Ltd., por possuir uma interface simples e intuitiva.

Na figura 3, vemos o resultado final do teste exploratório representado no Xmind 8 Pro. Nesta tela, estão informadas a data de realização do teste, a legenda e as áreas do teste. Data e legenda servem para orientar na busca e localização de informação, de forma simples e direta, evitando verbosidade e expressões muito longas. Já as áreas de teste, são agrupamentos de funcionalidades relacionadas, definidos em *Brainstorm*, baseadas no primeiro contato com

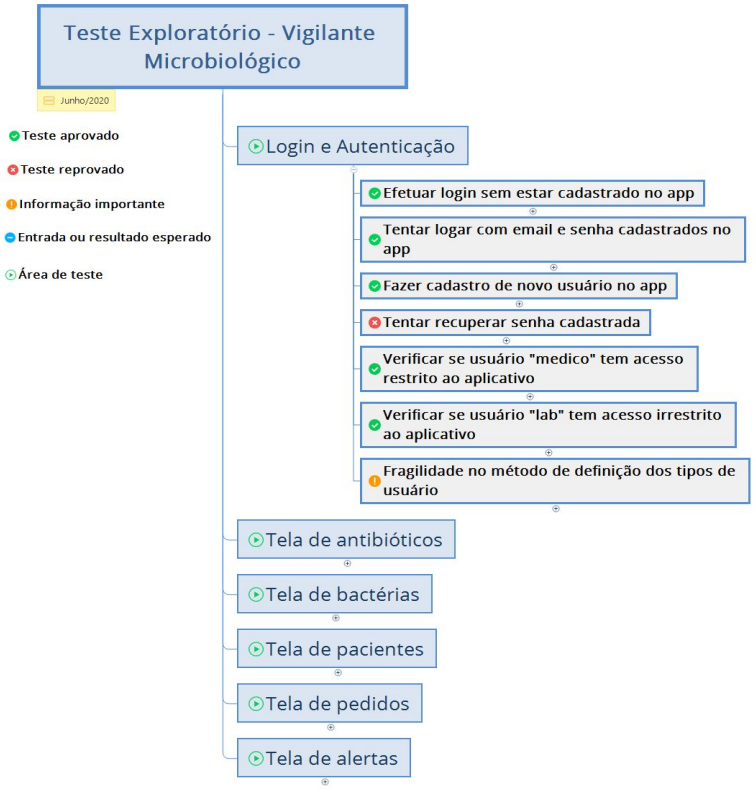
o aplicativo e nos requisitos. É a partir dessas áreas que os casos de teste são desenvolvidos e aplicados conforme o uso do software.

Figura 3 – Teste exploratório

Fonte: Próprio autor.

As principais telas do aplicativo Vigilante Microbiológico estão presentes no Anexo B. Dentro das áreas de teste estão registrados os casos de teste aplicados ao Vigilante Microbiológico (figura 4). Cada um deles é registrado na forma de frase, em linguagem natural, que resume o que está sendo testado.

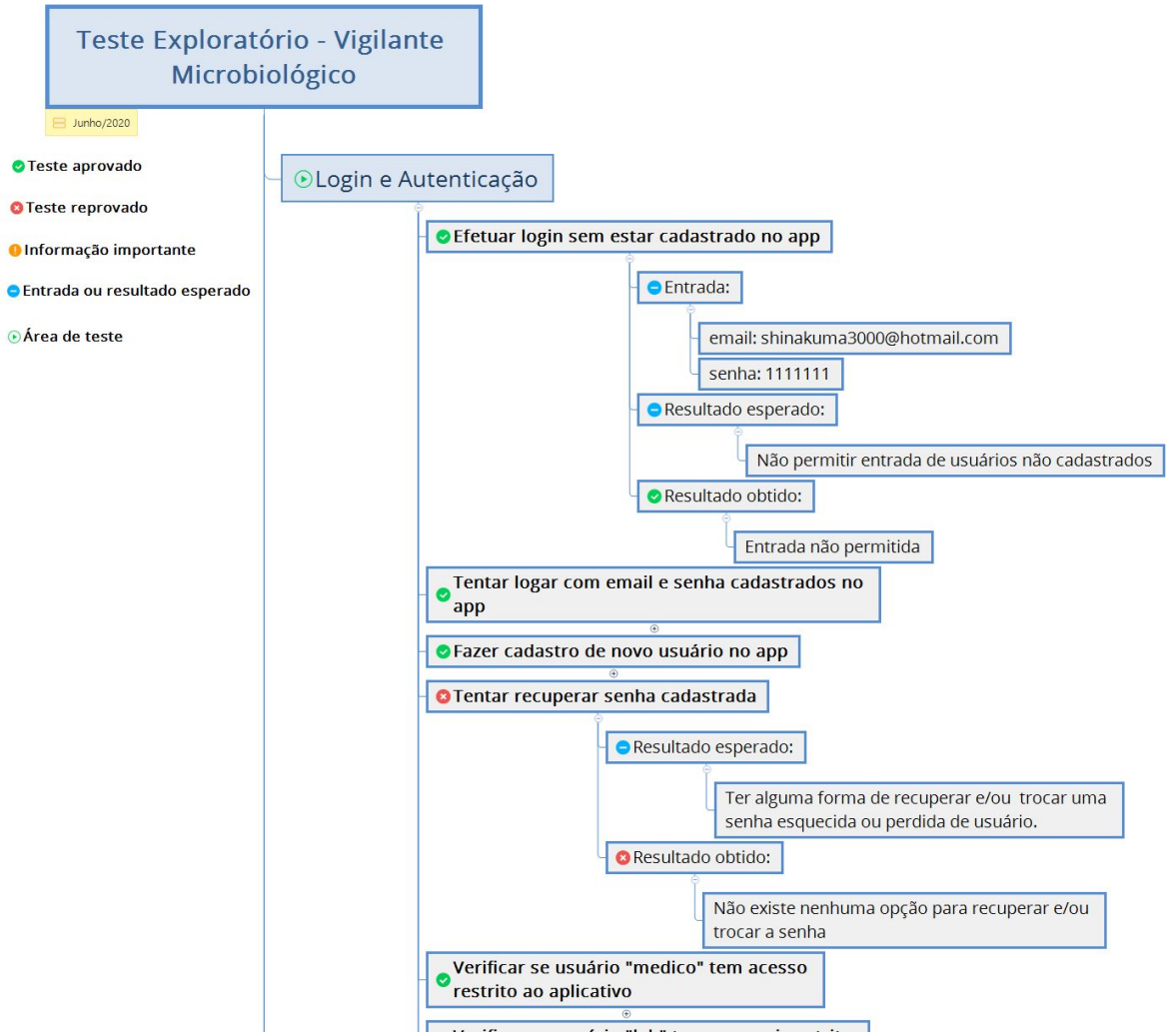
Figura 4 – Teste exploratório do login e autenticação



Fonte: Próprio autor.

Dentro de cada caso de teste, são registrados a entrada, resultado esperado e o resultado obtido (figura 5). A entrada representa os dados que tenham sido inseridos no software ou botões que tenham sido acionados, em um determinado caso de teste. Essa entrada deve gerar um resultado considerado correto para aquela ação, o que é definido em resultado esperado. E, por fim, o resultado real da ação de entrada é registrado em resultado obtido, sendo comparado com o resultado esperado. Caso o obtido seja diferente do esperado, o caso de teste é reprovado e devidamente marcado. Caso contrário, ele é aprovado.

Figura 5 – Entradas, resultado esperado e resultado obtido

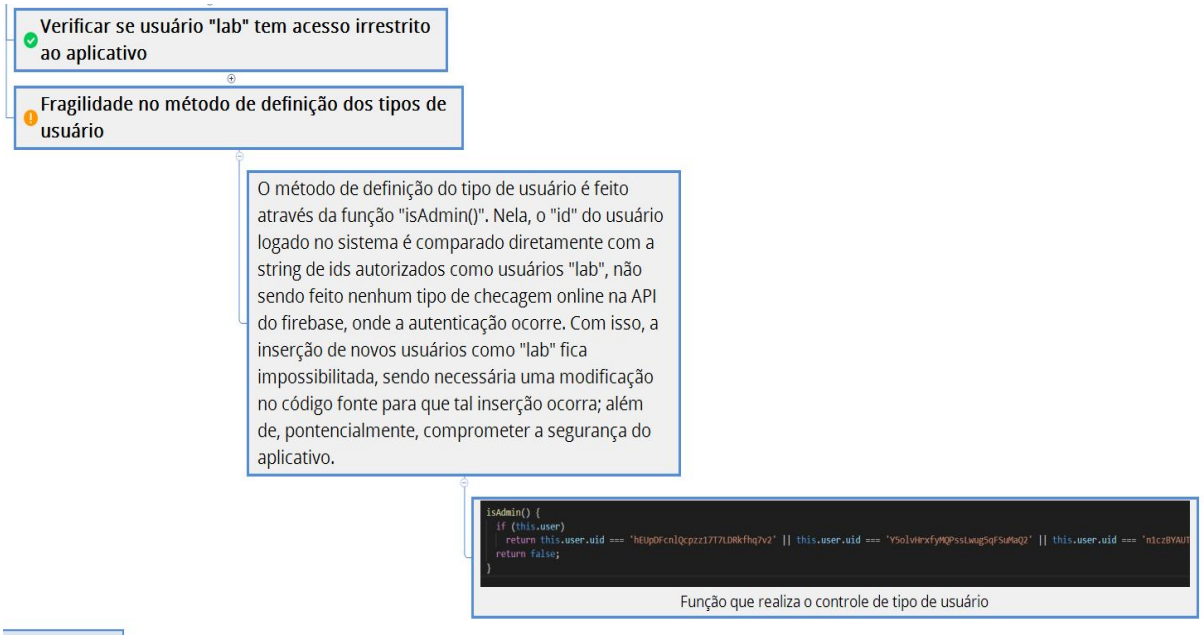


Fonte: Próprio autor.

Cada um desses casos de teste foi analisado, a nível de código, enquanto eram executados. Ao ser encontrada uma vulnerabilidade de segurança relacionada aos tipos de usuário do aplicativo, esta foi registrada, como informação importante, juntamente com uma descrição e um *print screen* do código (figura 6). Além disso, é importante ressaltar que os

testes foram feitos nas versões Android e WEB do aplicativo, tendo os mesmos resultados em ambas.

Figura 6 - Inserção de imagens no teste exploratório



Fonte: Próprio autor.

4.3 Arquitetura

Registrados os requisitos e feito o teste exploratório, foi então iniciada a diagramação da arquitetura do Vigilante Microbiológico. Esta etapa foi realizada durante os meses de julho e agosto de 2020, sendo baseada na análise do código fonte e uso prático do aplicativo.

Na arquitetura de um software, são documentados os seus componentes, suas propriedades, suas relações com softwares externos e as tecnologias empregadas na implementação. Essas informações formam a base de qualquer aplicação e precisam ser registradas para uso no desenvolvimento e manutenção do mesmo.

Tradicionalmente, a documentação de arquitetura é formada de uma série de diagramas representando diferentes escopos de informação do software. Apesar de não haver um consenso na linguagem de diagramação, como mostrado por Brown (2014), a UML é a que vem sendo mais utilizada ao longo dos anos.

Para Brown (2014), a indústria de desenvolvimento de software tem dificuldade na padronização e uso de ferramentas de visualização de arquitetura. Apesar da existência da UML e ArchiMate, por exemplo, muitos grupos de desenvolvimento acabam abandonando tais práticas e optando por uma linguagem própria, em busca de agilidade. O problema é que,

muitas vezes, a modelagem adotada é confusa e não representa corretamente o que está sendo desenvolvido.

Neste trabalho, visando representar a arquitetura do Vigilante Microbiológico de forma mais simples, direta e de fácil entendimento, foi utilizado o modelo C4, desenvolvido por Simon Brown. Este modelo tem como princípio básico a ênfase na abstração, ao invés da notação. Desta forma, segundo o seu criador, é possível focar na informação que se quer passar, deixando as estruturas de notação como foco secundário.

Com o C4, Brown (2014) criou uma forma de modelagem com apenas quatro níveis de abstração, como mostrado a seguir:

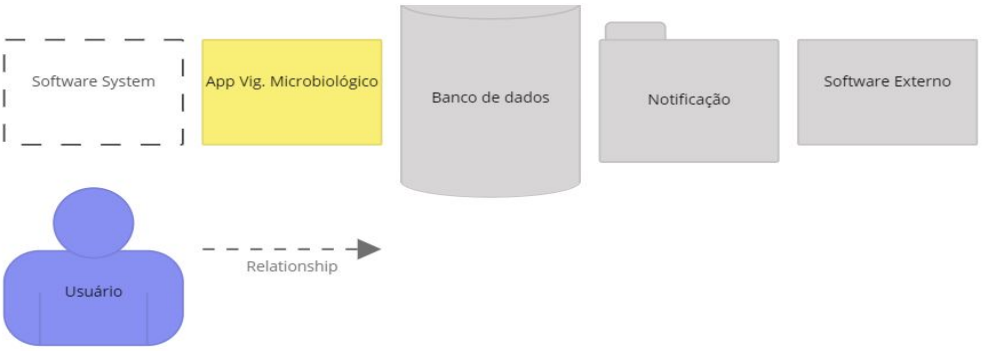
- **Contexto**: Apresenta uma visão de alto nível, composto por usuários e funcionalidades do software.
- **Container**: Mostra as tecnologias utilizadas no desenvolvimento do software, quais as responsabilidades de cada um e como se comunicam entre si.
- **Componente**: Cada container é composto por componentes lógicos que se relacionam entre si.
- **Classes**: É o mais baixo nível de representação. Serve para explicar detalhadamente como um determinado componente funciona, através de um diagrama de classes UML.

Cada nível representa uma visão diferente do software, abrangendo desde casos de uso até classes do código. O objetivo final é criar um único documento interativo e de fácil entendimento que tenha uma representação geral do sistema.

A ferramenta de documentação utilizada na arquitetura foi o Structurizr, versão 2120. Ela foi desenvolvida pelo próprio criador do modelo C4 e permite a criação dos diagramas segundo os quatro níveis de abstração deste modelo.

A figura 7 representa a notação utilizada na diagramação. Como descrito anteriormente, esta parte é feita por último e serve para explicar ao desenvolvedor o que cada estrutura, no diagrama, significa ou representa.

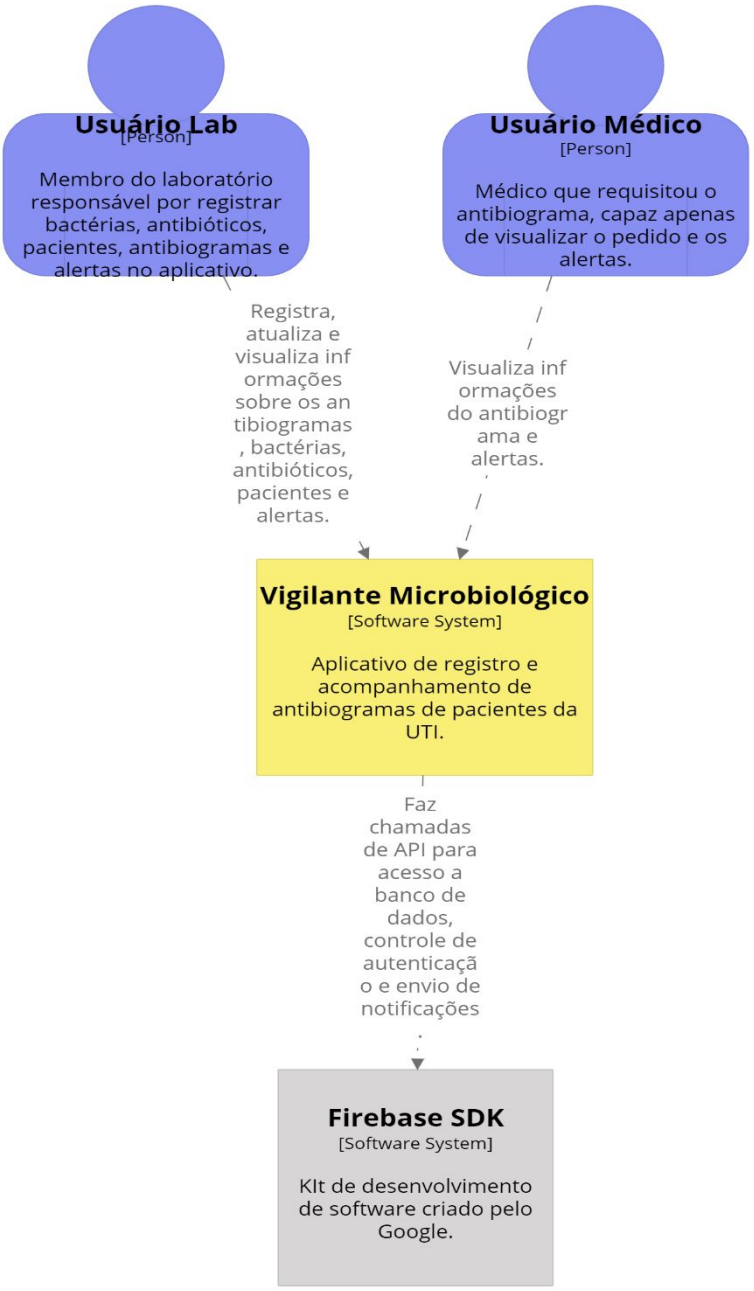
Figura 7 – Legenda dos diagramas C4



Fonte: Próprio autor.

Na figura 8, temos o diagrama do primeiro nível de abstração do modelo C4. Aqui, o objetivo é gerar uma visão de alto nível do software, mostrando os usuários do aplicativo interagindo com a aplicação, assim como a interação da aplicação com softwares externos.

Figura 8 – Diagrama de contexto



Fonte: Próprio autor.

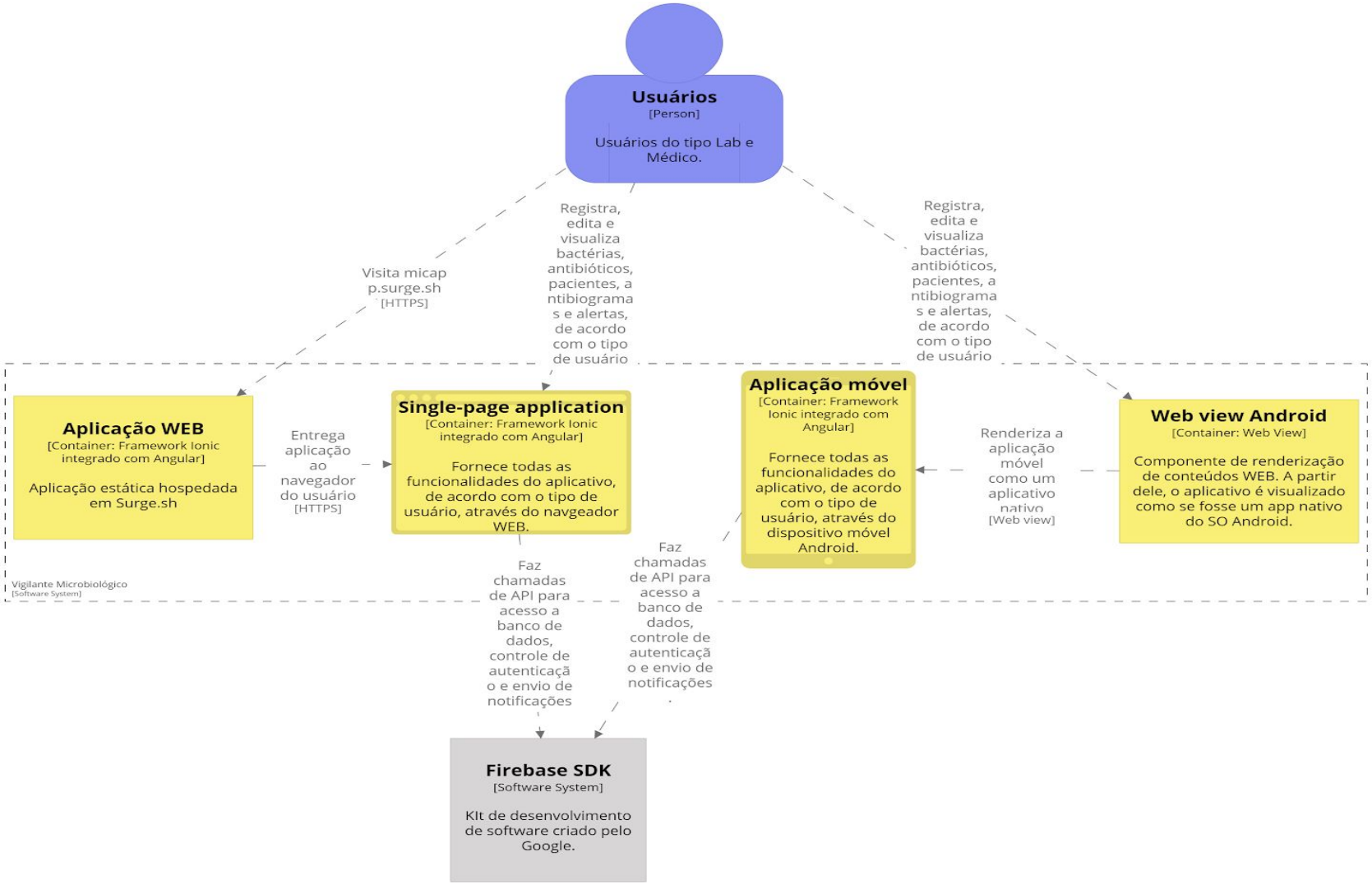
Neste primeiro nível de abstração, bem como no nível de container e componentes, é importante descrever as interações, os atores e os sistemas, e não, apenas, representá-los por frases curtas ou expressões, pois isso pode comprometer a compreensão do documento.

O nível seguinte de abstração é o nível de container. Os containers podem ser interpretados como partes do sistema, que armazenam dados ou executam um código com uma mesma finalidade sem detalhar as classes, funções e outras estruturas de mais baixo

nível. Dessa forma, esta parte da diagramação se concentra em demonstrar as tecnologias utilizadas no desenvolvimento do sistema, bem como as relações entre si.

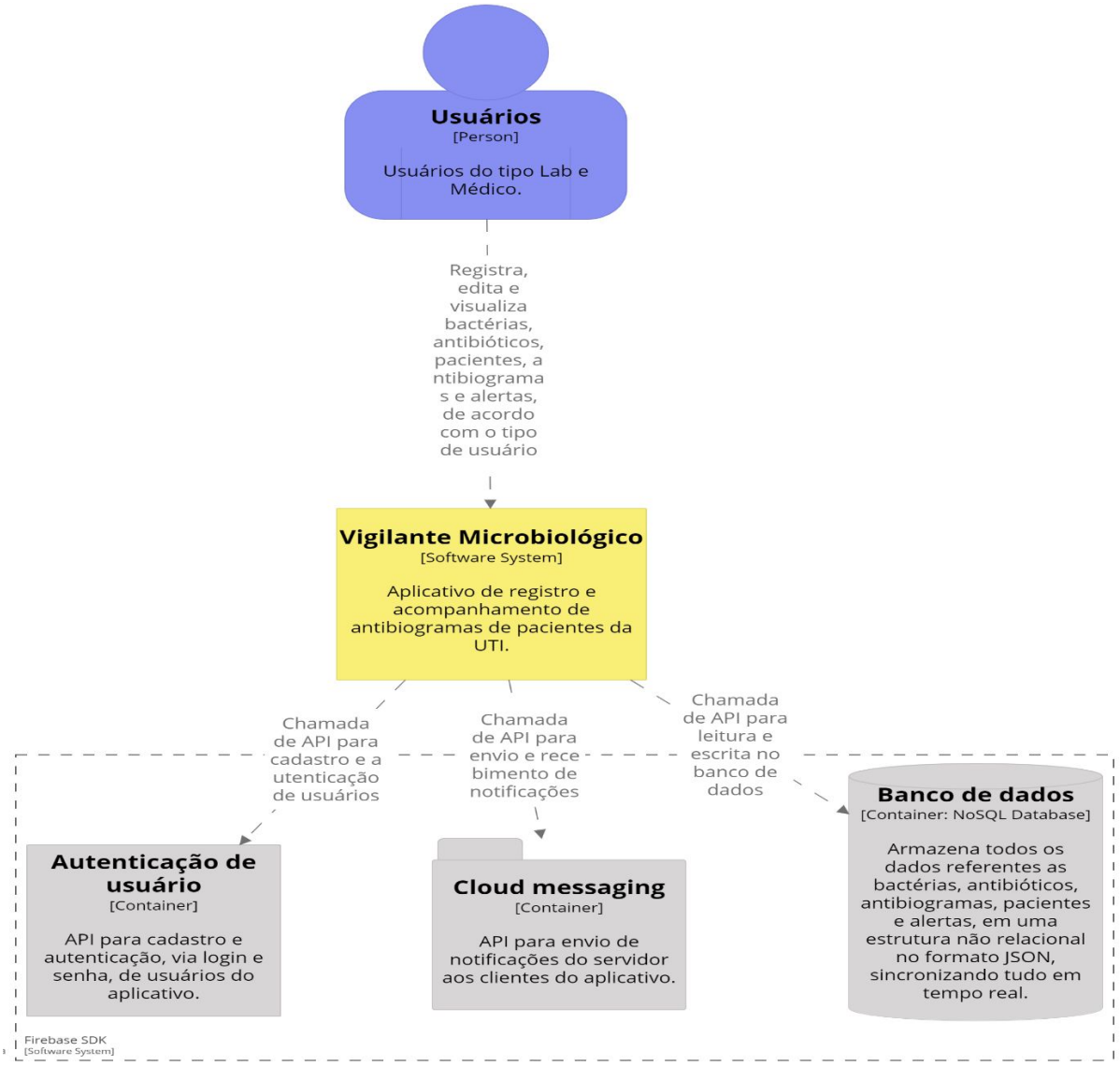
A figura 9 mostra o diagrama de container do Vigilante Microbiológico. Nele, são apresentadas as linguagens de programação, frameworks e interações entre os containers. Por existir o sistema externo do Firebase, versão 5.0.1, fornecendo funcionalidades através de API, um segundo diagrama de container foi gerado para ele (figura 10), com o intuito de detalhar melhor a sua relação com o aplicativo.

Figura 9 – Diagrama de container do Vigilante Microbiológico



Fonte: Próprio autor.

Figura 10 – Diagrama de container do SDK Firebase

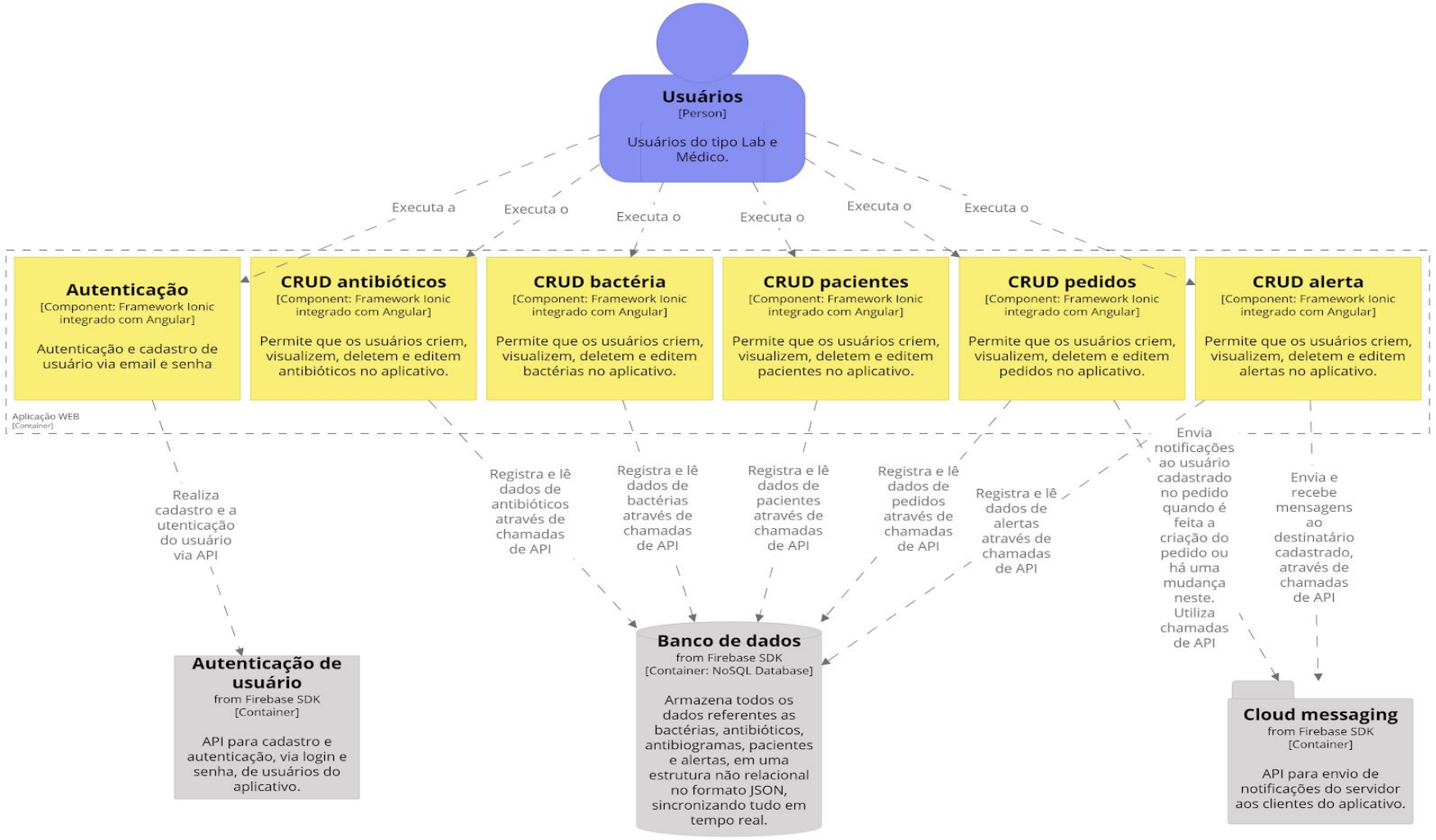


Fonte: Próprio autor.

No terceiro nível do modelo C4 temos os componentes. Os componentes representam as unidades lógicas que compõem os containers. Este nível apresenta uma visão mais próxima do código fonte, mostrando as funcionalidades principais da aplicação, e como elas interagem entre si e com outros sistemas, mas sem expor classes, atributos e métodos.

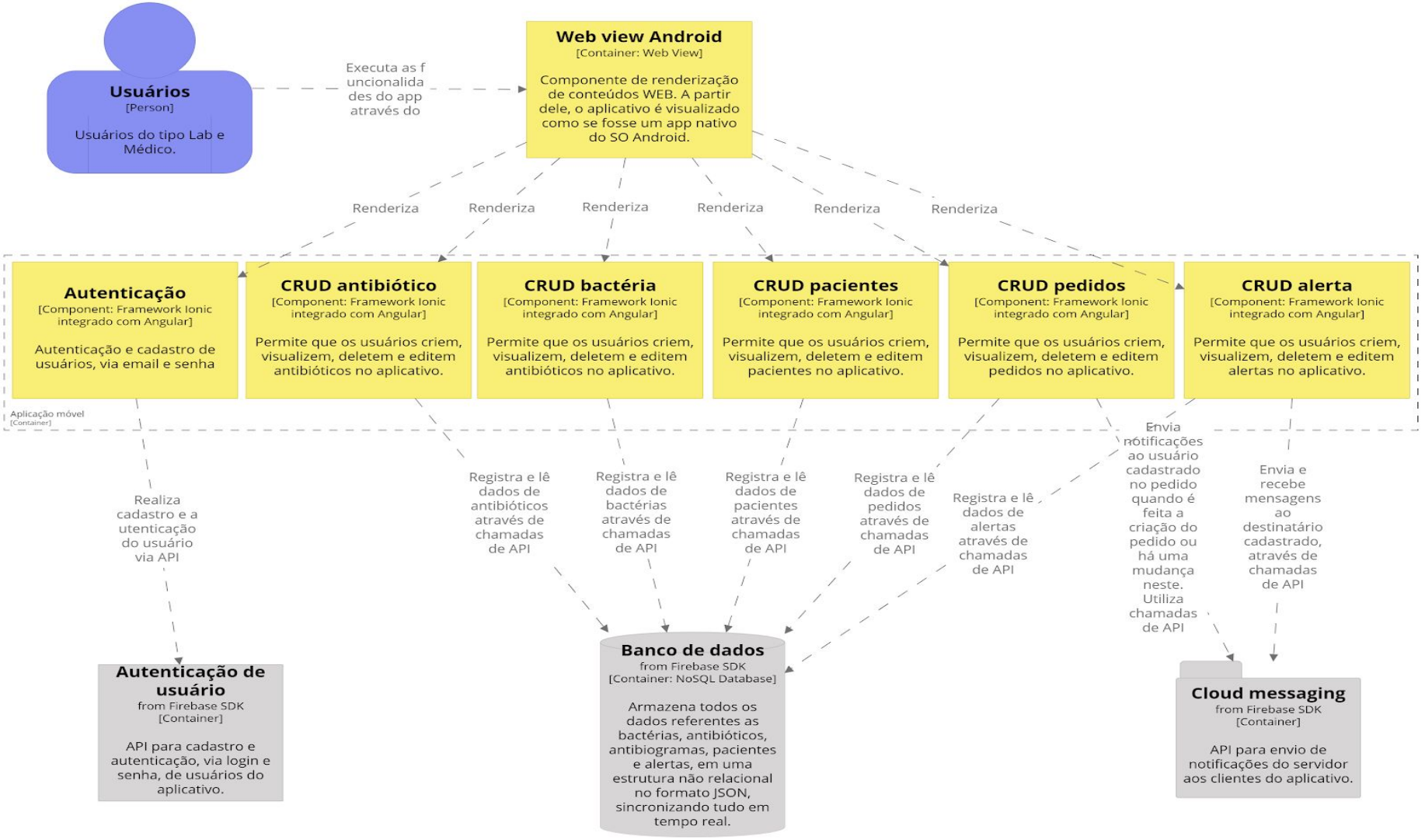
Nas figuras 11 e 12 temos os diagramas de componente feitos para este trabalho. Neles, é importante perceber que os componentes dos dois diagramas são exatamente os mesmos, pois a aplicação foi desenvolvida utilizando um framework multiplataforma que usa um único código fonte. Porém, o WEB utiliza o *browser* enquanto o móvel precisa da tecnologia *Web View* Android para visualização e execução, por isso a necessidade da criação de dois diagramas distintos.

Figura 11 – Diagrama de componentes do aplicativo WEB



Fonte: Próprio autor.

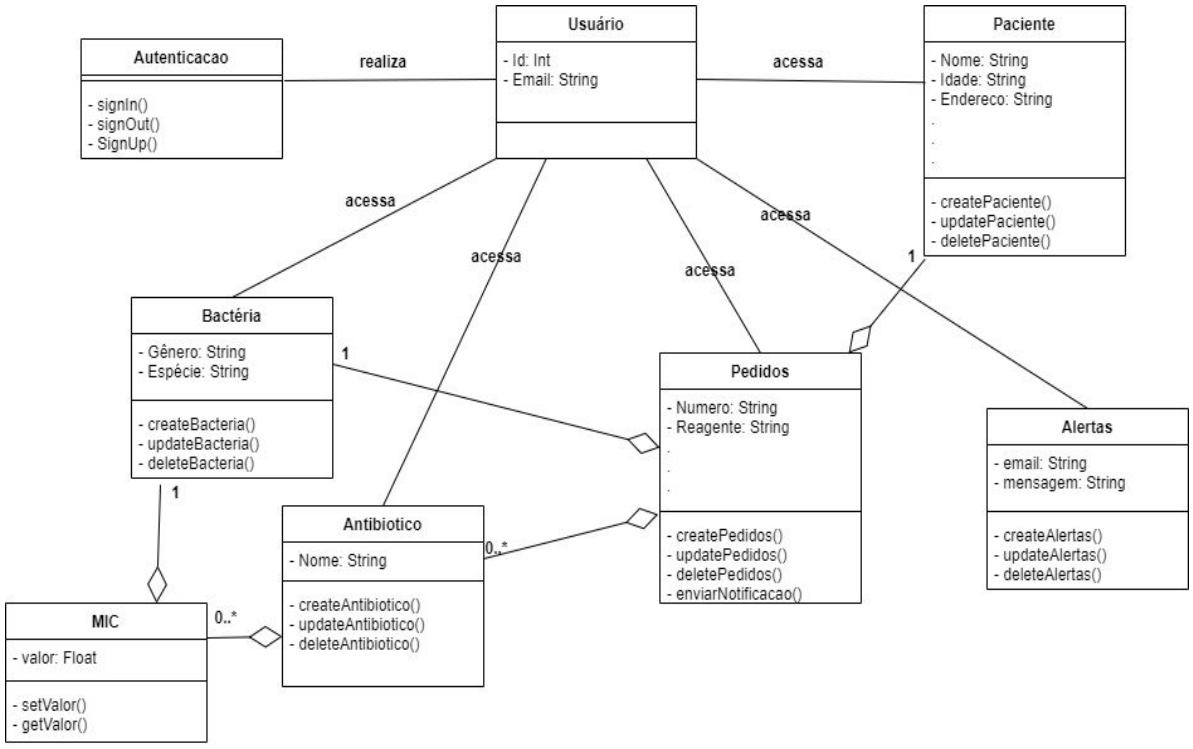
Figura 12 – Diagrama de componentes do aplicativo móvel



Fonte: Próprio autor.

Por último, temos o nível de código, ou classe, como também é chamado. Este é o mais baixo nível de abstração representado pelo modelo C4. É nesta etapa que classes, entidades, relacionamentos, métodos e atributos são diagramados utilizando a notação tradicional do UML, como pode ser visto nas figuras 13 e 14.

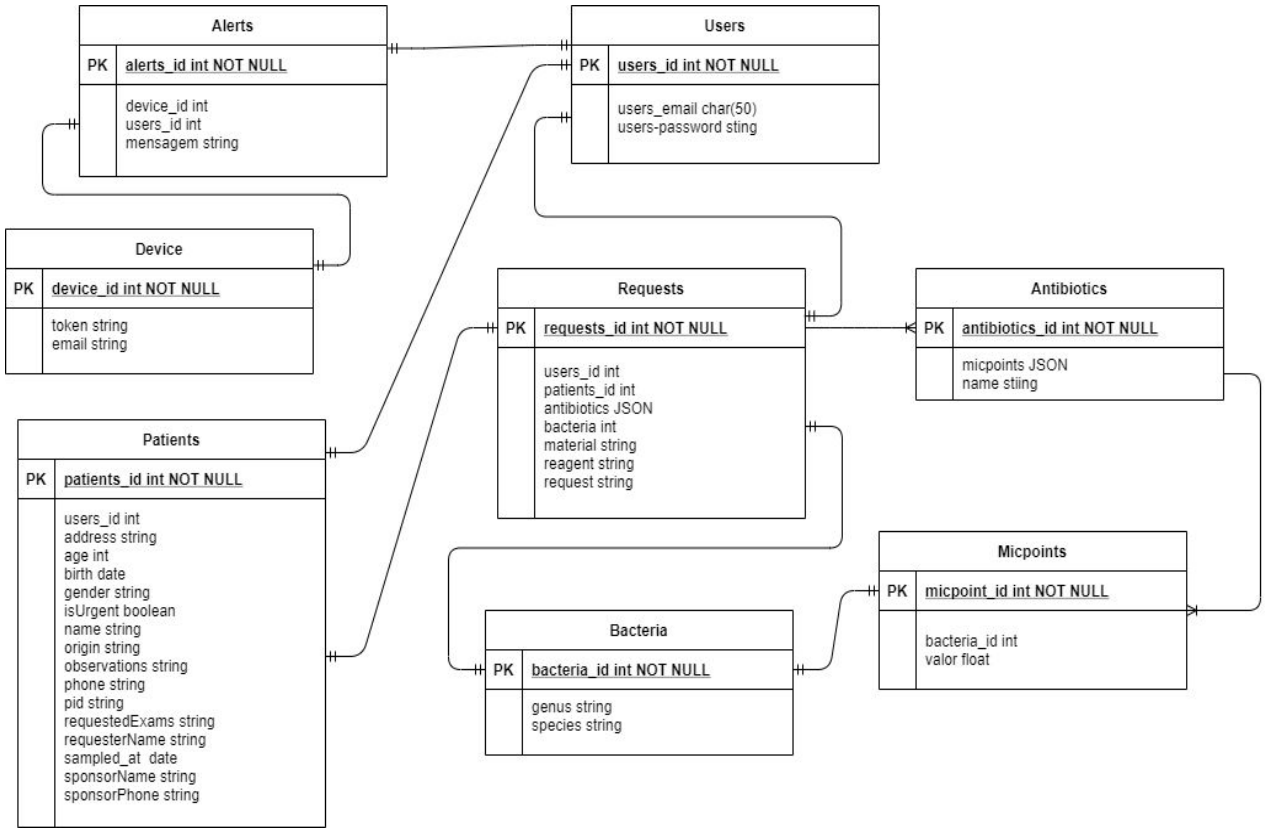
Figura 13 – Diagrama de classes



Fonte: Próprio autor.

Com a diagramação da arquitetura completa, a localização por informações arquiteturais tornou-se mais rápida e prática. A modelagem baseada no modelo C4 mostrou-se mais detalhada e, ainda assim, mais simples de visualizar e interpretar as informações, como constatado por Gladson Corrêa, que, apesar de ser o idealizador do software, não possui conhecimento na área de TI e, mesmo assim, conseguiu entender grande parte do funcionamento da aplicação ao visualizar os diagramas.

Figura 14 – Diagrama de entidade e relacionamento do BD Firebase



Fonte: Próprio autor

4.4 Teste de aceitação de usuário

O teste de aceitação de usuário (UAT) foi baseado na obra de Hambling e Goethem (2013). Neste livro, o UAT é constituído por criação do plano de teste, criação dos casos de teste e, por último, execução do teste e confirmação, ou não, de que a aplicação foi aprovada. Todas estas etapas geram documentação auxiliar, porém, para este trabalho, visando diminuir ao máximo o volume de documentos sem comprometer a compreensão geral do projeto, o teste foi adaptado gerando documentação apenas para a última etapa, de execução e confirmação.

Outra motivação para a criação do plano de teste e criação dos casos de teste serem feitas sem a geração de documentação é o tamanho do projeto e da equipe, que são considerados de pequeno porte. Além disso, a criação e execução do teste foi feita por apenas uma pessoa, não sendo necessário compartilhar informações, anteriores à etapa de execução, entre membros de equipe.

No plano de teste, foram definidos os usuários participantes, os dados que seriam usados, a forma e a execução do teste, e o critério de aceitação. Os usuários participantes

foram 13 estudantes e profissionais de farmácia, todos escolhidos pelo próprio idealizador da aplicação, assinando um termo de consentimento (Anexo A) autorizando o registro de dados pessoais e concordando em participar da atividade. Os dados de teste foram fornecidos, também, por Gladson Carvalho, diretamente aos testadores no ato da execução do teste. A forma de execução do UAT foi definida como formulário online, gerado no Google Docs de formulário, visando gerar um documento interativo, simples e de fácil compartilhamento. E, por fim, o critério de aceitação foi definido, junto com Gladson Carvalho, como sendo, no mínimo, 90% de aprovação em cada caso de teste.

A definição dos casos de teste foi feita com base nos requisitos da aplicação MICAPP (Apêndice A). A partir dos requisitos e em acordo com Gladson Carvalho, foram definidas as principais funcionalidades da aplicação, sendo elas: autenticação, registro de antibióticos, registro de bactérias, registro de pacientes, registro de pedidos e registro de alertas. A seguir, foram criados os testes de caso de cada uma dessas funcionalidades, segundo os requisitos funcionais da aplicação.

Definidos os casos de teste, foi criado o formulário para execução do UAT. Este documento foi organizado no formato de formulário padrão do Google Docs, sendo subdividido entre descrição e mais sete seções: autenticação, registro de antibióticos, registro de bactérias, registro de pacientes, registro de pedidos e registro de alertas (Quadro 2). Na descrição, foi explicada a organização do teste e como ele deveria ser feito, e, nas demais seções, foram colocados os casos de teste. Cada caso de teste possui uma descrição do que deve ser feito e um resultado esperado, o qual deveria ser igual ao resultado obtido pelo testador para que o caso fosse considerado aprovado. Além dos casos de teste, foi adicionado uma pergunta para avaliar o nível de satisfação do usuário durante o uso da aplicação em cada uma das seções.

Quadro 2 - Seções de teste do UAT

Seção	Objetivo	Número de casos de teste
Autenticação	Testar o cadastro de novos usuários; Testar o login de usuários; Avaliar o nível de satisfação do usuário.	2
Registro de antibióticos	Testar a criação, edição e deleção de antibióticos; Testar a edição da MIC nos antibióticos; Avaliar o nível de satisfação do usuário.	4
Registro de bactérias	Testar a criação, edição e deleção de bactérias; Avaliar o nível de satisfação do usuário.	3

Registro de pacientes	Testar a criação, edição e deleção de pacientes; Avaliar o nível de satisfação do usuário.	3
Registro de pedidos	Testar a criação, edição e deleção de pedidos; Avaliar o nível de satisfação do usuário.	3
Registro de alertas	Testar a criação, edição e deleção de alertas; Avaliar o nível de satisfação do usuário.	3

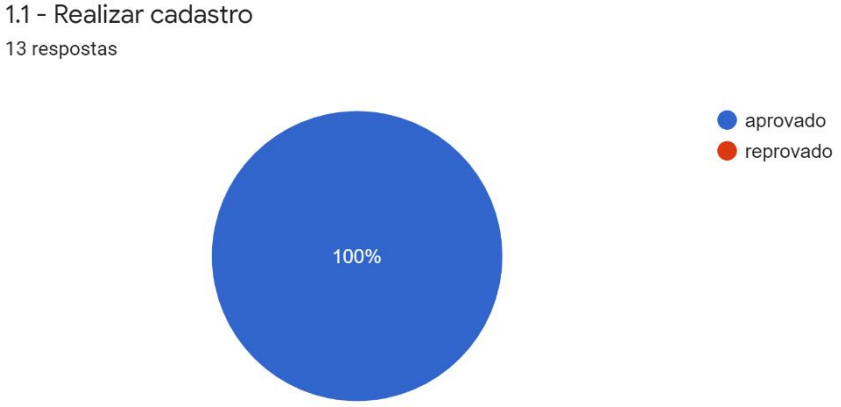
Fonte: Próprio autor.

O Apêndice B mostra como ficou o formulário pronto.

A execução do UAT ocorreu após a montagem do formulário de teste. Devido a pandemia de COVID-19, o teste foi feito de forma completamente online, a fim de evitar o risco de contágio entre os testadores e o aplicador. Sendo assim, foi enviado um e-mail a um grupo, criado a partir dos e-mails disponibilizados no termo de consentimento (Anexo A), contendo os *links* para acesso do formulário e da aplicação. O envio foi feito em 25 de setembro de 2020, marcando o início do UAT. Em 2 de outubro de 2020, foi finalizada a execução do teste e feita a análise dos resultados. Nas figuras 14 a 37, são mostrados os gráficos com a porcentagem de aprovação e reprovação, de cada caso de teste, e do nível de satisfação do usuário em cada seção.

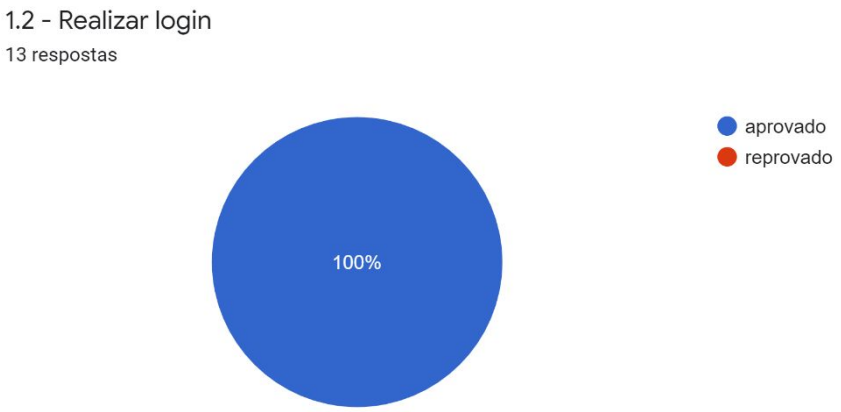
As figuras 15 e 16 são referentes aos casos de teste da seção de autenticação do MICAPP. Esta seção visa validar a capacidade de cada usuário de se cadastrar e acessar a aplicação como um usuário autenticado. Nestes casos de teste, os usuários tinham que realizar o cadastro e, depois, efetuar o login com o usuário recém criado, respectivamente. Como resultado, ambos os casos de teste receberam cem por cento de aprovação, ou seja, não tiveram *bugs*.

Figura 15 – Porcentagem de aprovação do caso de teste Realizar Cadastro



Fonte: Próprio autor.

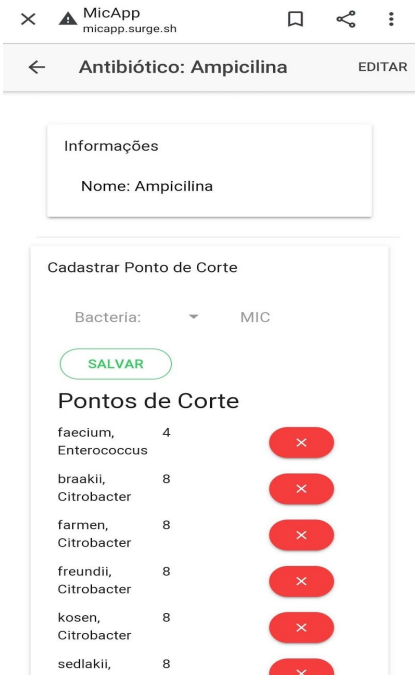
Figura 16 - Porcentagem de aprovação do caso de teste Realizar Login



Fonte: Próprio autor.

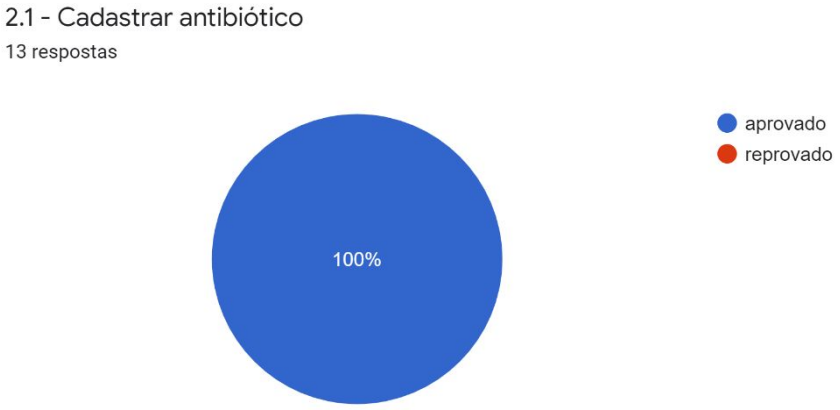
Na figura 18, abaixo, e nas figuras 1, 2 e 3, Apêndice C, temos a seção de antibióticos. Aqui, o objetivo é avaliar se o usuário consegue realizar o CRUD *(create, read, update e delete)*, sigla em inglês referente ao ato de criação, leitura, atualização e deleção de novos registros dentro de um banco de dados. Neste caso, o CRUD (tela de antibióticos no Anexo B) foi feito a partir de dados relacionados a antibióticos, sendo necessário ao usuário cadastrar o nome do antibiótico, cadastrar a concentração inibitória mínima (MIC) (figura 17), editar as informações cadastradas e deletar o antibiótico cadastrado, respectivamente.

Figura 17 - Tela de edição de antibióticos



Fonte: Próprio autor.

Figura 18 - Porcentagem de aprovação do caso de teste Cadastrar Antibiótico



Fonte: Próprio autor.

Assim como a seção de autenticação, todos os casos de teste alcançaram cem por cento de aprovação.

A seção seguinte, dentro do UAC, foi a de registro de bactérias. Esta seção testa o CRUD de informações bacterianas que, assim como os antibióticos, são necessárias para o posterior cadastro de antibiogramas e seus resultados. Como testes de caso, foram feitos os cadastros dos gêneros e espécies de bactérias, a edição destas informações cadastradas e a deleção da bactéria registrada, respectivamente.

Mais uma vez, o teste foi bem sucedido, e todos os casos de teste foram aprovados pelos usuários (figuras 4, 5 e 6 no Apêndice C).

Na seção de registro de pacientes, os usuários tinham que cadastrar um paciente fictício, preenchendo todos os campos mostrados na tela, incluindo nome do paciente, usuário ao qual o paciente está vinculado, idade, entre outras informações. A seguir, deveriam editar esses campos e deletar o paciente criado, completando o CRUD.

Como mostrado nas figuras 7, 8 e 9 do Apêndice C, todos os casos de teste alcançaram cem por cento de aprovação.

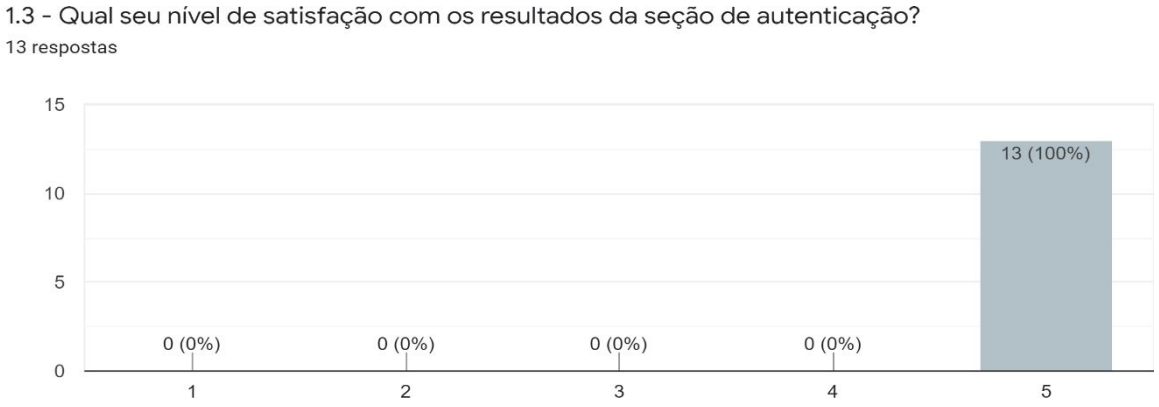
Analogamente às seções anteriores, a seção de cadastro de pedidos consistia em cadastrar, editar e deletar os pedidos de antibiogramas. Aqui, o usuário deve registrar um antibiograma e associá-lo a um usuário e um paciente registrados no sistema. Além disso, ainda no caso de teste de cadastro de pedidos, deve-se registrar valores de MIC para antibióticos cadastrados no sistema e associar esse valor a uma codificação referente a resistência, sensibilidade ou imunidade. Feito isto, o usuário deve editar as informações cadastradas no antibiograma e depois deletar o mesmo.

Os resultados desta seção são mostrados nas figuras 10, 11 e 12 do Apêndice C, revelando uma aprovação de cem por cento dos usuários.

Por último, temos a seção de alertas do MICAPP. Os alertas servem para enviar mensagens aos usuários cadastrados, servindo como um meio de comunicação direto entre os usuários membros do laboratório e aqueles que requisitaram o antibiograma.

Nesta seção, os usuários tinham que cadastrar, editar e deletar as notificações criadas. Nas figuras 13, 14 e 15 do Apêndice C, temos os resultados bem sucedidos do cadastro, edição e deleção de notificações, respectivamente, com cem por cento de aprovação.

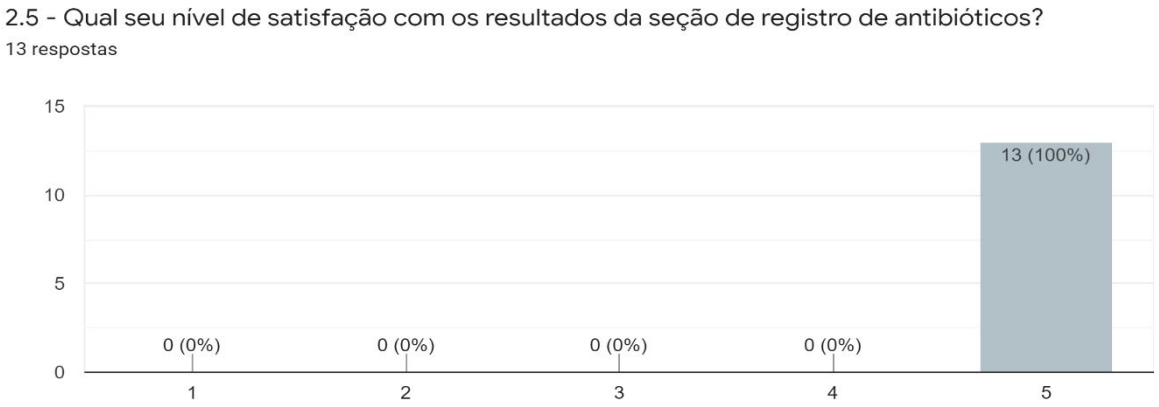
Figura 19 - Nível de satisfação com a seção de autenticação



Fonte: Próprio autor.

Em todas as seções do teste de aceitação do usuário, foi adicionada uma pergunta questionando o nível de satisfação do usuário com determinada seção (figura 19 a 24).

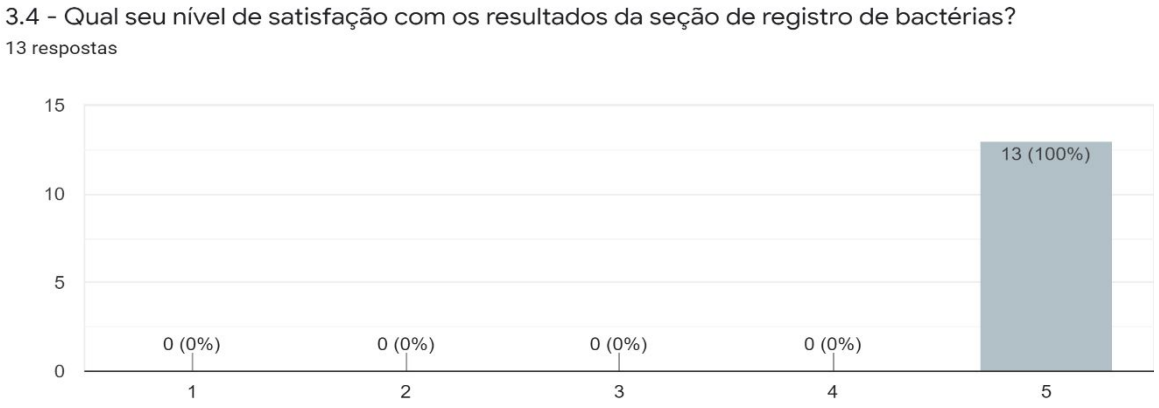
Figura 20 - Nível de satisfação com a seção de registro de antibióticos



Fonte: Próprio autor.

O intuito dessa questão é avaliar a experiência do usuário ao realizar os casos de teste no MICAPP, verificando se houve dificuldades durante a realização dos mesmos, se a interface é intuitiva e se ocorrem muitos erros. Com base nisso, os usuários deveriam dar uma nota de um a cinco, como é mostrado nos resultados.

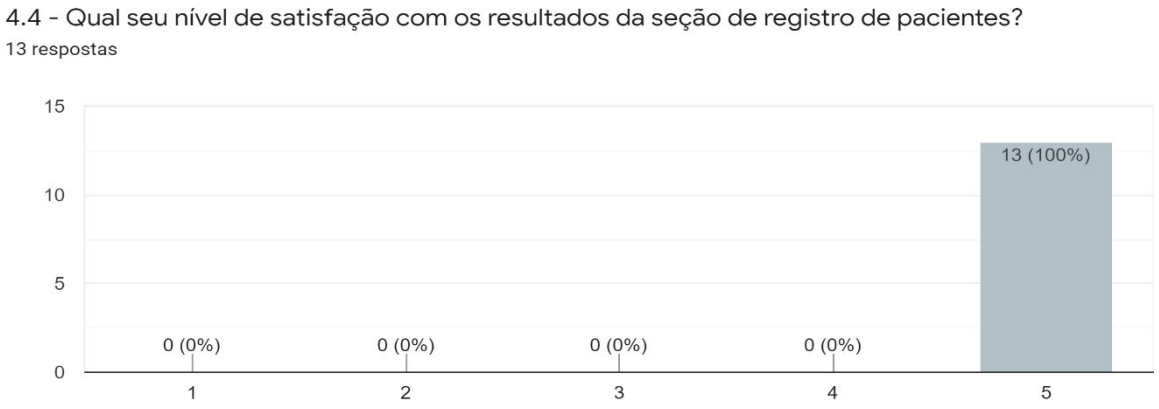
Figura 21 - Nível de satisfação com a seção de registro de bactérias



Fonte: Próprio autor.

É possível ver que todos os usuários responderam com cem por cento de grau de satisfação em todas as seções do UAC, evidenciando que não houve dificuldade ou problema para a realização dos testes.

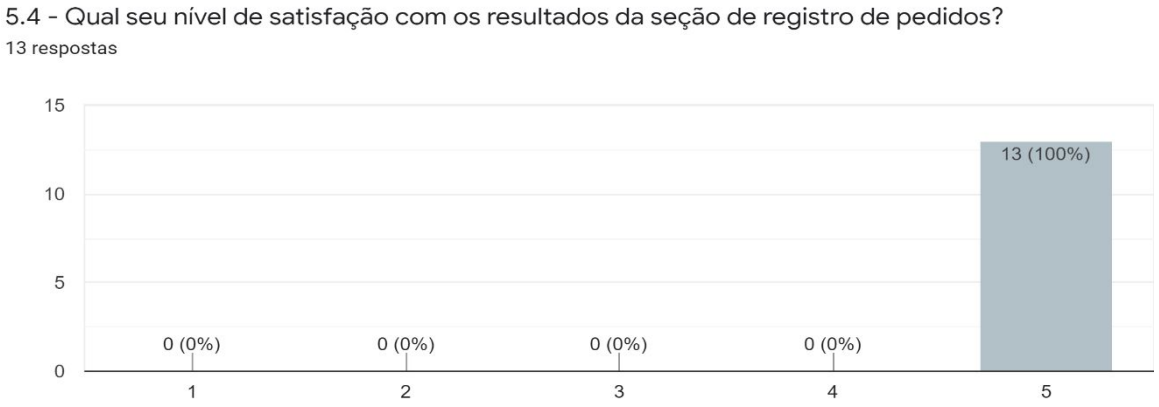
Figura 22 - Nível de satisfação com a seção de registro de pacientes



Fonte: Próprio autor.

Como mostrado nos gráficos, todos os casos de teste obtiveram 100% de aprovação pelos testadores, superando os 90% de aprovação mínimos determinados em acordo com Gladson Carvalho.

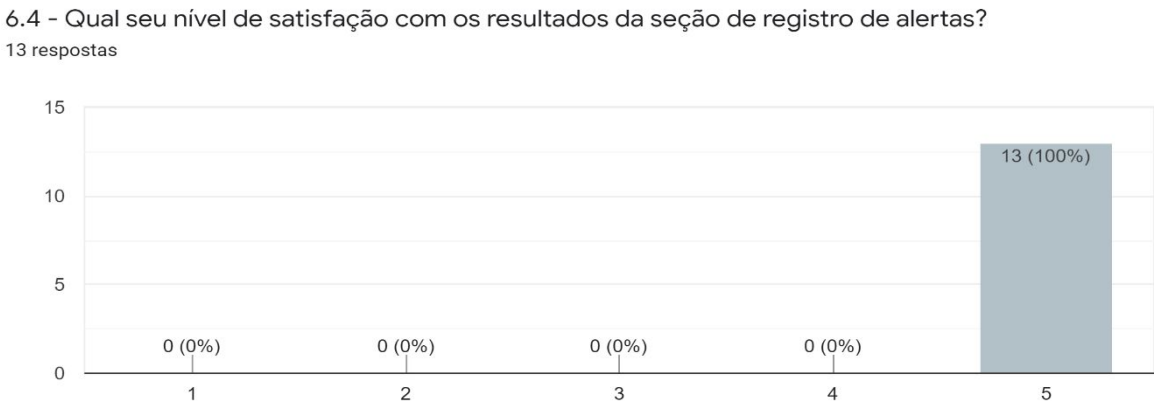
Figura 23 - Nível de satisfação com a seção de registro de pedidos



Fonte: Próprio autor.

Além disso, todos os usuários responderam com nível de satisfação cinco em todas as seções, o que revela que não houve problemas na execução dos testes ou dificuldades para a realização de tais. Sendo assim, o teste de aceitação de usuário foi considerado aprovado, e o software validado por usuários finais, podendo ser colocado em produção.

Figura 24 - Nível de satisfação com a seção de registro de alertas



Fonte: Próprio autor.

Os *bugs* encontrados durante o teste exploratório, como a implementação do requisito funcional RF03, vão ser corrigidos em versões futuras do aplicativo, por não afetarem as principais funcionalidades do MICAPP.

A implementação do UAT, usando o formulário do Google Docs, mostrou-se bem sucedida. A criação do teste foi rápida e intuitiva devido a interface bem estruturada do Google Docs. A execução do teste foi igualmente simples e rápida, apresentando os resultados em tempo real e já em formato de gráficos. Com a aplicação deste teste, foram concluídos todos os documentos do Vigilante Microbiológico.

5 CONCLUSÃO

O desenvolvimento deste trabalho possibilitou a descoberta e comparação das melhores práticas de documentação dentro da metodologia ágil. A partir da pesquisa bibliográfica, foi possível resumir as melhores e mais recentes pesquisas na área de documentação ágil, sendo que parte delas foi feita a partir de estudos de caso em empresas de desenvolvimento de software, o que enriqueceu o resultado do trabalho com dados reais de mercado.

Após a criação do arcabouço teórico, os objetivos específicos também foram alcançados, através da proposição e aplicação de ferramentas que dessem agilidade e facilitassem a criação e o entendimento da documentação de software. Todas as ferramentas utilizadas mostraram-se extremamente úteis, tornando o processo de documentação do aplicativo móvel Vigilante Microbiológico mais simples e inteligível.

No processo de desenvolvimento deste trabalho, é importante citar que também foram encontradas algumas dificuldades. O primeiro desafio foi encontrar trabalhos que abordassem a documentação segundo a metodologia ágil. Na pesquisa realizada, pouco material científico foi encontrado nesta área, o que pode indicar uma possível subestimação quanto ao assunto tratado e as suas implicações dentro do desenvolvimento ágil.

Ademais, alguns problemas também foram encontrados na parte prática do trabalho. Como a documentação do MICAPP foi gerada após a conclusão do desenvolvimento do mesmo, todo o entendimento do software, a nível de código e de usuário, dependeu de uma extensa exploração da interface e das linhas do código fonte, o que aumentou o tempo necessário para a criação dos documentos. Junto a isso, a documentação tardia também comprometeu o teste da aplicação, visto que os testes unitários, de acoplamento, de regressão, entre outros, não puderam ser feitos.

É importante ressaltar que, na etapa final da documentação do MICAPP, foi feito o teste de aceitação de usuário para validação da aplicação. Todo o teste foi feito com usuários escolhidos pelo cliente da aplicação, Gladson Carvalho, utilizando dados reais, também fornecidos por ele. O resultado foi a aprovação, com cem por cento em todos os testes realizados, liberando a versão atual do MICAPP para a entrega. Em versões futuras, no entanto, é recomendado um número maior de testadores no UAT, a fim de aumentar, ainda mais, a qualidade da validação.

Como contribuição principal deste trabalho, temos a abordagem da documentação no desenvolvimento ágil, que se mostrou pouco discutido, tanto na área acadêmica quanto na

profissional. Sendo assim, o resultado esperado é trazer maior relevância ao tema, inspirando novos trabalhos que deem prosseguimento ao que foi feito; adicionando novas ideias e gerando maior repercussão sobre a importância do bom planejamento na hora de gerar e registrar documentos do ciclo de vida de um software.

Para trabalhos futuros, encorajamos a proposição de novas ferramentas que facilitem ainda mais a criação dos documentos de software. Também propomos que seja feito um aprofundamento na área de documentação de testes, quem sabe até, propondo um *framework* de criação ágil para tais artefatos.

REFERÊNCIAS

ARYA, Deeksha. **Exploring the Correspondence Between Types of Documentation for Application Programming Interfaces**. 2019. Tese (Mestrado em Ciência da Computação) - School of Computer Science, McGill University, [*S. l.*], 2019.

BANOS, O. Design, implementation and validation of a novel openframework for agile development of mobile health applications. **BioMedical Engineering OnLine**, [*s. l.*], 2015.

BARRA, Daniela; ALMEIDA, Sônia; SASSO, Grace; PAESE, Fernanda; RIOS, Greize. Metodologia para Modelagem e Estruturação do Processo de Enfermagem Informatizado em Terapia Intensiva. **Texto Contexto Enferm.**, [*s. l.*], 2016.

BRIAND, Lionel. Software Documentation: How Much is Enough?. **Seventh European Conference On Software Maintenance And Reengineering (CSMR'03)**, [*s. l.*], 2003.

BROWN, Simon. **Software Architecture for Developers:** A developer-friendly guide to software architecture, technical leadership and the balance with agility. [*S. l.*: *s. n.*], 2014.

CARVALHO, Raimundo. **Desenvolvimento de Tecnologia Assistencial para Informe Rápido de Resultados de Microbiologia a Profissionais da Saúde (MICAPP)**. 2020. Tese (Mestrado em Análises Clínicas) - Instituto de Ciências Biológicas, Universidade Federal do Pará, [*S. l.*], 2020.

CHOO, Chun. **The Knowing Organization: How Organizations Use Information to Construct Meaning, Create Knowledge, and Make Decisions**. [*S. l.*: *s. n.*], 1998.

FORWARD, Andrew. **Software Documentation - Building and Maintaining Artefacts of Communication**. 2002. Tese (Mestrado em Ciência da Computação) - Ottawa-Carleton Institute for Computer Science, University of Ottawa, [*S. l.*], 2002.

FOWLER, Martin; HIGHSMITH, Jim. The Agile Manifesto. **Software Development**, [*s. l.*], 2001.

FOWLER, Martin *et al*. **Manifesto for Agile Software Development**. [*S. l.*], 2001. Disponível em: https://agilemanifesto.org/. Acesso em: 15 abr. 2020.

GAROUSI, Golara; GAROUSI-YUSIFOG˘LU, Vahid; RUHE, Guenther; ZHI, Junji; MOUSSAVI, Mahmoud; SMITH, Brian. Usage and usefulness of technical software documentation: An industrial case study. **Information and Software Technology**, [*s. l.*], 2015.

GUIMARÃES, Eliane; GODOY, Solange. Telenfermagem - Recurso para Assistência e Educação em Enfermagem. **REME - Rev Min Enferm.**, [*s. l.*], 2012.

HADAR, Irit; SHERMAN, Sofia; HADAR, Ethan; HARRISON JR., John. Less is More: Architecture Documentation for Agile Development. **2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)**, [*s. l.*], 2013.

HAMBLING, Brian; GOETHEM, Pauline. **User Acceptance Testing: A Step-by-step Guide**. [*S. l.*: *s. n.*], 2013.

HAZZAN, O.; DUBINSKY, Y. The Agile Manifesto. **SpringerBriefs in Computer Science**, [*s. l.*], 2014.

MARQUES, Lorena; CARVALHO, Renata; PAIVA, Thaisa; VIEIRA, Giovane; VITORIO, Aline. A tecnologia de informação em prol da segurança do paciente: o uso de aplicativos em dispositivos móveis na adesão ao checklist cirúrgico. **Revista Rede de Cuidados em Saúde**, [*s. l.*], 2017.

MATSUDA, Laura; ÉVORA, Yolanda; HIGARASHI, Ieda; GABRIEL, Carmen; INOUE, Kelly. Informática em Enfermagem: Desvelando o Uso do Computador por Enfermeiros. **Texto Contexto Enferm.**, [*s. l.*], 2015.

PERES, Heloisa; MARIM, Heimar. Informática em Enfermagem e Telenfermagem: desafios e avanços na formação e no cuidado. **J. health inform.**, [*s. l.*], 2012.

SATISH, C. J.; ANAND, Mahendran. Software Documentation Management Issues and Practices: a Survey. **Indian Journal of Science and Technology**, [*s. l.*], 2016.

SOMMERVILLE, Ian. Software Documentation. **Software Engineering**, [*s. l.*], 2001.

TRIPATHI, Vishvadeep; GOYAL, Arvind. A Document Driven Approach for Agile Software Development. **International Journal of Advanced Research in Computer Science and Software Engineering**, [*s. l.*], 2014.

VOIGT, Stefan; GARREL, Jörg; MÜLLER, Julia; WIRTH, Dominic. A Study of Documentation in Agile Software Projects. **ESEM '16: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement**, [*s. l.*], 2016.

WILLIAMS, Laurie. What Agile Teams Think of Agile Principles. **Communications of the ACM**, [*s. l.*], 2012. DOI 10.1145/2133806.2133823. Disponível em: https://dl.acm.org/doi/10.1145/2133806.2133823. Acesso em: 13 abr. 2020.

APÊNDICE A – Documentação de requisitos do aplicativo Vigilante Microbiológico

Documentação de Requisitos de Software

Aplicativo Vigilante Microbiológico

Idealizador: Raimundo Gladson Corrêa Carvalho
Programador: Amir Zahlan
Documentador: Felipe Gusmão Araújo

01/05/2020

1. Descrição geral do sistema

O aplicativo Vigilante Microbiológico é um software criado para registrar e controlar os resultados de cultura e antibiogramas de pacientes com infecções bacterianas. Ele foi idealizado para ser uma plataforma que agiliza a entrega de resultados, tornando as etapas de comunicação de resultados totalmente digitais e acessíveis através da internet.

O programa é acessado pelos membros do laboratório, que fazem a cultura e o antibiograma, e os médicos, que requisitam o exame. Os membros do laboratório têm acesso mais amplo ao aplicativo podendo, além de registrar o pedido de cultura e antibiograma e seu resultado, cadastrar antibióticos, bactérias, pacientes e alertas, utilizados para notificar os profissionais cadastrados. Já os médicos podem requisitar cultura e antibiograma e acompanhar o andamento e os resultados online, assim como receber alertas do laboratório.

2. Requisitos funcionais

RF001 - Requisitar login do usuário

O aplicativo deve possuir uma tela de login de usuário. Nela, devem ser requisitados login e senha de usuário, sendo permitido o acesso ao app apenas aqueles que estiverem cadastrados.

Restrição de usuário: nenhuma, disponível para os dois tipos de usuário.

RF002 - Permitir o cadastro de novos usuários

A tela de login do usuário deve possuir um botão que permita o cadastramento de

novos usuários.

Restrição de usuário: nenhuma, disponível para os dois tipos de usuário.

RF003 - Aprovação de cadastro de novo usuários

O cadastro de novos usuários deve gerar uma notificação aos usuários do tipo “lab”, cabendo a eles aprovar ou não esse cadastro. No ato da aprovação, o usuário “lab” que for analisar o pedido do cadastro deve, além de aprovar ou não, definir se o usuário que está se cadastrando é “lab” ou “médico”.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF004 - Possuir cadastro de antibióticos

O aplicativo deve possuir uma tela que permita o cadastro de antibióticos, além de listar, apenas pelo nome, os antibióticos já cadastrados.

O cadastro dos antibióticos deve conter o seguinte campo:

- Nome do antibiótico (permitir letras, números e sinais)

O cadastro das MIC (concentração inibitória mínima, em inglês) deste antibiótico, com as referentes bactérias, deve ser feita em uma tela separada ao clicar no nome do antibiótico listado na tela de antibióticos. (permitir apenas caracteres numéricos, ponto e vírgula).

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF005 - Permitir visualização de antibióticos

Na tela de antibióticos, deve ser possível visualizar todos os campos cadastrados de um antibiótico ao clicar no seu nome.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF006 - Permitir edição de antibióticos

O aplicativo deve possuir um botão que permita a edição do campo “nome” e das MIC do antibiótico.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF007 - Permitir deleção de antibióticos

O aplicativo deve possuir um botão que permita a deleção dos antibióticos registrados.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF008 - Possuir cadastro de bactérias

O aplicativo deve possuir uma tela que permita o cadastro de bactérias, além de listar as bactérias já cadastradas.

O cadastro das bactérias deve conter os seguintes campos:

- Gênero (permitir letras, números e sinais)
- Espécie (permitir letras, números e sinais)

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF009 - Permitir visualização de bactérias

Na tela de bactérias, deve ser possível visualizar todos os campos cadastrados de uma bactéria ao clicar na sua espécie.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF010 - Permitir edição de bactérias

O aplicativo deve possuir um botão que permita a edição dos campos “gênero” e “espécie” das bactérias.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF011 - Permitir deleção de bactérias

O aplicativo deve possuir um botão que permita a deleção das bactérias registradas.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF012 - Possuir cadastro de pacientes

O aplicativo deve possuir uma tela que permita o cadastro de pacientes, além de listar, apenas pelo nome, os pacientes já cadastrados.

O cadastro dos pacientes deve conter os seguintes campos:

- Usuário (usuários com cadastro no app)
- Mensagem de alerta (permitir letras, números e sinais)

- Urgência (apenas verdadeiro ou falso)
- Registro (permitir letras, números e sinais)
- Nome do paciente (permitir letras, números e sinais)
- Gênero (apenas masculino ou feminino)
- Data de nascimento (formato padrão de data [dd/mm/aaaa])
- Idade (apenas caracteres numéricos)
- Procedência do paciente (permitir letras, números e sinais)
- Contato (permitir letras, números e sinais)
- Endereço (permitir letras, números e sinais)
- Adicionar responsável (apenas verdadeiro ou falso)
- Nome do responsável (visível apenas se "adicionar responsável" for verdadeiro) (permitir letras, números e sinais)
- Contato do responsável (visível apenas se "adicionar responsável" for verdadeiro) (permitir letras, números e sinais)
- Nome do solicitante (permitir letras, números e sinais)
- Previsão de entrega (formato padrão de data [dd/mm/aaaa])
- Data da coleta (formato padrão de data [dd/mm/aaaa])
- Exames solicitados (permitir letras, números e sinais)
- Informações adicionais (permitir letras, números e sinais)

Restrição de usuário: Sim, disponível apenas para usuários "lab".

RF013 - Permitir visualização de pacientes

Na tela de pacientes, deve ser possível visualizar todos os campos cadastrados de um paciente ao clicar em seu nome.

Restrição de usuário: Sim, disponível apenas para usuários "lab".

RF014 - Permitir edição de pacientes

O aplicativo deve possuir um botão que permita a edição de todos os campos registrados no cadastro dos pacientes.

Restrição de usuário: Sim, disponível apenas para usuários "lab".

RF015 - Permitir deleção de pacientes

O aplicativo deve possuir um botão que permita a deleção dos pacientes registrados.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF016 - Possuir cadastro de pedidos

O aplicativo deve possuir uma tela que permita o cadastro de pedidos de antibiogramas, além de listar, apenas pelo número do pedido, os pedidos já cadastrados. O cadastro de um novo pedido deve gerar uma notificação no aplicativo do usuário que foi cadastrado.

O cadastro dos pedidos deve conter os seguintes campos:

- Usuário (permitir apenas usuários com cadastro no app)
- Paciente (permitir apenas pacientes cadastrados na tela de pacientes)
- Pedido (permitir letras, números e sinais)
- Material (permitir letras, números e sinais)
- Reagente (permitir letras, números e sinais)
- Bactéria (permitir apenas bactérias cadastradas na tela de bactérias)
- Antibióticos (mostrar todos os antibióticos cadastrados na tela antibióticos) (cada antibiótico)

O campo de antibióticos deve possuir três colunas de registro: Na primeira coluna deve-se registrar os MIC (permitir número, vírgula e ponto), a segunda deve possibilitar o registro de apenas um dos sinais “=”, “<=” ou “>=” e a terceira deve permitir o registro de apenas uma das seguintes letras: “R”, “I” e “S”.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF017 - Permitir visualização de pedidos

Deve ser possível visualizar todos os campos cadastrados de um pedido, além de mostrar os dados do paciente daquele pedido. Tudo isso, ao clicar no seu número de pedido.

O campo de antibióticos deve ser uma tabela com três colunas com os seguintes nomes: Antibiótico, MIC pedido e MIC bactéria. Na coluna “antibióticos”, devem ser mostrados apenas os antibióticos que tiveram os valores de MIC registrados no cadastro ou na edição de pedidos, no campo de antibióticos. Na coluna “MIC pedido”, devem constar o valor da MIC, um dos sinais “=”, “>=” e “<=” e uma das letras “R”, “S” e “I”, registrados no cadastro de pedido, no campo de antibióticos. Na última coluna, deve constar as MIC dos antibióticos para a bactéria daquele pedido,

registrados no cadastro de antibióticos.

Caso o valor do MIC na coluna “MIC pedido” seja maior que o valor do MIC na coluna “MIC bactéria”, a linha referente àquele antibiótico deve ser colorida de vermelho. Caso contrário, a linha deve ser colorida de verde. Em caso de não haver cadastro de “MIC bactéria” para um determinado antibiótico, a linha referente a este não deve ser colorida, e deve ser mostrada a seguinte sigla na coluna “MIC bactéria”: “*SRC”.

Restrição de usuário: nenhuma, disponível para os dois tipos de usuário.

RF018 - Permitir edição de pedidos

O aplicativo deve possuir um botão que permita a edição de todos os campos registrados no cadastro dos pedidos.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF019 - Permitir deleção de pedidos

O aplicativo deve possuir um botão que permita a deleção dos pedidos registrados.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF020 - Possuir cadastro de alerta

O aplicativo deve possuir uma tela que permita o cadastro de alertas, além de listar, apenas pelo email de destino, os alertas já cadastrados.

O cadastro dos alertas deve conter os seguintes campos:

- Destinatário (permitir apenas email de usuários cadastrados no app)
- Mensagem (permitir letras, números e sinais)

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF021 - Permitir visualização de alerta

Deve ser possível visualizar todos os campos cadastrados de um alerta, além de mostrar a data de criação deste. Tudo isso, ao clicar no email do usuário a quem ele foi destinado.

Restrição de usuário: nenhuma, disponível para os dois tipos de usuário.

RF022 - Permitir edição de alerta

O aplicativo deve possuir um botão que permita a edição de todos os campos registrados no cadastro de alertas.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

RF023 - Permitir deleção de alerta

O aplicativo deve possuir um botão que permita a deleção dos alertas registrados.

Restrição de usuário: Sim, disponível apenas para usuários “lab”.

3. Requisitos não-funcionais

RNF001 - O aplicativo deve ser multiplataforma

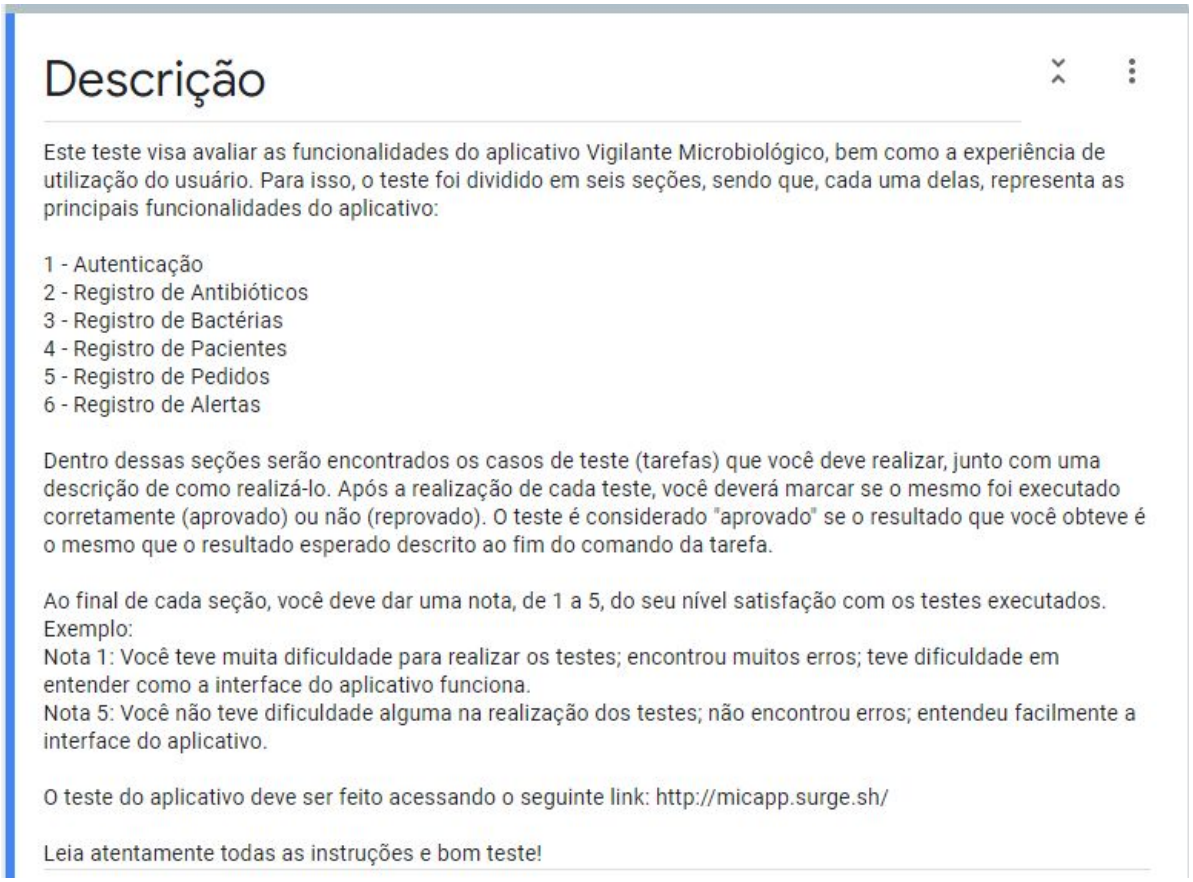
O aplicativo deve funcionar tanto no sistema operacional Android quanto em versão WEB, através de um Browser.

RNF002 - Devem existir dois tipos de usuário

O aplicativo deve possuir dois tipos de usuário: **“lab”** e **“médico”**; cada um com uma restrição de acesso específica. As restrições de acesso estão definidas nos requisitos funcionais.

APÊNDICE B - Formulário do teste de aceitação de usuário

Descrição do teste de aceitação de usuário (UAT)



Descrição

Este teste visa avaliar as funcionalidades do aplicativo Vigilante Microbiológico, bem como a experiência de utilização do usuário. Para isso, o teste foi dividido em seis seções, sendo que, cada uma delas, representa as principais funcionalidades do aplicativo:

1 - Autenticação
2 - Registro de Antibióticos
3 - Registro de Bactérias
4 - Registro de Pacientes
5 - Registro de Pedidos
6 - Registro de Alertas

Dentro dessas seções serão encontrados os casos de teste (tarefas) que você deve realizar, junto com uma descrição de como realizá-lo. Após a realização de cada teste, você deverá marcar se o mesmo foi executado corretamente (aprovado) ou não (reprovado). O teste é considerado "aprovado" se o resultado que você obteve é o mesmo que o resultado esperado descrito ao fim do comando da tarefa.

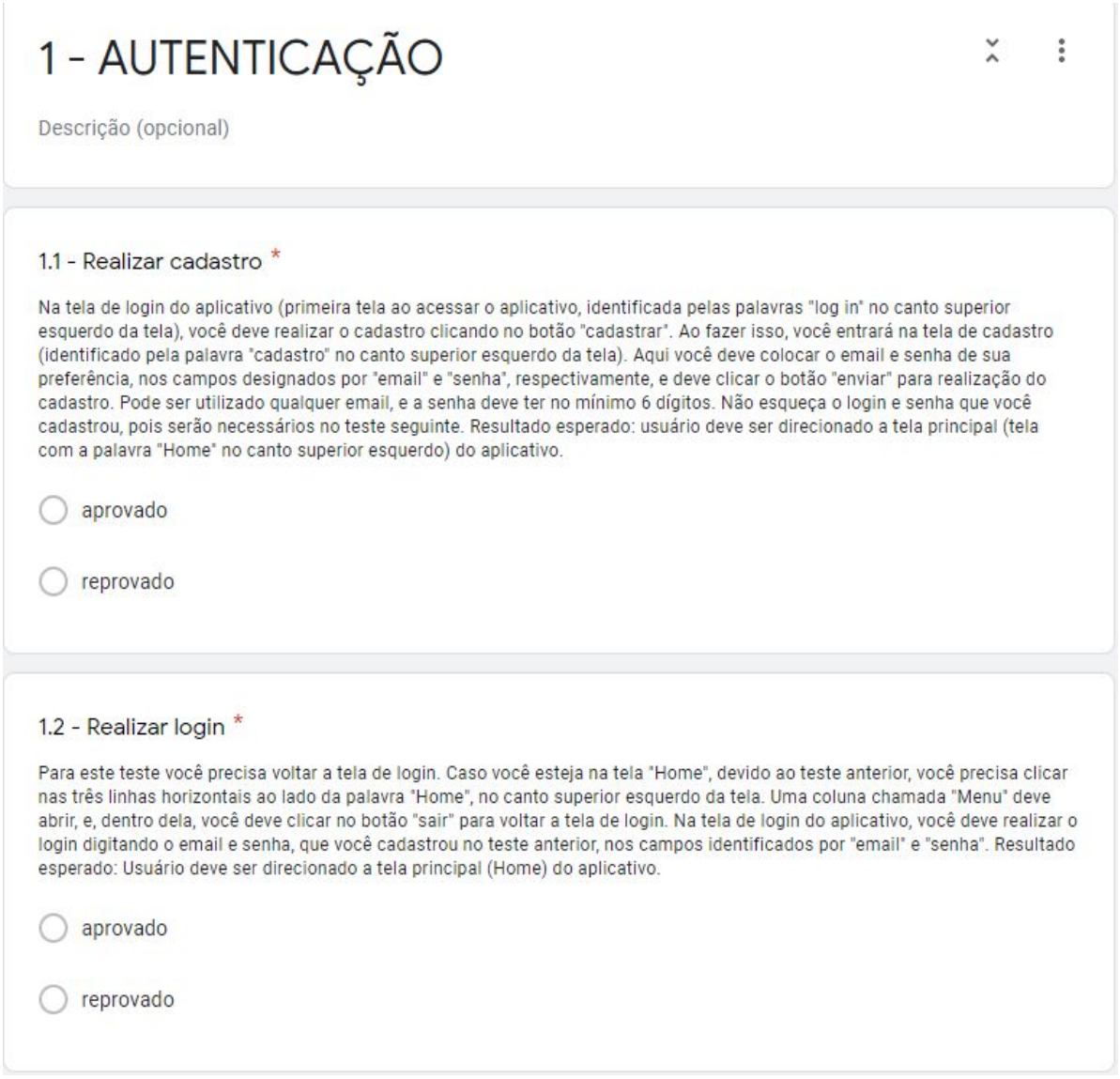
Ao final de cada seção, você deve dar uma nota, de 1 a 5, do seu nível satisfação com os testes executados.
Exemplo:
Nota 1: Você teve muita dificuldade para realizar os testes; encontrou muitos erros; teve dificuldade em entender como a interface do aplicativo funciona.
Nota 5: Você não teve dificuldade alguma na realização dos testes; não encontrou erros; entendeu facilmente a interface do aplicativo.

O teste do aplicativo deve ser feito acessando o seguinte link: http://micapp.surge.sh/

Leia atentamente todas as instruções e bom teste!

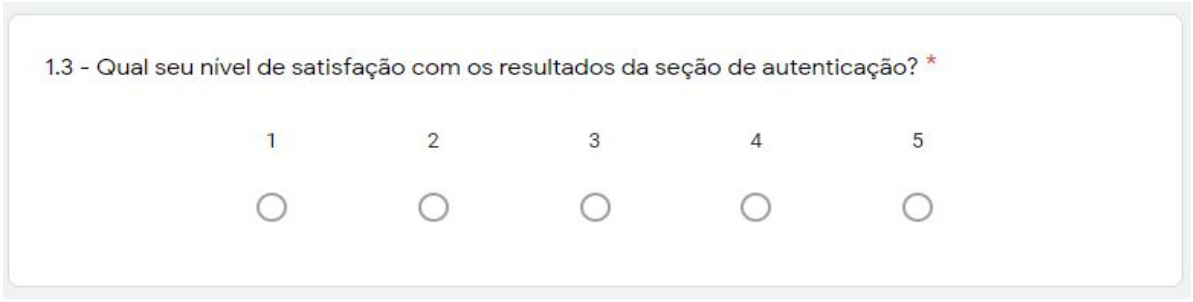
Fonte: Próprio autor.

Seção de autenticação do UAT



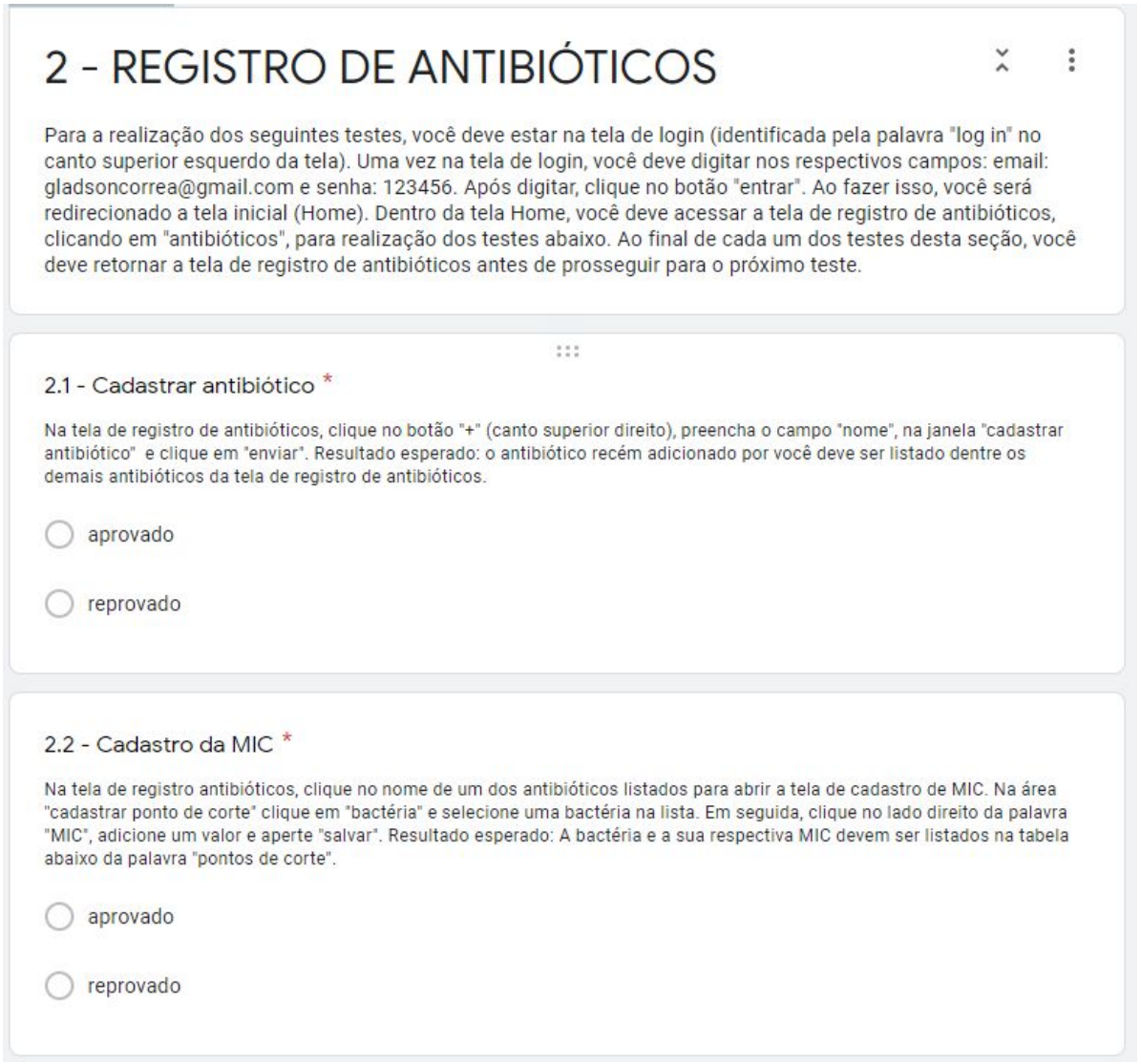
Fonte: Próprio autor.

Continuação da seção de autenticação do UAT



Fonte: Próprio autor.

Seção de antibióticos do UAT



2 - REGISTRO DE ANTIBIÓTICOS

Para a realização dos seguintes testes, você deve estar na tela de login (identificada pela palavra "log in" no canto superior esquerdo da tela). Uma vez na tela de login, você deve digitar nos respectivos campos: email: gladsoncorrea@gmail.com e senha: 123456. Após digitar, clique no botão "entrar". Ao fazer isso, você será redirecionado a tela inicial (Home). Dentro da tela Home, você deve acessar a tela de registro de antibióticos, clicando em "antibióticos", para realização dos testes abaixo. Ao final de cada um dos testes desta seção, você deve retornar a tela de registro de antibióticos antes de prosseguir para o próximo teste.

2.1 - Cadastrar antibiótico *

Na tela de registro de antibióticos, clique no botão "+" (canto superior direito), preencha o campo "nome", na janela "cadastrar antibiótico" e clique em "enviar". Resultado esperado: o antibiótico recém adicionado por você deve ser listado dentre os demais antibióticos da tela de registro de antibióticos.

aprovado

reprovado

2.2 - Cadastro da MIC *

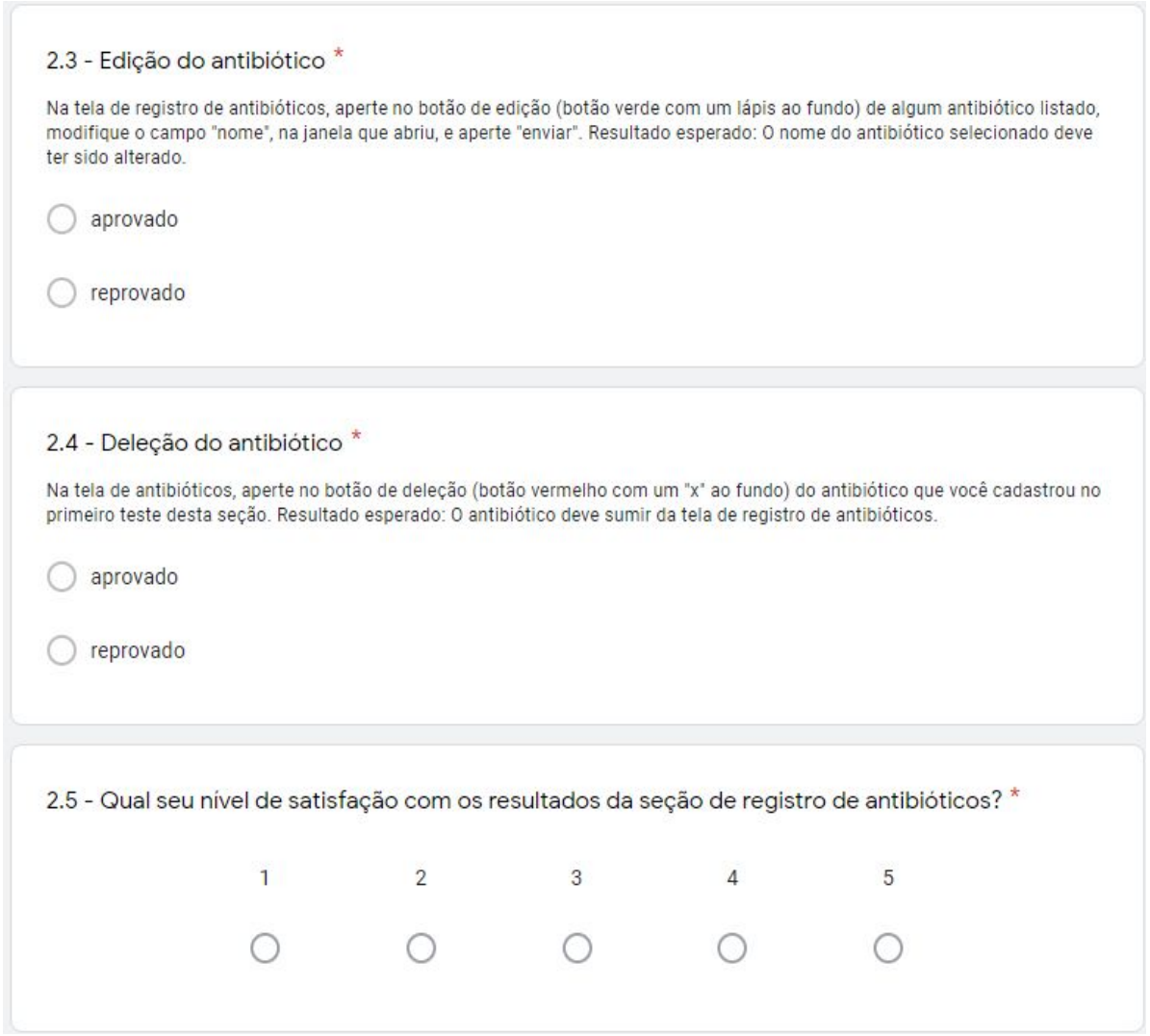
Na tela de registro antibióticos, clique no nome de um dos antibióticos listados para abrir a tela de cadastro de MIC. Na área "cadastrar ponto de corte" clique em "bactéria" e selecione uma bactéria na lista. Em seguida, clique no lado direito da palavra "MIC", adicione um valor e aperte "salvar". Resultado esperado: A bactéria e a sua respectiva MIC devem ser listados na tabela abaixo da palavra "pontos de corte".

aprovado

reprovado

Fonte: Próprio autor.

Continuação da seção de antibióticos do UAT



2.3 - Edição do antibiótico *

Na tela de registro de antibióticos, aperte no botão de edição (botão verde com um lápis ao fundo) de algum antibiótico listado, modifique o campo "nome", na janela que abriu, e aperte "enviar". Resultado esperado: O nome do antibiótico selecionado deve ter sido alterado.

aprovado

reprovado

2.4 - Deleção do antibiótico *

Na tela de antibióticos, aperte no botão de deleção (botão vermelho com um "x" ao fundo) do antibiótico que você cadastrou no primeiro teste desta seção. Resultado esperado: O antibiótico deve sumir da tela de registro de antibióticos.

aprovado

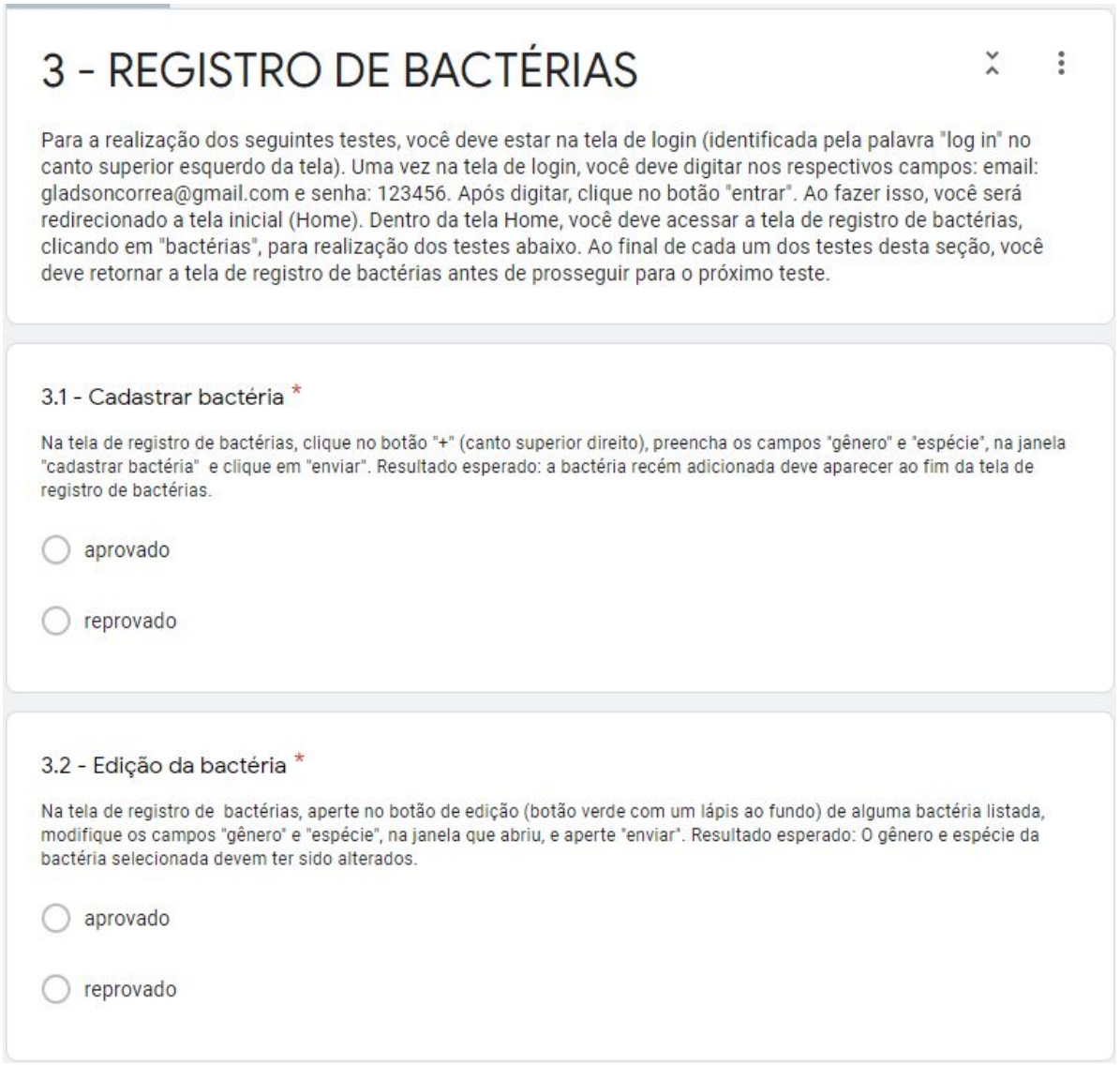
reprovado

2.5 - Qual seu nível de satisfação com os resultados da seção de registro de antibióticos? *

1 2 3 4 5

Fonte: Próprio autor.

Seção de bactérias do UAT



3 - REGISTRO DE BACTÉRIAS

Para a realização dos seguintes testes, você deve estar na tela de login (identificada pela palavra "log in" no canto superior esquerdo da tela). Uma vez na tela de login, você deve digitar nos respectivos campos: email: gladsoncorrea@gmail.com e senha: 123456. Após digitar, clique no botão "entrar". Ao fazer isso, você será redirecionado a tela inicial (Home). Dentro da tela Home, você deve acessar a tela de registro de bactérias, clicando em "bactérias", para realização dos testes abaixo. Ao final de cada um dos testes desta seção, você deve retornar a tela de registro de bactérias antes de prosseguir para o próximo teste.

3.1 - Cadastrar bactéria *

Na tela de registro de bactérias, clique no botão "+" (canto superior direito), preencha os campos "gênero" e "espécie", na janela "cadastrar bactéria" e clique em "enviar". Resultado esperado: a bactéria recém adicionada deve aparecer ao fim da tela de registro de bactérias.

aprovado

reprovado

3.2 - Edição da bactéria *

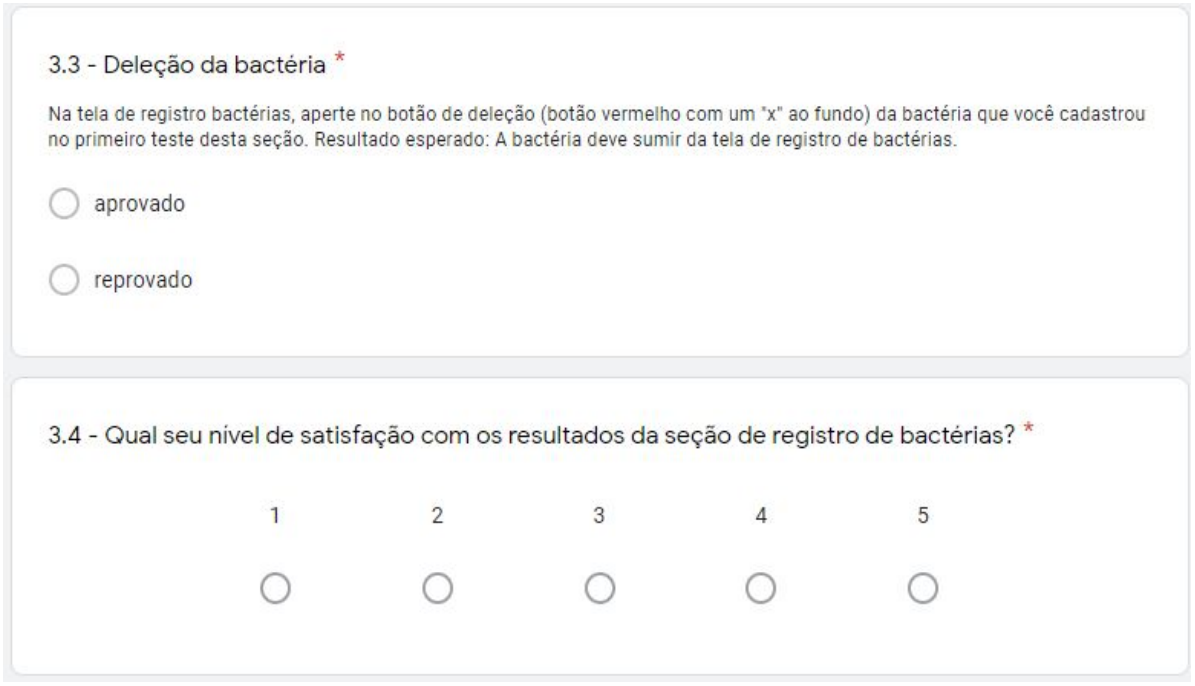
Na tela de registro de bactérias, aperte no botão de edição (botão verde com um lápis ao fundo) de alguma bactéria listada, modifique os campos "gênero" e "espécie", na janela que abriu, e aperte "enviar". Resultado esperado: O gênero e espécie da bactéria selecionada devem ter sido alterados.

aprovado

reprovado

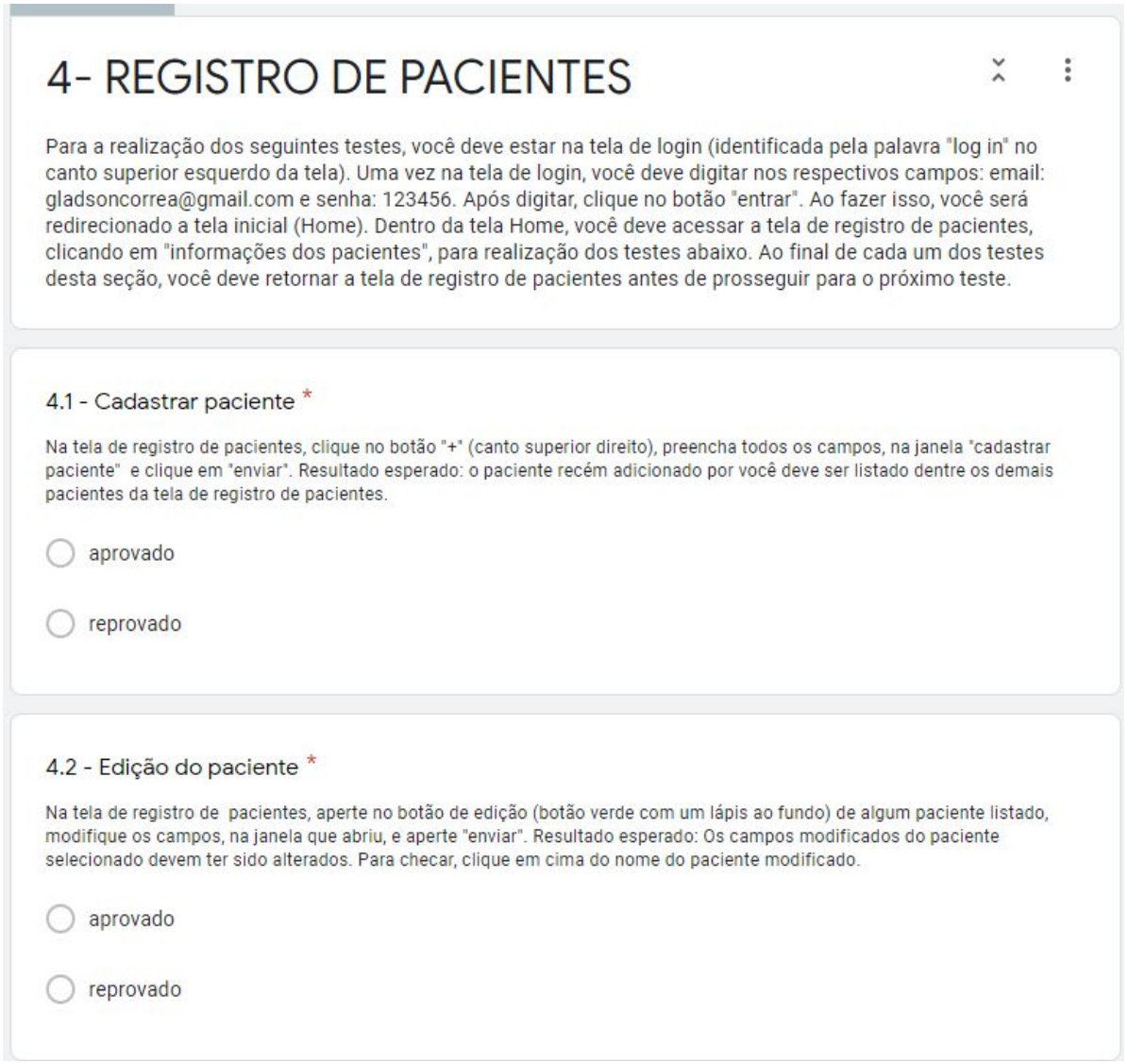
Fonte: Próprio autor.

Continuação da seção de bactérias do UAT



Fonte: Próprio autor.

Seção de pacientes do UAT



4- REGISTRO DE PACIENTES

Para a realização dos seguintes testes, você deve estar na tela de login (identificada pela palavra "log in" no canto superior esquerdo da tela). Uma vez na tela de login, você deve digitar nos respectivos campos: email: gladsoncorrea@gmail.com e senha: 123456. Após digitar, clique no botão "entrar". Ao fazer isso, você será redirecionado a tela inicial (Home). Dentro da tela Home, você deve acessar a tela de registro de pacientes, clicando em "informações dos pacientes", para realização dos testes abaixo. Ao final de cada um dos testes desta seção, você deve retornar a tela de registro de pacientes antes de prosseguir para o próximo teste.

4.1 - Cadastrar paciente *

Na tela de registro de pacientes, clique no botão "+" (canto superior direito), preencha todos os campos, na janela "cadastrar paciente" e clique em "enviar". Resultado esperado: o paciente recém adicionado por você deve ser listado dentre os demais pacientes da tela de registro de pacientes.

aprovado

reprovado

4.2 - Edição do paciente *

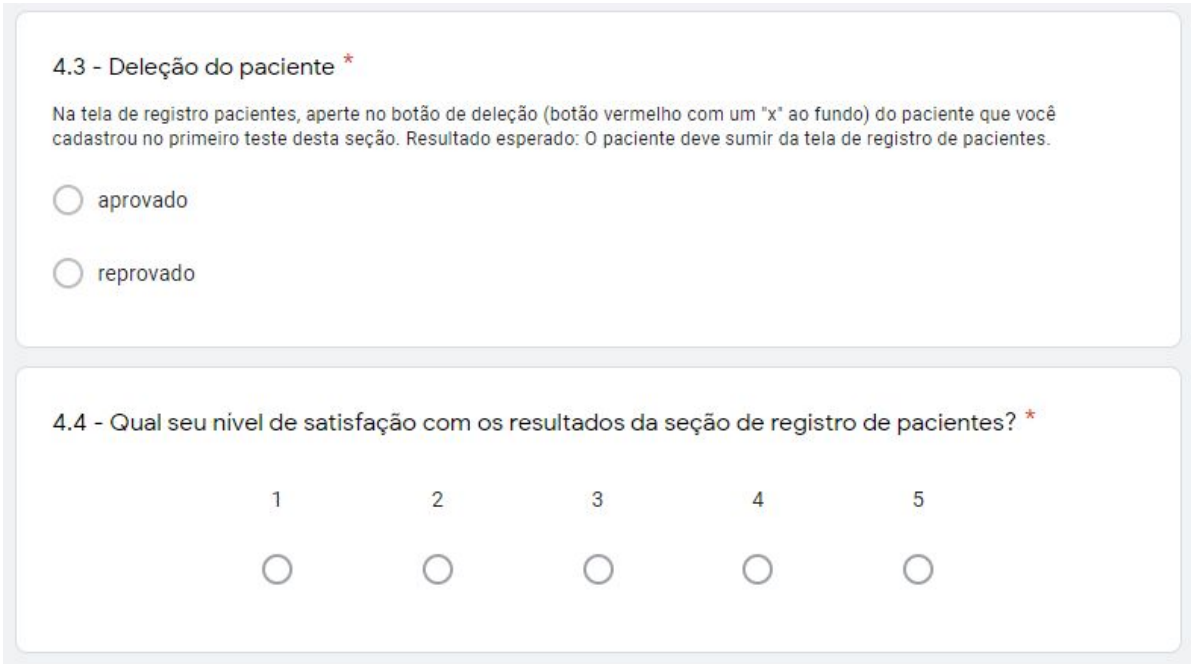
Na tela de registro de pacientes, aperte no botão de edição (botão verde com um lápis ao fundo) de algum paciente listado, modifique os campos, na janela que abriu, e aperte "enviar". Resultado esperado: Os campos modificados do paciente selecionado devem ter sido alterados. Para checar, clique em cima do nome do paciente modificado.

aprovado

reprovado

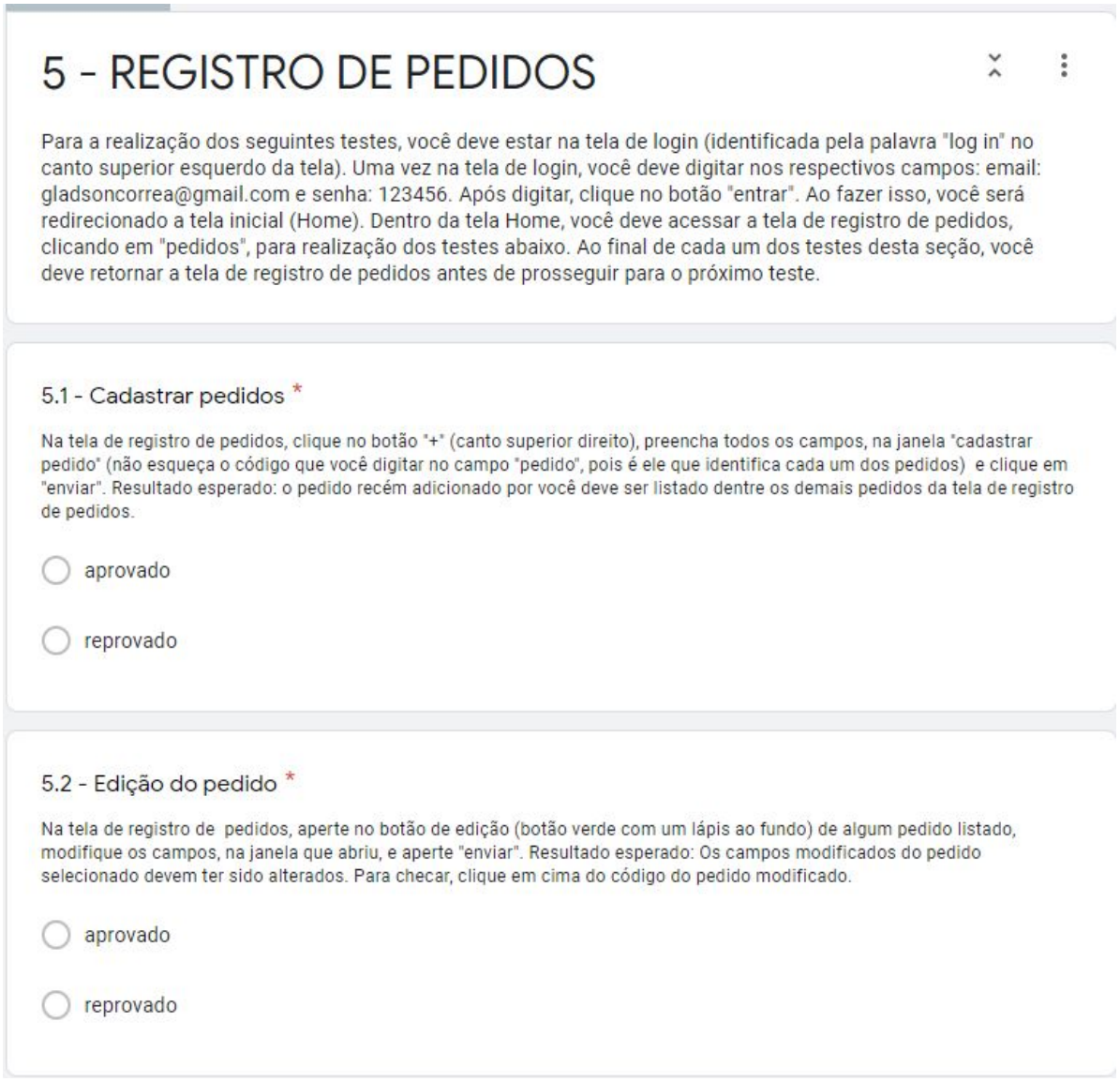
Fonte: Próprio autor.

Continuação da seção de pacientes do UAT



Fonte: Próprio autor.

Seção de pedidos do UAT



5 - REGISTRO DE PEDIDOS

Para a realização dos seguintes testes, você deve estar na tela de login (identificada pela palavra "log in" no canto superior esquerdo da tela). Uma vez na tela de login, você deve digitar nos respectivos campos: email: gladsoncorrea@gmail.com e senha: 123456. Após digitar, clique no botão "entrar". Ao fazer isso, você será redirecionado a tela inicial (Home). Dentro da tela Home, você deve acessar a tela de registro de pedidos, clicando em "pedidos", para realização dos testes abaixo. Ao final de cada um dos testes desta seção, você deve retornar a tela de registro de pedidos antes de prosseguir para o próximo teste.

5.1 - Cadastrar pedidos *

Na tela de registro de pedidos, clique no botão "+" (canto superior direito), preencha todos os campos, na janela "cadastrar pedido" (não esqueça o código que você digitar no campo "pedido", pois é ele que identifica cada um dos pedidos) e clique em "enviar". Resultado esperado: o pedido recém adicionado por você deve ser listado dentre os demais pedidos da tela de registro de pedidos.

aprovado

reprovado

5.2 - Edição do pedido *

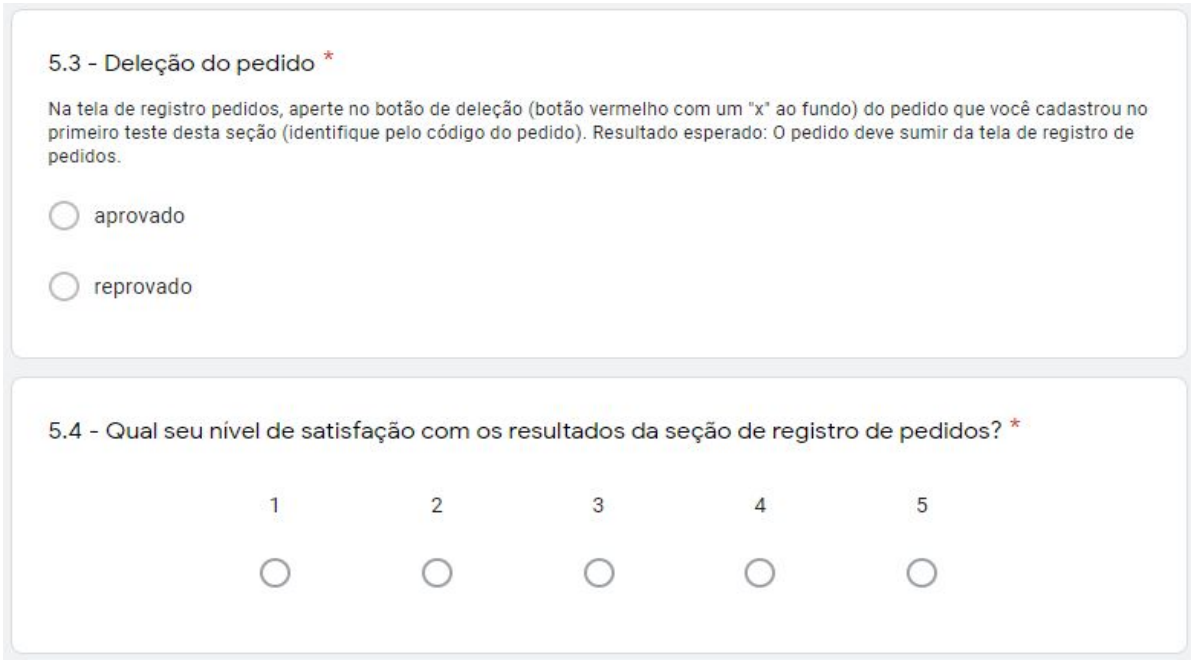
Na tela de registro de pedidos, aperte no botão de edição (botão verde com um lápis ao fundo) de algum pedido listado, modifique os campos, na janela que abriu, e aperte "enviar". Resultado esperado: Os campos modificados do pedido selecionado devem ter sido alterados. Para checar, clique em cima do código do pedido modificado.

aprovado

reprovado

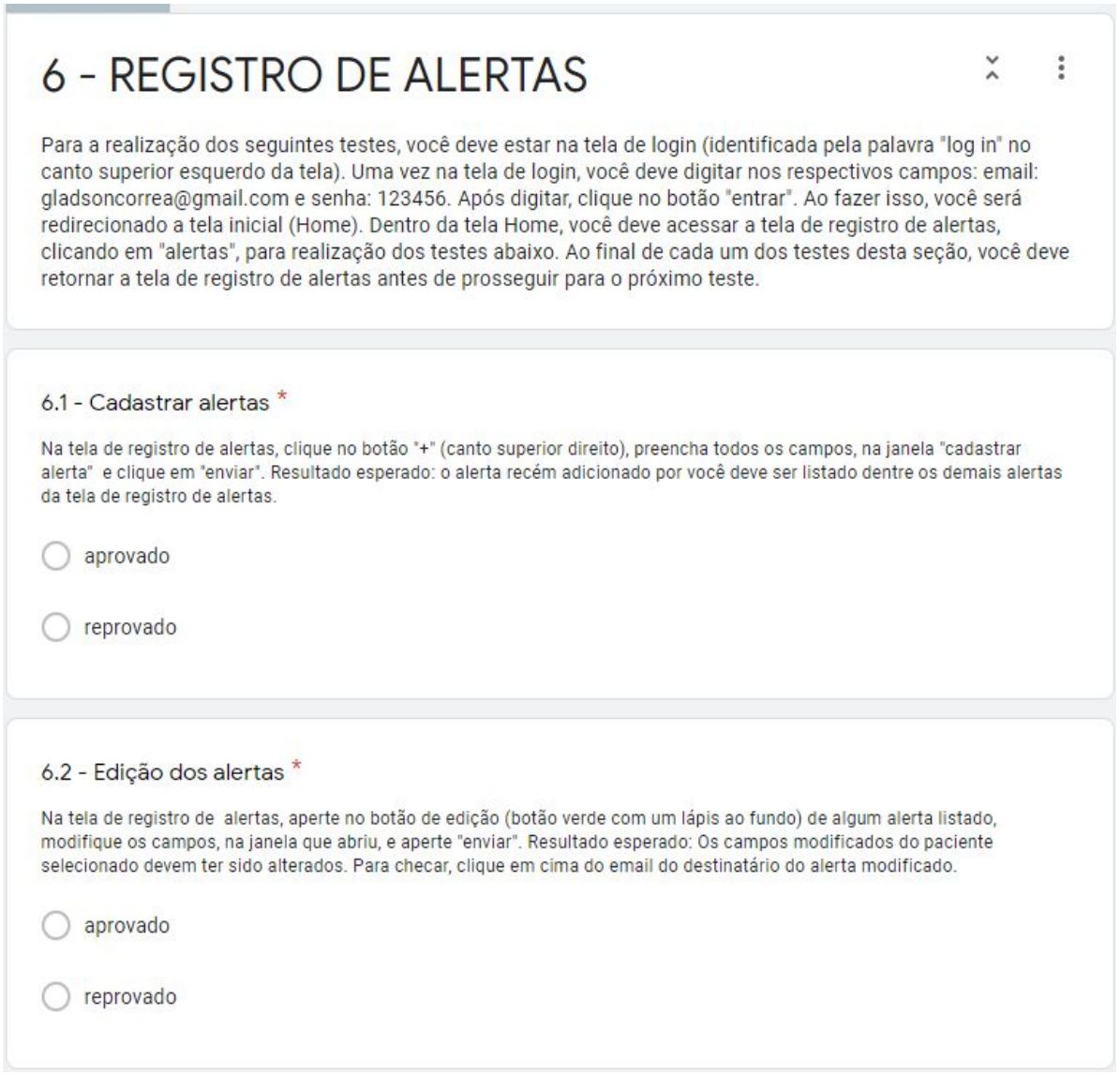
Fonte: Próprio autor.

Continuação da seção de pedidos do UAT



Fonte: Próprio autor.

Seção de alertas do UAT



6 - REGISTRO DE ALERTAS

Para a realização dos seguintes testes, você deve estar na tela de login (identificada pela palavra "log in" no canto superior esquerdo da tela). Uma vez na tela de login, você deve digitar nos respectivos campos: email: gladsoncorrea@gmail.com e senha: 123456. Após digitar, clique no botão "entrar". Ao fazer isso, você será redirecionado a tela inicial (Home). Dentro da tela Home, você deve acessar a tela de registro de alertas, clicando em "alertas", para realização dos testes abaixo. Ao final de cada um dos testes desta seção, você deve retornar a tela de registro de alertas antes de prosseguir para o próximo teste.

6.1 - Cadastrar alertas *

Na tela de registro de alertas, clique no botão "+" (canto superior direito), preencha todos os campos, na janela "cadastrar alerta" e clique em "enviar". Resultado esperado: o alerta recém adicionado por você deve ser listado dentre os demais alertas da tela de registro de alertas.

aprovado

reprovado

6.2 - Edição dos alertas *

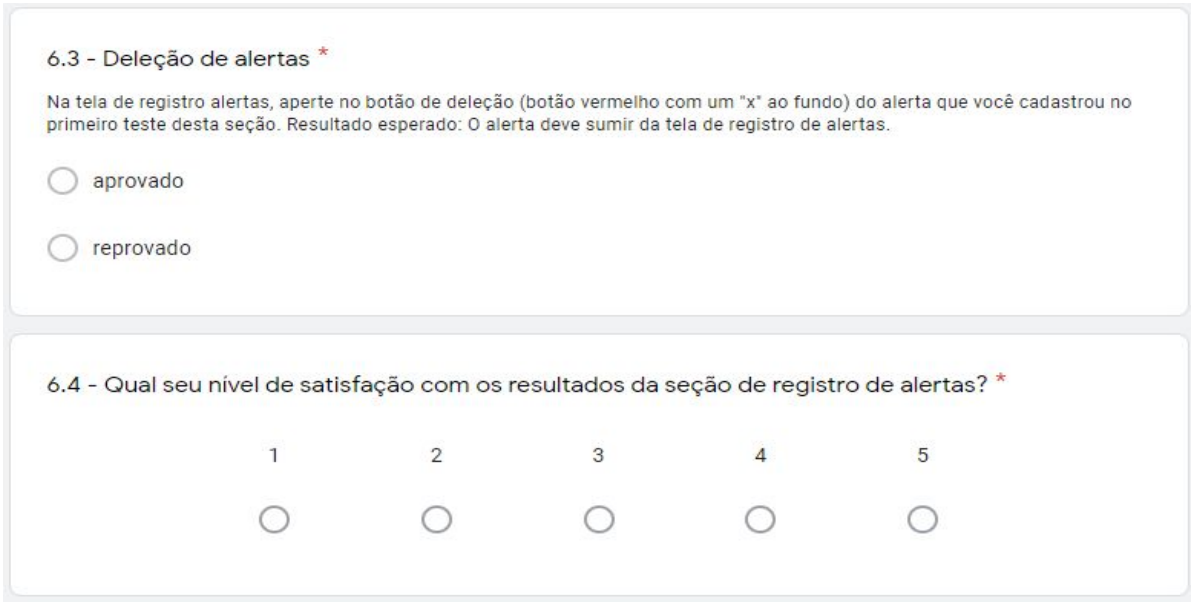
Na tela de registro de alertas, aperte no botão de edição (botão verde com um lápis ao fundo) de algum alerta listado, modifique os campos, na janela que abriu, e aperte "enviar". Resultado esperado: Os campos modificados do paciente selecionado devem ter sido alterados. Para checar, clique em cima do email do destinatário do alerta modificado.

aprovado

reprovado

Fonte: Próprio autor.

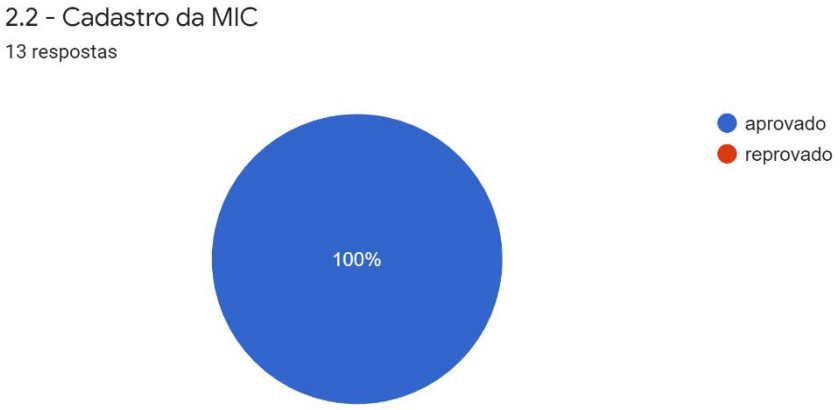
Continuação da seção de alertas do UAT



Fonte: Próprio autor.

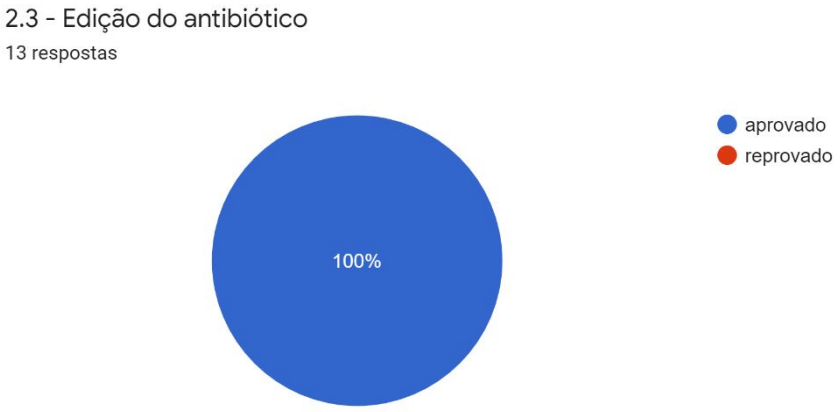
APÊNDICE C - Resultados do teste de aceitação de usuário (UAT)

Figura 1 - Porcentagem de aprovação do caso de teste Cadastro da MIC



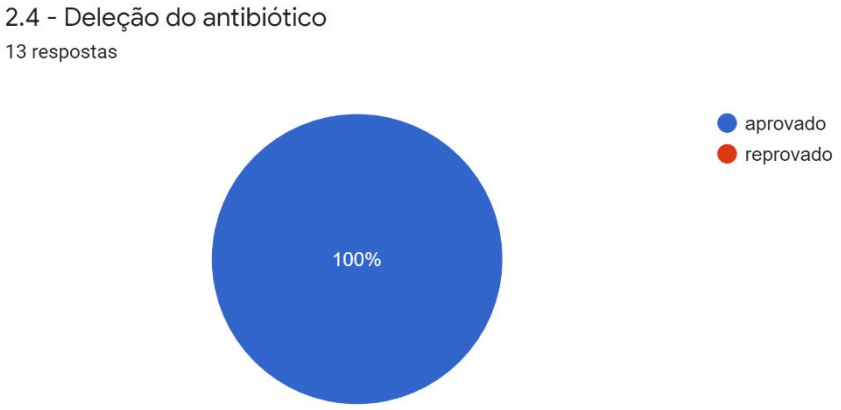
Fonte: Próprio autor.

Figura 2 - Porcentagem de aprovação do caso de teste Edição do Antibiótico



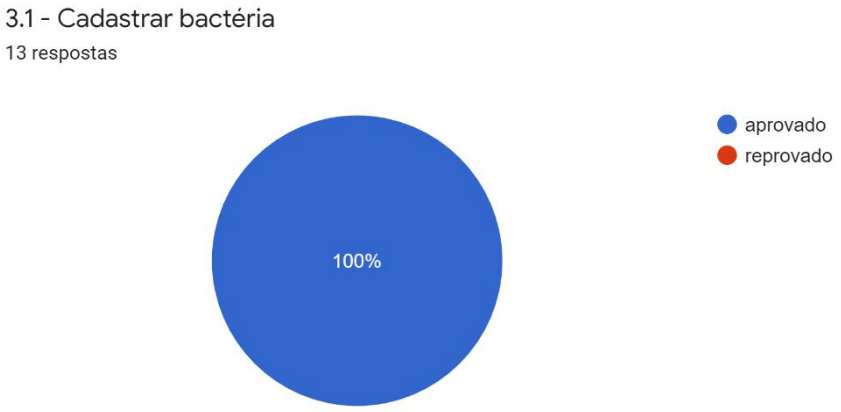
Fonte: Próprio autor.

Figura 3 - Porcentagem de aprovação do caso de teste Deleção do Antibiótico



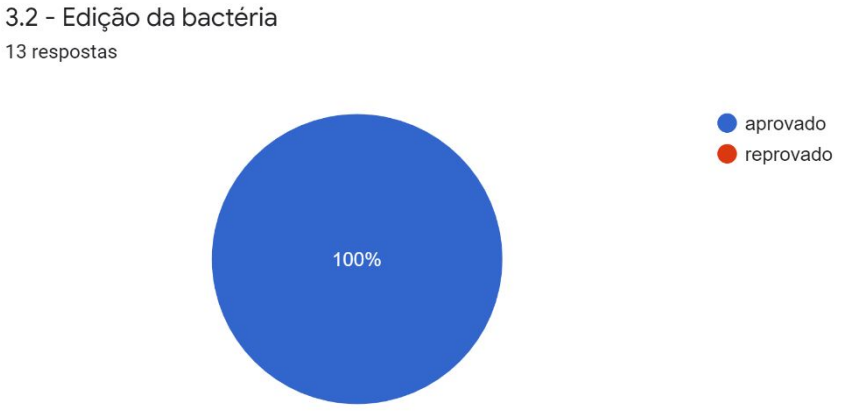
Fonte: Próprio autor.

Figura 4 - Porcentagem de aprovação do caso de teste Cadastrar Bactéria



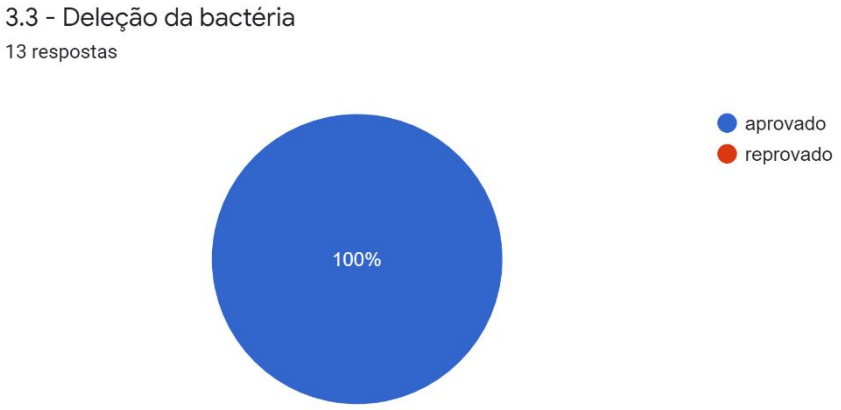
Fonte: Próprio autor.

Figura 5 - Porcentagem de aprovação do caso de teste Edição Bactéria



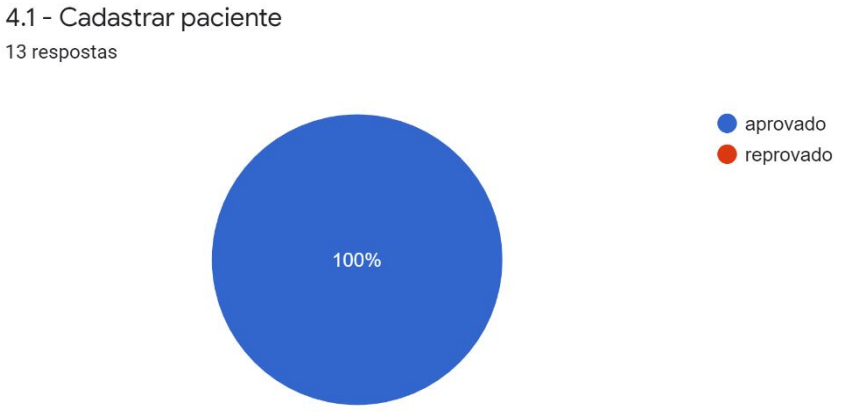
Fonte: Próprio autor.

Figura 6 - Porcentagem de aprovação do caso de teste Deleção da Bactéria



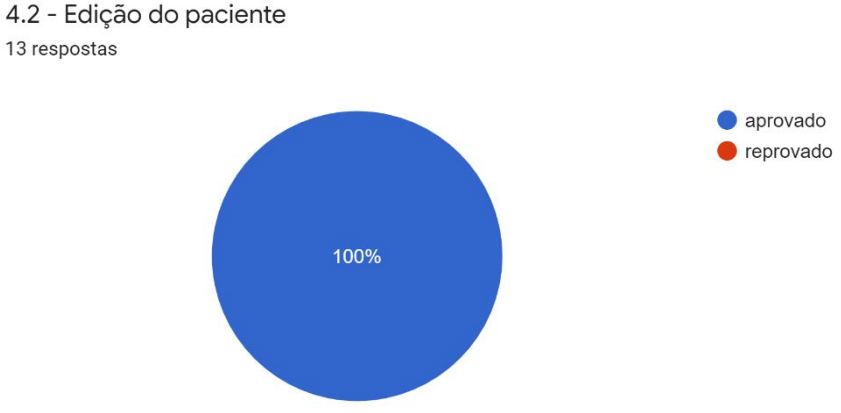
Fonte: Próprio autor.

Figura 7 - Porcentagem de aprovação do caso de teste Cadastrar Paciente



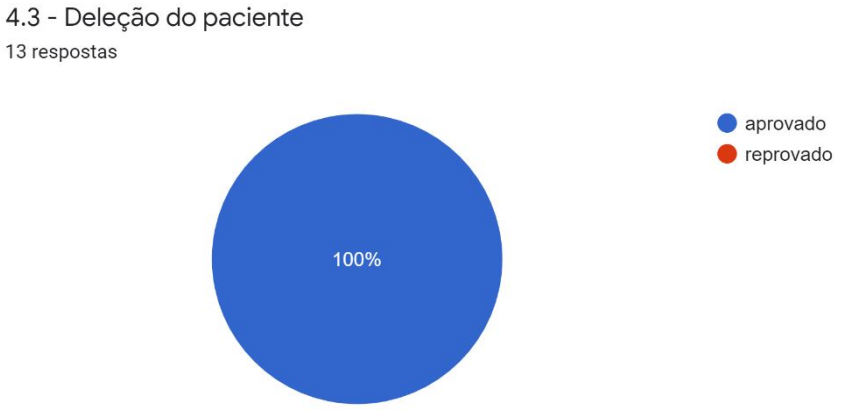
Fonte: Próprio autor.

Figura 8 - Porcentagem de aprovação do caso de teste Edição do Paciente



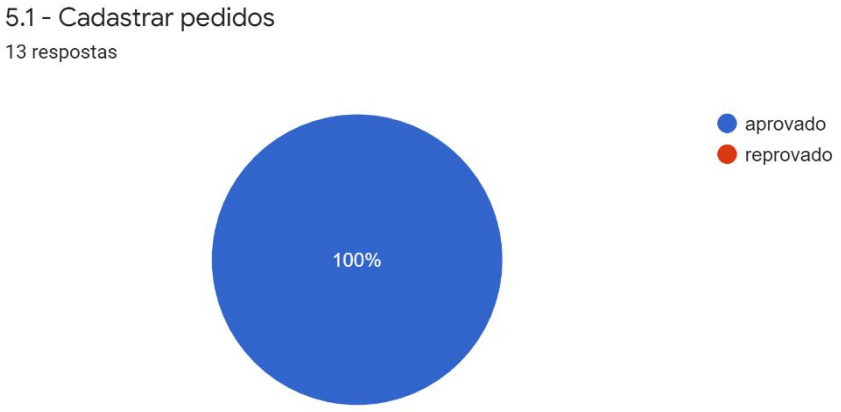
Fonte: Próprio autor.

Figura 9 - Porcentagem de aprovação do caso de teste Deleção do Paciente



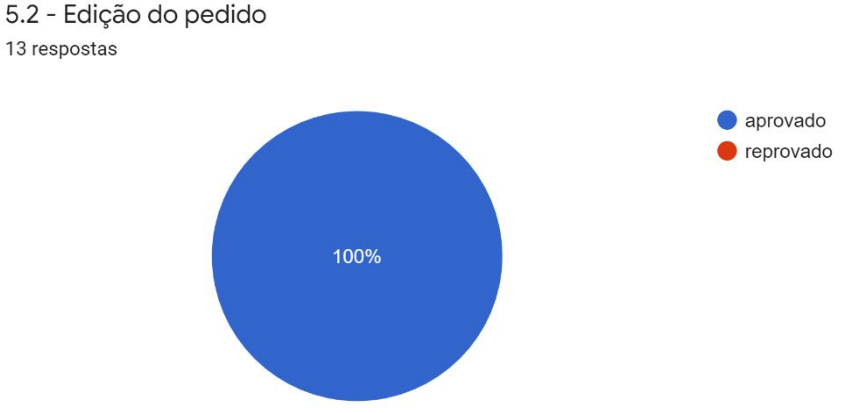
Fonte: Próprio autor.

Figura 10 - Porcentagem de aprovação do caso de teste Cadastrar Pedidos



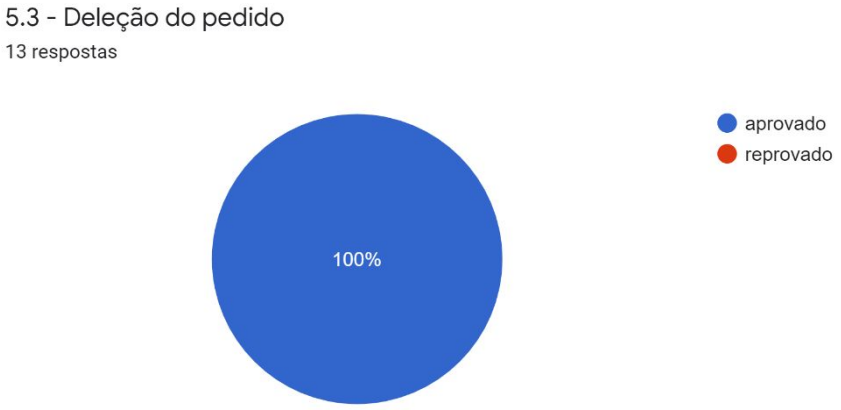
Fonte: Próprio autor.

Figura 11 - Porcentagem de aprovação do caso de teste Edição do Pedido



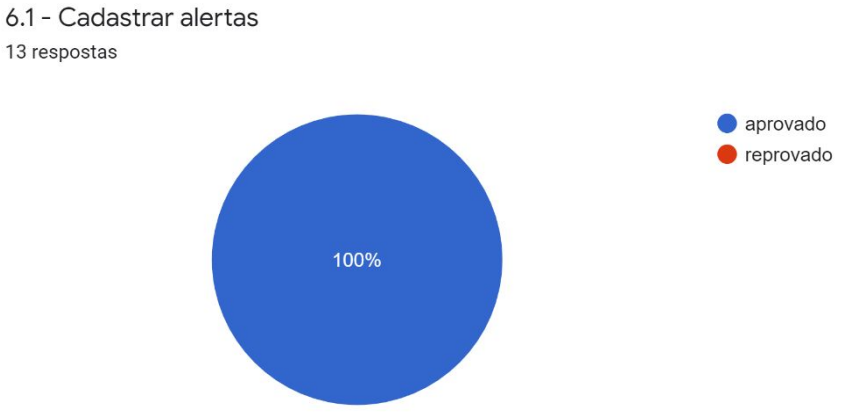
Fonte: Próprio autor.

Figura 12 - Porcentagem de aprovação do caso de teste Deleção do Pedido



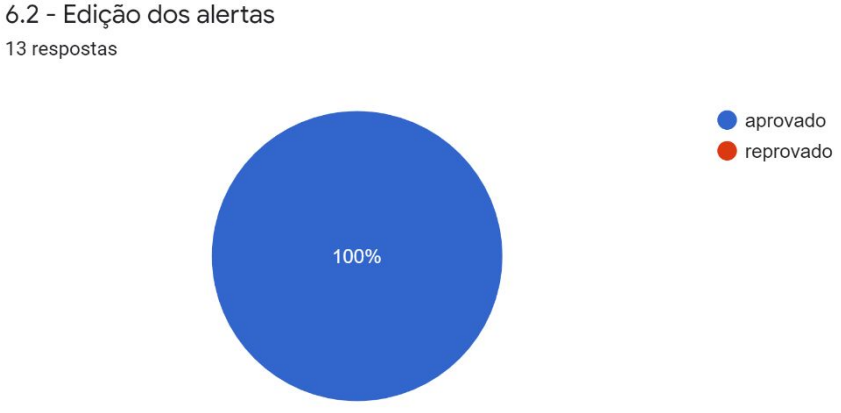
Fonte: Próprio autor.

Figura 13 - Porcentagem de aprovação do caso de teste Cadastrar Alertas



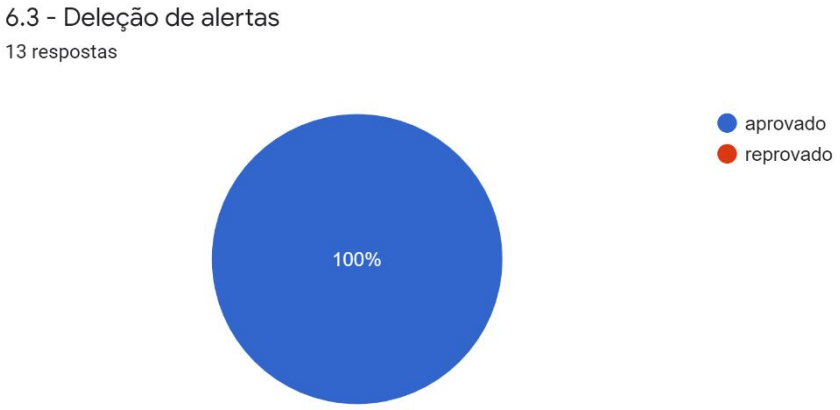
Fonte: Próprio autor.

Figura 14 - Porcentagem de aprovação do caso de teste Edição dos Alertas



Fonte: Próprio autor.

Figura 15 - Porcentagem de aprovação do caso de teste Deleção de Alertas



Fonte: Próprio autor.

ANEXO A – Termo de consentimento para a participação no teste de aceitação de usuário do aplicativo Vigilante Microbiológico

TERMO DE CONSENTIMENTO

Declaro concordar em fornecer os dados solicitados abaixo com o intuito de testar o aplicativo denominado MIC APP (Aplicativo de microbiologia / Vigilante microbiológico), desenvolvido por RAIMUNDO GLADSON CORREA CARVALHO.

NOME	E-MAIL	CONTATO	RG	ASSINATURA
Luciana Calandrine Gonçalves	lucalandrinegon@gmail.com	(91) 98093-4055	4861008	
Josielem Mendes	josielemmendes1@gmail.com	(91) 98128-5959	5319138	
Fernanda Fernandes	fernandamodestofernandes@gmail.com	(91) 98812-1195		
Paula Vanessa Reis	Paulavreis29@gmail.com	(91) 98237-1949		
Nelma Siqueira	nelmaassis@gmail.com	(91) 99128-7188		
Raquel Monteiro	Raquelmaraujo86@yahoo.com.br	(91) 99104-7271		
Mirtys Silva	myrtheskayo@gmail.com	(91) 98443-6247		
Rodrigo Pablo Barbosa	Rodrigopablo23@hotmail.com	(91) 98045-4537		
Ana Paula Paixão	Apsp17@gmail.com	(91) 98325-3542		
Erissandra Cabral	Eriss_ferreira@hotmail.com	(91) 99215-7710		
Ilanna Marques	Marquesilanna28@gmail.com	(91) 98123-3249		
Jessyca Lisboa	Jessycalisboa13@gmail.com	(91) 99363-9964		
Rosenira Torres	Nira_torres@hotmail.com	(91) 98234-8050		
Luciann Tavares	luciann@pauloazevedo.com.br	(91) 99349-1587		
Rhaisa Frota	Rhaisa.danielly@gmail.com	(91) 98150-6187		
Adria Jaqueline	Adriajaqueline2013@gmail.com	(91) 98959-9285		
Starley Matos	Starley_matos@hotmail.com	(91) 99189-4275		
Gessica Valeria Rocha	gessicavferro@gmail.com	(91) 99977-4238		
Fabricio Ribeiro	Ribeirofa2013@gmail.com	(91) 98138-3534		
Lacy Brito	Lcdbrito2@gmail.com	(91) 98891-6729		
Silvana Martins	Silvana.martins@pauloazevedo.com.br	(91) 99167-8906		
Éliri Nayelle Vasconcelos Soares	elrisoares@yahoo.com.br	(91) 98994-5739	7016062	

ANEXO B - Telas do aplicativo Vigilante Microbiológico (MICAPP)

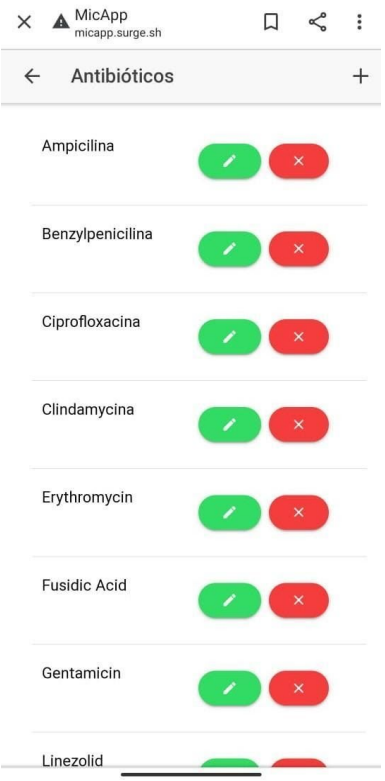
Tela de autenticação

Fonte: MICAPP (2020).

Tela Inicial

Fonte: MICAPP (2020).

Tela de antibióticos



Fonte: MICAPP (2020).

Tela de bactérias

Fonte: MICAPP (2020).

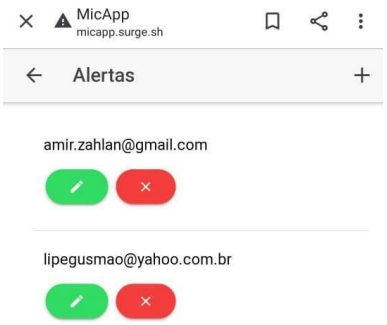
Tela de pacientes

Fonte: MICAPP (2020).

Tela de pedidos

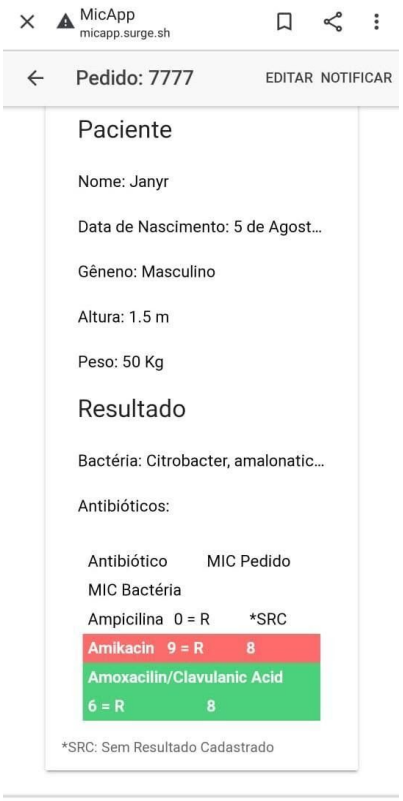
Fonte: MICAPP (2020).

Tela de alertas



Fonte: MICAPP (2020).

Tela de antibiogramas



Fonte: MICAPP (2020).