UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DO TOCANTINS/CAMETÁ
FACULDADE DE SISTEMAS DE INFORMAÇÃO

NATANAEL GARCIA WANZELER

MONITORA TOCANTINS: DESENVOLVIMENTO DO APLICATIVO PARA ANÁLISE DOS IMPACTOS DA COVID-19 NA REGIÃO DO BAIXO TOCANTINS

Cametá-PA
2020

NATANAEL GARCIA WANZELER

MONITORA TOCANTINS: DESENVOLVIMENTO DO APLICATIVO PARA ANÁLISE DOS IMPACTOS DA COVID-19 NA REGIÃO DO BAIXO TOCANTINS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação, Faculdade de Sistemas de Informação, Campus Universitário do Tocantins/Cametá, Universidade Federal do Pará, como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Ulisses Weyl da Cunha Costa.

Cametá-PA
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)

W244m Wanzeler, Natanael Garcia.

Monitora Tocantins: Desenvolvimento do Aplicativo para Análise dos Impactos da Covid-19 na Região do Baixo Tocantins / Natanael Garcia Wanzeler. — 2024.

LXXVII, 77 f. : il. color.

Orientador(a): Prof. Dr. Ulisses Weyl da Cunha Costa

Trabalho de Conclusão (Graduação) - Universidade Federal do Pará, Campus Universitário de Cametá, Curso de Sistemas de Informação, Cametá, 2024.

1. Desenvolvimento de aplicativo. 2. Cross-platform. 3. Pandemia. I. Título.

CDD 003

NATANAEL GARCIA WANZELER

MONITORA TOCANTINS: DESENVOLVIMENTO DO APLICATIVO PARA ANÁLISE DOS IMPACTOS DA COVID-19 NA REGIÃO DO BAIXO TOCANTINS

Data da Defesa: Cametá, PA, 27 de Novembro de 2024.

Membros da Banca Examinadora:

Professor Dr. Ulisses Weyl da Cunha Costa (Orientador)
Universidade Federal do Pará

Prof.Dr. Carlos dos Santos Portela (Membro interno)
Universidade Federal do Pará

Prof.Dr. Dalmi Gama dos Santos (Membro externo)
Universidade Federal do Pará

Dedico este trabalho à minha fámilia.

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus por toda a providencia que ele proporciona à minha vida. Em segundo lugar, agradeço à minha família, principalmente, à minha mãe Maria Raimunda Paes Garcia que sempre que sempre me incentivou a buscar o melhor pra mnha vida, dedicando-se com paciência e carinho, me ajudando a prosseguir.

Aos meus irmãos, muito obrigado por tudo que fazem por mim, pelos bons conselhos e por toda ajuda que me deram até aqui.

“Aí está um verdadeiro israelita, em quem não há falsidade”
(João 1:47)

RESUMO

Com a propagação da pandemia de Coronavírus, os dois hospitais da cidade de Cametá, no Baixo Tocantins, além dos postos de saúde, ficaram cheios, pois os casos foram acumulando, sem que houvesse atendimento para toda a população doente que se apresentava nas emergências destes hospitais. Diante de tal problemática, observou-se a necessidade de desenvolver um aplicativo para monitorar os casos de COVID-19 no município, para fazer orientações de cuidados e prevenção ao vírus e mapear os lugares com altos níveis de contaminação. Para isso, foi criado o aplicativo Monitora Tocantins, com o objetivo de fazer o monitoramento dos sintomas da doença através dos dados coletados por meio do formulário presente no aplicativo. O conjunto de atividades que permitiu alcançar os objetivos do projeto foi pautado numa abordagem aplicada buscando cumprir as principais etapas do processo de desenvolvimento que envolvem o levantamento de requisitos, que serve para detalahar as funcionalidade do aplicativo; a escolha das ferramentas que serão utilizadas na criação do aplicativo é outra etapa importante; além disso, a elaboração do diagrama de caso de uso é parte essencial para a compreensão daquilo que não ficou claro nos requisitos; assim como, o modelo lógico que serve para planejar quais dados serão armazenados no aplicativo. Dessa forma, o aplicativo multiplataforma Monitora Tocantins é o principal resultado da execução dos objetivos específicos necessários para se atingir esse objetivo geral. Os resultados esperados foram alcançados com a conclusão do desenvolvimento e a estabilidade do aplicativo.

Palavras-chave: Pandemia. Monitora Tocantins. desenvolvimento multiplataforma.

ABSTRACT

With the propagation of the Coronavirus pandemic, the two hospitals in the city of Cametá, in Baixo Tocantins, as well as the health centers, became full, due to the cases, without there being any care for the entire sick population that presented in the emergency rooms of these hospitals. In light of this problem, it was necessary to develop an application to monitor COVID-19 cases in the city, to provide guidance of care and prevention of the virus and to map the places with high levels of contamination. In order to achieve this, the Monitora Tocantins application was created, with the objective of monitoring the symptoms of the disease through the data collected by the form present in the application. The set of activities that granted achieving the project objectives was based on an applied approach aiming to fulfill the main stages of the development process that involve the gathering of requirements, which serves to detail the functionality of the application; the choice of the tools that will be used in the creation of the application is another important step; moreover, the elaboration of the use case diagram is an essential part of understanding what was not clear in the requirements; as well as the logical model that is meant to plan which data will be stored in the application. Therefore, the Monitora Tocantins multiplatform application is the main result of the execution of the specific objectives necessary to achieve this general objective. The expected results were attained with the completion of the development and the stability of the application.

Keywords: Pandemic. Monitora Tocantins. Multiplatform Development.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API	Interface de programação de aplicações
CSS	Cascading Style Sheets
DER	Diagrama de Entidade-Relacionamento
GPS	Sistema de posicionamento global
HTML	HyperText Markup Language
HTTP	Protocolo de Transferência de Hipertexto
IDE	*Integrated Development Environment*
iOS	iPhone Operating System
JDK	*Java SE Development Kit*
JS	*JavaScript*
JSON	JavaScript Object Notation
JSX	*JavaScript XML*
LGPD	Lei Geral de Proteção de Dados Pessoais
RF	Requisito Funcional
RNF	Requisito não Funcional
RN	Regras de Negócio
SDK	Software Development Kit
SI	Sistema de Informação
TCC	Trabalho de Conclusão de Curso
UBS	Unidade Básica de Saúde
UFPA	Universidade Federal do Pará
UML	Unified Modeling Language
UX	User Experience Design

SUMÁRIO

1. INTRODUÇÃO

Este trabalho tem como principal objetivo abordar o desenvolvimento do aplicativo Censo para a análise dos impactos da covid-19 no baixo Tocantins, o qual foi criado por alunos do curso de graduação de SI, da Universidade Federal do Pará – Campus Cametá, durante o período pandêmico. Pois devido a propagação do vírus no município, os locais de atendimento apresentavam super demandas de pessoas infectadas, sem que houvesse uma estrutura adequada para atender a população doente que dava entrada nas emergências dos hospitais. Além disso, a secretaria de saúde do município não conta com um suporte adequado para a obtenção de informações precisas a respeito do alto índice de contaminação da doença, ou seja, o município não dispõe de uma ferramenta para realizar a coleta dos dados necessários à execução de planos de ação no combate à pandemia na cidade. Por isso, justifica-se a implementação do aplicativo Monitora Tocantins que tem por principal finalidade monitorar os possíveis casos/sintomas de infecção causada pelo Coronavírus.

O aplicativo está pautado nos padrões de desenvolvimento do processo de criação da aplicação que segue as etapas necessárias da metodologia de desenvolvimento e sua relevância para o processo, apresentando conceitos fundamentais à compreensão do tema, as definições da componentização do software, a linguagem, tipos de plataformas, os modelos de processo, aplicativos semelhantes, entre outros. Inicialmente, realizou-se uma pesquisa bibliográfica de abordagem exploratória buscando compreender a problemática, objetivando identificar os aspectos constitutivos do problema; após isso, foi feito um esboço estrutural da aplicação sugerindo suas interfaces e funcionalidades do aplicativo, em seguida, inicia-se a etapa do levantamento de requisitos para nortear todo o desenvolvimento da aplicação; em seguida criou-se um esboço estrutural da apliação apresentando suas interfaces e funcionalidades, as quais passaram pela avaliação dos usuários do aplicativo, para verificar os pontos positivos que a serem relatados e os pontos negativos, para que sejam feitas possíveis sugestões ou melhorias para o bom aproveitamento do apalicativo.

O trabalho tem a seguinte estrutura: No capítulo 1, introdução; No capítulo 2, detalhamento dos objetivos geral e específico; No capítulo 3, fundamentação teórica; No capítulo 4, metodologia para a construção do aplicativo, destacando as principais características cross-platform da ferramenta React Native; No capítulo 5, desenvolvimento: compreendendo a comparação de trabalhos semelhantes, modelo de criação de software, levantamento e análise de requisitos, diagramação (caso de uso, MER), ferramentas, prototipação; No capítulo 6, os

resultados alcançados, No capítulo 7, considerações finais e trabalhos a serem realizados no futuro.

1.1 **OBJETIVOS**

1.1.1 Geral

– Expor as etapas relacionadas ao desenvolvimento do aplicativo censo.

1.1.2 Especifico

– Definir as etapas do processo de desenvolvimento;

– Levantar os requisitos;

– Definir ferramentas para o desenvolvimento do aplicativo;

– Elaborar diagramas (caso de uso, modelo lógico);

– Criar o banco de dados;

– Criar protótipo;

– Mostrar funcionalidade;

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção tem como objetivo apresentar a revisão bibliográfica mediante a exposição do referencial teórico para fundamentar o trabalho, assim como apresentar os trabalhos semelhantes ao aplicativo desenvolvido.

2.1 CONTEXTUALIZAÇÃO

O avanço tecnológico consolidou a era da cultura de dados, melhorando as experiencias humanas no mundo digital, trazendo soluções eficientes, facilitando o dia a dia e melhorando a qualidade de vida de muitos.

A pandemia de COVID-19 foi um grande demarcador desta década, pois com o distanciamento social obrigatório, houve uma mudança drástica na rotina de todos. E a tecnologia tornou possível a execução de atividades necessárias e tão importantes para a sociedade, pois através da engenharia de *software*, que possibilita criar ferramentas robustas que auxiliam na comunicação, estruturação, organização de dados e informações. Além disso, possibilitam armazenamento, processamento, acesso em tempo real ou remoto e compartilhamento destes.

Poster e Shapiro (1999) apontam a tecnologia como campo de interação entre técnicas e relações sociais que reconfigura a analogia entre tecnologia e cultura. Brittos (2002) acrescenta que as tecnologias geram impacto econômico, político e sociais.

Os dispositivos moveis evoluíram de uma tecnologia incorporada por um sistema de voz analógico, para uma comunicação baseada em aplicativos, serviços e dados que o ecossistema de internet conseguiu fornecer de forma eficiente até o presente momento.

2.2 AS ETAPAS ESSENCIAIS PARA A CRIAÇÃO DE UM SOFTWARE

De acordo com Sommerville (2011.p,5) a engenharia de *software* é uma disciplina que se preocupa com todas as etapas, atividades e processos relacionados ao planejamento, construção e manutenção de aplicações que são conhecidas e agregam conceitos importantes que são trabalhados desde o início do *software* até a evolução do sistema, como por exemplo, a prototipagem e o levantamento de requisitos. Para ele, as atividades relacionadas à produção de um software envolvem diferentes processos, porém todos incluem as quatro atividades fundamentais:

> **1 – Especificação de *Software*:** Definição de funcionalidade e restrições de seu funcionamento

> **2 – Projeto e Implementação de *Software*:** O Software deve atender às especificações.
>
> **3 – Validação do *Software*:** O *Software* deve ser validado para garantir as demandas do cliente.
>
> **4 – Evolução do *Software*:** O *Software* deve evoluir para atender as necessidades de mudança dos clientes.

Estas atividades incluem subatividades como validação de requisitos, projeto de arquitetura, testes unitários, etc. Além de atividades de apoio como documentação e gerenciamento de configuração de *software*.

Conforme afirma Sommerville, a engenharia de software é importante por dois motivos:

> **1 –** Cada vez mais, indivíduos e sociedades dependem dos sistemas de software avançados. Temos de ser capazes de produzir sistemas confiáveis econômica e rapidamente;
>
> **2 –** Geralmente é mais barato, a longo prazo, usar métodos e técnicas da engenharia de software para sistemas de software, em vez de simplesmente escrever os programas como se fossem algum projeto pessoal. Para a maioria dos sistemas, a maior parte do custo é mudar o software depois que ele começa a ser usado. SOMMERVILLE (2011. p, 5)

2.3 MODELOS DOS PROCESSOS DE SOFTWARE

De acordo com Sommerville (2018, p.31), um modelo de processo de *software* — às vezes, chamado de ciclo de vida do desenvolvimento de software — é uma representação simplificada de um processo de *software*.

Os modelos dos processos de *software* são utilizados para explicar as diferentes abordagens do desenvolvimento de *software*. Os modelos de processo apresentados aqui são:

2.3.1 MODELO EM CASCATA

Representa as atividades fundamentais do processo, como especificação, desenvolvimento, validação e evolução, na forma de fases de processo distintas, como especificação de requisitos, projeto de software, implementação e testes.

Os principais estágios do modelo em cascata citados por Sommerville, refletem diretamente as atividades fundamentais do desenvolvimento:

> **1. Análise e definição de requisitos.** Os serviços, restrições e metas do sistema são estabelecidos por meio de consulta aos usuários. Em seguida, são definidos em detalhes e funcionam como uma especificação do sistema.

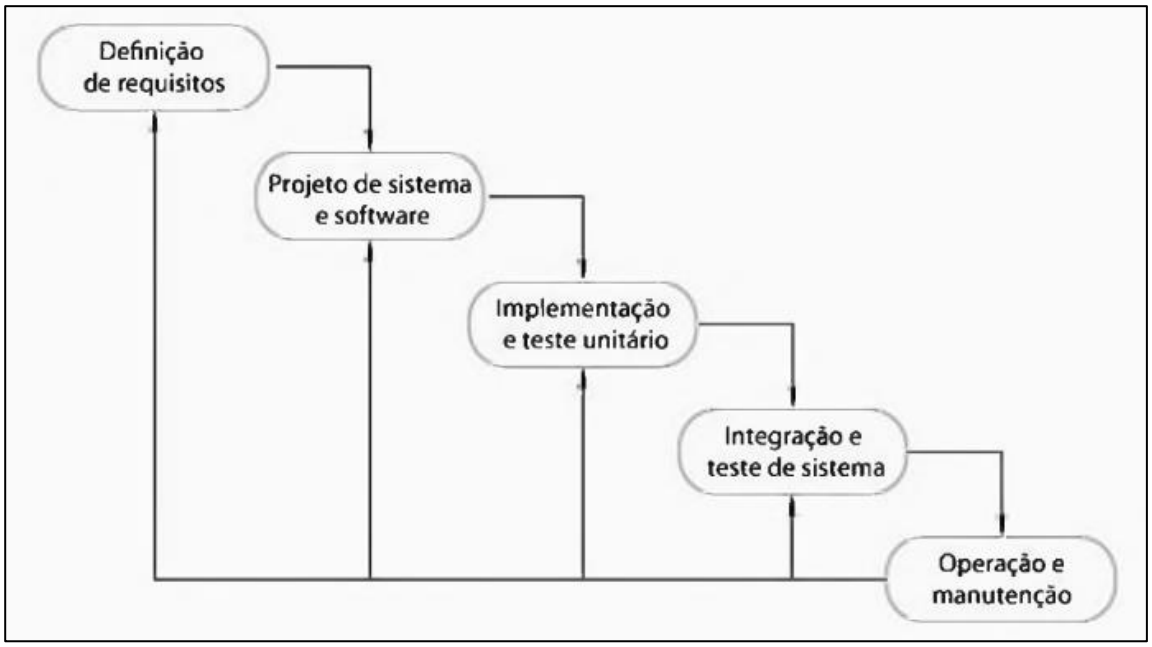
2. Projeto de sistema e software. O processo de projeto de sistemas aloca os requisitos tanto para sistemas de hardware como para sistemas de software, por meio da definição de uma arquitetura geral do sistema. O projeto de software envolve identificação e descrição das abstrações fundamentais do sistema de software e seus relacionamentos.

3. Implementação e teste unitário. Durante esse estágio, o projeto do software é desenvolvido como um conjunto de programas ou unidades de programa. O teste unitário envolve a verificação de que cada unidade atenda a sua especificação.

4. Integração e teste de sistema. As unidades individuais do programa ou programas são integradas e testadas como um sistema completo para assegurar que os requisitos do software tenham sido atendidos. Após o teste, o sistema de software é entregue ao cliente.

5. Operação e manutenção. Normalmente (embora não necessariamente), essa é a fase mais longa do ciclo de vida. O sistema é instalado e colocado em uso. A manutenção envolve a correção de erros que não foram descobertos em estágios iniciais do ciclo de vida, com melhora da implementação das unidades do sistema e ampliação de seus serviços em resposta às descobertas de novos requisitos. (SOMMERVILLE 2011. p,20)

Figura 1 – Modelo em Cascata.



Fonte: Sommerville(2011).

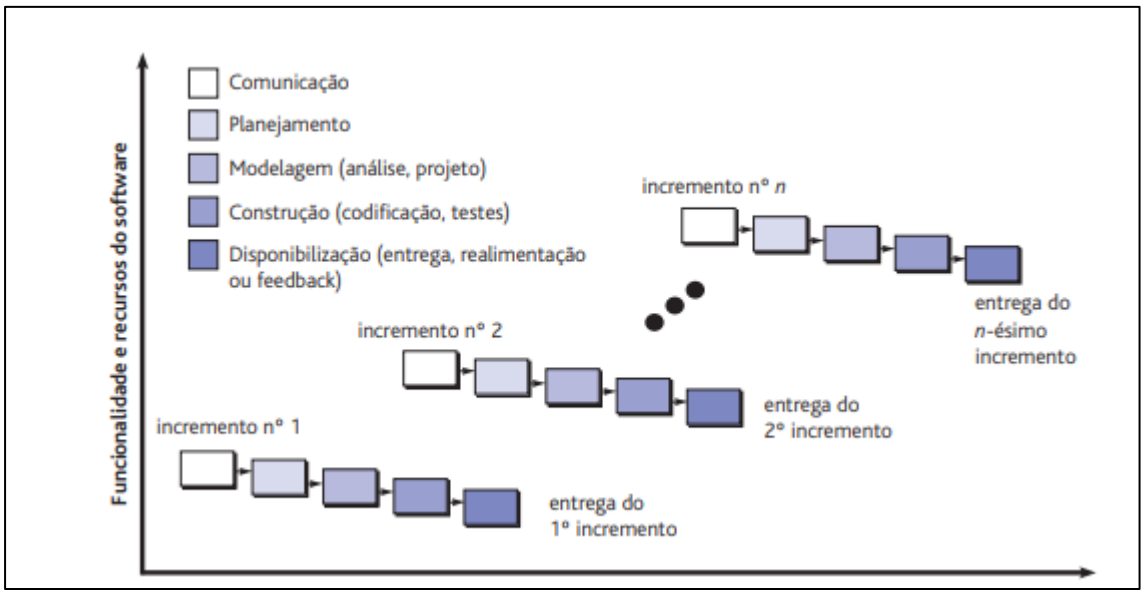
2.3.2 MODELO INCREMENTAL

Intercala as atividades de especificação, desenvolvimento e validação. O sistema é desenvolvido como uma série de versões (incrementos), com cada uma delas acrescentando funcionalidade à versão anterior.

Para Pressman e Maxim (2016. p,44), o modelo incremental libera uma série de versões, denominadas incrementos, que oferecem, progressivamente, maior funcionalidade ao

cliente à medida que cada incremento é entregue, ou seja, a cada entrega, os requisitos são refinados para que haja a agregação das funcionalidades, além de combinar elementos dos fluxos de processos lineares e paralelos, aplicam-se sequências lineares, de forma escalonada para o funcionamento do *software*, à medida que o tempo avança, como se observa na Figura 2.

Figura 2 – Modelo Incremental



Fonte: Pressman e Maxim (2016).

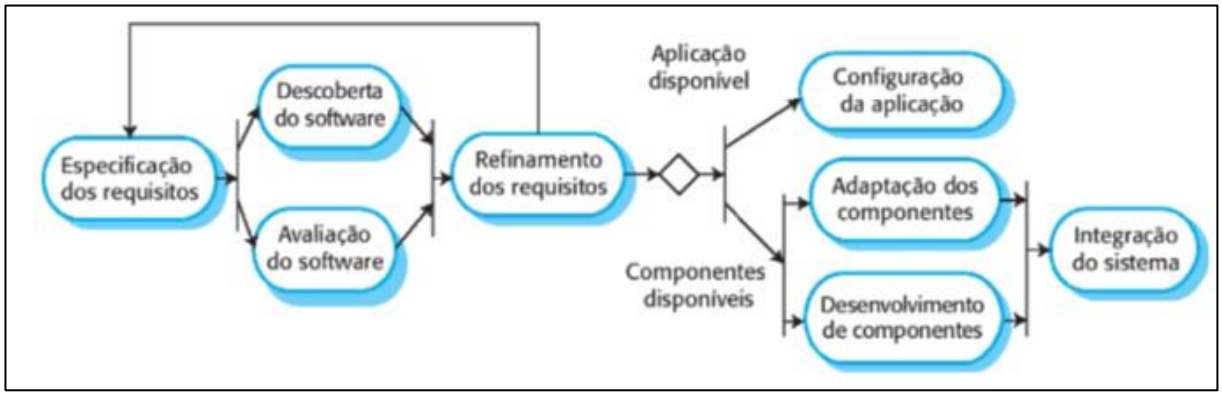
2.3.3 INTEGRAÇÃO E CONFIGURAÇÃO

Baseia-se na disponibilidade de componentes ou sistemas reusáveis. O processo de desenvolvimento de sistemas se concentra na configuração desses componentes, para que sejam utilizados em um novo contexto, e na integração deles em um sistema, pois na maioria dos projetos há algum reuso de software.

De acordo com Sommerville (2019. p,37), os três tipos de componentes de software reusados frequentemente são:

> **1.** Sistemas de aplicação stand-alone configurados para utilização em um ambiente particular. Esses sistemas são de uso geral e possuem muitas características, mas precisam ser adaptados para uso em uma aplicação específica.
>
> **2.** Coleções de objetos desenvolvidos como um componente ou como um pacote a ser integrado a um *framework* de componentes, como o Java Spring framework (WHEELER; WHITE, 2013).
>
> **3.** Web services desenvolvidos de acordo com os padrões de serviço e que estão disponíveis para uso remoto na internet.

Figura 3 – Modelo Orientado ao Reuso.



Fonte: Sommerville (2011).

Para Sommerville (2018, p.31,32), não existe modelo de processo universal aplicável a todos os tipos de desenvolvimento de software. Pois, depende do cliente, dos requisitos, do ambiente, e do tipo de software a ser desenvolvido.

2.4 SISTEMAS DE INFORMAÇÃO

O conceito de Sistema de Informação é aplicável a qualquer dispositivo desenvolvido para coletar, processar, armazenar e transmitir informações, afim de facilitar o acesso de usuários interessados, solucionar problemas e atender às suas necessidades, utilizando-se de quaisquer aplicativos, sites de notícia, enviar e receber informações por meio de sistemas desenvolvidos para esse fim, auxiliam nas tomadas de decisão do cotidiano.

De acordo com Kenneth e Laudon (KENNETH; LAUDON, 2011), um sistema de informação é um conjunto de componentes interligados que processam dados e os utilizam na tomada de decisão, de modo que estes sejam processados e se tornem totalmente úteis a uma organização.

2.5 DESENVOLVIMENTO MOVEL

Segundo Ferreira et al. (2018), aplicação móvel é todo e qualquer software destinado a ser executado em um dispositivo móvel. Segundo Hjort (2020), as aplicações são classificadas em: Aplicações nativas e aplicações multiplataforma.

Ao iniciar o desenvolvimento de um aplicativo para dispositivos móveis, as empresas e desenvolvedores precisam preocupar-se em gerar uma aplicação que possa atender, pelo menos, os sistemas com maior relevância no mercado, que são *Android* e *iOS* que dominam

aproximadamente 99% desses dispositivos. Ambos possuem ambientes de desenvolvimento diferentes. Nesses modelos de desenvolvimento necessitava-se de dois times de desenvolvedores para alcançar as duas plataformas, gerando assim, um desperdício de tempo e dinheiro para a construção de aplicações para ambos os sistemas operacionais. Atualmente, o modelo cross-platform surge como uma alternativa, onde o desenvolvimento é feito em uma única linguagem e depois, o código é compilado ou gerado para sistemas operacionais distintos, reduzindo custos e tempo de criação. Para o desenvolvimento da aplicação Monitora Tocantins foi utilizado o conceito do modelo cross-platform que é aplicado pelo framework React Native.

2.6 APLICAÇÕES NATIVAS

Para Monde (2011):

> As aplicações nativas são desenvolvidas para uma única plataforma, com a capacidade de executar e explorar todas as funções que tal plataforma em questão dispõe. Ou seja, os aplicativos que são criados exclusivamente para *iOS*, *Android* e/ou *Windows Phone*, entre outras, conseguem acessar todo o potencial do dispositivo através da própria arquitetura do sistema operacional, como câmera, calendário, álbum de fotos, *GPS*, entre outros. É dito que um aplicativo nativo é todo programa construído sob medida para uma única plataforma, com o intuito de funcionar sob medida para os dispositivos em conjunto com suas especificidades. (apud REIS, 2019).

Os aplicativos nativos geralmente, são rápidos, responsivos, oferecem melhor desempenho, experiência e são capazes de fazer uso completo dos recursos de hardware e *software*, uma vez que se comunicam diretamente com os componentes de interface de usuário e tais componentes são personalizados especificamente para sua plataforma (HJORT, 2020). O ambiente de desenvolvimento integrado (IDE) utilizado pelo sistema *Android* é o *Android Studio* (ANDROID, 2024), e dos dispositivos *iOS* a *IDE* oficial é denominada *Xcode*. O *android* studio utiliza a linguagem de programação *Java* ou *Kotlin*, enquanto os sistemas *iOS* utilizam o *Swift e Objective-C*. Diferentemente das aplicações *cross-platform* que utiliza-se da criação de um código único para a sua implementação, o aplicativo nativo exige uma equipe específica de desenvolvedores para criar códigos distintos para cada plataforma.

2.7 APLICAÇÕES CROSS-PLATFORM

Para El-Kassas (EL-KASSAS et al., 2017), o conceito principal de multiplataforma é desenvolver um aplicativo uma vez para executá-lo em qualquer lugar, ou seja, criar um único código que pode ser compilado para mais de um SO, pois suas etapas só precisam ser

desenvolvidas e testadas uma vez evitando o retrabalho nos processos de criação de um software.

De acordo com El-Kassas (EL-KASSAS et al., 2017), os tipos de aplicações móveis são aplicações *Web*, aplicações nativas e aplicações híbridas.

- Aplicações *Web* não precisam ser instaladas e podem ser acessadas pela URL de um navegador e são aplicações que utilizam *HTML*, *CSS*, *Javascript* no seu desenvolvimento.
- Aplicações nativas podem ser instaladas por downloads ou pela própria store. São aquelas desenvolvidas com o uso de linguagens e ferramentas fornecidas pela própria plataforma (*Android* e *iOS*).
- As aplicações híbridas podem ser instaladas por downloads ou pela própria store. São renderizadas dentro de uma aplicação nativa (Web) utilizando uma Web View.

2.8 REACT NATIVE

O React Native foi introduzido em 2015 pelo Facebook, e é um framework de código aberto para desenvolver aplicativos mobile (Android e iOS). Foi criado como uma solução para o desenvolvimento multiplataforma e um dos principais objetivos é fazer com que o desenvolvedor não necessite saber duas linguagens de programação para facilitar o seu desenvolvimento pois utiliza somente uma base de código. Com o uso do React Native, a interface gráfica oferece uma melhor experiência com a renderização de componentes de interface nativa por utilizar recursos específicos da plataforma, como o microfone ou a câmera do telefone, acessando a API da plataforma de interfaces. Uma das grandes vantagens do React Native é o JavaScript e o seu subconjunto TypeScript, visto que são linguagens amplamente utilizadas por desenvolvedores web. Pois torna-se prático criar aplicativos móveis com técnicas já conhecidas, uma vez que a linguagem e o ambiente de desenvolvimento de cada plataforma podem ser reaproveitados.

2.9 APLICATIVOS SEMELHANTES

Os três trabalhos semelhantes apresentados nesta seção serviram de base para o presente trabalho por se tratar de estudos que possuem o mesmo tema: o desenvolvimento mobile com a metodologia *crossplatform,* cujas abordagens e técnicas utilizadas pelos autores fizeram uso do *framework React Native*.

2.9.1 APLICATIVO GESTÃO DE VACINAÇÃO

A criação de um aplicativo desenvolvido por Lima (2021), que tem como título “Multiplataforma para gestão de vacinação”, tem como ideia principal criar uma ferramenta para auxiliar seus usuários a fazerem a gestão de suas vacinas e manter esses registros no aplicativo, e com isso se manter imunizado das doenças que possuem vacinas fornecidas pelo SUS. Isso será realizado através de notificações e avisos que serão exibidos no aplicativo. Também, o usuário poderá verificar todas as vacinas que já foram aplicadas, do mesmo modo que ocorre quando se utiliza um cartão de vacinação.

2.9.2 APLICATIVO TURISTANDO BEBERIBE

A principal proposta do aplicativo criado por Falcão (2022), Turistando Beberibe, é oferecer um catálogo de experiências turísticas na região devido à carência de aplicações voltadas ao turismo na cidade Beberibe/CE, através da contratação de passeios e experiências turísticas. Com o objetivo de abranger a máxima quantidade de pessoas, foi decidido que o aplicativo seria disponível para as plataformas *Android* e *iOS*.

2.9.3 APLICATIVO NAVEG

É um aplicativo criado por Araujo (2019), que busca auxiliar veganos e vegetarianos de Fortaleza a acompanhar os principais eventos, produtos e serviços relacionados a esse estilo de vida. A proposta do NaVeg é ser *cross-platform,* executando-se nos Sistemas Operacionais Androide *iOS*

Os três aplicativos descritos acima têm como base o desenvolvimento multiplataforma que traz a vantagem na economia de recursos e de tempo, uma vez que não é preciso mais de uma equipe para desenvolver mais de uma vez um mesmo aplicativo. Eles descrevem todo o processo de criação do aplicativo desde a sua conceituação até o seu desenvolvimento, destacando as características *cross-platform* da ferramenta de desenvolvimento usada, o *React Native.*

3. METODOLOGIA

3.1 CONTEXTO

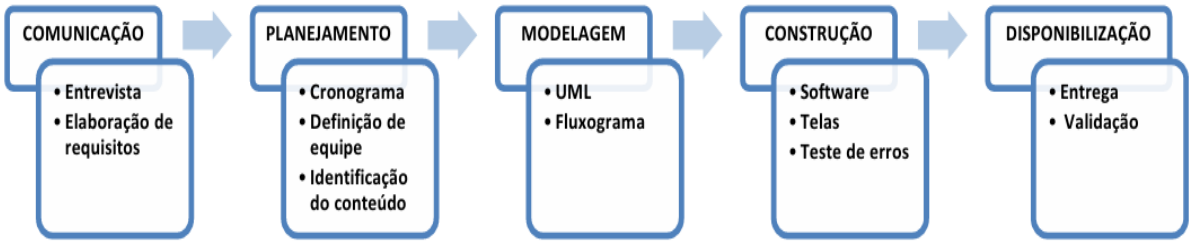
Esta é uma pesquisa do tipo aplicada com enfoque na produção tecnológica, conforme define Gil (2019), que a pesquisa aplicada é a geração ou utilização de conhecimentos de base para construção ou aperfeiçoamento de produtos para aplicação imediata com o objetivo de contribuir para fins práticos de problemas locais ou regionais. Este estudo visa, por meio de métodos computacionais, criar um aplicativo para monitorar os casos de COVID-19, tendo em vista a grande demanda de usuários que apresentaram sintomas compatíveis com a doença no município, e a necessidade de se ter estimavas adequadas das notificações e subnotificações, através do mapeamento dos lugares com altos índices de contaminação, bem como, fazer a orientação ao primeiro atendimento, reforçar as medidas de prevenção e levar à tomada de decisão mais efetiva, proporcionando ao sistema de saúde local os dados necessários para a execução de planos de ação de combate à pandemia. Inicialmente, buscou-se realizar uma pesquisa bibliográfica com abordagem exploratória, conforme preconizado por Pereira et al. (2018), objetivando identificar os aspectos constitutivos do problema. Além do mais, entende-se que o aplicativo é uma importante ferramenta para futuras pesquisas de profissionais da área da saúde. Este trabalho está alinhado à utilização da UML como parte da metodologia para o desenvolvimento do aplicativo para dispositivos moveis, possibilitando a organização das etapas do desenvolvimento do software de uma forma eficiente, pois segundo Bezerra (2007. p,14), a UML é uma caixa de ferramentas utilizada pelos desenvolvedores para realizar tarefas, independente da linguagem de programação utilizada.

3.2 MODELAGEM DO SISTEMA

Para a organização das atividades do desenvolvimento deste aplicativo foi utilizado o modelo incremental de Sommerville (2018, p.35), que se baseia na ideia de desenvolver uma implementação inicial, obter feedback dos usuários ou terceiros e fazer o software evoluir através de várias versões, até alcançar o sistema finalizado. Uma das vantagens do modelo escolhido é ter a possibilidade de determinar, ainda durante uma das etapas de desenvolvimento das funcionalidades do sistema, se os requisitos previstos para os níveis subsequentes estão corretos, caso sejam detectadas as falhas pode-se planejar a correção antes do inicio do novo módulo.

O desenvolvimento do aplicativo seguiu o formato do diagrama utilizado a partir do referencial de Pressman e Maxim (2016), seguindo-se as fases: comunicação, planejamento, modelagem, construção e disponibilização, conforme representado na Figura 1. Cada atividade utilizada nesta pesquisa é descrita nas subseções a seguir, em cinco fases.

Figura 4 – Fases do Desenvolvimento do Aplicativo.



Legenda: UML: Linguagem de Modelagem Unificada.

Fonte: Adaptado de Pressman e Maxim (2016).

Fase 1 – Comunicação: Na fase inicial do projeto foram realizadas reuniões (via Google Meet) com os coordenadores do projeto de desenvolvimento do aplicativo Monitora Tocantins, nas quais foram discutidos assuntos referentes a prototipagem, apresentando-se as interfaces e funcionalidade para serem validadas pelos profissionais da área da saúde, conhecido como stakeholders do sistema. Esta validação foi marcada pela verificação dos pontos positivos ou vantagens e os pontos negativos referentes a uma interface limpa e objetiva dos textos e imagens para facilitar a aprendizagem do usuário na utilização do aplicativo. Além disso, destaca-se ainda, a validação do formulário que foi baseado nos termos técnicos citados pelo Ministério da Saúde.

Fase 2 – Planejamento: A partir de outras reuniões realizadas entre coordenadores e alunos responsáveis pelo projeto estabeleceu-se um cronograma de atividades a serem executadas para atingir os objetivos da pesquisa, sendo definidas as tarefas de cada um dos membros do projeto. As reuniões aconteceram de forma remota, via Google Meet, para realização e execução do cronograma proposto.

Fase 3 – Modelagem: Nesta fase, utilizou-se a Linguagem Unificada de Modelagem (UML), linguagem única, cujo papel é auxiliar a equipe a visualizar todas as etapas do desenvolvimento do aplicativo, facilitando a compreensão do seu funcionamento. Os fluxogramas do aplicativo foram construídos a partir da linguagem UML, que geraram

clareza ao escopo, possibilitando que todos os envolvidos na pesquisa entendessem cada fase do processo.

Fase 4 – Construção: O aplicativo foi desenvolvido somente para a plataforma Android, utilizando-se as ferramentas de software e as linguagens:

- **React Native (Framework):** biblioteca para desenvolvimento do aplicativo;
- **Software Development Kit (SDK) completo**: conjunto de ferramentas de desenvolvimento de software que permite a criação de aplicativos;
- **CSS** (Cascading Style Sheets): utilizada para favorecer um visual mais agradável e garantir a usabilidade do aplicativo;
- **TypeScript**: adiciona recursos, como tipagem e orientação a objetos que não estão presentes do Javascript.
- **PostgreSQL**: utilizado para armazenar e gerenciar os dados coletados pelo dispositivo.
- **VScode**: IDE oferece diversas extensões que facilitam o desenvolvimento, a organização e leitura do código
- **Figma**: permite a criação de interfaces, protótipos realistas e wareframes para sites e aplicativos etc;

Após a fase de construção foram realizados testes para identificação de possíveis erros, para verificar se o aplicativo estava de acordo com as especificações determinadas para o projeto.

Fase 5 – Disponibilização: Após a conclusão, o aplicativo foi disponibilizado aos bolsistas do laboratório para a realização de testes. Antes, porém, os bolsistas receberam um treinamento para conhecerem as funcionalidades e configurações, pois não foi disponibilizado uma documentação que detalhasse as funcionalidades do aplicativo.

3.3 DESCOBERTA DE REQUISITOS

A descoberta de requisitos nada mais é que o processo de reunir informações sobre o sistema requerido apresentando assim os requisitos funcionais, não funcionais e as regras de negócio. Durante a fase de descoberta de requisitos, as fontes de informação incluem documentação, *stakeholders* do sistema e especificações de sistemas similares.

Para Sommerville (2011) “os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferecem e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada”.

3.4 REQUISITOS FUNCIONAIS

Descrevem as funcionalidades que deverão ser implementadas na aplicação, conforme define Reinehr (2020) “o requisito funcional como sendo “uma funcionalidade requerida pelos Stakeholders para cumprir algum objetivo de negócio. Representa o que os desenvolvedores deverão implementar para que os usuários possam realizar as suas atividades”.

Quadro 1 – Requisitos Funcionais.

ID	NOME	DESCRIÇÃO
RF01	*Login* do usuário	O aplicativo deve ter uma tela de login com a Logo do centralizada em tamanho padrão com um texto curto e objetivo para demonstrar a segurança no controle nos dados do usuário. A tela de login deve ter dois *TextInput*, ou seja, dois campos que possibilite ao usuário digitar seu e-mail e senha (campos obrigatórios). Na tela de login deve ter um botão de confirmação personalizado que siga os padrões de design. Ela deve ter ainda, um botão de cadastro de usuário para direcionar o usuário à tela de cadastro. Obs: Ao não preenchimento de um dos campos obrigatórios (*e-mail* e senha), o aplicativo deve mostrar uma mensagem de erro, mostrando também, qual dos campos não foi preenchido; deve ser implementada no aplicativo a funcionalidade de validação dos dados fornecidos pelo usuário, pois sempre que o usuário clicar em “Avançar”, é obrigatório que o aplicativo mostre um loading de 5 segundos, possibilitando ao aplicativo verificar se o dispositivo possui *internet*, caso não possua, ele deve mostrar uma mensagem ao usuário informando o erro; em segunda hipótese, caso o aplicativo tenha internet, ele deverá fazer uma consulta ao bando de dados e realizar

		a validação entre as informações fornecidas pelo usuário. Caso as informações não coincidam, o aplicativo deve mostrar uma mensagem de erro, do contrário o aplicativo deve mostrar um modal com uma mensagem: “Logado com sucesso” e em seguida carregar a tela inicial para o usuário.
RF02	Cadastrar usuário	O aplicativo deve permitir que o usuário forneça sua identificação (nome, cpf) que deve ser digitado nos *TexInputs* para efetivar o cadastro. Em seguida ao clicar em avançar, o aplicativo deve mostrar um loading de 5 segundos, enquanto o aplicativo realiza a consulta ao banco de dados para verificar se o usuário já possui cadastro. Se sim, não será perguntado novamente os dados (endereço, data de nascimento e sexo) já mostrará de imediato a tela Inicial. Se não, o usuário deve preencher os campos de endereço, data de nascimento, sexo, obrigatoriamente. Obs: Todos os campos obrigatórios devem ser verificados pelo aplicativo, caso um desses campos não seja preenchido, o aplicativo deve mostrar uma mensagem de erro, e destacar o campo que não foi preenchido. A tela de cadastro deve ter um botão de Voltar (o detalhamento desse requisito pode ser visualizado no requisito “RF17”)
RF03	Tela de *splash*	Ao abrir o aplicativo pela primeira vez aparecerá a tela com a Logo do aplicativo, centralizada, assim como um loading de 5 segundos dando a impressão de que algo está sendo carregado.
RF04	Tela de Boas vindas (slid)	O aplicativo deve ter uma tela onde seja mostrado *cards/* ícones/imagens, ilustrando a prevenção contra a COVID-19. Nessa tela é preciso ter um texto de conscientização que seja curto e objetivo. Logo abaixo do texto deve ter uma barra de progresso em formato de barras,

		possibilitando ao usuário clicar nelas para pular para a próxima tela, a tela de boas vindas. Obs: Nessa tela é preciso ter um botão que possibilite passar para a próxima tela, além disso, é preciso criar uma padronização em todas as telas de boas vindas, utilizando-se ícones e textos representativos.
RF05	Tela Inicial	O aplicativo deve ter uma tela onde o usuário possa acessar/visualizar todas as informações gerais do aplicativo, como: 1ª – Logo do aplicativo (deve ficar na parte superior esquerda do aplicativo); 3ª – Botão de configuração (ver "RF06"); 4ª – *Cards* informando o número de pessoas infectadas pela COVID-19 (ver "RF12"); 5ª – Aplicar Questionário da sua Região (ver "RF16"); 6ª – *Links* de notícias relacionadas à COVID-19. As notícias devem estar no formato de listagem, na vertical, com título, imagem e descrição. Nesta listagem é preciso ter uma barra de rolagem para facilitar ao usuário navegar entre as notícias.
RF06	Configuração	O aplicativo deve ter um menu no canto superior direito da tela, contendo um ícone com formato de usuário. Neste menu suspenso o usuário poderá visualizar as categorias "Conta e Geral". Na categoria Conta, o aplicativo deve ter dois botões chamados: Opção 1ª – Dados Pessoais (ver"RF07"); Opção 2ª – Endereço (ver "RF08"); Na categoria Geral, o aplicativo deve ter os botões: Opção 1ª – Suporte (ver "RF09") Opção 2ª – Sobre ("RF10") Opção 3ª – Sair do Aplicativo (ver "RF14");

		Obs: Todos esses botões devem ter ícones e tamanhos padronizados de acordo com a sua finalidade.
RF07	Dados Pessoais	O aplicativo deve ter um botão chamado Dados Pessoais que direcionado o usuário para uma tela que contém os campos nome, cpf e *e-mail*. Obs: Se o usuário editar algum campo, o aplicativo fará uma verificação em todos os campos buscando identificar os campos alterados. Em seguida, ele mostrará um modal com a seguinte mensagem "Foram alterados os campos x, y, z, você deseja fazer a alteração?", Se a resposta for sim, aparecerá um loading de 5 segundos demonstrando que o aplicativo está processando a ação, no final será mostrado uma mensagem ao usuário: "Informações alteradas com sucesso". Este modal também precisa ter um botão "Cancelar". Caso o aplicativo apresente alguma instabilidade inesperada, os dados não serão alterados e o aplicativo irá mostrar uma mensagem de erro e voltar para a tela Dados Pessoais com os dados do usuário.
RF08	Endereço	O aplicativo deve ter um botão Endereço, que direcionará o usuário a uma tela que contém os campos rua, número, bairro, distrito e cidade. Obs: Se o usuário editar algum campo, o aplicativo fará uma verificação em todos os campos buscando identificar os campos alterados. Em seguida, ele mostrará um modal com a seguinte mensagem "Foram alterados os campos x, y, z, você deseja fazer a alteração?", Se a resposta for sim, aparecerá um loading de 5 segundos demonstrando que o aplicativo está processando a ação, no final será mostrado uma mensagem ao usuário: "Informações alteradas com sucesso". Este modal também precisa ter um botão "Cancelar".

		Caso o aplicativo apresente alguma instabilidade inesperada, os dados não serão alterados e o aplicativo irá mostrar uma mensagem de erro e voltar para a tela Endereço com os dados do usuário.
RF09	Suporte	O aplicativo deve ter um botão chamado Suporte que ao clicar nele o usuário será direcionado a uma tela com as seguintes estruturas: 1ª – Deve ter um *TextInpt* (caixa de texto), onde o usuário possa pesquisar algum assunto que ele esteja com dúvidas. 2ª – Deve ter dois *links*: um para sugerir melhorias e o outro para relatar *bugs*. Ao clicar no primeiro o usuário será direcionado a uma tela onde poderá escrever um texto sugerindo possíveis melhorias ao aplicativo; O segundo *link* direcionará o usuário a uma tela, na qual ele informará o possível problema enfrentado durante o uso do aplicativo; Obs: Em ambas as telas (Sugerir Melhorias e Relatar B*ug*), o usuário deve visualizar um campo, no qual possa escrever seu texto, assim como um botão para enviá-lo. Sempre que o usuário clicar em enviar, o aplicativo deve verificar se o dispositivo possui internet, caso não possua, o aplicativo deve mostrar uma mensagem de erro, ao contrário deve mostrar uma mensagem "Enviado com sucesso"
RF10	Botão Sobre	O aplicativo deve ter um botão chamado, "Sobre". Ao clicar neste botão o usuário será direcionado a outra tela onde visualizará as informações relacionadas: 1ª – Logo do aplicativo (deve ser centralizada na tela e em tamanho padrão); 2ª – Versão do aplicativo (deve ser centralizada na tela e em tamanho padrão. Ainda nesta tela o aplicativo deve colocar o "*Copyright* do laboratório"); 3ª – Convidar um amigo (ver "RF11");

		4ª – Botão de Termos e Condições (ver “RF12”); 5ª – Botão de Política de privacidade (ver “RF13”);
RF11	Convidar um amigo	O aplicativo deve ter um botão/*link* para compartilhar o aplicativo com outras pessoas e para que isso seja possível é preciso que o aplicativo esteja disponível em uma plataforma segura para que não ocorra nenhum contratempo. Sempre que o usuário clicar nesta funcionalidade o aplicativo fará uma verificação se o usuário possui internet, caso não haja acesso à internet o aplicativo mostrará uma mensagem de erro, do contrário, o aplicativo mostrará um modal com os aplicativos de compartilhamento/de trocas de mensagens, dessa forma, o usuário do Monitora Tocantins poderá enviar o *link* para o usuário fazer o *download*.
RF12	Termos e Condições	O aplicativo deve ter um botão chamado Termos e Condições, por meio do qual o usuário será direcionado a uma tela onde poderá ler os termos e condições de uso de seus dados fornecidos aos responsáveis do aplicativo. Ainda nesta tela o aplicativo deve ter um botão com a funcionalidade “Voltar” a tela anterior (ver “RF17”);
RF13	Política de Privacidade	O aplicativo deve ter um botão chamado Política de Privacidade, por meio do qual o usuário será direcionado a uma tela onde poderá ler a “Política de privacidade” de uso de seus dados por parte dos responsáveis do aplicativo. Ainda nesta tela o aplicativo deve ter um botão com a funcionalidade “Voltar” a tela anterior (ver “RF17”);
RF14	Sair do aplicativo	O aplicativo deve ter um botão chamado “Sair do Aplicativo”, esta funcionalidade fará com que o usuário seja deslogado de sua conta no aplicativo. Obs: Ao clicar nesse botão o aplicativo deve verificar se o dispositivo possui internet para poder deslogar a conta do usuário, caso não tenha acesso a internet, o aplicativo deve mostrar uma mensagem de erro, do contrário o aplicativo

		deve verificar se o dispositivo possui algum formulário pendente. Se sim, o usuário será obrigado a fazer a sincronização dos formulários pendentes. Após realizar a sincronização dos formulários será mostrado um modal com a seguinte mensagem “Formulários enviados com sucesso. Usuário deslogado com sucesso”
RF15	Visualizar casos da COVID-19	Após o usuário ter inserido seus dados de acesso (e-mail e senha) e o aplicativo ter permitido seu acesso, será mostrado em um campo fixo na tela inicial a quantidade de casos de COVID-19 no munícipio separados em *cards*: um conterá o total de infectados (suspeitos em análise); o outro com o total de casos confirmados, e por último o número de óbitos ocorridos em consequencia da doença. Esses *cards* devem conter uma cor e ícones que representem o nível de gravidade do quadro epdemiológico. Obs: Neste campo os *cards* devem estar delimitados. Além disso, deve ter um botão para que o usuário possa visualizar em um modal ou em uma tela de forma detalhada todos os casos de COVID-19. Ao efetuar essa ação, o aplicativo deve ter um botão “Voltar” (ver “RF17”), ou pode ser criado um botão “Voltar”, na parte inferior da tela, centralizado.
RF16	Aplicar questionário da sua região	O aplicativo deve ter um botão na tela inicial com ícones, cores e textos que remeta ao conteúdo do questionário. Nesta tela temos as seguintes funcionalidades: 1ª – Botão Voltar a tela anterior (ver RF17”); 2ª – Título da tela (ver RF18); 3ª – Botão de Sincronização dos dados (ver RF19); 4ª – *Cards* ilustrando o total de formulários (ver “RF20”); 5ª – Listagem dos formulários coletados (ver “RF21”); 6ª – Telas do questionário (ver “RF22”);

RF17	Navegação	O aplicativo deve ter um botão na parte superior no canto esquerdo, que deve ter um ícone de uma seta para funcionalidade de voltar à tela inicial;
RF18	Título da tela	Na parte superior, o aplicativo deve ter um texto com o título – Censo 2022 – com alinhamento centralizado e tamanho padrão.
RF19	Sincronização dos dados	Na parte superior, no canto direito, o aplicativo deve ter um ícone de sincronização/envio de dados, que servirá para o usuário enviar os formulários pendentes, ou seja, os formulários que foram coletados durante o aplicativo estar *offline*, pois ao clicar nesse botão todos os formulários pendentes serão enviados ao servidor do banco de dados. Antes que o aplicativo finalize o envio dos formulários é preciso que a aplicação faça umas validações: 1ª – Caso o usuário clique no botão e o dispositivo esteja sem internet, o aplicativo deve mostradar uma mensagem de erro. 2ª – Ao clicar no botão sincronizar todos os formulários pendentes, o aplicativo deve verificar se um dos formulários não foi cadastrado, caso ainda não esteja, o sistema deve cadastrar (será mostrado um modal com uma mensagem informando que o formulário foi cadastrado com sucesso). Outra hipótese seria o formulário já estar cadastrado no bando de dados. O sistema deve verificar cada uma das perguntas para saber qual delas foram alteradas, com isso será mostrado um modal informando que o usuário já possui cadastro e deverá perguntar se ele deseja fazer as alterações, para isso, esse modal deve ter um botão “Confirmar”, e outro para “Cancelar”.
RF20	*Cards* ilustrando o total de formulários	O aplicativo deve mostrar o nome do usuário e abaixo do nome do usuário terão três *cards*; o primeiro mostrará o subtotal de formulários enviados ao servidor; o segundo mostrará a quantidade de formulários pendentes (são

		formulários que foram coletados quando o aplicativo estava sem internet e foram salvos apenas no celular); o terceiro mostrará o total dos formulários que estavam pendentes e foram enviados ao servidor com sucesso. Cada *card* deve ter uma cor específica para representar a sua finalidade, exemplo: azul seria o subtotal, alaranjado seria os pendentes, verde seriam os formulários pendentes que foram enviados com sucesso;
RF21	Listagem dos formulários coletados	O aplicativo deve mostrar a listagem dos formulários que foram coletados pelo usuário, além disso, cada formulário deve ser personalizado com uma borda lateral esquerda de acordo com a cor de cada card (alaranjado seriam os pendentes, verde seriam os formulários completos, ou os formulários pendentes que foram enviados com sucesso para o banco de dados). Além disso, é preciso ser implementado dentro de cada *card* um pequeno detalhamento mostrando a data da criação desse formulário e também informar se o formulário foi enviado; se sim, deve ser adicionado a data do envio. Outro ponto importante que precisa ser implementado no *card* é ter um texto – pendente e/ou completo – à direita, no canto superior do *card*, nas opções de listagem pendente e/ou completo; a cor do texto deve ser a mesma que é representada pelo *card*, ou seja, se o card for completo o texto deve ser verde, se for pendente deve ser alaranjado.
RF22	Telas do questionário	O aplicativo deve ter um botão que direcionará o usuário para as telas que contém as perguntas do formulário. Algumas dessas telas poderão ter padrões e funcionalidades semelhantes. 1ª – Sempre que o entrevistador iniciar um novo formulário, o aplicativo deve ter uma tela onde seja mostrado um texto descritivo informando qual o objetivo da coleta de dados. Ainda nesta tela o aplicativo deve ter

		dois *ChekList*, ou seja, duas caixinhas: Uma para o entrevistador e a outra para o entrevistado. É preciso que ambos aceitem o "Termo de Confidencialidade e Sigilo", assim como, o "Termo de consentimento de Tratamento de Dados Pessoais", assim, os envolvidos estarão concordando com a Lei Geral de Proteção de Dados – LGPD, só assim o aplicativo deverá permitir que se avance para as próximas telas do formulário. Ainda nesta tela, os dois *ChekList* devem possuir um link para que ambos os usuários possam ler as informações a respeito do que trata esses termos. Esses termos são obrigatórios no formulário por se tratar de uma coleta de dados sensíveis. O detalhamento desses dois *ChekList* pode ser visualizado nos requisitos RF12 e RF13; 2ª – Na parte superior, no canto esquerdo do aplicativo deve ter o botão "Voltar" a tela anterior, representado pelo ícone de uma seta, ver RF17; 3ª – O aplicativo deve ter um texto com o título da tela e uma pergunta curta e objetiva. Deve ser implementado na parte superior, alinhamento centralizado e tamanho padrão; 4ª – Abaixo do título e da pergunta, o aplicativo dever ter as alternativas que podem variar de opções de múltipla escolha, que apresenta varias opções de respostas que podem ser assinaladas a partir de uma até a totalidade das opções a serem marcadas; a outra alternativa é colocar um menu *Drop Down*, onde temos várias perguntas e o entrevistado deve escolher somente uma alternativa. 5ª – O aplicativo deve implementar um mecanismo de detecção para verificar se as perguntas obrigatórias do formulário foram selecionadas, caso não tenham sido selecionadas, será mostrado um modal com uma

		mensagem informando que é preciso selecionar uma das alternativas. Obs: Todas as perguntas obrigatórias que não forem selecionadas/marcadas dentro do formulário devem possuir uma borda avermelhada sobre o/os campo/s dando destaque nas alternativas que precisam ser selecionadas, possibilitando ao usuário identificar melhor o que ele deve fazer. 6ª – Na tela de endereço, o aplicativo deve ter a funcionalidade de automatizar o preenchimento das informações, ou seja, buscar automaticamente o endereço do usuário através da geolocalização. Ao visualizar a tela "endereço", o aplicativo deve mostrar um modal que tenha um texto perguntando ao usuário se ele deseja buscar o seu endereço pela geolocalização, caso ele concorde, será ativado a sua localização; outra maneira seria através das configurações do seu aparelho, ativando a sua localização manualmente. Após fazer isso, o aplicativo deve fazer uma verificação se o dispositivo possui *internet* e atrelado a isso, deve ser mostrado na tela um *loading* de 5 segundos, enquanto isso o aplicativo realiza a verificação, caso não possua, deverá ser mostrado uma mensagem de erro; do contrário todos os campos serão preenchidos automaticamente, mas estes campos podem ser editados, se necessário. Como dito anteriormente, caso o dispositivo não possua internet, o recenseador poderá preencher os campos manualmente. 7ª – Nas telas que solicitarem data (dia/mês/ano), o aplicativo deve permitir que o entrevistador digite manualmente, ou clique no ícone de calendário e escolha a respectiva data.

		8ª – Todas as telas do formulário devem ter um botão padronizado – tamanho da fonte, altura, largura, cor de fundo e cor dos textos. 9ª – Ao iniciar o formulário, o aplicativo deve ter um enumerador de tela mostrando o total de telas do formulário, isto é, todas as telas terão um número identificando em qual tela o usuário está. O enumerador deve ficar no canto superior da tela e possuir um círculo e uma cor que dei destaque nos valores que estão dentro do círculo; 10ª – Ao chegar na última tela do formulário, o aplicativo deve permitir que o entrevistador possa voltar à tela anterior para editar algumas das perguntas respondidas pelo entrevistado, pois isso possibilita que o entrevistado corrija uma resposta que ele tenha fornecido de forma errada. 11ª – Ao finalizar o formulário, o aplicativo deve mostrar ao usuário o nível do seu sintoma (leve, moderado e/ou grave), de acordo com o que foi respondido no formulário, ainda nessa opção, terá um emoji representando o sintoma. 12ª – Ainda nesta tela, o aplicativo deve ter os locais de atendimento contra a COVID-19, por isso, ele deve ter botões com ícones ilustrativos, juntamente com informações do local, buscando mostrar a identificação de cada posto de atendimento à COVID-19: Ao clicar neste botão o aplicativo deve mostrar um *alert* perguntando se o usuário deseja abrir o *google Maps* e ao aceitar, o aplicativo deve verificar se o dispositivo possui acesso a internet, caso não tenha, o aplicativo deve mostrar uma mensagem de erro, do contrário, o mesmo poderá visualizar o endereço através do *google Maps*. OBS: O *google Maps* não abre dentro do app. É aberto o próprio App do *google Maps*, o aplicativo deverá linkar o

		endereço do usuário com o endereço dos postos de atendimento, com isso será mostrada na tela do dispositivo o local onde ele pode encontrar o posto de atendimento mais próximo. 13ª – Por fim o aplicativo deve ter um botão para Finalizar formulário. Ao clicar nesta opção o aplicativo deve ter um *loading* de 5 segundos mostrando que os dados estão sendo analisados, possibilitando a aplicação fazer algumas verificações: A primeira é fazer com que o aplicativo faça uma requisição ao banco de dados para verificar se o formulário que está sendo enviado ao banco de dados já possui algum cadastro; se sim, o sistema deve verificar quais das alternativas foram alteradas; ao finalizar a comparação o aplicativo deve mostrar ao entrevistador uma mensagem informando quais os campos estão sendo alterados, assim como um botão possibilitando ao entrevistador concordar ou discordar com as alteração; A segunda verificação seria consultar o acesso a internet no dispositivo, caso o dispositivo tenha acesso, o formulário deve ser enviado ao servidor de banco de dados retornando com a mensagem de "Enviado com Sucesso", caso contrário o aplicativo deve mostrar uma mensagem informando que o formulário será armazenado no dispositivo, ou seja, o formulário só será enviado ao servidor de banco de dados após o dispositivo ter acesso à internet. Após o aplicativo fazer todas as verificações possíveis e ter salvo o formulário, o entrevistador deve ser direcionado a tela de listagem dos formulários respondidos.

Fonte: Autor do Trabalho (2024).

3.5 REQUSITOS NÃO FUNCIONAIS

Os requisitos não funcionais definem as características de qualidade que o sistema deve ter e que estão relacionadas às suas funcionalidades. De acordo com SOMMERVILLE (2018), esses requisitos não funcionais normalmente especificam ou restringem as características do sistema como um todo.

Quadro 2 – Requisitos Não Funcionais.

ID	Nome	Descrição
RNF01	Aplicativo com interface mobile	O aplicativo será implementado para acesso via *mobile*.
RNF02	Segurança de acesso	O aplicativo utilizará criptografia md5 ou *hash* para geração de *token*, que servirá para fazer autenticação do usuário no aplicativo e verificar se ele já possui algum cadastro.
RNF03	Banco de dados	O aplicativo deverá usar o banco de dados *PostgresSQL* onde serão armazenadas as informações e os formulários dos usuários.
RNF04	*Design* da *interface*	O aplicativo deverá ter uma interface responsiva com "textos, imagens e ícones" adaptados ao tamanho de cada tela.
RNF05	Interface amigável	O aplicativo deve ter uma interface amigável com as cores, o posicionamento de botões, texto, imagens e ícones bem simples para facilitar a ação e entendimento do usuário.
RNF06	Tempo de resposta do aplicativo	O aplicativo deverá ter um tempo de resposta para todas as requisições que o usuário executar. No caso de algum erro na hora da requisição, o aplicativo deverá mostrar um *alert* com uma mensagem explicando o que o usuário deve fazer.
RNF07	Facilidade no uso	O aplicativo deve ser implementado considerando uma *interface* gráfica amigável que respeite as padronizações de Usabilidade e Experiência do Usuário;

RNF08	Aplicativo em manutenção	O aplicativo deverá mostrar uma mensagem de manutenção mostrando que o aplicativo está fora do ar. Essa mensagem deve ser mostrada em um *alert* explicando sobre a possível manutenção.
RNF09	Restrições de dispositivos	O aplicativo deverá ter restrições de dispositivos, ou seja, só será permitido o uso do aplicativo em um único dispositivo, por usuário. Uma das propostas de utilizar o *token* é para saber se o usuário já possui algum cadastro em algum dispositivo.
RNF10	Ambiente de Desenvolvimento	O ambiente de desenvolvimento utilizado será *oVisual Studio Code*, que permite a criação de aplicações *mobile*, entre outras.
RNF11	Linguagem de Programação	Será utilizada a biblioteca *React Native* que utiliza *JavaScript* e *TypeScript* para criar aplicações *mobile* e *desktop*.
RNF12	Controle de versionamento	Deve-se utilizar a plataforma *GitHub* para fazer o controle do código-fonte, arquivos e versões do app. Essa ferramenta permite que os programadores possam versionar seus códigos, ou seja, tenham um histórico do processo de desenvolvimento do seu app. Além disso, possibilita que os desenvolvedores trabalhem de forma remota no projeto, enviando as alterações para o repositório do *GitHub*.
RNF13	Linguagem de programação do *BackEnd*	O backend deve ser escrito utilizando o *NodeJS*.
RNF14	Linguagem de programação	O aplicativo deve ser desenvolvido utilizando o *framework React Native*.
RNF15	Principios RESTfull	As operações do backend devem ser consumidas via serviço *RESTful*.
RNF16	Formato de comunição (*Back e FrontEnd*)	A estrutura de comunicação entre o aplicativo e o *backend* deve ser o *JSON*.

RNF17	Arquitetura	O aplicativo deve ser na arquitetura cliente/servidor
RNF18	Hospedagem	O aplicativo deve estar disponível 99.99% ao mês, operando 24 horas por dia e 7 dias da semana, desconsiderando-se apenas as paradas programadas para manutenção preventiva, a fim de manter a API e os demais arquivos do aplicativo operando normalmente.

Fonte: Autor do trabalho (2024).

3.6 REGRA DE NEGÓCIO

As regras de negócio definem como o sistema deverá se comportar em ocasiões diferentes. Conforme afirma Kamada (2006), as regras de negócio, que delimitam e governam a realização dos negócios da organização e são fundamentais, pois, no conjunto, representam o seu comportamento. Elas podem ser simples ou complexas e podem envolver outros requisitos.

Quadro 3 – Regras de Negócio.

ID	Nome	Descrição
RN01	Formulário	Quando o usuário clicar no botão Finalizar formulário, o aplicativo precisa fazer um levantamento das perguntas respondidas pelo entrevistado, porque de acordo com a quantidade de respostas fornecidas o aplicativo deve mostrar o grau em que a doença se encontra (leve, moderado, ou grave).
RN02	Obrigatoriedade dos Termos do Aplicativo	Ao clicar no botão iniciar formulário, o aplicativo deverá mostrar uma tela com dois *ChekList*: um para o Termo de Confidencialidade e Sigilo e o outro com o Termo de Consentimento de dados Pessoais, nesse sentido, o formulário só deve iniciar após entrevistador e entrevistado estarem cientes destes termos.
RN03	Armazenamento local	Para que os formulários possam ser aplicados em locais ou áreas de difícil acesso a uma internet de qualidade, principalmente na região de ilhas, vilas, comunidades quilombolas, faz-se necessário implementar no aplicativo uma função que permita a aplicação destes

		formulários utilizando a estratégia *OfflineFirst,* isto é, sem acesso à internet Obs: Após clicar em finalizar formulário o aplicativo mostra um loading de 5 segundos permitindo que o aplicativo verifique e confirme que não há conectividade de internet, com isso ele salvará as respostas coletadas no próprio dispositivo.
RN04	Cadastro	Para o usuário ter acesso às informações e funcionalidades do aplicativo, primeiro ele precisa realizar seu cadastro no aplicativo.
RN05	Listagem dos Formulários	A tela de listagem dos formulários deve ter os componentes personalizados, isto é, a listagem deve ter uma cor que represente a finalidade daquele formulário:se o formulário estiver pendente deverá ficar com a cor alaranjada; caso tenha sido enviado ao banco de dados, deve ficar na cor verde.
RN06	Níveis dos Sintomas	De acordo com o levantamento feito pelos desenvolvedores juntamente com alguns enfermeiros que trabalham nas UBS em Cametá, o nível dos sintomas (leve, moderando, e grave) são classificados de acordo com o número e a gravidade dos sintomas informados. O formulário deve possuir um mecanismo para analisar as respostas do usuário e mostrar o grau da doença.
RN07	Conexão com a Internet	Sempre que o usuário acessar o aplicativo, o sistema deve verificar a conectividade do aparelho com a *internet*, caso o dispositivo esteja sem internet ele apresentará mensagens já programadas, por exemplo: 1ª – Ao clicar em Notícias Relacionadas a COVID-19 – "você está sem internet". 2ª – Casos de COVID-19 em Cametá – ao clicar na tela inicial para verificar a atualização dos casos, aparecerá a mensagem "você está sem internet".

		3ª – Ao iniciar o formulário de perguntas o aplicativo deve mostrar a mensagem: aparecerá a mensagem "você está sem internet", em seguida o usuário será informado de que os dados serão salvos no dispositivos. 4ª – Caso o usuário deixe a opção de dados móveis ou *wi-fi* ligado, automaticamente o aplicativo irá tentar enviar o formulário, entretanto, se o tempo de requisição do envio ultrapassar 5 segundos o aplicativo deve mostrar uma mensagem ao usuário informando que o formulário foi salvo no dispositivo.
RN08	Duplicidade de conta	Só pode existir uma conta de usuário por e-mail.

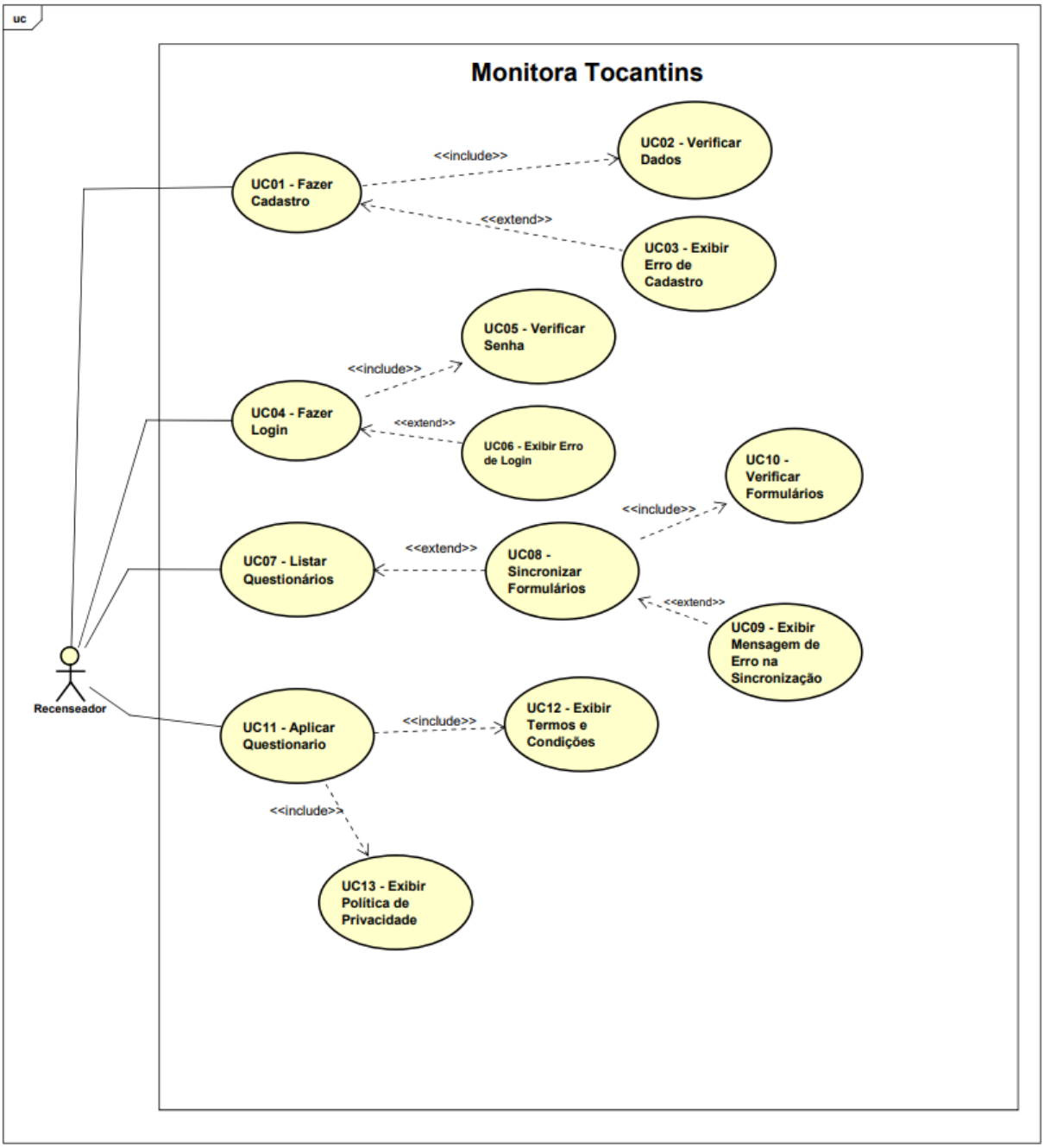
Fonte: Autor do trabalho (2024)

3.7 DIAGRAMA DE CASO DE USO

O modelo de Casos de Uso é uma representação das funcionalidades do sistema, que podem ser observáveis externamente. Segundo BEZERRA (2015), O modelo de caso de uso molda os requisitos funcionais do sistema. Para a criação do diagrama de caso de uso da aplicação, foi utilizada a ferramenta ASTAH (2024), focada na elaboração de diagramas da *UML*.

A Figura 5 abaixo representa o diagrama de casos de uso do Recenseador do Projeto Monitora Tocantins, onde são ilustrados os papeis de interação com o aplicativo e ações que podem ser executadas.

Figura 5 – Diagrama *UML*.



Fonte: Autor do trabalho (2024)

Na Figura 5, podemos observar o diagrama de caso de uso, onde o recenseador é responsável pelo uso do aplicativo para cadastro de usuário, aplicação de formulário, atualização das informações do usuário, listar formulários, envio/sincronização dos formulários para o banco de dados.

Casos de Uso relacionado ao ator Recenseador

Caso de uso 01: Cadastrar.

- ✓ Contexto de uso: O usuário não possuir uma conta cadastrada no banco de dados do aplicativo.
- ✓ Precondições: Devido o caso de uso “Cadastrar” possuir a obrigatoriedade do caso de uso “Verificar Dados”, Os dados precisam ser validados antes de fazer o cadastro.
- ✓ Condições e sucesso: CPF e e-mail não ter sido registrado no banco de dados, todos os dados serem preenchidos de forma correta, com isso, o recenseador será redirecionado para tela de login.
- ✓ Fluxo alternativo: Devido o caso de uso “Cadastrar” possuir o caso de uso “Exibir ERRO de Cadastro”, se o CPF já possuir cadastro, ou os dados estão inválidos, conta não é registrada e uma mensagem de erro é exibida ao realizar o cadastro.
- ✓ Ator principal: Recenseador.
- ✓ Condição para começar: Clicar na opção “Criar Conta”.
- ✓ Dados Coletados:
  - – Nome
  - – CPF
  - – Email
  - – Senha
  - – Endereço
  - – Data de nascimento
  - – Gênero

Caso de Uso 04 – Login.

- ✓ Contexto de uso: Utilizado para fazer a autenticação do recenseador para que ele possa acessar o aplicativo.
- ✓ Precondições: Devido o caso de uso “Login” possuir a obrigatoriedade do caso de uso “Verificar senha”, é preciso que usuário do aplicativo Possua cadastro no banco de dados.
- ✓ Condições sucesso: Inserir e-mail e senha corretos, com isso será validado as informações fornecidas para que em seguida o recenseador seja redirecionado a sua conta no aplicativo.

- ✓ Fluxo alternativo: Devido o caso de uso “Login” possuir o caso de uso “Exibir ERRO de Login”, ao inserir e-mail ou senha incorretos ou inválidos, uma mensagem informará ao recenseador sobre o erro.
- ✓ Ator principal: Recenseador.
- ✓ Condição para começar: Iniciar o aplicativo ou fazer logout.

Caso de Uso 07 – Listar Questionário.

- ✓ Contexto de uso: Utilizado para listar todos os questionários coletrados pelo rescenseador.
- ✓ Precondições: Devido o caso de uso “Listar Questionário” possui as obrigatoriedades dos casos de uso “Sincronizar Formulário e Verificar Formulários”, é precoso o restar logado na sua conta, e clicar no botão realizar “Sincronização” dos formulários. Outro ponto importante é fazer a verificação de cada formulário.
- ✓ Condições sucesso: Ter formulários cadastrados, tanto no banco de dados do servidor como no próprio dispositivo (pedentes).
- ✓ Fluxo alternativo: Caso os formulário estejam incompletos, ou o dispositivo não tenha internet para fazer a sincronização dos formularios, será exibida uma mensagem informando ao recenseador sobre o possível erro.
- ✓ Ator principal: Recenseador.
- ✓ Condição para começar: Clicar no botão “IoT Imuniza – 2022”.

Caso de Uso 11 – Aplicar Questionário.

- ✓ Contexto de uso: Utilizado para coletar as respostas dos entrevistados.
- ✓ Precondições: Devido o caso de uso “Aplicar Questionário” possuir os casos de uso Obrigatórios “Termos e Condições e Politica de Privacidade”, o recenseador precisar estar logado na sua conta antes de iniciar o formulário propriamente dito, o recenseador e entrevistado precisam estar cientes com os “Termos e Condições e Politica de Privacidade”.
- ✓ Condições sucesso: Precisa ser respondida todas as perguntas do formulários.
- ✓ Fluxo alternativo: Caso os formulário estejam incompletos, será exibida uma mensagem informando ao recenseador sobre o possível erro.
- ✓ Ator principal: Recenseador.

- ✓ Condição para começar: Clicar no botão com formato de uma (+), que significa novo “Formulário”.

3.8 MODELO LÓGICO

Para o planejamento do armazenamento dos dados do aplicativo foi realizadas a elaboração de um diagrama do modelo de entidade relacionamento que mostra as entidades e relacionamentos envolvidos na especificação do aplicativo. A Figura 6 apresenta o modelo de entidade relacionamento gerado através da ferramenta *MySQL Workbench*, ilustrando as entidades e relacionamentos utilizados no sistema.

As informações contidas no diagrama apresentam uma estruturação que utiliza a língua inglesa como padrão na sua escrita, buscando alcançar um maior número de colaboradores.

Desta forma, o modelo lógico do projeto Banco de Ideias é estruturado em 5 (cinco) tabelas, sendo 4 (quatro) de entidades e 1 (uma) de relacionamento, contendo seus atributos, tipos de variáveis e tamanho de cada variável. As tabelas são: Formulários (forms_responses), Arquivos exportados (exported_files), Usuários (users), Endereços (addresses) e Formulários e Respostas Sobre Usuários (forms_responses_on_users).

Todas as tabelas deste diagrama possui campos com as mesmas finalidade, exemplo: o atributo id, identificado com o ícone de uma chave amarela na frente do nome, é do tipo VARCHAR com tamanho 191 caracteres tem a função de permitir que a apliação faça uma validação dos dados do usuário ou realize uma consulta no banco de dados. Outro ponto importante a se detalhar são os campos created_at, updated_at e deleted_at, esses atributos todos do tipo DATETIME com tamanho de 3 (três) caracteres com a função de monitorar os eventos que são realizado no banco de dados, desde uma inserção de dados, atualização e até a exclusão dos dados do banco, pois eles possibilitam o controle das ações que são requisitadas pelos usuários.

Na tabela forms_responses há os campos: age_group, do tipo VARCHAR com tamanho 191 caracteres; reason_not_cpf do tipo VARCHAR com tamanho 191 caracteres; schol_level do tipo VARCHAR com tamanho 191 caracteres; religion do tipo VARCHAR com tamanho 191 caracteres; comorbidity do tipo TINYINT com tamanho de 1 caractere; diabetes do tipo TINYINT com tamanho de 1 caractere; heart_problem do tipo TINYINT com tamanho de 1 caractere; kidney_disease do tipo TINYINT com tamanho de 1 caractere; thyroid do tipo TINYINT com tamanho de 1 caractere; obesity do tipo TINYINT com tamanho de 1 caractere; other_comorbidity do tipo VARCHAR com tamanho 191 caracteres; affected_covid_19 do tipo

TINYINT com tamanho de 1 caractere; diagnostic_confirmation do tipo TINYINT com tamanho de 1 caractere; time_interval do tipo VARCHAR com tamanho 191 caracteres; diagnostic_method do tipo VARCHAR com tamanho 191 caracteres; treatment_place do tipo VARCHAR com tamanho 191 caracteres; hospital_treatment do tipo VARCHAR com tamanho 191 caracteres; covid_sequelae do tipo VARCHAR com tamanho 191 caracteres; vaccinated do tipo TINYINT com tamanho de 1 caractere; vaccine_doses do tipo VARCHAR com tamanho 191 caracteres; reason_not_to_take do tipo VARCHAR com tamanho 191 caracteres; lost_family_member do tipo TINYINT com tamanho de 1 caractere; affected_covid_after_vaccinated do tipo TINYINT com tamanho de 1 caractere; rehabilitation_sequelae do tipo TINYINT com tamanho de 1 caractere; treatment_rehabilitation_sequelae do tipo VARCHAR com tamanho 191 caracteres; opnion_prevention_measures do tipo VARCHAR com tamanho 191 caracteres; covid_information do tipo VARCHAR com tamanho 191 caracteres; maintained_family_income do tipo TINYINT com tamanho de 1 caractere; received_social_assistance do tipo TINYINT com tamanho de 1 caractere; recovered_family_income do tipo TINYINT com tamanho de 1 caractere; Family_in_debt do tipo TINYINT com tamanho de 1 caractere; e como chave-estrangeira, user_common_id do tipo VARCHAR com tamanho 191 caracteres, que referencia a tabela users.

Temos ainda a tabela exported_files com os campos, user_id do tipo VARCHAR com tamanho 191 caracteres, que é chave-estrangeira da tabela users; e o campo filename do tipo VARCHAR com tamanho 191 caracteres.

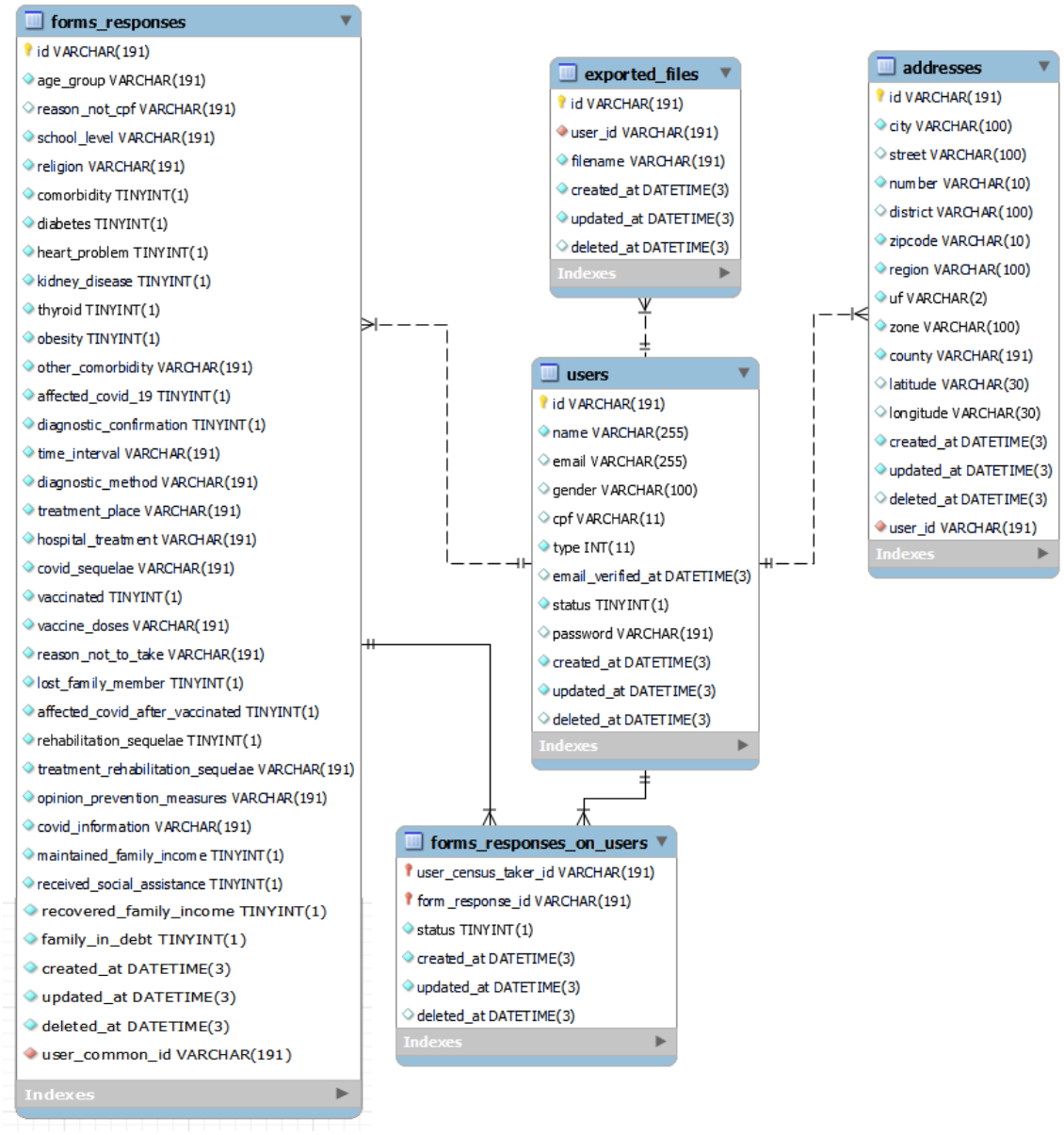
Na tabela users temos os atributos, name do tipo VARCHAR com tamanho 255 caracteres; email do tipo VARCHAR com tamanho 255 caracteres; gender do tipo VARCHAR com tamanho 100 caracteres; cpf do tipo VARCHAR com tamanho 11 caracteres; type do tipo INT com tamanho de 11 caracteres; email_verified_at do tipo DATETIME com tamanho de 3 caracteres; status do tipo TINYINT com tamanho de 1 caractere; password do tipo VARCHAR com tamanho 191 caracteres;

Há também a tabela adresses com os campos, city do tipo VARCHAR com tamanho 100 caracteres; street do tipo VARCHAR com tamanho 100 caracteres; number do tipo VARCHAR com tamanho 10 caracteres; district do tipo VARCHAR com tamanho 100 caracteres; zipcode do tipo VARCHAR com tamanho 10 caracteres; region do tipo VARCHAR com tamanho 100 caracteres; uf do tipo VARCHAR com tamanho 2 caracteres; zone do tipo VARCHAR com tamanho 100 caracteres; count do tipo VARCHAR com tamanho 191

caracteres; latitude do tipo VARCHAR com tamanho 30 caracteres; longitude do tipo VARCHAR com tamanho 30 caracteres; como chave-estrangeira temos o campo user_id do tipo VARCHAR com tamanho 191 caracteres que faz referencia a tabela users;

Por fim temos a tabela forms_responses_on_users que surgiu do relacionamento de N para N, entre as tabelas forms_responses e users. Essa tabela é composta por 2 (dois) campos que são considerados chaves-compostas, sendo um o user_census_taker_id do tipo VARCHAR com tamanho 191 caracteres; o outro atributo é o form_response_id do tipo VARCHAR com tamanho 191 caracteres; temos também o status, do tipo TINYINT com tamanho de 1 caracter.

Figura 6 – Diagrama entidade-relacionamento.



Fonte: Autor do trabalho (2024)

3.9 DELIMITAÇÕES DO DESENVOLVIMENTO DO APLICATIVO

Inicialmente:

1 O aplicativo não estará disponível nas lojas de dispositivos moveis.
2 O aplicativo não conterá quaisquer tipos anúncios.
3 O aplicativo não estará disponível para versão *web*.
4 O aplicativo não será disponibilizado para a versão do sistema *iOS*.
5 O aplicativo não estará disponível aos usuários comuns.

4. DESENVOLVIMENTO

Neste capítulo será apresentado a descrição das ferramentas e tecnologias que foram utilizadas, além do cronograma de atividades para coleta de requisitos, a criação do diagrama de caso de uso *UML*, elaboração do modelo físico do banco de dados, e por fim criação do protótipo de telas do aplicativo.

4.1 FERRAMENTAS E TECNOLOGIAS

4.1.1 FIGMA

O *Figma*, Lançado em 2016, revolucionou no processo de *design*, proporcionando uma abordagem inovadora e eficiente para a criação de interfaces de usuário e experiências digitais, por ser uma plataforma de design colaborativo baseada na nuvem que se tornou uma ferramenta essencial para profissionais da área de design, pois, oferece uma variedade de ferramentas poderosas (*design* de *interface*, prototipagem e *design* de experiência do usuário (*UX*)).

Essa ferramenta permite que os usuários acessem e colaborem em projetos de qualquer lugar, a qualquer momento. Isso elimina as limitações geográficas, permitindo que equipes distribuídas trabalhem de maneira eficiente, contribuindo para uma colaboração mais integrada GARRETT (2023).

O figma também se destaca na criação de componentes reutilizáveis que são bastante utilizados pelo *framework React Native*, o que é crucial para manter consistência visual em projetos extensos. Os designers podem compartilhar protótipos interativos com partes interessadas e receber comentários diretamente no projeto, melhorando a interação entre os membros da equipe e *stakeholders*.

Figura 7 – Logo *Figma*.

Fonte: https://www.figma.com/community/file/1252952214981489076

4.1.2 GITHUB

Sua finalidade é compartilhar e fazer gerenciamento das versões do código-fonte, por isso, decidiu-se utilizar o *Github*, que é uma das principais plataformas para este fim. Github (2024).

O *GitHub* é uma plataforma online que hospeda projetos *Git* e permite o compartilhamento de projetos entre os desenvolvedores. Nele estão hospedados mais de 96 milhões de projetos, desde trabalhos escolares até sistemas de código aberto, como o Linux. (CONTA, 2018).

Figura 8 – Logo do Github.

Fonte: https://github.com/logos

4.1.3 GIT

Criado por *Linus* Torvalds em 2005, o Git oferece um ambiente eficiente e seguro para monitorar mudanças em arquivos e projetos. Com ele, os desenvolvedores podem rastrear e desfazer alterações, trabalhar simultaneamente em ramos de código diferentes e simplificar a colaboração entre membros de equipes de desenvolvimento Git (2023).

Git é um sistema de controle de versão distribuído, gratuito e de código aberto, projetado para lidar com tudo, desde projetos pequenos até muito grandes, com rapidez e eficiência Git (2024).

É uma ferramenta importantíssima para o desenvolvimento de softwares nos dias de hoje, principalmente quando se trata de trabalhar em equipe e há a necessidade de fazer alterações no mesmo projeto por todos os colaboradores de forma simultânea e organizada.

Figura 9 – Logo do Git.

Fonte: https://git-scm.com/downloads/logos

4.1.4 CSS

É responsável em trazer uma melhor aparência na criação do *layout* das telas, tornando-as mais simples, por ser uma ferramenta que é utilizada para estilizar páginas *web*. Nesse sentido, os desenvolvedores que já estão acostumados a trabalhar na estilização de *sites*, ao construírem telas no *React Native* terão uma experiência muito parecida com a forma em que se estrutura e trabalha em projetos *web*. Existem três formas de estilizar os elementos da interface em *React Native*, são elas:

> 1 – Propriedade "*styles*" que é encontrada na grande maioria dos componentes fornecidos pela própria ferramenta.
>
> 2 – *StyleSheet* é uma ferramenta especial disponibilizada para definir estilos fora do *JSX* e, em seguida, referenciando dentro da propriedade "*styles*".
>
> 3 – *Styled-components* é uma biblioteca externa e através dela é possível de forma sucinta escrever *CSS* dentro do código *JS*. Esse método é denominado *CSS-in-JS*.

Segundo FERREIRA (2016), o *CSS* é um conjunto de regras de estilos aplicado ao conteúdo para trabalhar o design da página. O *CSS* tem como pontos fortes controlar a posição das camadas utilizadas, controlar a formação, o tamanho e a cor dos elementos da página.

Figura 10 – Logo do CSS.

Fonte: https://www.flaticon.com/br/icones-gratis/logotipo-css

4.1.5 JAVASCRIPT

Segundo FLANAGAN (2014), a linguagem *JavaScript* é derivada da linguagem *Java*, das funções de primeira classe *Scheme* e da herança baseada em protótipos de *Self*. *JavaScript* é uma linguagem de alto nível, dinâmica, interpretada e não tipada, conveniente para estilos de programação orientados a objetos e funcionais." Por isso, não há dúvidas de que o uso do *JavaScript* é uma das grandes vantagens do *React Native*, visto que é uma linguagem comumente difundida e amplamente utilizada por desenvolvedores *web*. Dessa forma torna-se prático construir aplicativos móveis com técnicas já consolidadas em vez de ter que aprender uma nova linguagem e ambiente de desenvolvimento para cada plataforma (HJORT, 2020).

Figura 11 – Logo do JavaScript.

Fonte: https://www.javascript.com/

4.1.6 TYPEACRIPT

É uma linguagem que é um superconjunto de *JavaScript*. Ela apresenta ferramentas e formas mais eficientes para criar códigos. Ele adiciona recursos, como tipagem e orientação a objetos que não estão presentes do *Javascript* (MELO, 2021). No código-fonte ela é utilizada para auxiliar principalmente na tipagem das variáveis, garantindo segurança durante a fase de desenvolvimento, tanto no *backend* quanto no *frontend*.

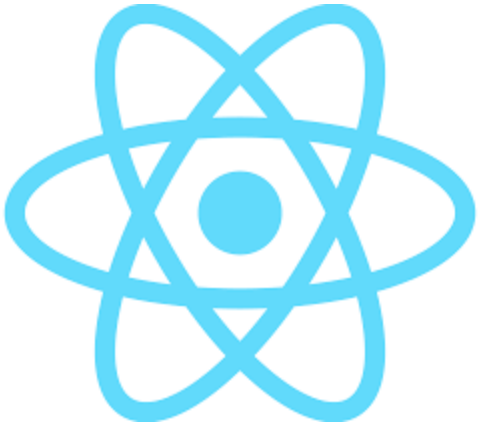
Figura 12 – Logo do TypeScript.

Fonte: TypeScript: JavaScript With Syntax For Types.

4.1.7 REACT NATIVE

É um *framework* que consiste em uma série de ferramentas que viabilizam a criação de aplicações móveis nativas capazes de converter seu código *Javascript* e *JSX* para o código nativo da linguagem do SO desejado (*Android* ou *iOS*). Essa biblioteca não é dependente de tecnologias *Web* como muitos outros *frameworks* de desenvolvimento multiplataforma. Pois ela permite acesso a componentes nativos como câmera, *GPS*, e-mail e agenda. De acordo com a documentação NATIVE (2024), o React Native usa o *Javascript* para acessar a API e criar a aparência e o comportamento da interface de usuário (*UI*) usando componentes da biblioteca *React do Javascript*.

Figura 13 – Logo do React Native.



Fonte: https://reactnative.dev/

4.1.8 NODE JS

É uma plataforma projetada para criar aplicativos de rede escalonáveis que possibilita aos desenvolvedores usarem a mesma linguagem em ambas as camadas de aplicação. Conforme Lopes (2014), *Node.js* é uma plataforma de desenvolvimento de aplicações do lado do servidor *(server-side)* que se baseia na utilização do interpretador V8 *JavaScript Engine* do *Google* que

utiliza apenas código em *JavaScript*, o mesmo utilizado pelo navegador *Google Chrome*. (apud CAMPOS; PEREIRA; CARVALHO; et al, 2016).

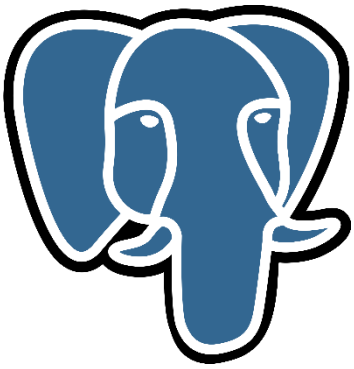
Figura 14 – Logo do Nodejs.

Fonte: https://nodejs.org/en/

4.1.9 POSTGRESQL

É um banco de dados relacional de código aberto que utiliza um sistema de gerenciamento amplamente utilizado em aplicações modernas do backend. Os principais benefícios do *PostgreSQL* são disponibilizar uma estrutura robusta, eficiente e confiável, por meio do prisma.

Figura 15 – Logo PostgresSQL.



Fonte: https://www.postgresql.org/

4.1.10 VSCODE

É um Ambiente de Desenvolvimento Integrado mais conhecido como *IDE*, que possui integração facilitada com todo o ferramental que permite a instalação de diversas bibliotecas amplamente utilizadas pela comunidade de desenvolvedores, o que possibilita o reaproveitamento de código e, consequentemente, acelera o processo de desenvolvimento.

Visual Studio Code ou *VS code* como conhecemos, é um editor de código destinado ao desenvolvimento de aplicações. Além de ser gratuito, ele é open source (código aberto) e oferece suporte a várias linguagens de programação, além de ser altamente configurável,

permitindo que os desenvolvedores personalizem a interface e incluam funcionalidades de acordo com suas preferências.

Figura 16 – Logo VSCode.

Fonte: https://code.visualstudio.com/

4.1.11 JAVASCRIPT OBJECT NOTATION (JSON)

É uma formatação utilizada para estruturar dados em formato de texto e transmiti-los de um sistema para outro, como em aplicações cliente-servidor ou em aplicativos móveis, é um formato leve para troca de informações e dados entre sistemas. Mesmo contendo o nome *JavaScript*, ele não precisa ser utilizado somente com *javascript*, pode utilizá-lo com outras linguagens. (GAMA, 2011).

Figura 17 – Logo JSON.

Fonte: https://devtools.com.br/

4.1.12 API RESTFUL

É uma interface que possibilita a comunicação entre diferentes sistemas de *software* pela internet através de protocolos (*GET*, *POST*, *PUT*, *DELETE*), seguindo os princípios da arquitetura *REST*.

É uma condição que permite a aplicação dos princípios de *REST* que é um estilo arquitetônico que estabelece padrões entre sistemas de computador na web, facilitando a comunicação entre eles.

De acordo com o *RedHat* (2024), para que uma *API* seja considerada *RESTful*, ela deve atender a alguns critérios:

- ✓ Arquitetura cliente-servidor composta por clientes, servidores e recursos, com solicitações gerenciadas por *HTTP*.
- ✓ Comunicação stateless entre cliente e servidor significa que nenhum dado do cliente é armazenado entre requisições *GET*. Cada requisição é independente e não depende das anteriores.
- ✓ Dados armazenáveis em cache que simplificam as interações entre cliente e servidor.
- ✓ Ter uma interface consistente entre componentes para que as informações sejam transferidas de maneira padronizada. Para isso, é necessário que:
  - ➢ Os recursos solicitados sejam identificáveis e distintos das representações enviadas ao cliente.
  - ➢ Os recursos e suas representações sejam diferentes, mas a representação contenha as informações necessárias para o cliente manipular o recurso.
  - ➢ As mensagens autoexplicativas retornadas para o cliente tenham informações suficientes para Detalhar como ele deve processá-las.
  - ➢ Exista um hipertexto ou hipermídia disponível, de forma que, após acessar um recurso, o cliente possa usar hiperlinks para localizar todas as outras ações disponíveis.
- ✓ Um sistema em camadas que organiza cada tipo de servidor (responsável pela segurança, balanceamento de carga etc.) envolvido na recuperação das informações solicitadas em hierarquias invisíveis ao cliente.

Código sob demanda (opcional): capacidade de enviar código executável do servidor para o cliente quando solicitado, ampliando sua funcionalidade.

4.1.13 ANDROID STUDIO

Essa *IDE* oferece suporte total ao *Kotlin*, *Java* e *C++*, além de oferecer suporte a recursos avançados, como a criação de layouts responsivos para adaptação a diferentes tamanhos de tela. A acessibilidade no *Android* nativo é tratada de uma maneira mais clara, pois fornece um editor de código robusto, facilitando a criação e correção de código. Por permitir uma criação de uma interface de usuário intuitiva e amigável o *Android Studio* contribui para uma experiencia de desenvolvimento mais intuitivo. É o emulador *Android* incluso no *Android Studio* que possibilita aos desenvolvedores testarem seus aplicativos em uma variedade de dispositivos virtuais, simulando diferentes tamanhos de tela. Isso é crucial para garantir que os aplicativos ofereçam uma experiência consistente em diversos dispositivos, Fernando (2020).

É uma *IDE* indispensável para os desenvolvedores na elaboração e construção de aplicativos de alta qualidade, pois de acordo com Adash (2023), essa ferramenta permite que os desenvolvedores identifiquem e resolvam problemas de desempenho em seus aplicativos, otimizando a eficiência e garantindo uma experiência de usuário.

Figura 18 – Logo Android Studio.

Fonte: https://developer.android.com/?hl=pt-br

4.1.14 ASTAH COMMUNITY

O *Astah Community* é um *software* gratuito para a criação de diagramas e modelagens de acordo com a *UML* (*Unified Modeling Language* – Linguagem de Modelagem Unificada). Essa ferramenta possui uma interface de fácil utilização, dividida em várias seções, cada uma com sua finalidade, que possibilita o desenvolvimento dos diagramas: classes, casos de uso, sequência, de atividades, entre outros.

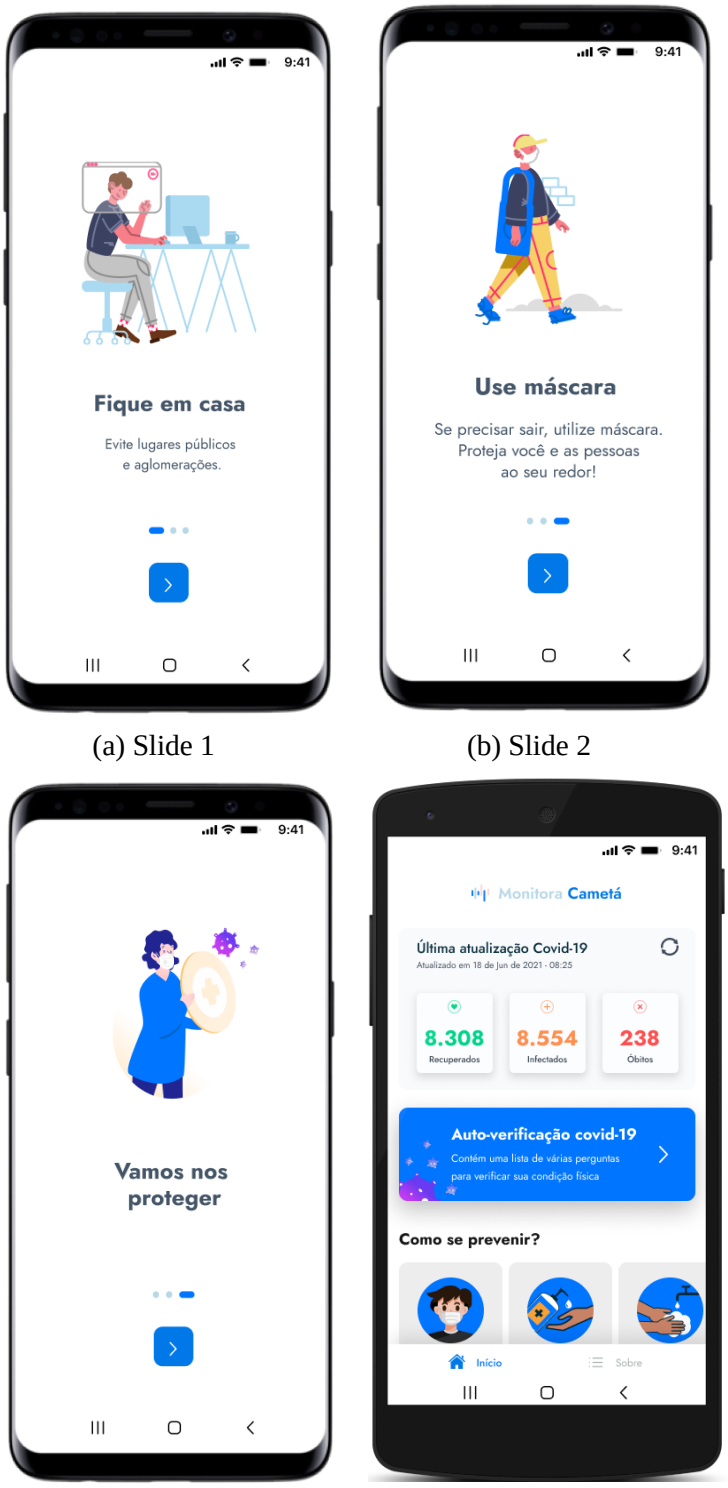
Figura 19 – Logo Astah.

Fonte: https://astah.net/

4.2 PROTOTIPAGEM DO APLICATIVO

Outra etapa importante do desenvolvimento do aplicativo foi a prototipagem realizada utilizando a ferramenta Figma (2022), onde foi construído o layout do aplicativo Monitora Tocantins, com uma interface objetiva e simples, ícones e imagens representativos, como mostra o quadro de figuras abaixo:

Figura 20 – Exemplos da Prototipagem da Interface do Aplicativo Monitora Tocantins.



(a) Slide 1 (b) Slide 2

(c) Slide 3 (d) Tela Inicial

Fonte: Autor do trabalho (2024)

Nas figuras ***a, b, c*** temos telas que trazem informações referentes à COVID-19, além de orientações sobre prevenção da doença.

d) Tela Inicial: Na parte superior encontra-se um cartão que contém informações atualizadas sobre os casos confirmados, recuperados e número de óbitos ocorridos pela COVID-19. Abaixo, o usuário poderá selecionar o botão de “Autoverificação COVID-19”, no qual terá acesso ao questionário sobre os sintomas da doença. Na sequência há duas listas na horizontal, a primeira fala sobre as medidas de prevenção e a segunda, mostra os principais sintomas da doença. Além disso, podemos visualizar na parte inferior da tela os botões “Início e Sobre”. Ao clicar no primeiro temos a funcionalidade de acessar a tela inicial do aplicativo; ao clicar no botão “Sobre” o usuário poderá visualizar os Termos e Condições, Política de Privacidade do aplicativo, e ainda poderá reportar falhas, bugs ou fazer sugestões de melhorias.

No decorrer do desenvolvimento foram feitas algumas alterações visuais que se observaram necessárias para a melhoria do aplicativo, como por exemplo, a criação do botão de configuração referente às informações do usuário e do aplicativo; na tela inicial, a lista (vertical) de notícias sobre a COVID-19, a lista dos formulários que foram aplicados, botão de sincronização dos formulários e *cards* ilustrativos que mostram a contagem dos formulários coletados.

5. RESULTADOS

Os principais resultados obtidos nesse estudo foram o desenvolvimento de um aplicativo censo voltado para a análise dos impactos da COVID-19 na região do Baixo Tocantins, utilizando modelos importantes de processos e métodos consolidados desde o início da criação do software até a sua evolução. Nesse sentido, a organização das atividades do desenvolvimento do aplicativo tomou como base para essa criação o modelo incremental que intercala as atividades de especificação, desenvolvimento e validação.

Outro resultado importante foi alcançar a estabilidade do aplicativo, pois, a partir do momento em que se fizeram as alterações necessárias, ele se manteve estável, ou seja, não apresentou mais problemas que interferissem em suas funcionalidades. Dessa forma apresentamos o aplicativo Monitora Tocantins apto a realizar coleta de dados dos usuários do Baixo Tocantins e fornecer informações concretas a respeito da COVID-19.

5.1 OS TESTES

Após finalizar a criação do aplicativo foram realizados testes, os quais foram divididos em 2 (duas) etapas, sendo a primeira feita durante a produção e a segunda, durante a implementação do aplicativo. Durante a criação do produto, os testes eram feitos pelos próprios desenvolvedores de forma incremental, ou seja, a cada módulo finalizado os teste eram feitos de maneira técnica para se observar o fluxo das funcionalidades criadas. As falhas e erros encontrados eram logo corrigidas ou implementavam-se as melhorias necessárias. A segunda etapa dos testes coube aos alunos bolsistas do projeto para observarem possíveis erros gramaticais, imagens com resoluções ruins/distorcidas, mau posicionamento e possíveis *bugs* (botão não funcionando corretamente, rotas com conflito ou sem interação, erros de requisições, entre outros), e assim, reportá-los aos desenvolvedores do aplicativo para as devidas correções/melhorias.

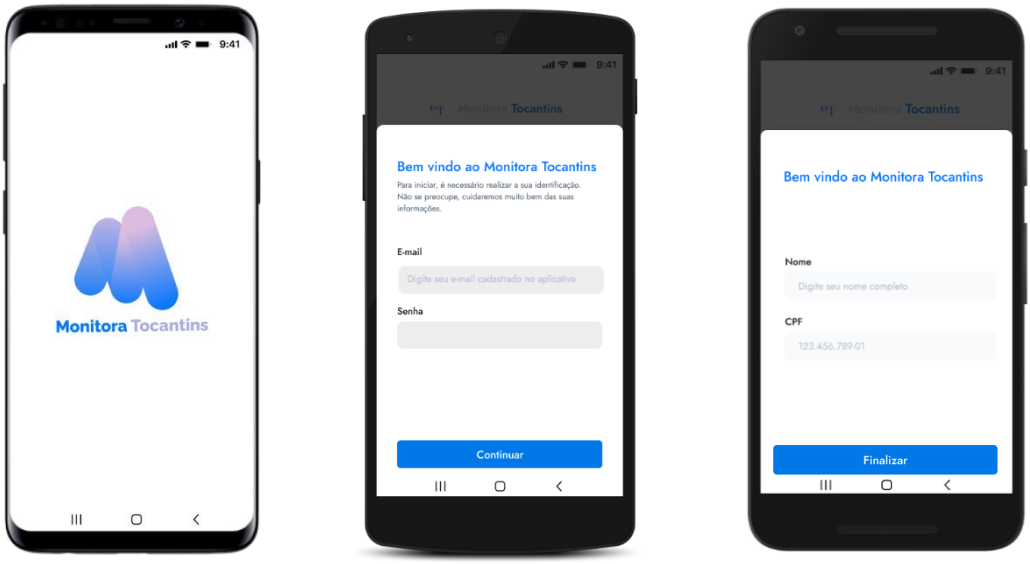
A fase de testes foi importante para desenvolver uma interface limpa, simples, de fácil entendimento e utilização para proporcionar aos usuários uma interação atrativa e confortável.

5.2 O APLICATIVO

Esta versão do aplicativo foi desenvolvida para os dispositivos *Android* utilizando o *framwork React Native*, banco de dados *PostgreSQL*. As imagens do aplicativo exibidas abaixo foram capturadas através do emulador presente no *Android Studio*, que simula o dispositivo

Google Pixel 3. As telas principais foram selecionadas e descritas neste trabalho, as demais telas estão no Apêndice 1.

Figura 21 – Telas Iniciais do Monitora Tocantins.



(a) Splash Screen (b) Tela de Login (c) Tela de Cadastro

Fonte: Autor do trabalho (2024)

Todos os dados exibidos nesta seção, presentes nas figuras são meramente ilustrativos, apenas para exemplificar e simular o uso do aplicativo.

a) Splash screen: a tela splash screen é mostrada durante a inicialização do aplicativo, possibilitando o aplicativo carregar as suas configurações necessárias para inicializar, após finalizar este carregamento o usuário é direcionado para a tela seguinte.

b) Tela de Login: a tela seguinte a ser exibida é a tela de *login*, na qual há necessidade de inserir senha e email do usuário como credencial para o uso do aplicativo. Caso os dados obrigatórios não sejam inseridos ou sejam inseridos de forma incorreta, o aplicativo exibirá uma mensagem de erro. Se o usuário não for cadastrado, ele pode ser direcionado para a tela de cadastro para realizar esse registro.

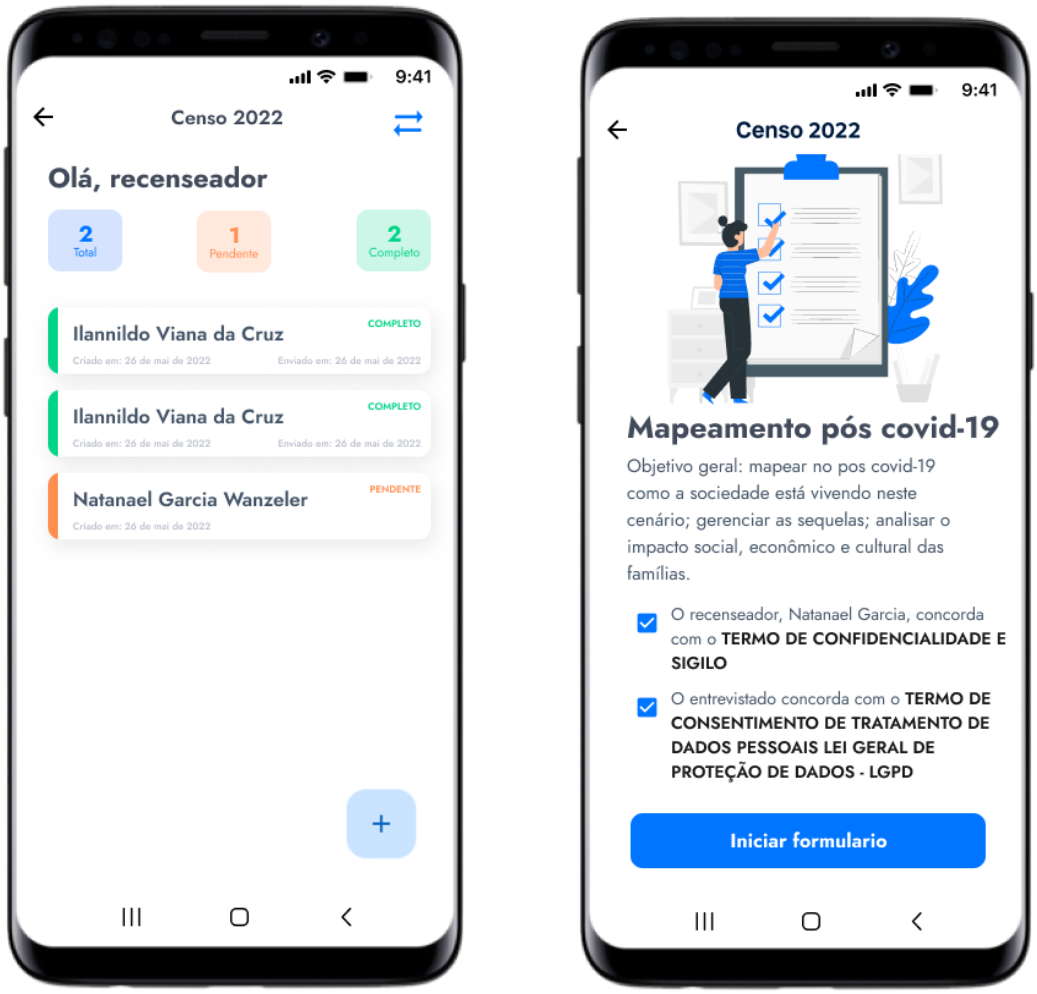
c) Tela de Cadastro: Nesta tela, o usuário visualizará vários campos de texto onde ele deverá inserir seus dados pessoais (nome, email, senha, CPF, endereço, data de nascimento e genero). O único dado que não é obrigatório para o cadastro é o CPF. Caso os dados obrigatórios não sejam inseridos ou inseridos de forma incorreta, o aplicativo exibirá uma mensagem de erro.

Figura 22 – Tela Principal do Monitora Tocantins.

(a) Tela Inicial
Fonte: Autor do trabalho (2024)

Na parte superior esquerda desta tela temos a logo do aplicativo, na parte superior direita, encontra-se o botão de Configuração, no qual o usuário poderá visualizar informações pessoais do próprio usuário, e os Termos e Condições, Política de Privacidade do aplicativo, e ainda poderá reportar falhas, bugs, ou fazer sugestões de melhorias; logo abaixo temos as informações atualizadas sobre os casos confirmados, recuperados e número de óbitos ocorridos pela COVID-19 em Cametá-PA; em seguida, o usuário poderá selecionar o botão "*IoT* Imuniza - 2022", onde terá um campo contendo a listagem de formulários já coletados e um botão para iniciar o questionário a ser preenchido, como se observa nas figuras abaixo.

Figura 23 – Listagem de Formulários Coletados e Telas do Formulário.



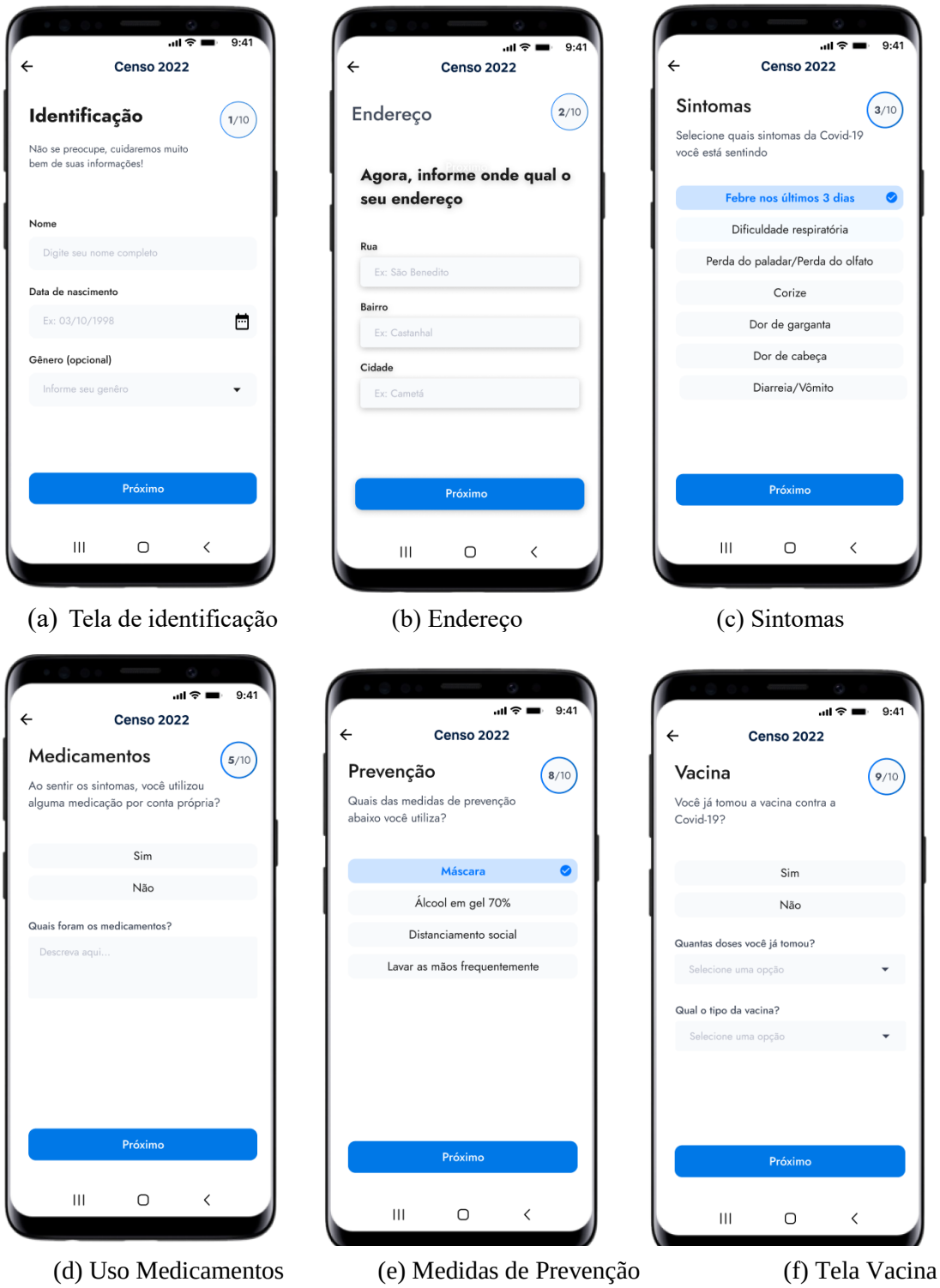
(a) Listagem de Formulários (b) Termos do Formulário do Aplicativo

Fonte: Autor do trabalho (2024)

Na Figura 23a, na parte inferior direita da tela, temos o botão para iniciar um novo formulário, com isso o usuário será direcionado para a tela da Figura 23b, onde temos dois botões (*CheckList*) que são obrigatórios e precisam ser assinalados pelo usuário antes de clicar no botão “Iniciar formulário”. A primeira opção do botão diz respeito ao Termo de confidencialidade e Sigilo, no qual o recenseador assegura ao entrevistado o sigilo das informações prestadas. A segunda opção trata do Termo de Consentimento de Tratamento de Dados Pessoais pela Lei Geral de Proteção de Dados – LGPD, na qual o entrevistado permite aos responsáveis do aplicativo, o uso dos dados fornecidos na entrevista para fins de pesquisa.

Nas figuras a seguir apresentamos as perguntas que foram elaboradas e implementadas no formulário do aplicativo.

Figura 24 – Telas do Formulário.



(a) Tela de identificação (b) Endereço (c) Sintomas

(d) Uso Medicamentos (e) Medidas de Prevenção (f) Tela Vacina

Fonte: Autor do trabalho (2024)

a) **Tela de identificação:** Nessa tela é feita a solicitação dos dados pessoais do usuário como nome completo, data de nascimento, além da opção de gênero. Após isso, deve clicar em “Próximo” para prosseguir para a tela seguinte.

b) **Tela de endereço:** Nessa tela as informações podem ser preenchidas de duas formas: na primeira, se houver internet, o aplicativo mostra uma mensagem perguntado se o usuário deseja utilizar a sua localização atual para preencher os campos da tela de endereço, automaticamente; a segunda maneira de preencher os campos seria manualmente, preenchendo os campos com nome da rua/travessa/avenida, bairro, e cidade onde mora. Após isso, deve clicar em "Próximo" para prosseguir para a tela seguinte.

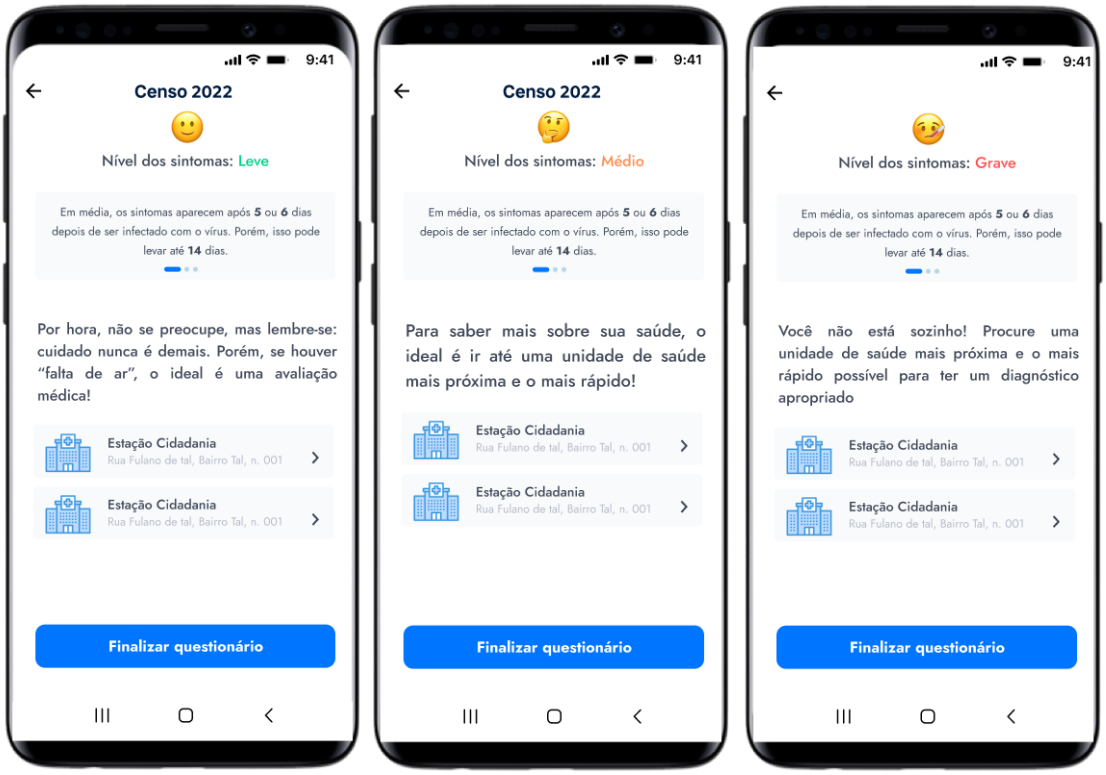
c) **Tela de sintomas:** Essa tela apresenta os principais sintomas da COVID-19, nela o usuário deverá selecionar todos os sintomas que forem percebidos por ele até o momento. Na tela seguinte "Início de sintomas", o usuário informará o período do início dos sintomas (nos últimos 10 dias/nos últimos 15 dias). Após isso, deve clicar em "Próximo" para prosseguir para a tela seguinte.

d) **Uso de medicamentos:** O entrevistado deverá informar se fez automedicação, selecionando as opções "sim/não"; se sim, ele deverá descrever o tipo de medicamento utilizado. Ao clicar em "Próximo", o usuário verá as telas "Contato com a COVID" e "Aglomeração", nas quais deverá selecionar uma das opções "sim/não" para informar se teve contato com alguém que testou positivo para a doença e/ou se frequentou lugares com aglomeração. Informando também, o local onde esteve mais frequentemente. Após isso, deve clicar em "Próximo" para prosseguir para a tela seguinte.

e) **Tela de medidas prevenção:** Nessa tela o usuário irá assinalar as medidas de prevenção utilizada por ele durante a pandemia. Após isso, deve clicar em "Próximo" para prosseguir para a tela seguinte.

f) **Tela vacina:** O entrevistado deverá selecionar "sim/não" para responder se já tomou vacina para a COVID e deverá descrever no espaço em branco quantas doses e o tipo de vacina recebido por ele. Após isso, deve clicar em "Próximo" para prosseguir para a tela "Reação à Vacina", na qual irá selecionar "sim/não" para responder se teve reação à vacina; se sim, deverá descrever no campo em branco abaixo quais reações apresentou.

Figura 25 – Tipos de Telas para Finalizar o Formulário.



(a) Caso Leve (b) Caso Moderado (c) Caso Grave

Fonte: Autor do trabalho (2024)

A Figura 25 acima apresentas três tipos de telas relacionadas ao nível dos sintomas e à localização dos postos de pronto atendimento da COVID. Após coletar todas as respostas relacionadas aos sintomas citados pelo usuário, o aplicativo realiza uma classificação para definir o nível da doença:

a) Caso leve: caracterizado a partir da presença de sintomas não específicos: tosse, dor de garganta e/ou coriza, diarreia, dor de cabeça.

b) Caso moderado: inclui desde sinais leves até sinais de piora.

c) Caso grave: inclui os casos leve e moderado, e também apresenta dificuldade respiratória, desconforto respiratório, pneumonia, etc.

6. CONSIDERAÇÕES FINAIS

Considerando a proposta deste trabalho de fazer uma abordagem sobre o desenvolvimento do aplicativo Censo Monitora Tocantins para fins de análise dos impactos da COVID na região tocantina, ressaltamos que buscamos mostrar com fidelidade todo o processo de criação do aplicativo, desde a sua conceituação até o seu desenvolvimento, destacando as características *crossplatform* da biblioteca utilizada, o *React Native*. O aplicativo alcançou um resultado satisfatório a respeito de sua usabilidade. De maneira geral, observou-se que as ferramentas usadas no desenvolvimento do aplicativo facilitaram a sua criação, aumentando a produtividade.

O aplicativo foi finalizado e testado para que pudesse fazer a coleta das informações a respeito da COVID-19, possibilitando fornecer dados concretos sobre o quadro epidemiológico da região. Além disso, vale ressaltar que o aplicativo apresenta-se como uma grande ferramenta de medidas preventivas e ações de combate à doença causada pelo vírus.

Além do que foi exposto nesse trabalho, muitas etapas do processo de desenvolvimento ainda precisam ser desenvolvidas e melhoradas para que possam atender seus usuários da melhor forma possível. Neste sentido, deseja-se implementar para os trabalhos futuros os itens abaixo:

1 Disponibilizar o aplicativo nas lojas de dispositivos móveis.
2 Desenvolver uma versão *web* do sistema para fazer o gerenciamento e exibição dos dados coletados.
3 Disponibilizar o aplicativo para versão do sistema *iOS*.
4 Habilitar uma versão do aplicativo para os usuários comuns.

7. DIFICULDADE ENFRENTADAS

Durante a criação do aplicativo foram enfrentadas algumas dificuldades, dentre as quase citamos o excesso de tarefas a serem cumpridas pelos dois membros da equipe; a incompatibilidade de horários entre o horário do desenvolvimento do aplicativo, o horário de trabalho e o horário de aulas dos desenvolvedores, o que tornava a situação improdutiva em todas essas tarefas.

8. LIÇÕES APREENDIDAS

As lições aprendidas a partir do desenvolvimento do aplicativo Monitora Tocantins, além da assimilação de conceitos, métodos sobre a abordagens sobre o tema trabalhado, certamente serviram de base e exemplo para nosso desenvolvimento profissional na área de TI. Podemos citar como exemplo: processo de criação de interfaces, elementos visuais, usabilidade, prototipagem e avaliação de usabilidade, são conteúdos que foram abordados neste trabalho.

Outra lição importante refere-se ao desenvolvimento de aplicativos, que possibilitou um maior entendimento do *framework React Native*, que permite criar aplicações multiplataforma, ou seja, tanto para *android* como *iOS.*

REFERÊNCIAS

ADASH, Fernando. Android Studio 4.0. Disponível em: <Código Google: Android Studio 4.0>. Acesso em: 12 outubro 2024.

ARAUJO, Gabriel Rodrigues de. *Desenvolvimento Cross-platform com React Native: Um estudo de Caso do Aplicativo Naveg.* 2019. Disponível em: <https://repositorio.ufc.br/bitstream/riufc/45892/3/2019_tcc_graraujo.pdf> Acesso em: 15 outubro 2024.

BEZERRA, Eduardo. *Princípios de Análise e Projeto de Sistemas com UML*.3.ed. Rio de Janeiro: Elsevier, 2015.

CAMPOS, Diego P. G.; PEREIRA, Kaique E.; CARVALHO, Samuel A.; et al. *APLICAÇÃO DE ARMAZENAMENTO EM NUVEM UTILIZANDO A PLATAFORMA NODE.JS*. RE3C - Revista Eletrônica Científica de Ciência da Computação, v. 11, n. 1, 2016. Disponível em: <https://revistas.unifenas.br/index.php/RE3C/article/view/160>. Acesso em: 12 outubro. 2024.

CONTA, Gabriel Sebastian von. *Identificação e visualização de rastros entre artefatos no GitHub*. 2018. Monografia (Curso Ciência da Computação) – Universidade Federal do Rio Grande do Norte. Disponível em: <https://repositorio.ufrn.br/handle/123456789/33042>. Acesso em: 05 outubro 2024.

Documentação de introdução ao GitHub, Github. Disponível em: <https://docs.github.com/pt/get-started> Acesso em: 20 outubro. 2024.

EL-KASSAS, W. S. et al. Taxonomy of cross-platform mobile applications development approaches. Ain Shams Engineering Journal, Elsevier, v. 8, n. 2, p. 163–190, 2017.

FALCÃO, Filipe Dourado. *Desenvolvimento do aplicativo Turistando Beberibe utilizando React Native*. 2022. Disponível em:<https://repositorio.ufc.br/bitstream/riufc/69029/3/2022_tcc_fdfalc%c3%a3o.pdf>.Acesso em: 18 outubro 2024.

FERNANDO, Adash. *Android Studio 4.0. Google Developers*. Disponível em: <https://developers-br.googleblog.com/2020/06/android-studio-40.html>. Acesso em: 16 setembro 2024.

FERREIRA, Cristiane MS et al. *Uma avaliação de frameworks multiplataforma para desenvolvimento de aplicativos móveis multimídia.* IEEE LATIN AMERICA TRANSACTIONS, [S.I], v. 16, n. 4, 2018. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8362158>. Acesso em: 10 outubro 2024.

FERREIRA, Daniela Carvalho Monteiro. *Website Aural: aplicações sonoras com CSS 2*. v. 2, n. 4, 2016. Disponível em: <https://d1wqtxts1xzle7.cloudfront.net/51406786/637-1657-1-PB-libre.pdf?1484736407=&response-content-disposition=inline%3B+filename%3DWebsite_Aural_aplicacoes_sonoras_com_CSS.pdf&Expires=1730259120&Signature=U3kWKOxkKL8itUzSjQapGG1UaRPSQMeBrmePbvZhIGlajcqxwm86XEsQd4uN63Yd2IZye5Zi7NC8bGK-PQIuuQpVzdmnTGEjXdt51QN3X6zIvFqjyujBK12F0iNpf902epp7NkB6xXddrS6WDym5OMYPH~aUGk2qMXkFq3b4vKwz~4ecKvDs1orWWl43cxK~4VB2QUbRNaqeFSB6eMNCS729oXX98iHBz3NvKfkH80ll0SSeB~1QHIBrmXRM8BcoavMYemiPjGYZhRLnKbvhYdE8Xp9c3Ps1~NrVSF~nvZXIzjCcyXVffyMh5uuUpenvhsgRjabD2Auszn6L9BCq~w__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>. Acesso em: 19 setembro 2024.

GAMA, A. *O que é JSON: Do Básico ao Avançado com JSON* - Leia agora! Disponível em: <https://www.devmedia.com.br/o-que-e-json/23166>. Acesso em: 12 outubro. 2024.

GARRETT, Filepe. *O que é Figma? Quatro Perguntas Sobre como Usar o Site*. 05/06/2021. Disponível em: <https://www.techtudo.com.br/listas/2021/06/o-que-e-figma-quatro-perguntas-sobre-como-usar-o-site.ghtml >.Acesso em: 20 outubro. 2024.

GIL, Antonio Carlos. *Métodos e técnicas de pesquisa social*. 6. ed. Ediitora Atlas SA, 2008.

HJORT, E. *Evaluation of React Native and Flutter for Cross-platform Mobile Application Development*. Abo Akademi University, 2020. Disponível em: <http://urn.fi/URN:NBN:fi-fe2020112392758>. Acesso em: 20 outubro 2024.

KAMADA, Aqueo. *Execução de Serviços Baseada em Regras de Negócio*. 2006. Tese (Doutorado em Engenharia Elétrica) – Universidade Estadual de Campinas. Disponível em: <https://repositorio.unicamp.br/acervo/detalhe/395513>. Acesso em: 16 setembro 2024.

KENNETH, C. L.; LAUDON, J. P. *Sistemas de Informação Gerenciais*. Editora Person. São Paulo, 2011.

LANAGAN, David. *JavaScript: O Guia Definitivo*. 6. Porto Alegre Bookman 2014. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788565837484/pageid/17>. Acesso em: 06 setembro 2020.

LIMA, Guilherme Ferreira Faioli. *Desenvolvimento de um Aplicativo Multiplataforma para Gestão de Vacinação*. 2021. Disponível em: <https://monografias.ufop.br/bitstream/35400000/3009/6/MONOGRAFIA_DesenvolvimentoAplicativoMultiplataforma.pdf>. Acesso em: 20 outubro 2024.

MELO, D. *O que é TypeScript? [Guia para iniciantes] | Aplicativos e Software | Tecnoblog*. Disponível em: <https://tecnoblog.net/responde/o-que-e-typescript-guia-para-iniciantes/>. Acesso em: 06 setembro 2024.

OLIVEIRA, Figueiredo de Oliveira. *Sistemas de Informação: Um enfoque gerencial inserido no contexto empresarial e tecnológico*. 3ª ed. São Paulo: Érica 2002.

Pereira,A.S.,et al (2018). *Methodology of Cientific Research*. UFSM Editors

Pressman RS, Maxim BR. *Engenharia de Software uma Abordagem Profissional*. 8a ed. São Paulo: AMGH; 2016.

NATIVE, React. *Learn Once, Write Anywhere*. React native. Disponível em: <https://reactnative.dev/>. Acesso em: 06 agosto 2024.

NATIVE, React. *New Architecture is Here*. React native. 23 de outubro de 2024 <https://reactnative.dev/docs/getting-started>. Acessado em: 26 de setembro 2024.

Red Hat. *O que é uma API REST?*. 17 de agosto de 2023. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api#:~:text=Para%20que%20uma%20API%20seja,todas%20as%20outras%20a%C3%A7%C3%B5es%20dispon%C3%ADveis>. Acesso em: 11 outubro 2024.

REINEHR, Sheila. Engenharia de requisitos. Porto Alegre SAGAH, 2020.

REIS, Antonio Carlos Serafim. *UM ESTUDO COMPARATIVO ENTRE MODELOS DE DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS*. 2019. Monografia (Curso Engenharia de Software) – Universidade Federal do Ceará. Disponível em: <http://repositorio.ufc.br/bitstream/riufc/49707/1/2019_tcc_acsdosreis.pdf>. Acesso em: 20 outubro 2024.

SHAPIRO, A. *The control revolution: how the internet is putting individuals in charge and changing the word we know*. New York: A Century Foundation Book, 1999.

SOMMERVILLE, Ian. *Engenharia de Software*. 9ª Edição. São Paulo: Pearson Prentice, 2011.

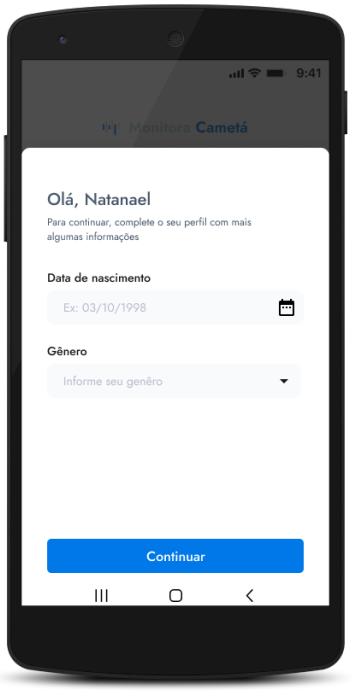
SOMMERVILLE, Ian. *Engenharia de software*.10 ed - São Pauto: Pearson Education do Brasil, 2018.

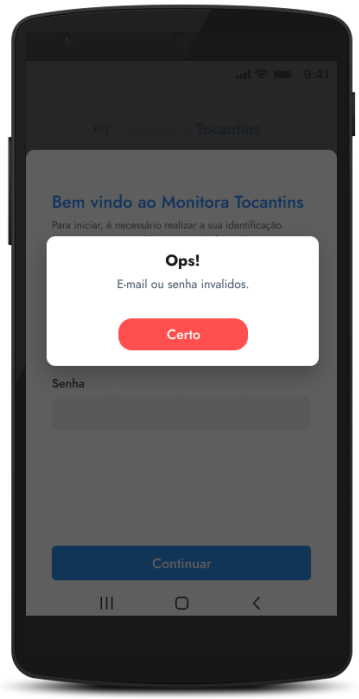
WHEELER,W; J. WHITE.*Spring in practice. Greenwich*, CT: Manning Publications, 2013.

APÊNDICES

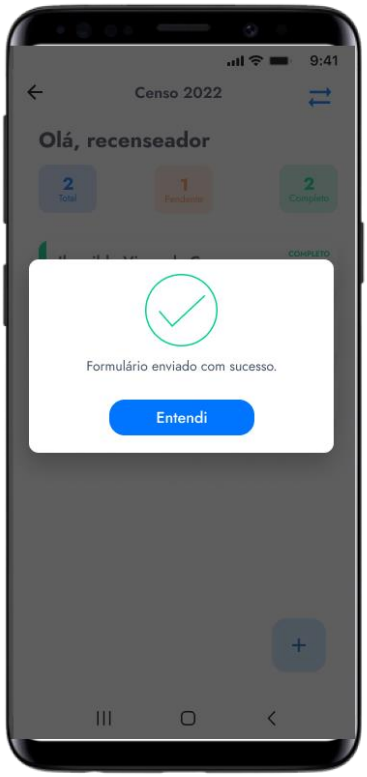
APÊNDICE 1 – TELAS DO APLICATIVO

Neste apêndice estão as telas que não foram incorporadas ao trabalho, por se tratar de telas de facíl compreensão visual.



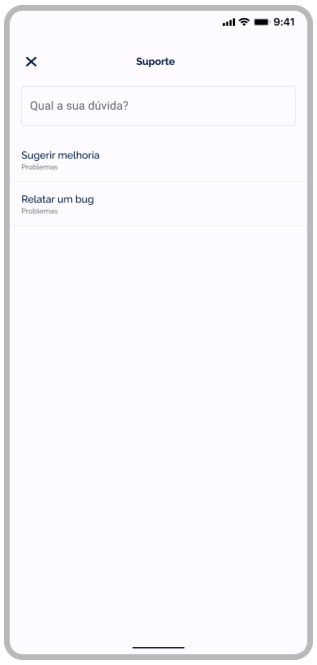


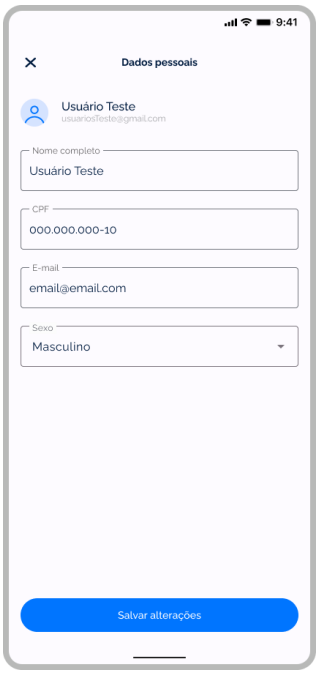

9:41
Censo 2022
Olá, recenseador
2
Total
1
Pendente
2
Completo
Ilannildo Viana da Cruz
COMPLETO
Ops!
Você está sem acesso a internet.
Formulário salvo no seu dispositivo
Certo

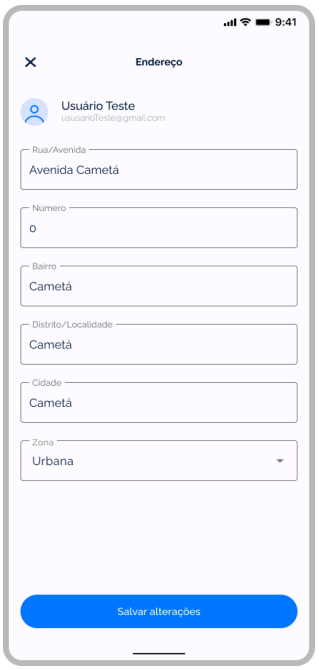

9:41
Censo 2022
Olá, recenseador
2
Total
1
Pendente
2
Completo
Formulário enviado com sucesso.
Entendi


9:41
Censo 2022
Ops!
Você precisa concordar com os termos do aplicativo
Certo
famílias.
O recenseador, Natanael Garcia, concorda com o TERMO DE CONFIDENCIALIDADE E SIGILO
O entrevistado concorda com o TERMO DE CONSENTIMENTO DE TRATAMENTO DE DADOS PESSOAIS LEI GERAL DE PROTEÇÃO DE DADOS - LGPD
Iniciar formulario

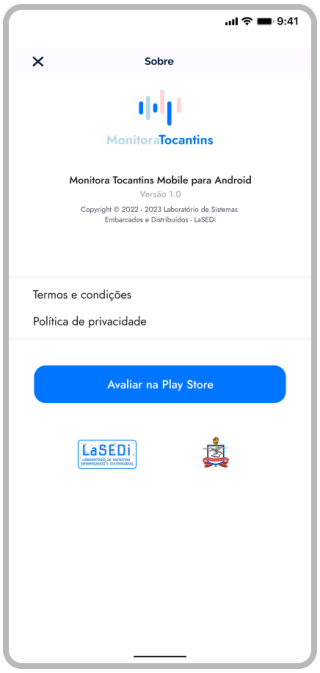

9:41
Censo 2022
Sintomas
3/10
Selecione quais sintomas da Covid-19 você está sentindo
Febre nos últimos 3 dias
Ops!
Você precisa selecionar umas das alternativas
Certo
Dor de garganta
Dor de cabeça
Diarreia/Vômito
Próximo

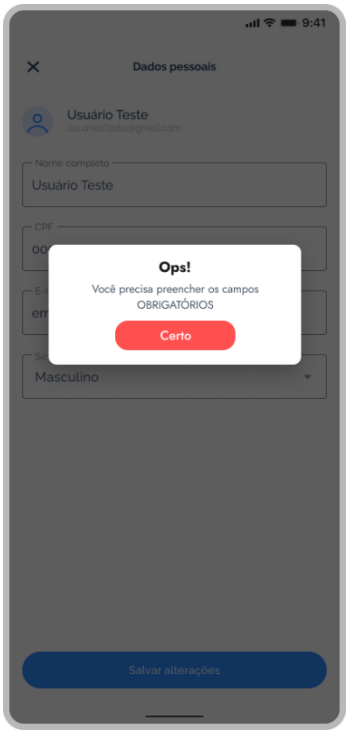

9:41
Suporte
Qual a sua dúvida?
Sugerir melhoria
Problemas
Relatar um bug
Problemas


9:41
Dados pessoais
Usuário Teste
usuariosTeste@gmail.com
Nome completo
Usuário Teste
CPF
000.000.000-10
E-mail
email@email.com
Sexo
Masculino
Salvar alterações


9:41
Endereço
Usuário Teste
usuariosTeste@gmail.com
Rua/Avenida
Avenida Cametá
Número
0
Bairro
Cametá
Distrito/Localidade
Cametá
Cidade
Cametá
Zona
Urbana
Salvar alterações


9:41
Perfil
Usuário Teste
usuariosTeste@gmail.com
Conta
Dados pessoais
Endereço
Geral
Suporte
Sobre
Convidar amigos
Sair
Monitora Tocantins v1.0.6 - LaSEDi


9:41
Sobre
MonitoraTocantins
Monitora Tocantins Mobile para Android
Versão 1.0
Copyright © 2022 - 2023 Laboratório de Sistemas Embarcados e Distribuídos - LaSEDi
Termos e condições
Política de privacidade
Avaliar na Play Store
LaSEDi


9:41
Dados pessoais
Usuário Teste
Nome completo
Usuário Teste
CPF
Ops!
Você precisa preencher os campos OBRIGATÓRIOS
Certo
Masculino
Salvar alterações


9:41
Dados pessoais
Usuário Teste
Nome completo
Usuário Teste
Dados editados com sucesso
Entendi
Salvar alterações