APLICAÇÃO MOBILE PARA REGISTRO E DISPONIBILIZAÇÃO DE DADOS DE PROBLEMAS URBANOS

Rodrigo Lucas Alves de Oliveira

POSSE– GO
2023

RODRIGO LUCAS ALVES DE OLIVEIRA

APLICAÇÃO MOBILE PARA REGISTRO E DISPONIBILIZAÇÃO DE DADOS DE PROBLEMAS URBANOS

Trabalho apresentado como requisito parcial para a Conclusão do Curso de Bacharelado em Sistemas de Informação da Universidade Estadual de Goiás.

Orientadora: Cristiane Batista Xavier
Coorientador: Ronaldo Ferreira da Silva

POSSE– GO
2023

COMISSÃO EXAMINADORA

Prof. Esp. Cristiane Batista Xavier
Universidade Estadual de Goiás
Orientadora

Prof. Ms. Ronaldo Ferreira da Silva
Universidade Estadual de Goiás
Coorientador

Profª. Givanilde de Assis dos Santos Oliveira
Universidade Estadual de Goiás
Avaliador

Prof. Roberto Felício de Oliveira
Universidade Estadual de Goiás
Avaliador

Posse, 23 de Janeiro de 2023

FICHA CATALOGRÁFICA

AOL48a

Alves de Oliveira, Rodrigo Lucas
APLICAÇÃO MOBILE PARA REGISTRO E DISPONIBILIZAÇÃO DE DADOS DE PROBLEMAS URBANOS / Rodrigo Lucas Alves de Oliveira; orientador Cristiane ; co-orientador Ronaldo Ferreira da Silva. -- Iaciara, 2023.
40 p.

Graduação - Sistemas de Informação -- Unidade de Posse, Universidade Estadual de Goiás, 2023.

1. Problemas Urbanos. 2. Gestão Pública. 3. Aplicativo Móveis. I. , Cristiane, orient. II. Ferreira da Silva, Ronaldo , co-orient. III. Título.

DEDICATÓRIA

A Deus, agradeço por todas as bênçãos e orientações ao longo desta jornada. A minha esposa, obrigado por seu amor e apoio incondicional. Aos meus orientadores, muito obrigado por sua sabedoria e orientação valiosa. Sem vocês, eu não teria chegado até aqui.

AGRADECIMENTOS

Agradeço imensamente a todos os professores e orientadores que me ajudaram ao longo desse projeto de TCC. Em especial, agradeço ao meu orientador que me guiou, me incentivou e me acompanhou desde o início até a conclusão deste trabalho. Agradeço a todos os meus colegas de classe que contribuíram para o meu aprendizado e me deram valiosos feedbacks. Agradeço também a minha família e amigos por todo o apoio e incentivo que me deram durante esse período. Este trabalho não teria sido possível sem a ajuda de todos vocês.

EPÍGRAFE

“O temor do Senhor é o princípio da sabedoria,

e o conhecimento do Santo é prudência”.

Provérbios 9.10

RESUMO

Com o objetivo de criar uma ferramenta que possa auxiliar cidadãos e gestão pública na resolução de problemas urbanos, o projeto de software apresentado a seguir tem seu principal objetivo de unir cidadãos e gestão pública para um bem comum, tornar a cidade um local mais agradável de se viver. Por intermédio de pesquisa bibliográfica, foi possível entender como os problemas urbanos surgiram e como a gestão pública tem dificuldade em lidar com tais problemas, ao longo dos anos o crescimento desordenado das cidades proporcionou ainda mais problemas urbanos, e impossibilitando ainda mais a gestão pública corrigir esse fato. Por esse motivo o aplicativo desenvolvido neste projeto intitulado de ResolveAí visa unir cidadãos e gestão, para que a gestão possa trabalhar e agir exatamente onde o cidadão precisa, afinal, uma gestão trabalha em função da sociedade, então nada mais justo do que a sociedade ajudar a gestão pública enviando informações valiosas para que a gestão possa executar seu trabalho de forma mais assertiva criando uma cidade com menos problemas urbanos. O aplicativo é dividido em dois módulos: *backend* (responsável por toda parte lógica do aplicativo) e *frontend* (responsável por toda parte visual do aplicativo) em suma essa modularização tem como resultado final um aplicativo *mobile* compilado para *Android*, para que pessoas possam fazer registro de problemas urbanos.

Palavras-chave: Problemas Urbanos, Gestão Pública, Aplicativo Móveis

ABSTRACT

With the goal of creating a tool that can assist citizens and public management in resolving urban problems, the software project presented below has its main objective of bringing citizens and public management together for a common good, making the city a more pleasant place to live. Through bibliographic research, it was possible to understand how urban problems arose and how public management has difficulty dealing with such problems. Over the years, the disordered growth of cities has caused even more urban problems, making it even more difficult for public management to correct this fact. Therefore, the application developed in this project entitled ResolveAí aims to unite citizens and management, so that management can work and act exactly where the citizen needs, after all, management works for society, so nothing is fairer than society helping public management by sending valuable information so that management can perform its work more effectively, creating a city with fewer urban problems. The application is divided into two modules; backend (responsible for all the logical part of the application) and frontend (responsible for all the visual part of the application) in short, this modularization results in a mobile application compiled for Android, so that people can register urban problems.

Key Words: *Urban problems, Public Management, Mobile Application.*

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SIGLAS, ABREVIAÇÕES E SÍMBOLOS

ONU	-	Organização das Nações Unidas
PNAD	-	Pesquisa Nacional por Amostra de Domicílios
RF	-	Requisitos Funcionais
RNF	-	Requisitos Não Funcionais
JWT	-	*Json Web Token*
JSON	-	*JavaScript Object Notation*
ER	-	Entidade de Relacionamento
API	-	*Application Programming Interface*
SOAP	-	*Simple Object Access Protocol*
RPC	-	*Remote Procedure Calls*
APK	-	*Android Package*

LISTA DE QUADROS

Sumário

1 INTRODUÇÃO

1.1 Apresentação

Problemas urbanos são situações que acontecem em zona urbana, geralmente, devido ao crescimento desordenado das cidades e pela má administração pública (SANTOS, 1993). O Relatório do Programa Habitat, órgão ligado à ONU, revela que 52,3 milhões de brasileiros, cerca de 28% da população, vivem nas 16.433 favelas cadastradas no país, contingente que chegou a 55 milhões de pessoas em 2020. Buracos em ruas, lixo em locais públicos, praças quebradas, ponto de ônibus sem proteção contra sol ou chuva, são exemplos de problemas urbanos.

A gestão pública possui desafios de variadas ordens que impactam em toda a sua operação (SCHLEICHER, 2014). Por isso, alguns contextos demoram muito para serem analisados e resolvidos, tais como adaptar-se às mudanças apresentadas pelo mundo atual, melhorar, continuamente, os serviços oferecidos pelo Estado e implementar políticas públicas eficientes que atendam às expectativas e bem-estar dos clientes (sociedade) (PONTONI, 2020).

Devido a tal dificuldade, normalmente as análises e debates sobre a administração pública recorrem a pontos de partida mais gerais e comuns à literatura especializada (SCHLEICHER, 2014). É necessário para superar os desafios acima citados, a utilização de várias ferramentas, são elas: orçamento participativo, qualificação profissional do servidor público, modernização da mão-de-obra do Estado, motivação dos servidores públicos (PONTONI, 2020).

E assim, está essa monografia visa apresentar a documentação de desenvolvimento e de um aplicativo mobile com a finalidade de auxiliar na comunicação entre os gestores públicos e a população local, desta forma a comunidade auxiliará a gestão a ter em primeira mão os fatos que ocorrem na cidade. Logo, os cidadãos poderão registrar os problemas que estão enfrentando auxiliando os gestores na coleta de dados nos pontos onde o problema está ocorrendo.

Para alcançar os objetivos foram realizados estudos, com intuito de entender como os problemas urbanos surgiram, para isso utilizamos de pesquisas bibliográficas, após isso foi feito uma pesquisa para entender como a gestão pública lida com os problemas urbanos a fim de extrair informações que

afirmam a necessidade do aplicativo *mobile*. Logo após essa etapa foram definidos os requisitos do aplicativo. E por fim foram definidos as tecnologias e plataformas para o desenvolvimento do aplicativo *mobile*

1.3 Justificativa

Na atualidade, mais da metade da população mundial reside em áreas urbanas. De acordo com dados da Pesquisa Nacional por Amostra de Domicílios (PNAD, 2015), a maior parte da população brasileira, 84,72%, vive em áreas urbanas, e contrapartida, 15,28% dos brasileiros vivem em áreas rurais, o que indica que o mundo atual está em constante mudança para um local predominantemente urbano (PEREIRA, 2017).

Com isso existem inúmeras deficiências com a gestão pública brasileira. (PONTONI, 2020) devido aos perímetros urbanos possuírem uma quantidade enorme de problemas, tais como, grande quantidade de lixos nas ruas, iluminação pública danificada, má conservação dos espaços públicos, carência de controle e fiscalização de espaços privados, como lotes, buraco no asfalto e dentre tantos outros (PEREIRA, 2017). Desta forma, para manter o controle de todos esses problemas muitas vezes se torna uma tarefa difícil. Uma alternativa viável seria uma comunicação direta e efetiva entre cidadãos e o gestor público para que os próprios cidadãos informem os problemas urbanos mais pertinentes para a gestão.

A maioria dos cidadãos buscam meios alternativos como as redes sociais (Instagram e Facebook) para informar os problemas urbanos enfrentados por eles. Porém, esses meios podem não ser tão eficientes, pois pode ocasionar diversas limitações como: (i) a denúncia muitas vezes pode não chegar até a gestão, (ii) não receber uma resposta da gestão (iii) rede social não é um meio formal de comunicação.

Neste contexto, fica evidente a necessidade de uma aplicação que auxilie cidadãos no processo de comunicação com a gestão, e dessa forma realizar os registros dos problemas urbanos enfrentados por eles. Com isso, foi desenvolvida uma aplicação mobile para auxiliar os cidadãos a relatar problemas urbanos e estabelecer uma comunicação específica com a gestão pública.

1.4 Objetivos

1.4.1 Objetivos Geral

Visando melhorar o processo de comunicação entre os cidadãos e os gestores públicos, este projeto desenvolve um aplicativo mobile que auxilia os munícipes a relatar os problemas urbanos que enfrentam no dia a dia em seus bairros e disponibilizá-los para acesso da gestão local.

1.4.2 Objetivos Específicos

Essa seção visa delimitar nossa pesquisa e expõe as especificações mais aprofundadas sobre as etapas que foram seguidas com o propósito de termos condições de atingirmos nosso objetivo geral. Deste modo, apresentamos nossos objetivos específicos:

- Consultar a legislação sobre as regras para gerenciamento de problemas urbanos.
- Realizar a documentação e análise de requisitos para identificação dos problemas e implementação do aplicativo
- Identificar, analisar e aplicar padrões de projeto e tecnologias (Linguagens de programação, plataformas, bibliotecas) necessárias para o desenvolvimento do sistema.
- Desenvolver a estrutura básica do sistema aplicando os requisitos funcionais, não funcionais e regras de negócio necessários.
- Aplicar testes de usabilidade para avaliar a qualidade da interação entre usuários e o aplicativo mobile;
- Disponibilizar a aplicação para que os cidadãos possam baixar e instalar em seus smartphones.

2 REVISÃO TEÓRICA

2.1 Urbanização e crescimento desordenado das cidades

Urbanização é o processo de crescimento das áreas urbanas mais rápido do que as áreas rurais (SANTOS, 1993). Segundo a ONU, pela primeira vez na história a população mundial, torna-se majoritariamente urbana. A ONU aponta que em 2030 a população urbana será de 60%. A urbanização brasileira se tornou generalizada a partir do terceiro terço do século XX, evolução quase contemporânea da fase atual de macro urbanização e metropolização. (SANTOS, 1993).

> Ao longo do século XX o Brasil se desenvolveu, industrializou passando a ser urbano. No entanto, esse desenvolvimento foi marcado por diversas contradições, dentre elas a mais grave: concentração de riquezas em detrimento da exclusão social de uma maioria da população. Os problemas sociais urbanos têm se agravado no território nacional e tanto nas metrópoles como nas cidades de médio porte, tem se tornado comum um cotidiano de violência e de descaso com os direitos fundamentais do cidadão que é garantido pela Constituição de 1988. (PACHECO, 2013, pg: 2)

Como é difundido pelo autor, o grande desenvolvimento urbano brasileiro é marcado por diversas contradições e problemas. Mesmo em cidades de médio porte, a gestão pública não consegue diminuir a quantidade de problemas urbanos enfrentados pela população. A corrupção, a falta de transparência, o excesso de burocracia, a obsolescência das normas legais, a deficiente comunicação entre governo e sociedade, os escassos investimentos em ferramentas gerenciais e em tecnologia da informação. (PONTONI, 2020), são apenas alguns dos problemas entre a sociedade e a gestão pública.

O êxodo rural no Brasil foi um movimento que ocorreu por volta de 1950, este movimento aconteceu no Brasil na década de 1960, no governo de Juscelino Kubitschek, quando houve um grande investimento no desenvolvimento industrial, principalmente na região Sudeste. A consequência disto foi um grande movimento migratório do Nordeste para o Sudeste do país. Este processo se estendeu durante as décadas de 70 e 80. (MARTINS, 2010). Porém as cidades não estavam preparadas para receber essa quantidade de migrantes. Causando assim inúmeros problemas urbanos.

> [...]o êxodo rural se acelerou, chegando, no período 1970–1980, a transferir,

> para o meio urbano, o equivalente a 30,0% da população rural existente em 1970, ano em que migraram 12,5 milhões de pessoas. [...] (ALVES, 2011).

Como se pode observar houve um crescimento extremamente desordenado no meio urbano, justamente pelo fato dos moradores rurais estarem em busca de melhores condições de vida, porém a realidade desse crescimento urbano nem sempre proporcionou essa melhor qualidade de vida, desencadeando os famosos problemas urbanos já mencionados nesse projeto.

2.2 Tecnologias e Padrões utilizados

2.2.1 Aplicativos *mobile* nativos e híbridos

Aplicativos *mobile* ou aplicações móveis, são sistemas criados para serem instalados em *smartphones* e *iPads* (BODUCH, 2017). Aplicativos *mobile* são normalmente encontrados e disponíveis para *download* em plataformas como: *Google Play Store* ou *App Store*. De forma geral os aplicativos são desenvolvidos em sua linguagem nativa, por exemplo o Android utiliza da linguagem de programação *Java* e o *iOS* utiliza o *Swift*. (ROCHA, 2020).

Já aplicações híbridas utilizam de recursos da *web* para desenvolver uma aplicação mobile que tenha um comportamento extremamente similar a uma aplicação nativa (ROCHA, 2020). É com isso podendo utilizar recursos como Sistema de posicionamento global (GPS) ou câmera de forma nativa, como é destacado por Rocha (2020). Só que essa abordagem possui algumas limitações, tais como: performance inferior, limitação de *design*, limitação de funcionalidades e usabilidade mais baixa.

> O termo *Cross-Platform Development* (Desenvolvimento Multiplataformas), também conhecido como Desenvolvimento Híbrido, é a prática de desenvolver produtos de software para múltiplas plataformas ou ambientes de software. Engenheiros e desenvolvedores utilizam vários métodos para acomodar diferentes sistemas operacionais ou ambientes para uma única aplicação ou produto (*CROSS-PLATFORM DEVELOPMENT*, 2020).

Destarte, para o desenvolvimento do projeto foi escolhida uma tecnologia que não possui grandes limitações como as aplicações híbridas. E nesse cenário foi escolhido o *React Native*, que será apresentado e discutido nos próximos passos do projeto.

2.2.2 React Native

O *React Native* é um *framework JavaScript* desenvolvido e mantido pelo *Facebook* (*META*) sua principal característica é criar aplicativos para *Android* e *iOS*, compartilhando praticamente todo o código de uma plataforma para outra (BODUCH, 2017). Assim os desenvolvedores ganham bastante vantagens no desenvolvimento, tais como ganho de tempo, não precisam aprender duas linguagens e podem usar os sistemas operacionais *Windows*, *Linux* e *MacOs* para desenvolver os *apps*.

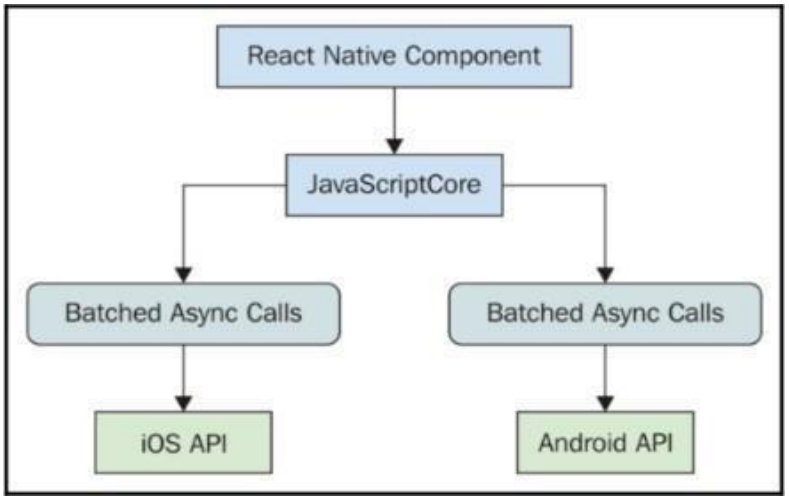
> De acordo com o livro *Learning React Native* (EISENMAN, 2018, p. 17) "*React Native* é um framework *JavaScript* utilizado para escrever aplicações reais e de renderização nativa para *iOS* e *Android*". Possui grandes semelhanças com o *ReactJS*, seu *framework* equivalente para escrever aplicações web. Ambos os *frameworks* são *open source*, e foram criados pela empresa *Facebook*. (MENDES, 2020)

O *React Native* pode não parecer tão impressionante pois já existem *frameworks* que já empacotam aplicações *web* em uma espécie de "*browser"* como o *Cordova*, ou o *Manifold.js*. Contudo o *React Native* é diferente pois todo código produzido é convertido nativamente para cada uma das plataformas, e com isso não possui os problemas que aplicações híbridas tem.

> [...] O *React Native* usa uma técnica que faz chamadas assíncronas para o SO móvel subjacente, que chama as *APIs* do *widget nativo*. Há um mecanismo *JavaScript* e a *API* do *React* é basicamente a mesma do React para a *Web*. A diferença é principalmente com o alvo; em vez de um *DOM*, há chamadas de API assíncronas. (BODUCH, 2017, pg 211).

Então todo código produzido em *React Native* é convertido para código nativo por intermédio de uma chamada da API do sistema operacional. O *React Native* possui sintaxe muito similar ao *Reactjs* da *web,* só que diferente da web ele não usa *tags* HTML (Linguagem de Marcação de HiperTexto), e sim suas próprias *tags* que no fim serão convertidas para código nativo do SO. A seguir a Figura 1 exemplifica como funcionam as chamadas assíncrona para a *API* de cada SO.

Figura 5: Conceito de *React Native*



Fonte*:* *React and React Native,* 2021

Foi escolhido o *React Native* como tecnologia para desenvolvimento do aplicativo *mobile*, com esse framework é possível ter uma boa produtividade, e ganharemos inúmeras vantagens, de início será uma aplicação para *Android*, porém com a evolução do *App* podemos disponibilizar ele para *iOS*, já que com *React Native* poderemos fazer isso com extrema facilidade.

2.2.3 Disponibilização de dados via API (*Application Programming Interface*)

APIs são mecanismos que permitem a comunicação entre softwares usando um conjunto de definições e protocolo como é descrito pela documentação da *Amazon AWS.* Então com uma API podemos fazer a comunicação de dois *softwares* diferentes e até mesmo *software* que não usam a mesma linguagem de programação (JASSE, 2017).

> *API* significa *Application Programming Interface* (Interface de Programação de Aplicação). No contexto de *APIs,* a palavra Aplicação refere-se a qualquer *software* com uma função distinta. A interface pode ser pensada como um contrato de serviço entre duas aplicações. Esse contrato define como as duas se comunicam usando solicitações e respostas. A documentação de suas respectivas *APIs* contém informações sobre como os desenvolvedores devem estruturar essas solicitações e respostas. (JASSE, *2017*)

Existem 4 (quatro) maneiras de criar APIs:

- **APIs SOAP**: Essas APIs usam o *Simple Object Access Protocol* (Protocolo de Acesso a Objetos Simples). Cliente e servidor trocam mensagens usando XML. Esta é uma API menos flexível que era mais popular no passado.
- **APIs RPC**: Essas APIs são conhecidas como *Remote Procedure Calls* (Chamadas de Procedimento Remoto). O cliente conclui uma função (ou um procedimento) no servidor e o servidor envia a saída de volta ao cliente.
- **APIs *WebSocket***: API *Websocket* é outro desenvolvimento de API da Web moderno que usa objetos JSON para passar dados. Uma API *WebSocket* oferece suporte à comunicação bidirecional entre aplicativos cliente e o servidor. O servidor pode enviar mensagens de retorno de chamada a clientes conectados, tornando-o mais eficiente que a API REST.
- **APIs *REST*:** Essas são as APIs mais populares e flexíveis encontradas na *Web* atualmente. O cliente envia solicitações ao servidor como dados. O servidor usa essa entrada do cliente para iniciar funções internas e retorna os dados de saída ao cliente.

2.2.4 APIs *REST*

REST significa Transferência Representacional de Estado. O *REST* define um conjunto de funções como *GET, PUT, POST, DELETE* e assim por diante (JASSE, 2017). Isso significa que outros aplicativos podem usar o recurso para acessar dados do servidor. Os clientes e servidores utilizam do protocolo de comunicação *HTTP* para se comunicarem.

> A principal característica da *API REST* é a ausência de estado. A ausência de estado significa que os servidores não salvam dados do cliente entre as solicitações. As solicitações do cliente ao servidor são semelhantes às URLs que você digita no navegador para visitar um site. A resposta do servidor corresponde a dados simples, sem a renderização gráfica típica de uma página da Web.(JASSE*, 2017*).

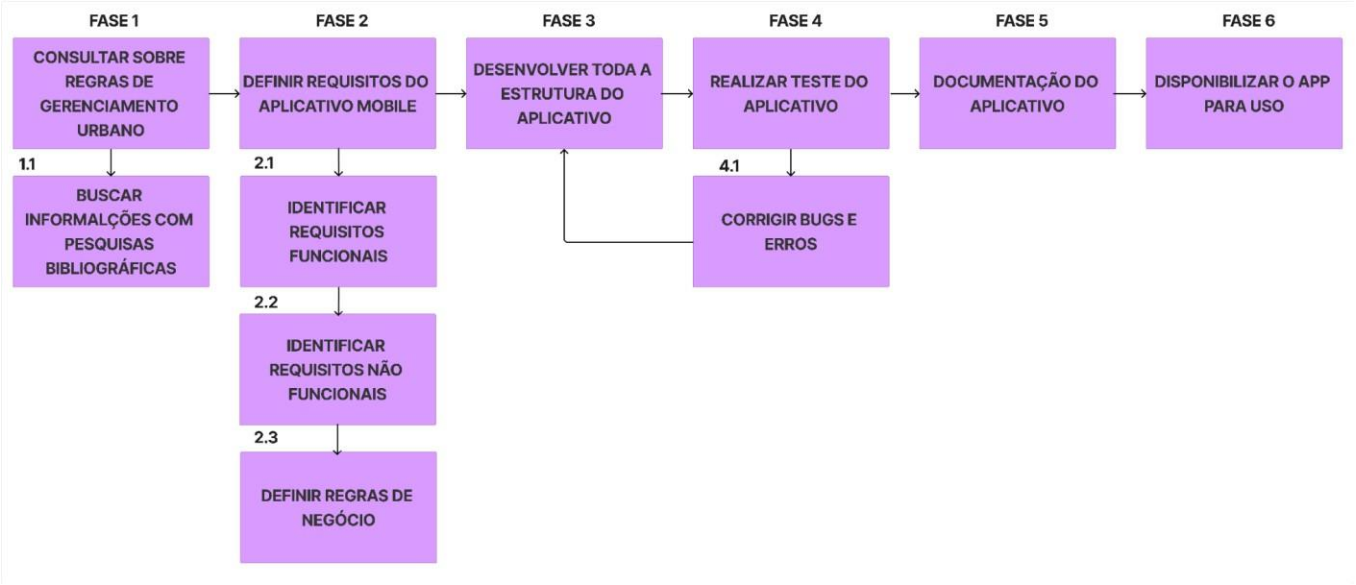
Então para criação do aplicativo *mobile*, foi criada uma *API* que irá conter os dados de cadastros de problemas urbanos e disponibilizar esses dados para a gestão pública utilizar em seus próprios sistemas. Com isso foi definido o conjunto de tecnologias para o desenvolvimento do *app*.

A implementação com base nas tecnologias apresentadas na seção "*4.2 Tecnologias e Padrões Utilizados*", resultou como produto o aplicativo compilado em arquivo *Android Package* (APK) que permite ser instalado em sistemas operacionais *Android.* As telas a seguir apresentam o aplicativo em funcionamento.

3 MÉTODOS

Para a realizar o objetivo principal dessa monografia foram definidas 6 (seis) fases que estão representadas cada uma por uma cor conforme a **Figura 1** a seguir:

Figura 1: Procedimentos Metodológicos



Fonte: Autor

- **A Fase 1 –** Foi entendido como funcionam as regras para gerenciamento de problemas urbanos. E para isso foi utilizado de pesquisas bibliográficas
- **A Fase 2 –** Foi a responsável por definir os requisitos funcionais, não funcionais e regras de negócio da aplicação.
- **A Fase 3 –** nesta fase foi desenvolvida toda a estrutura da aplicação, a fim de ter uma primeira versão da aplicação para iniciar a fase de testes.
- **A Fase 4** – Foi destinada a realização dos testes de usabilidade para observar usuários reais usando o produto para descobrir problemas e pontos de melhorias.

A Fase 5 – Foi desenvolvida toda a documentação de software.

A Fase 6 – Foi disponibilizada a aplicação para que os usuários possam ter

acesso e usá-la.

4 RESULTADOS

4.1 Elementos da Documentação de *Software*

Estão presentes nessa seção, todos os elementos da documentação de software, são eles Requisitos de software, como requisitos funcionais e não funcionais; Diagrama de Caso de Uso; Diagrama de Entidades de Relacionamento; e por fim Diagrama de classes.

4.1.1 Requisitos de *Software*

Requisitos de software tem por finalidade descrever o comportamento do sistema. Segundo Machado, 2016, engenharia de requisitos são todas as atividades realizadas para identificar, analisar, especificar e definir as necessidades de negócio que um aplicativo deve prover para solução do problema levantado. Para a criação do aplicativo foi levantada uma série de requisitos funcionais e não funcionais, para melhor compreensão do que deve ou não ser feito. A engenharia de requisitos foi necessária pois com ela foi possível entender estreitamente como o aplicativo deve funcionar, e com isso entregar algo que realmente possa ajudar pessoas e assim ajudar a gestão pública a diminuir a quantidade de problemas urbanos. Os requisitos expressam as características e restrições do produto de software do ponto de vista de satisfação das necessidades do usuário e, em geral independente da tecnologia empregada na construção da solução
(MACHADO, 2016).

> Um conjunto de requisitos pode ser definido como uma condição ou capacidade necessária que o software deve possuir para que o usuário possa resolver um problema ou atingir um objetivo ou para atender as necessidades ou restrições da organização ou dos outros componentes do sistema (Machado, 2016).

As próximas sessões são dedicadas a explanar os requisitos funcionais e não funcionais do aplicativo para maior compreensão e entendimento do aplicativo desenvolvido. A partir desses requisitos foi possível extrair todas as informações necessárias para a construção do mesmo e após todos os requisitos definidos foram

escolhidas as tecnologias necessárias para a construção do aplicativo.

4.1.1.1 Requisitos Funcionais (RF)

Essa seção diz a respeito dos requisitos funcionais do aplicativo mobile desenvolvido. Segundo ADAM (2002), requisitos constituem uma declaração completa do que o software irá fazer sem referir-se a como irá fazê-lo. Os requisitos funcionais são os que descrevem o comportamento do sistema, suas ações para cada entrada, ou seja, é aquilo que descreve o que tem de ser feito pelo sistema, assim como deve sair do sistema (MACHADO, 2016).

Nos RF são descritos todos os comportamentos do aplicativo, desde efetuar o cadastro do usuário até o cadastro de problemas, esses requisitos ajudam a entender toda a estrutura do aplicativo e principalmente entender quais as características e funcionalidades temos dentro do aplicativo, além de mostrar exatamente qual o objetivo do mesmo.

Tabela 1: Requisitos Funcionais

Identificação	Descrição
RF1 - Login	É necessário efetuar *login* no aplicativo com *e-mail* e senha para acessar todas as funcionalidades do aplicativo.
RF2 - Cadastro de dados do usuário	O aplicativo deve permitir o usuário se cadastrar com nome, e-mail, e senha
RF3 - Cadastro de problemas	O aplicativo deve permitir que o usuário registre/cadastre o problema urbano com descrição, localização e imagem
RF4 - Listagem de problemas cadastros pelo usuário	O aplicativo deve permitir o acesso dos problemas urbanos já cadastros pelo usuário
RF5 - Visualizar detalhes do problema cadastrado	Na página inicial do app após efetuar login o usuário poderá visualizar todas as informações referentes os problemas cadastrados.
RF6 – Manter cadastro problema	O aplicativo deve permitir acessar, editar e remover os problemas urbanos já cadastrados pelo usuário
RF7 - Visualizar informações sobre o perfil	O usuário poderá visualizar as informações sobre seu perfil como seu nome e e-mail.

do usuário	
RF8 - Editar informações de perfil do usuário	O usuário pode editar suas informações como trocar o nome e sua senha.

Fonte: Autor

4.1.1.2 Requisitos Não Funcionais

Essa seção dispõe dos requisitos não funcionais (RNF) do aplicativo mobile. Os RNF são as funcionalidades que não estão ligadas diretamente com a necessidade de algum usuário, ou seja, não são as funções que um usuário utiliza diretamente e sim que o próprio sistema utiliza (DIEDRICH, 2011). Por Exemplo: O sistema deverá ter servidores de redundância.

Os RNF são aqueles que expressam como deve ser feito. Em geral se relacionam com padrões de qualidade como confiabilidade, performance, robustez. São muito importantes, pois define se o sistema será eficiente para a tarefa a que se propõe a fazer ou não. (MACHADO, 2016).

Tabela 2: Requisitos Não Funcionais

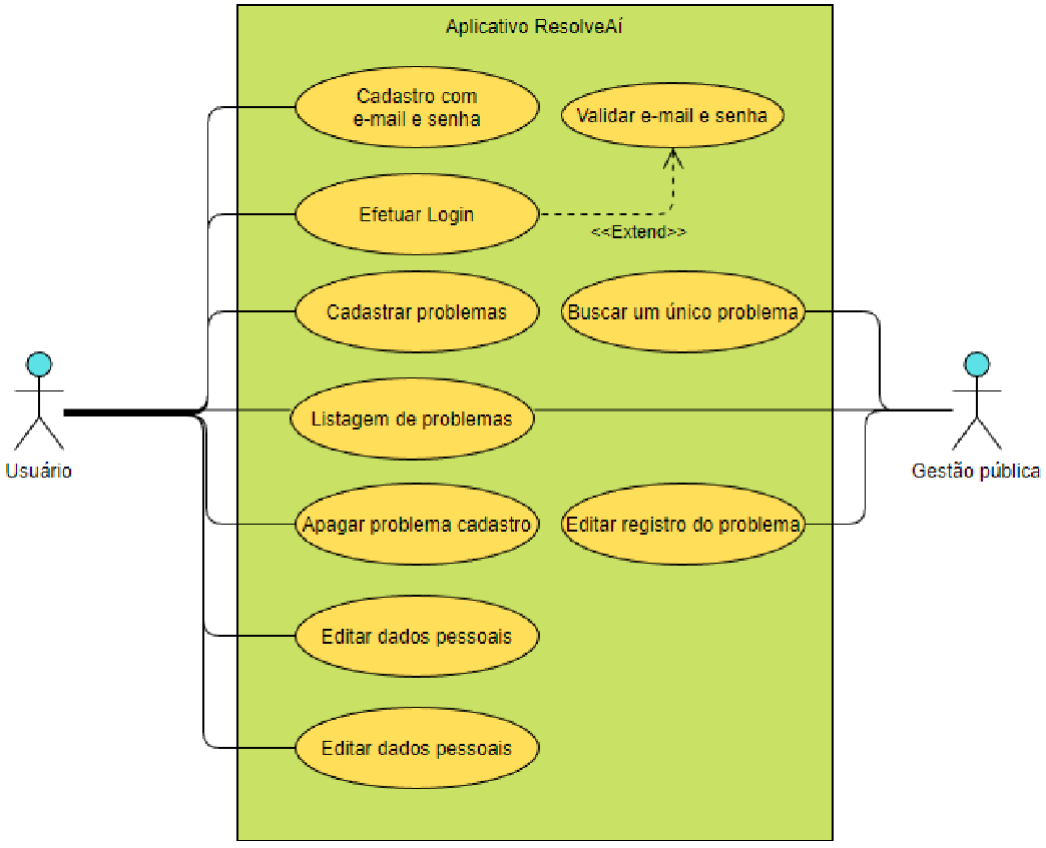
Identificação	Descrição	Módulo	Prioridade
RNF-01	O aplicativo deve ser escrito com a linguagem de programação *Javascript*.	Criação	Alta
RNF-02	O aplicativo deve possuir uma aplicação *backend* que contém toda a regra de negócio e comunicação com banco de dados e serviços externos.	Integração	Alta
RNF-03	O aplicativo deve possuir uma aplicação *frontend* que é responsável por exibir toda a parte visual do aplicativo.	Integração	Alta
RNF-04	O aplicativo deve possuir autenticação via *JWT* (*Json Web Token)* para a comunicação entre backend e frontend	Cadastro	Alta
RNF-05	Os dados trafegados entre *backend* e *frontend* devem estar em formato *JSON*.	Integração	Alta

RNF-06	O aplicativo deve estar disponível 24 horas por dia e 7 dias por semana.	Disponibilidade	Alta
RNF-07	O aplicativo deve disponibilizar uma rota para que a entidade possa visualizar as informações cadastradas pelos usuários do aplicativo.	Listagem	Alta
RNF-08	O aplicativo deve disponibilizar uma rota para que a entidade possa mandar informações sobre o problema cadastrado.	Cadastro	Alta

4.2 Diagrama de Caso de Uso

Essa seção se dispõe em detalhar os casos de uso do aplicativo. Um caso de uso envolve a interação dos atores com o sistema (UML: Guia de Usuário, 2006). Um ator representa um conjunto coerente de papéis que os usuários dos casos de uso desempenham quando interagem com esses casos (UML: Guia de Usuário, 2006). Os atores podem ser humanos ou sistemas automatizados (MACHADO, 2006). Por exemplo, na modelagem do aplicativo o processo de resolução do problema urbano envolve a interação entre o cidadão que faz o cadastro do problema e gestão pública que resolve o problema. A figura 2 a seguir, apresenta o caso de uso do aplicativo.

Figura 2: Caso de Uso



Fonte*:* Autor

4.2.1 Atores

Quadro 1 - Atores

ATORES	
Nome	**Descrição**
Cidadão	Usuário responsável por utilizar o aplicativo mobile, cadastrando os problemas urbanos, além de seus dados pessoais.
Gestor Público	Usuário que terá acesso aos problemas cadastrados pelos cidadãos.

Fonte: Autor.

4.2.2 Ações do Cidadão

Quadro 2 – Realizar cadastro

Cidadão	
CS01 – Realizar cadastro do Usuário	
Descrição	Responsável por realizar o cadastro do usuário.
Pós-condições	Liberar acesso ao aplicativo.
Fluxo Principal	
Usuário	**Sistema**
Acesse a tela de cadastro.	
	Exibe formulário.
Preenche os dados.	
	Valida os dados.
	Redireciona para a tela de autenticação.
Exceções	
Dados de cadastro incorretos.	Sistema exibe uma mensagem informando os erros encontrados.
Dados de cadastro não preenchidos.	O sistema exibe uma mensagem informando a obrigatoriedade dos campos.

Fonte: Autor

Quadro 3 – Realizar autenticação

Cidadão	
CS02 – Realizar autenticação	
Descrição	Responsável por realizar autenticação do usuário.

Pré-condições	Usuário deve estar registrado no sistema.
Pós-condições	Liberar acesso ao aplicativo.
Fluxo Principal	
Usuário	**Sistema**
Acessa a tela de autenticação.	
	Exibe formulário.
Preenche os dados.	
	Valida os dados.
	Retorna o *token* de autenticação do usuário.
	Redireciona para a tela inicial do aplicativo.
Exceções	
Dados de autenticação incorretos.	Sistema exibe uma mensagem informando os erros encontrados.
Dados de autenticação não preenchidos.	O sistema exibe uma mensagem informando a obrigatoriedade dos campos.

Fonte: Autor

Quadro 4 – Realizar cadastro de Problema

Cidadão	
CS03 – Realizar autenticação	
Descrição	Responsável pelo cadastro de problemas no aplicativo
Pré-condições	O usuário deve estar autenticado no sistema. **(CS02)**
Pós-condições	Exibe funcionalidades para registro de problemas.
Fluxo Principal	
Usuário	**Aplicativo**
Acessa a tela de cadastro	
	Exibe tela com mapa
Seleciona Posição no mapa	
Clica no botão de próximo	
	Exibe formulário para preencher dados.
Preenche os dados	.
	Valida os dados.
	Redirecionar usuário para tela de listagem

	de problemas.
FAT01 – Remover Problema	
Acessa tela de detalhes do problema	
	Exibe informações do problema.
Seleciona opção de remoção	
	Exibe janela de confirmação.
Confirmar operação	
	Realiza operação de remoção do problema.
	Redirecionar usuário para tela de listagem.
Exceções	
Dados de formulário incorretos.	Sistema exibe uma mensagem informando os erros encontrados.
Sessão do usuário inexistente.	Sistema redireciona para página de autenticação.

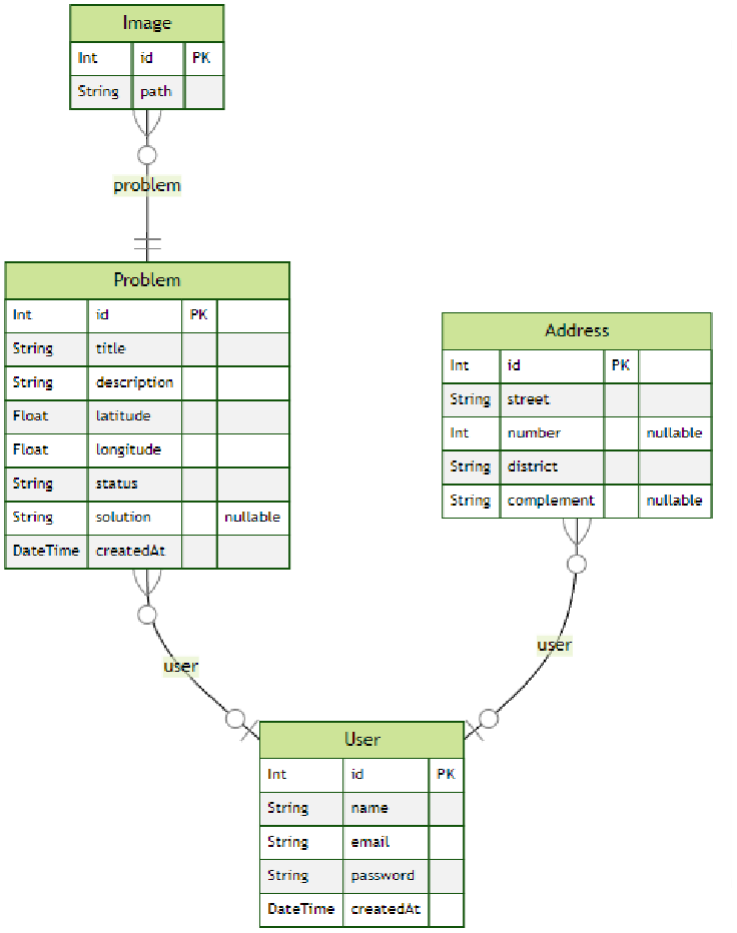
Fonte: Autor

4.3 Diagrama de Entidade de Relacionamento (ER)

Esta seção é dedicada ao diagrama de entidade de relacionamento onde pode ser observado as tabelas que existem no banco de dados do aplicativo. Um diagrama entidade relacionamento (ER) é um tipo de fluxograma que ilustra como “entidades”, pessoas, objetos ou conceitos, se relacionam entre si dentro de um sistema (NOGUEIRA, 1988).

Foram necessárias apenas 4 (quatro) tabelas no banco de dados, *User, Address, Problem e Image.* Toda a modelagem do banco de dados está fundamentada nestas tabelas. Essas Tabelas possuem relacionamentos ente si, sendo que relacionamento da principal e da tabela *User* com a tabela *Problem*, sendo um relacionamento 1 – N onde um Usuário pode estar relacionado a vários problemas, como pode ser observado na **Figura 3**

Figura 3: Diagrama de Entidade de Relacionamento (ER)



Fonte: Autor

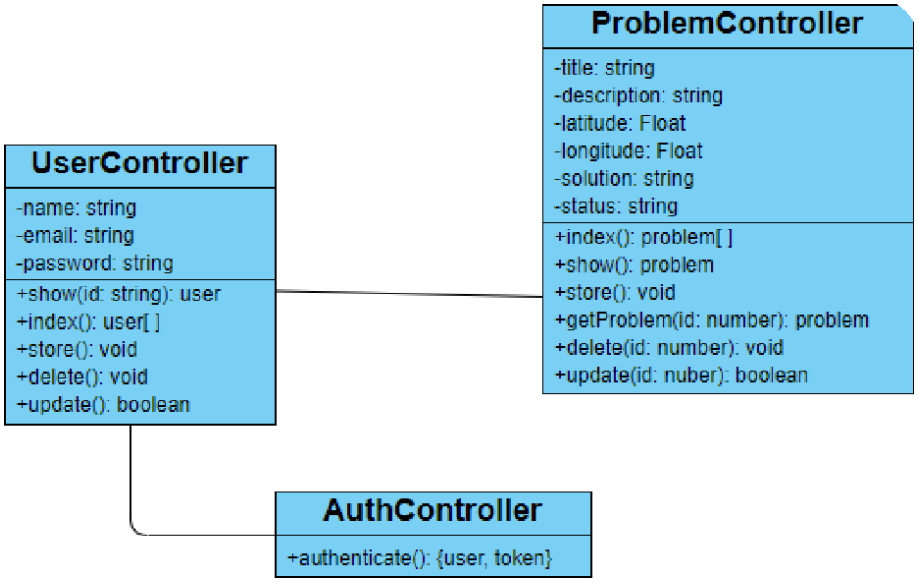
4.4 Diagrama de Classes

Esta seção é dedicada aos diagramas de classes, com o objetivo de observar as classes com suas operações e atributos. O diagrama de classes é composto de suas classes e associações existentes entre elas, ou seja, os relacionamentos entre as classes. Alguns métodos de desenvolvimento de *software*, como o Processo Unificado, recomendam que se utilize o diagrama de classes ainda durante a fase de análise, produzindo-se um modelo conceitual a respeito das informações necessárias ao software.

Na **figura 4** pode ser observada as classes que compõem o aplicativo, a classe *ProblemController* e onde estão as funcionalidades principais do aplicativo, como criação de um problema, listagem e busca. As demais classes são *UserController* e *AuthController*, sendo *UserController* responsável por todas as funções ligadas ao

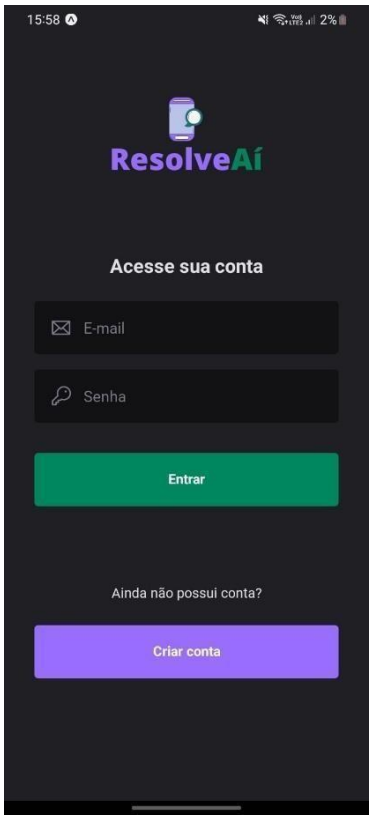
usuário, e o *AuthController* sendo uma classe apenas para validação e autenticação de usuários.

Figura 4: Diagrama de Classes

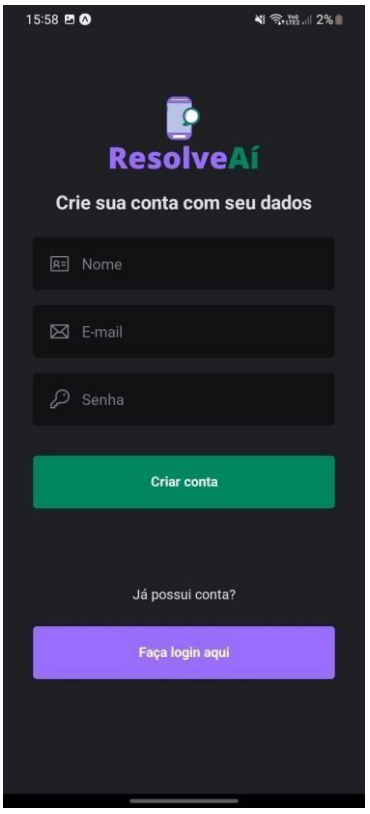


5. Telas

Figura 6: **Tela de cadastro e autenticação**



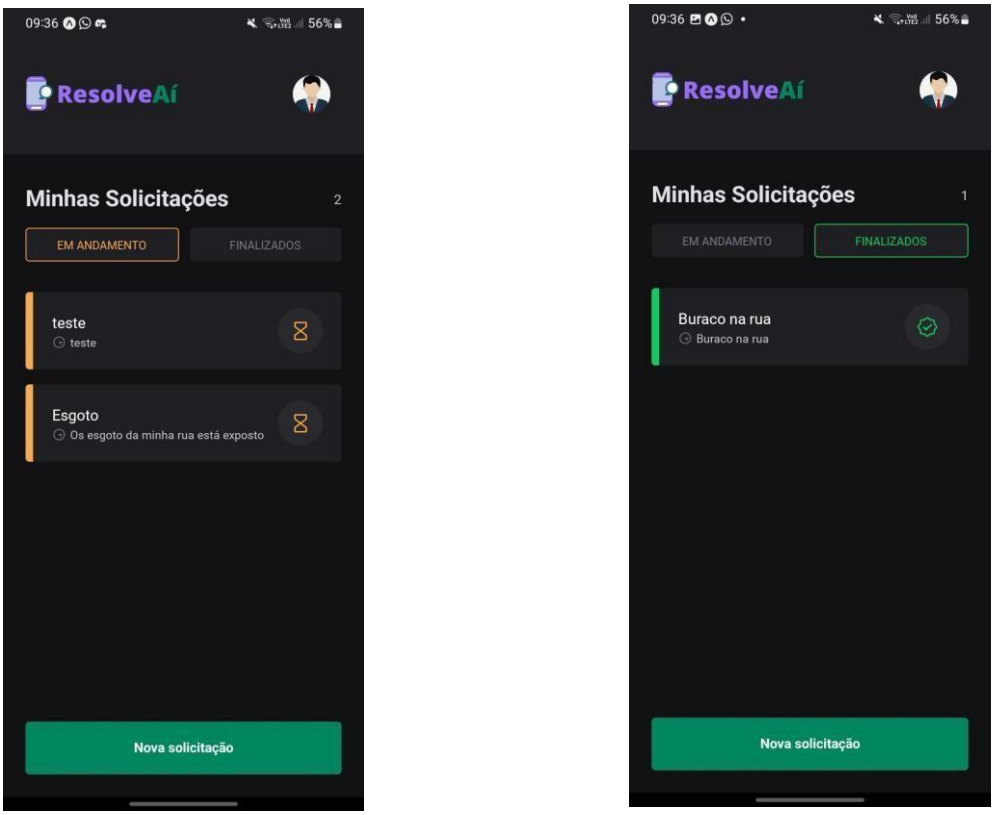
(a) – Tela de login



(b) – Tela de Cadastro

A ação inicial para utilização do aplicativo e o cadastro do usuário usando suas informações pessoais, como nome, e-mail e senha (Figura 6 – b), após realizar o cadastro, o usuário será redirecionado para a tela de login (Figura 6 – a).

Figura 7: Telas de listagem de problemas

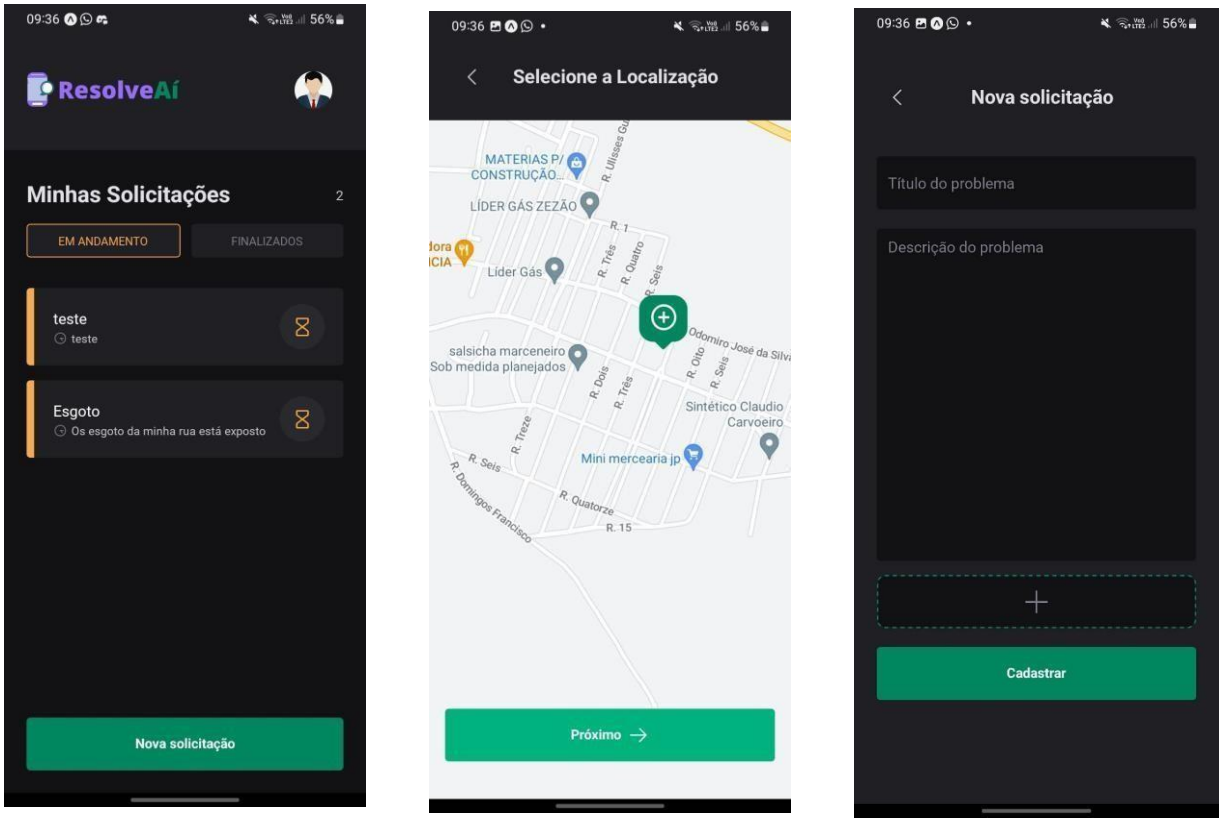


(a) – lista de problemas não solucionados **(b) – lista de problemas solucionados**

Fonte: Autor

Logo após o usuário se autenticar no aplicativo, será redirecionado para a tela de listagem de problemas (Figura 7 – a), nesta tela tem duas opções; opção 1 listagem dos problemas que estão em andamento, ou seja, que ainda não foram solucionados. E opção 2, listagem de problemas finalizados, ou seja, que já foram solucionados (Figura 7 - b).

Figura 8: Telas de cadastro de problemas



(a) – Lista de problemas **(b) – Selecionar Posição no mapa.** **(c) - Preencher dados**

Para iniciar o cadastro de um novo problema e necessário clicar no botão “nova solicitação” (Figura 8 - a) que levará o usuário para tela de selecionar a posição no mapa (Figura 8 – b), e logo em seguida na tela da Figura 8 – c, o usuário pode preencher os dados problema a ser cadastrado.

6. CONCLUSÃO

Com as pesquisas e esforços feitos para a criação desta pesquisa científica foi obtido uma grande evolução individual, devido a necessidade do tema proposto, e com aprendizado sobre as tecnologias utilizadas para o desenvolvimento do aplicativo, bem como também pode ser entendido como problemas urbanos afetam tanto uma população como uma gestão pública.

O aplicativo tem o principal intuito de ser apenas um mediador de informações, rápido e objetivo, para que assim ele seja de fato algo que as pessoas realmente usem em seu dia a dia e ajude cada vez mais pessoas e solucionar problemas urbanos.

Após o estudo em relação aos problemas urbanos, fica evidente como a população de uma cidade pode vir a sofrer graves consequências quando exposta a tais problemas, infelizmente esta é uma realidade que afeta inúmeras pessoas.

Em suma, a ferramenta foi desenvolvida usando as tecnologias propostas e, com base nos resultados obtidos, foi possível verificar que os objetivos e justificativas para sua criação foram alcançados. É importante mencionar que o software elaborado neste estudo é uma opção básica e ainda está em sua fase de testes, logo, a sua evolução e aprimoramento é uma necessidade para projetos futuros.

REFERÊNCIAS

Flanagan, David. JavaScript: O Guia Definitivo. Bookman Editora 6ª Edição. Porto Alegre 1996.

AWS, Amazon: **O que é uma API? 2022**. Disponível em: <https://aws.amazon.com/pt/what-is/api/> Acesso em: 06 de agosto de 2022.

BONA, Eduardo: **Programação Backend I**, Disponível em: <https://docplayer.com.br/82749869-Programacao-back-end-i.html> Acesso em: 01 de agosto de 2022.

ROCHA, Gustavo: **Análise de ferramentas computacionais para planejamento estratégico do uso do solo e transportes:** Disponível em: <https://www.teses.usp.br/teses/disponiveis/18/18144/tde-17112010-163004/publico/Rocha_2010.pdf> Acesso em: 03 de agosto de 2022.

REICHELIS, Raquel: **Gestão Pública e a Questão Social na Grande Cidade**. Disponível em: <https://www.scielo.br/j/ln/a/ywJskBcfMPLjtqGf69cDYmt/?format=pdf&lang=pt> Acesso em 05 de agosto de 2022.

CARNEIRO, **Ricardo: Gestão pública no século XXI as reformas pendentes.** Disponível em: <https://books.scielo.org/id/895sg/pdf/noronha-9788581100159-06.pdf> Acesso em 05 de agosto de 2022.

WOLTER, Alzirene Pontoni. VELHO, Altemir da Silva. **Gestão Pública no Brasil: Desafios e perspectivas. Revista Científica Multidisciplinar Núcleo do Conhecimento**. Ano 05, Ed. 02, Vol. 02, pp. 18-27. Fevereiro de 2020. ISSN: 2448-0959.

SANTOS, Milton: **A Urbanização Brasileira**. Editora HUCTEC. São Paulo, 1993.

SANTOS, Reinaldo: **Crescimento desordenado, Segregação Social Nas Cidades Médias Brasileiras: O Caso da Cidade de JUAZEIRO/BAHIA/BRASIL.** Disponível em: <http://observatoriogeograficoamericalatina.org.mx/egal14/Geografiasocioeconomica/Geografiaurbana/066.pdf> Acesso em 05 de agosto de 2022.

BATISTA, Mariana: **Transparência, corrupção e má gestão: uma análise dos municípios brasileiros.** Disponível em: <https://www.scielo.br/j/rap/a/SrM6ZBpBk8czChjjtDZ9ZGr/?lang=pt#> Acesso em: 04 de agosto de 2022.

MARTINS, Cecilia: **Êxodo Rural.** Disponível em: <https://cesad.ufs.br/ORBI/public/uploadCatalago/14353018122013Geografia_Rural_aula_06.pdf> Acesso em: 05 de agosto de 2022

ALVES, Eliseu: **Êxodo e sua contribuição à urbanização de 1950 a 2010**. Disponível em: <https://www.alice.cnptia.embrapa.br/alice/bitstream/doc/910778/1/Exodoesuacontribuicao.pdf> Acesso em: 04 de agosto de 2022.

BODUCH, Adam: **React and React Native.** Editora PACKT 2017.

PORTAL UOL EDUCAÇÃO, **Urbanização do Brasil: Consequências e características das cidades**. Disponível em: <https://educacao.uol.com.br/disciplinas/geografia/urbanizacao-do-brasil-consequencias-e-caracteristicas-das-cidades.htm#:~:text=Faveliza%C3%A7%C3%A3o%20e%20outros%20problemas%20da%20urbaniza%C3%A7%C3%A3o&text=Dentre%20eles%20destacam%2Dse%20o,do%20ar%20e%20da%20%C3%A1gua> Acesso em: 14 de agosto de 2022.

SCHLEICHER, Rafael: **Os Desafios da Administração Pública no Brasil e a Capacitação dos Servidores Públicos.** Disponível em: <https://repositorio.enap.gov.br/bitstream/1/1449/8/Os%20desafios%20da%20administra%C3%A7%C3%A3o%20p%C3%BAblica%20no%20Brasil%20e%20a%20capacita%C3%A7%C3%A3o%20dos%20servidores%20p%C3%BAblicos.pdf> Acesso em: 14 de agosto de 2022.

JASSE, Erminio, **Application programming interface - API para integração de dados em agricultura de precisão**. Disponível em: <http://repositorio.utfpr.edu.br/jspui/handle/1/3420>. Acesso em: 14 de agosto de 2022.

ADAM, Rosangela Aguiar, **Abordando O Problema De Análise De Requisitos Não Funcionais Em Engenharia De Software.** Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/83358/222844.pdf?sequence=1&isAllowed=y>. Acesso em: 12 de dezembro de 2022.

DIEDRICH, Lucas Guilherme, **Integração Da Metodologia Ágil Openup Nos Processos De Engenharia De Software.** Disponível em: <https://riut.utfpr.edu.br/jspui/bitstream/1/23525/2/MD_ENGESS_I_2012_14.pdf> Acesso em: 08 de dezembro de 2022.

TECHOPEDIA: **Educating IT Professionals To Make Smarter Decisions. CROSS-PLATFORM DEVELOPMENT.** Disponível em: <https://www.techopedia.com/definition/30026/cross-platform-development> Acesso em: 29 de novembro de 2022.

NEVES, Jonathan, **Uma Análise Comparativa Entre Flutter E React Native Como Frameworks Para Desenvolvimento Híbrido De Aplicativos Mobile: Estudo De Caso Visando Produtividade.** Disponível em: <ttps://repositorio.animaeducacao.com.br/handle/ANIMA/15960>. Acesso em: 11 de dezembro de 2022.

MACHADO, Filipe Nery. **Analises e Gestão de Requisitos de Software**. 3ª Edição Editora: Erica São Paulo - 2016