MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE PERNAMBUCO CAMPUS GARANHUNS

COORDENAÇÃO DO CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CLOVES JOSÉ DUARTE SILVA

CIDADE MONITOR: PLATAFORMA DE MONITORAMENTO DE AMEAÇAS URBANAS NA CIDADE DE CAJAZEIRAS

GARANHUNS – PE

2025

CLOVES JOSÉ DUARTE SILVA

CIDADE MONITOR: PLATAFORMA DE MONITORAMENTO DE AMEAÇAS URBANAS NA CIDADE DE CAJAZEIRAS

Trabalho de conclusão de curso apresentado à Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Ciência e Tecnologia de Pernambuco – IFPE-Campus Garanhuns, como parte das exigências para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Dr. Jair Galvão de Araújo

Coorientador: Prof. Me. Humberto Beltrão da Cunha Júnior

Coorientador: Prof. Dr. Evaldo de Lira Azevêdo

GARANHUNS – PE

2025

S586c Silva, Cloves José Duarte

Cidade monitor: plataforma de monitoramento de ameaças urbanas na cidade de Cajazeiras / Cloves José Duarte Silva ; orientador Jair Galvão de Araújo, 2025.

70f. : il.

Orientador: Jair Galvão de Araújo.

Trabalho de Conclusão de Curso (Graduação) – Instituto Federal de Pernambuco. Pró-Reitoria de Ensino. Diretoria de Ensino. Campus Garanhuns. Coordenação do Curso de Tecnólogo em Análise e Desenvolvimento de Sistemas. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, 2025.

1. Software – Desenvolvimento. 2. Software de aplicação – Desenvolvimento. 3. Sistemas de comunicação móvel. 4. Computação móvel – Programação. 5. Cidades inteligentes – Inovações Tecnológicas. I. Título. II. Araújo, Jair Galvão de (orientador). III. Instituto Federal de Pernambuco.

CDD 005.1

Louise Machado Freire Dias – CRB4/2267

CLOVES JOSÉ DUARTE SILVA

CIDADE MONITOR: PLATAFORMA DE MONITORAMENTO DE AMEAÇAS URBANAS NA CIDADE DE CAJAZEIRAS

Trabalho de Conclusão de Curso, apresentado à Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, IFPE-Campus Garanhuns, como parte das exigências para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Garanhuns-PE, 09 de abril de 2025

BANCA EXAMINADORA

Prof. Dr. Jair Galvão de Araujo

Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (IFPE)

Campus Garanhuns-PE

Prof. Dr. David Alain do Nascimento

Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (IFPE)

Campus Garanhuns-PE

Prof. Dr. Paulo André da Rocha Pérris

Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (IFPE)

Campus Garanhuns-PE

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais, Cloves José da Silva e Jane Verônica Duarte Silva, pois, sem eles, minha chegada até aqui não teria sido possível de maneira alguma. Agradeço também à minha companheira, Cláudia Fernanda Luna Cavalcante, por me acompanhar e me incentivar ao longo de toda essa jornada. Dirijo meus agradecimentos aos meus irmãos, Cleverton Anderson Duarte Silva e Cataliny Andreza Duarte Silva, por serem meus grandes exemplos desde a infância até a vida adulta. Sou profundamente grato ao Prof. Me. Humberton Beltrão da Cunha Junior por toda a orientação, paciência e apoio, bem como ao Prof. Dr. Evaldo Lira de Azevedo, pelo acom- panhamento, orientação e suporte durante a construção deste trabalho. Por fim, agradeço aos meus amigos Thiago Monteiro de Melo, Lucas dos Santos, Maurício e Wanderley, por estarem ao meu lado em todos os momentos da minha graduação, tornando os dias mais leves e prazerosos.

RESUMO

As ameaças urbanas são provenientes de alterações e movimentações de natureza humana que venham a ser realizadas em um espaço, causando perigos à sociedade por meio da infraestrutura ou de forma ambiental. Tais inovações são criadas devido ao crescimento populacional de cidades e pelo alto grau de urbanização que é visto atualmente. Contudo, essas alterações, em muitas situ- ações, acontecem de maneira acelerada e desordenada, fazendo com que não exista um controle eficiente sobre tais modificações, acarretando em danos ao meio ambiente e à população. Visando melhorar o controle e gestão de ameaças ambientais urbanas, foi proposto o desenvolvimento de um software que permite mapear, armazenar e documentar o maior número possível dessas amea- ças, com base na contribuição popular. O sistema busca fortalecer a segurança e a qualidade de vida da população nas áreas afetadas, ao possibilitar que os moradores participem ativamente no mapeamento dos riscos. O estudo de caso escolhido foi a cidade de Cajazeiras, no interior da Pa- raíba, onde as principais ameaças ambientais foram identificadas e categorizadas. O sistema opera em dois módulos principais: (1) um aplicativo móvel que permite à população registrar uma ame- aça, como deslizamentos de terra, alagamentos, poluição do ar, descarte irregular de resíduos ou erosão do solo, de acordo com uma categoria específica; e (2) um painel de controle para a admi- nistração pública gerenciar esses registros, incluindo informações sobre geolocalização, priori- zando as ações de mitigação de acordo com a gravidade e o impacto de cada ameaça. Essa abor- dagem visa promover uma gestão mais eficiente e colaborativa das ameaças urbanas, tornando a cidade mais resiliente aos desafios ambientais. No entanto, é importante salientar que uma coleta de informações relacionadas à usabilidade do sistema não fez parte do escopo deste projeto. Con- tudo, todo o seu desenvolvimento foi realizado com base no retorno de especialistas de gestão ambiental e administrativa da cidade de Cajazeiras- PB. No aspecto técnico, o sistema denominado Cidade Monitor, foi desenvolvido utilizando uma arquitetura simples e escalável, tendo em vista a otimização de código e as boas práticas de desenvolvimento de software. É esperado que a pla- taforma ajude no gerenciamento ambiental das cidades em que for implementado, gerando a cola- boração da população e da gestão no combate a ameaças urbanas.

Palavras-chave: aplicação móvel; monitoramento participativo; cidades inteligentes.

ABSTRACT

Urban threats arise from human-induced changes and activities carried out in a given area, posing risks to society through infrastructure or environmental means. These innovations are largely driven by the growing population of cities and the high degree of urbanization seen today. However, such changes often occur rapidly and without proper planning, resulting in a lack of effective control and causing harm to both the environment and the population. To improve the control and management of urban environmental threats, the development of software was pro- posed to map, store, and document as many of these threats as possible, based on community input. The system aims to enhance safety and quality of life for people living in affected areas by enabling residents to actively participate in identifying and mapping local risks. The case study selected was the city of Cajazeiras, located in the interior of the state of Paraíba, Brazil, where the main environmental threats were identified and categorized. The system operates through two main modules: (1) a mobile application that allows the public to report threats— such as landslides, flooding, air pollution, illegal waste disposal, or soil erosion—according to specific categories; and (2) a control panel for public administration to manage these reports, including geolocation data, and prioritize mitigation efforts based on the severity and impact of each threat. This approach aims to foster a more efficient and collaborative management of urban threats, making the city more resilient to environmental challenges. However, it is im- portant to note that collecting information on the system's usability was not within the scope of this project. Nonetheless, the entire development process was guided by feedback from envi- ronmental and administrative management experts from the city of Cajazeiras. From a technical perspective, the system—named Cidade Monitor—was developed using a simple and scalable architecture, with a focus on code optimization and software development best practices. The platform is expected to support environmental management in any city where it is implemented, promoting collaboration between the population and local authorities in tackling urban threats.

Keywords: mobile application; participatory monitoring; smart cities.

LISTA DE FIGURAS

Figura 1 - Linha do tempo de desenvolvimento.....**18**
Figura 2 - Diagrama de casos de uso da aplicação web.....**20**
Figura 3 - Diagrama de caso de uso do app mobile.....**25**
Figura 4 - Arquitetura da aplicação.....**28**
Figura 5 - Modelo entidade relacional.....**31**
Figura 6 - Cadastro na plataforma.....**34**
Figura 7 - Cadastro de ameaça.....**35**
Figura 8 - Cadastro de agente.....**36**
Figura 9 - Cadastro de cliente.....**37**
Figura 10 - Abertura de chamado.....**38**
Figura 11 - Resultado dos testes.....**40**
Figura 12 - Página de login na plataforma web.....**57**
Figura 13 - Tela inicial do Cidade Monitor.....**58**
Figura 14 - Página de visualização de chamados abertos.....**59**
Figura 15 - Página de cadastro de categorias.....**60**
Figura 16 - Página de cadastro de agentes.....**61**
Figura 17 - Página de cadastro de município.....**62**
Figura 18 - Página de cadastro de cliente.....**63**
Figura 19 - Página inicial do aplicativo.....**64**
Figura 20 - Página de descrição de ameaça.....**65**
Figura 21 - Página de abertura de chamado.....**66**

LISTA DE QUADROS

Quadro 1 - Modelo de dados da tabela tb_municipality.....**48**
Quadro 2 - Modelo de dados da tabela tb_agent.....**49**
Quadro 3 - Modelo de dados da tabela tb_client.....**51**
Quadro 4 - Modelo de dados da tabela tb_threat......**53**
Quadro 5 - Modelo de dados da tabela tb_call.....**55**
Quadro 6 - Modelo de dados da tabela tb_image.....**56**

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
ER	Entidade Relacionamento
POO	Programação Orientada a Objeto
SGBD	Sistema de Gerenciamento de Banco de Dados
SPA	*Single Page Application*
SSR	*Server Side Rendering*
UML	*Unified Modeling Language*

SUMÁRIO

1. INTRODUÇÃO.....12

2. OBJETIVOS.....14

2.1. GERAL.....14

2.2. ESPECÍFICOS.....14

3. RELEVÂNCIA DO ESTUDO.....15

3.1. CONCLUSÃO.....15

4. METODOLOGIA.....17

5. DOCUMENTAÇÃO.....19

6. SISTEMA WEB.....20

6.1. VISUALIZAR CHAMADOS.....20

6.2. FINALIZAR CHAMADOS.....21

6.3. CADASTRAR CATEGORIA.....21

6.4. EDITAR CATEGORIA.....22

6.5. DELETAR CATEGORIA.....23

6.6. EFETUAR LOGIN.....24

6.7. REALIZAR CADASTRO.....24

7. APLICAÇÃO MOBILE.....25

7.1. EFETUAR CADASTRO.....25

7.2. EFETUAR LOGIN.....25

7.3. ABRIR CHAMADO.....26

8. ARQUITETURA.....28

8.1. FRONT-END WEB.....29

8.2. FRONT-END MOBILE.....29

8.3. BACK-END.....29

8.4. ARMAZENAMENTO DE DADOS.....**29**

8.5. AUTENTICAÇÃO.....**29**

8.6. RESUMO DA ARQUITETURA.....**30**

9. MODELAGEM DO PROJETO.....**31**

10. MODELO DE DADOS.....**33**

11. DIAGRAMA DE SEQUÊNCIA.....**34**

11.1. CADASTRO DE MUNICÍPIO.....**34**

11.2. CADASTRO DE AMEAÇAS.....**35**

11.4 . CADASTRO DE AGENTE.....**35**

11.5 . CADASTRO DE CLIENTES.....**36**

11.6 . ABERTURA DE CHAMADOS.....**37**

12. RELATÓRIO DE EXECUÇÃO DE TESTES.....**39**

12.1. OBJETIVO GERAL DOS TESTES.....**39**

12.2. AMBIENTE DE TESTES.....**39**

13. RESULTADOS DOS TESTES.....**40**

13.1. TESTE DO MÉTODO GETTHREAT.....**40**

13.2. TESTE DO MÉTODO GETTHREATBYID.....**40**

13.3. TESTE DO MÉTODO POSTTHREAT.....**41**

13.4. TESTE DO MÉTODO GETCALL.....**41**

13.5. TESTE DO MÉTODO FINALIZECALL.....**41**

13.6. TESTE DO MÉTODO GETAGENTS.....**42**

13.7. TESTE DO MÉTDO GETAGENTSBYID.....**42**

13.8. TESTE DO MÉTODO POSTAGENTS.....**43**

13.9. TESTE DO MÉTODO POSTMUNICIPALITY.....**43**

13.10. TESTE DO MÉTODO SIGNIN.....**43**

13.11. TESTE DO MÉTODO POSTCLIENT.....**44**

13.12. TESTE DO MÉTODO SIGNINCLIENT....44
14. CONCLUSÃO....45
15. REFERÊNCIAS....46
APÊNDICE A – MODELO DE DADOS DA TABELA TB_MUNICIPALITY....48
APÊNDICE B – MODELO DE DADOS DA TABELA *TB_AGENT*....49
APÊNDICE C – MODELO DE DADOS DA TABELA *TB_CLIENT*....51
APÊNDICE D – MODELO DE DADOS DA TABELA *TB_THREAT*....53
APÊNDICE E – MODELO DE DADOS DA TABELA *TB_CALL*....55
APÊNDICE F – MODELO DE DADOS DA TABELA *TB_IMAGE*....56
APÊNDICE G – PÁGINA DE LOGIN NA PLATAFORMA WEB....57
APÊNDICE H – PÁGINA INICIAL DA PLATAFORMA....58
APÊNDICE I – PÁGINA DE VISUALIZAÇÃO DE CHAMADOS ABERTOS....59
APÊNDICE J – PÁGINA DE CADASTRO DE CATEGORIAS....60
APÊNDICE K – PÁGINA DE CADASTRO DE AGENTES....61
APÊNDICE L – PÁGINA DE CADASTRO DE MUNICÍPIO....62
APÊNDICE M – PÁGINA DE CADASTRO DE CLIENTE....63
APÊNDICE N – PÁGINA INICIAL DO APLICATIVO....64
APÊNDICE O – PÁGINA DE DESCRIÇÃO DA AMEAÇA....65
APÊNDICE P – PÁGINA DE ABERTURA DE CHAMADO....66
APÊNDICE Q – REPOSITÓRIO FRONT-END WEB....67
APÊNDICE R – REPOSITÓRIO FRONT-END MOBILE....68
APÊNDICE S – REPOSITÓRIO BACK-END....69

1. INTRODUÇÃO

O êxodo rural deu-se início diante da alta globalização e revoluções industriais que afe- taram a sociedade, processo esse em que as pessoas migraram do campo para a cidade no intuito de melhores condições de vida. Durante o século XIX, a taxa de emprego e demanda por pro- dução aumentou consideravelmente, fazendo com que a população migrasse cada vez mais para as grandes cidades (Campos & Branco, 2021).

No entanto, não estava previsto que essa movimentação ocorresse de forma desorde- nada, fazendo com que as áreas urbanas não tivessem capacidade ideal para acomodar a todos. Uma das consequências é o surgimento de favelas, composta por famílias de baixa renda, pelo adensamento e intensidade na ocupação do solo, pela carência de infraestrutura, pela dificul- dade no acesso aos serviços e equipamentos sociais ofertados pela cidade e pela insalubridade da moradia (Pires, 2016).

Tais modificações na sociedade e na estrutura local impactaram, portanto, na degradação do ambiente, como os buracos nas vias públicas, risco de incêndio e enchentes, infraestrutura mal elaborada e outros casos (Duarte & Farias Filho, 2020), os quais, prejudicam o direito de ir e vir das pessoas e a organização de órgãos públicos para correção desses problemas (Duarte & Farias Filho, 2020).

É fundamental que exista uma forma de mapeamento dessas ameaças urbanas, para que os órgãos responsáveis possam identificar quais são elas. Projetos como ECOSIM (Colléter et al., 2025) e *Smart Healthy Enviroment* (Ullo & Sinha, 2020) demonstram diferentes abordagens para integrar tecnologias de monitoramento ambiental urbano. Enquanto o primeiro enfatiza o uso de *Geographical information systems (GIS)*, modelagem e interfaces analíticas em uma arquitetura cliente-servidor (Fedra, 1999), o segundo foca em soluções participativas, que uti- liza redes de sensores e dados gerados pela população para promover cidades mais inteligentes e saudáveis (Bacco et al., 2017). Essas abordagens oferecem diretrizes valiosas para o desen- volvimento de aplicações que combinam tecnologias emergentes e engajamento comunitário no enfrentamento de desafios urbanos.

Sendo assim, o objetivo deste estudo foi desenvolver um *software* que fosse capaz de mapear e monitorar as ameaças ambientais urbanas, por meio de um aplicativo mobile, voltado à sociedade civil, e uma aplicação *web* voltada aos órgãos gestores. Para a construção do projeto foram utilizadas as bibliotecas *Next.js*, para a criação do *front-end web*, *Expo* para a criação do *front-end mobile* e *Nest.js* para o desenvolvimento do *back-end.*

O intuito desse trabalho foi a criação de um relatório técnico, para que fosse possível a aplicação prática de certos conceitos de engenharia de *software* no desenvolvimento de uma

aplicação *web*, bem como a elaboração de um aplicativo *mobile* que obtenha as funções relaci- onadas à proposta apresentada.

2. OBJETIVOS

2.1. GERAL

O objetivo deste documento é elaborar um aplicativo chamado “Cidade Monitor”, fundamentado nos princípios da Engenharia de Software, os quais serão explicados em suas funcionalidades. O “Cidade Monitor” é um software concebido com foco no gerenciamento de problemas urbanos, e sua principal base foi o projeto “Aquameaça” (Vieira et al. 2019), que propunha criar uma forma de monitoramento de ameaças aquáticas.

2.2. ESPECÍFICOS

Desenvolver uma aplicação web para monitoramento e registro de novas ameaças ambientais urbanas em grandes centros, de acordo com a categorização das ameaças disponibilizadas.

Desenvolver uma aplicação mobile para o registro de ameaças urbanas existentes na cidade.

Desenvolver uma *API* para comunicação entre aplicação *mobile* e o sistemas *web*.

Auxiliar na colaboração participativa na resolução de problemas ambientais comuns aos principais centros urbanos.

3. RELEVÂNCIA DO ESTUDO

A escolha deste tema se dá pela importância de monitoramento de ameaças ambientais que possam se tornar danosas para populações que habitam áreas urbanas. Tendo em vista a atual situação ambiental em que vivemos, um mecanismo de mapeamento de ameaças é funda- mental para que exista uma resposta rápida das autoridades, minimizando os riscos e tornando o ambiente mais seguro para a sociedade e a natureza. A luz desse aspecto, é possível pontuar a relevância desse estudo por meio dos seguintes pontos: importância ambiental, tecnologia, contribuição popular e políticas públicas.

Observando do ponto de vista ambiental, tendo em vista a atual situação climática em que vivemos, um sistema desenvolvido com o objetivo de registrar e documentar ameaças am- bientais é de grande importância para mitigar os danos causados ao meio ambiente, promovendo também uma melhoria no bem-estar da população. Pode-se pontuar também que por meio da centralização de informações é possível promover uma investigação acurada das principais cau- sas e consequências dos riscos urbanos, tanto para o ecossistema quanto para a população.

Pode-se observar também que a aplicação da tecnologia no auxílio à prevenção de ame- aças urbanas e ambientais é uma maneira eficaz e inteligente de utilizar recursos tecnológicos como veículo de melhoria na qualidade de vida e de melhora ambiental.

Deve-se salientar também que, por meio da criação do aplicativo, será possível criar um ecossistema colaborativo entre população e gestão, onde o cidadão também é agente de cidada- nia, pois colabora diretamente com o cuidado e manutenção do ambiente em que vive. Desse modo é possível dizer que população e gestão trabalharam juntas em favor de uma melhora coletiva da localidade.

É válido salientar também que o mapeamento e monitoramento de ameaças ambientais é de grande valia durante a formulação de políticas públicas, criando um equilíbrio entre áreas passíveis de ameaça e áreas menos afetadas.

3.1. CONCLUSÃO

Conclui-se, portanto, que o desenvolvimento de um sistema de mapeamento e monito- ramento de ameaças ambientais representa uma importante contribuição para a gestão pública e a sustentabilidade urbana. Por meio da integração entre tecnologia, participação popular e

ações governamentais, é possível criar uma estrutura robusta que não apenas identifica e mitiga riscos ambientais, mas também promove o engajamento cidadão e a conscientização coletiva. Esse modelo colaborativo fortalece o elo entre população e gestão, incentivando a preservação ambiental e a melhoria da qualidade de vida nas áreas urbanas.

Além disso, a centralização e análise de dados obtidos por meio do aplicativo viabilizam uma resposta mais ágil e eficaz às emergências ambientais, bem como subsidiam a formulação de políticas públicas baseadas em evidências. Ao unir tecnologia e cidadania, o projeto poten- cializa os esforços de proteção ao meio ambiente e à sociedade, estabelecendo um caminho promissor para a construção de cidades mais resilientes, equilibradas e sustentáveis.

Em síntese, o monitoramento de ameaças ambientais em áreas urbanas apresenta-se como uma ferramenta essencial para a promoção da segurança e do bem-estar tanto da popula- ção quanto do meio ambiente. A atual crise climática e os desafios urbanos exigem respostas ágeis e eficazes, e um sistema de mapeamento de riscos ambientais surge como uma solução viável para mitigar danos e prevenir desastres. Ao centralizar informações e utilizar tecnologia avançada, é possível identificar, analisar e agir sobre as principais causas e consequências des- sas ameaças, beneficiando tanto o ecossistema quanto as comunidades que nele vivem.

4. METODOLOGIA

Durante o período de desenvolvimento da aplicação, algumas metodologias que contri- buíram para a realização de um processo de produção eficaz, sendo elas: análise de requisitos, desenvolvimento de interfaces de uso, desenvolvimento *back-end*, teste e documentação.

A realização de uma pesquisa com maior profundidade foi de grande importância para definição dos requisitos funcionais e não funcionais que deveriam existir, tanto no aplicativo *mobile*, quanto no sistema *web*, sendo essa etapa denominada de análise de requisitos. Durante esse processo foram organizados encontros quinzenais diretamente com a secretaria de meio ambiente do município de Cajazeiras, esses encontros foram essenciais para que o aplicativo atendesse as principais necessidades do município.

Para a criação das interfaces de usuário foram utilizadas ferramentas de desenvolvimento *front-end*, tanto *web* quanto *mobile*, todas a telas foram criadas de modo a atender da melhor maneira as regras de negócio solicitadas pela gestão do município.

Já na parte de desenvolvimento *back-end* foi implementada uma *Application Programming Interface* (*API*) com as regras de negócio relatadas durante a análise de requisitos, essa *API* será utilizada na comunicação entre aplicativo *mobile* e aplicação *web*.

A criação de testes unitários foi relevante para a verificação e a criação de novas regras de negócio. Também é possível pontuar que os testes foram imprescindíveis para o desenvolvi- mento de uma aplicação segura, atendendo aos requisitos não funcionais de segurança.

Vale destacar que, para ampliar a abrangência da análise, foram consultados artigos científicos que abordam os principais tipos de ameaças ambientais urbanas no Brasil, como a poluição da água (Brovini et al., 2021) e a contaminação do solo por metais pesados resultantes do processo de urbanização (Penteado et al., 2021) e suas consequências para crianças em fase de desenvolvimento, como pneumonia, baixo peso ao nascer e comprometimentos no desen- volvimento psicomotor e mental (Froes Asmus et al., 2016), o risco pode ser associado às no- ções de incerteza, exposição ao perigo, perda prejuízos em decorrência de processos naturais ou associados às relações humanas (Castro et el., 2005, p.12).

Por fim, a documentação detalhada foi produzida para facilitar a incorporação e utiliza- ção da plataforma, aumentando assim a adesão dos usuários finais. Com essa abordagem, pla- nejou-se um ciclo de desenvolvimento eficiente, onde todas as etapas foram cuidadosamente definidas e planejadas, resultando na entrega de um *software* escalável.

Figura 1 - Linha do tempo de desenvolvimento

Fonte: O autor (2024)

5. DOCUMENTAÇÃO

Os atores e os casos de uso da plataforma estão representados a seguir:

- atores:
  - agente (usuário da plataforma *web*)
  - cliente (usuário da plataforma *mobile*)
- identificação dos casos de uso:
  - visualizar registro;
  - finalizar chamado;
  - cadastro de categorias;
  - edição de categorias;
  - exclusão de categorias;
  - *login*;
  - criação de conta;
- identificação dos casos de uso do cliente:
  - cadastro de cliente;
  - *login*;
  - cadastro de chamado;

a) identificação dos casos de uso do cliente:

a. criar conta na aplicação;

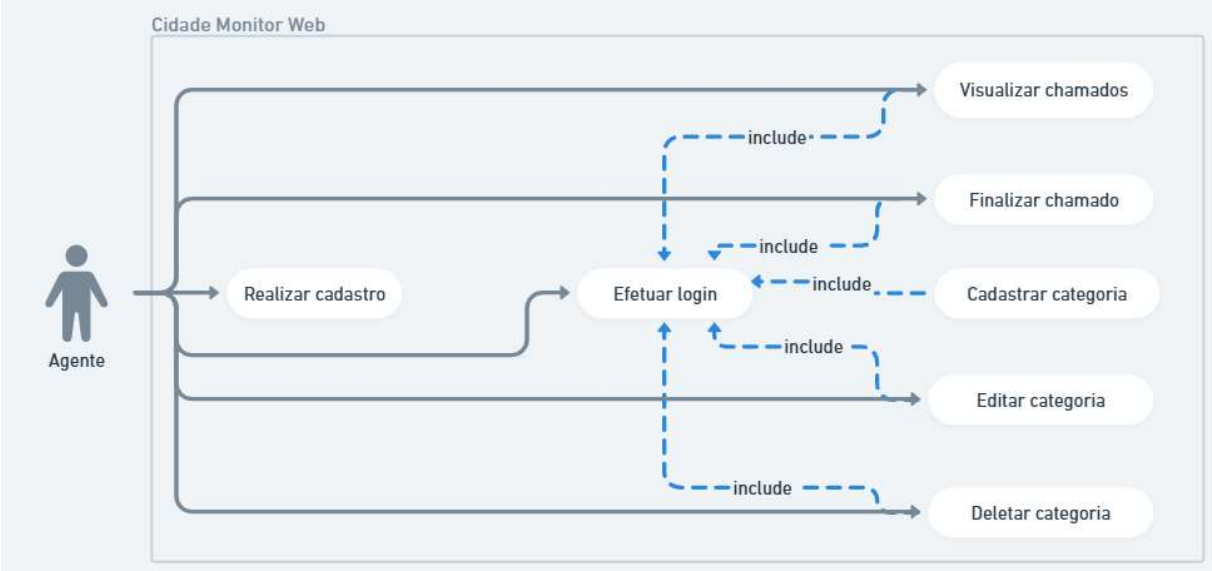
b. realizar *login*;

c. cadastrar chamado;

6. SISTEMA *WEB*

Na Figura 2 é possível visualizar o diagrama de uso da aplicação *web*.

Figura 2 - Diagrama de casos de uso da aplicação web



Fonte: O autor (2024)

Conforme é possível observar na Figura 2, estão presentes os casos de uso para que o sistema *web* venha a atingir seu objetivo. A seguir serão descritos todos os casos de uso representados no diagrama.

6.1. VISUALIZAR CHAMADOS

O modelo de caso de uso para a funcionalidade de visualização de chamados está descrito logo abaixo:

- ator:
  - agente
- descrição:
  - o agente poderá visualizar no mapa todos os chamados abertos.
- pré-condições:
  - o agente deverá estar autenticado no sistema;
  - devem existir chamados já cadastrados naquela região.
- fluxo principal:
  - realizar login na plataforma
- pós-condições:
  - visualizar todos os chamados registrados.

O modelo apresentado anteriormente descreve como é possível visualizar os chamados efetuados pelos clientes.

6.2. FINALIZAR CHAMADOS

O modelo de caso de uso para a funcionalidade de finalização de chamados está descrito logo abaixo.

- ator:
  - agente
- descrição
  - o agente poderá ir até a opção de “Chamados”, no menu lateral da aplicação. Clicando nessa opção, ele será redirecionado para a página de chamados, onde será possível visualizar todos os chamados abertos pelos usuários no aplicativo. Clicando na coluna “Ações”, presente na listagem, o agente poderá finalizar um chamado caso ele tenha sido concluído.
- pré-condições:
  - devem existir chamados já cadastrados naquela região.
- fluxo principal
  - navegar até a opção “Chamados”, no menu lateral do sistema.
- pós-condições
  - deverá ser exibido um modal de confirmação contendo o título do chamado.
  - após a sua finalização a listagem deverá ser atualizada, deixando de exibir o chamado finalizado.
  - deverá ser exibido um aviso na parte superior direita indicando o sucesso ou falha durante o processo.

O modelo apresentado anteriormente descreve como é efetuado o processo de finalização de chamados abertos.

6.3. CADASTRAR CATEGORIA

O modelo de caso de uso para a funcionalidade de cadastro de ameaças está descrito logo abaixo:

- ator:
  - agente.
- descrição:

    - o agente poderá navegar até a opção “Categorias”, no menu lateral da aplicação. Clicando nessa opção ele será redirecionado para a página de categorias, onde será possível visualizar a listagem de categorias já cadastradas.
- pré-condições:
    - navegar até a opção “Categorias”, no menu lateral do sistema;
    - clicar no botão “Adicionar nova categoria”
    - preencher corretamente os campos obrigatórios do formulário;
    - clicar no botão “Cadastrar”
- pós-condições:
    - o modal deverá ser fechado e na parte superior direita deverá ser exibido um aviso indicando o sucesso ou falhar durante o processo;
    - a listagem deverá ser atualizada exibindo a nova ameaça cadastrada.

O modelo descrito anteriormente descreve a ação de cadastrar ameaças.

6.4. EDITAR CATEGORIA

O modelo de caso de uso para a funcionalidade de edição de ameaças está descrito logo abaixo:

- ator:
    - agente.
- descrição:
    - o agente poderá navegar até a opção “Categorias”, no menu lateral da aplicação. Clicando nessa opção ele será redirecionado para a página de categorias, onde será possível visualizar a listagem de categorias já cadastradas.
- pré-condições:
    - navegar até a opção “Categorias” no menu lateral;
    - existir categorias já cadastradas.
- fluxo principal
    - navegar até a opção “Categorias” no menu lateral;
    - na coluna “Ações” clicar no botão com ícone de três pontos e selecionar a opção “Editar categoria”;
    - deve ser aberto um modal de edição contento todas as informações relacionadas a ameaça selecionada;

    - após alteração dos campos, o agente poderá finalizar o processo de edição clicando em “Cadastrar”.
- pós-condições:
    - o modal deverá ser fechado e na parte superior direita deverá ser exibido um aviso indicando o sucesso ou falha durante o processo;
    - a listagem deverá ser atualizada exibido a ameaça editada.

A descrição acima relatada descreve acerca da ação de edição de ameaças já cadastradas.

6.5. DELETAR CATEGORIA

O modelo de caso de uso para a funcionalidade de exclusão de ameaças está descrito logo abaixo:

- ator:
    - agente.
- descrição:
    - o agente poderá navegar até a opção “Categorias”, no menu lateral da aplicação. Clicando nessa opção ele será redirecionado para a página de categorias, onde será possível visualizar a listagem de categorias já cadastradas.
- pré-condições:
    - navegar até a opção “Categorias” no menu lateral;
    - existir categorias já cadastradas.
- fluxo principal:
    - realizar login na plataforma;
    - navegar até a opção “Categorias” no menu lateral;
    - na coluna “Ações” clicar no botão com o ícone de três pontos e selecionar a opção “Deletar categoria”;
    - deve ser aberto um modal de confirmação contendo o título da categoria que será deletada.
- pós-condições:
    - o modal deverá ser fechado e na parte superior direita deverá ser exibido um aviso indicando o sucesso ou falha durante o processo;
    - a listagem deverá ser atualizada exibindo somente as categorias não deletadas.

Acima foi descrita a funcionalidade de deletar categorias.

6.6. EFETUAR *LOGIN*

O modelo de caso de uso para a funcionalidade de *login* de ameaças está descrito logo abaixo:

- ator:
  - agente.
- descrição:
  - ao digitar o endereço do site no navegador, o agente poderá ter acesso aos servi- ços oferecidos pelo sistema realizando o *login*. Para isso ele deverá informar o e-mail e senha e clicar em “Entrar”.
- pré-condições:
  - ter um cadastro ativo na plataforma.
- fluxo principal:
  - acessar o endereço de hospedarem do sistema no navegador *web*;
  - digitar as credenciais de *login*.
- pós-condições:
  - o usuário será redirecionado para a página principal do sistema.

O modelo apresentado descreve a ação de login na aplicação *web*.

6.7. REALIZAR CADASTRO

O modelo de caso de uso para a funcionalidade de cadastro no sistema está descrito logo abaixo:

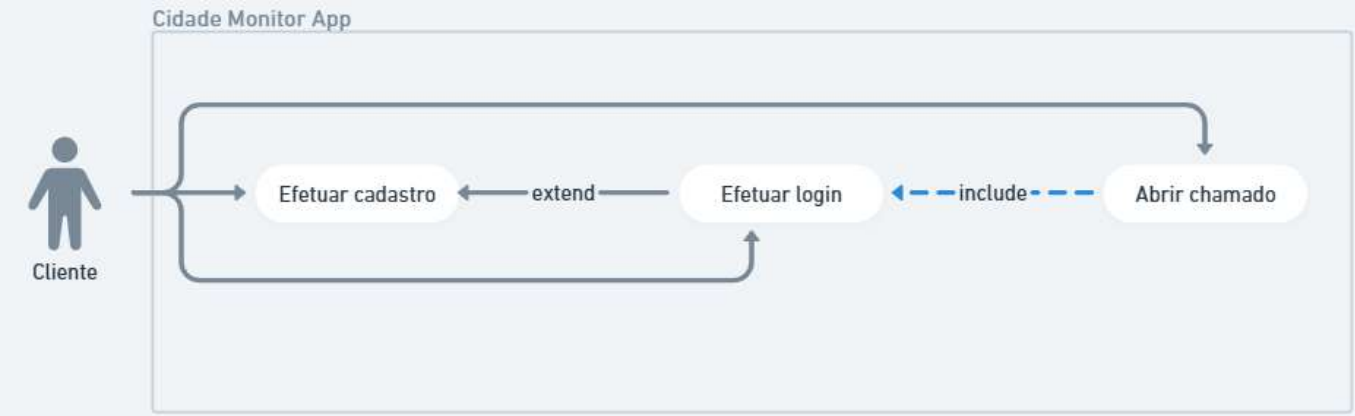
- ator:
  - agente.
- descrição:
  - ao acessar o sistema, caso não exista nenhuma conta cadastrada, o usuário poderá criar uma conta a partir da página “Cadastre-se”.
- pré-condição:
  - acessar o endereço da plataforma;
  - acessar a página “Cadastre-se”
- fluxo principal:
  - acessar a página “Cadastre-se”;
  - efetuar preenchimento do formulário de cadastro;
- pós-condições:
  - O usuário terá acesso às funcionalidades da plataforma.

A descrição acima explica como o agente poderá efetuar o seu cadastro.

7. APLICAÇÃO *MOBILE*

Abaixo é possível visualizar o diagrama de uso da aplicação *mobile.*

Figura 3 - Diagrama de caso de uso do *app mobile*



Fonte: O autor (2024)

Dado o exposto, é possível visualizar na Figura 3 todos os casos de uso presentes no sistema *mobile*. Adiante serão descritos todos os casos de uso representados no diagrama.

7.1. EFETUAR CADASTRO

O modelo de caso de uso para a funcionalidade de cadastro está descrito logo abaixo:

- ator:
  - cliente
- descrição
  - ao realizar o download do aplicativo em seu celular, o cliente poderá efetuar o registro no aplicativo pressionando o botão “Cadastre-se”.
- pré-condição:
  - ter efetuado o *download* do aplicativo em seu celular.
- fluxo principal:
  - acessar o aplicativo já instalado no celular;
  - pressionar o botão “Cadastre-se”.
- pós-condições:
  - o cliente poderá efetuar o login no aplicativo.

Acima foi descrito o modelo de cadastro no aplicativo *mobile*.

7.2. EFETUAR *LOGIN*

O modelo de caso de uso para a funcionalidade de login está descrito logo abaixo:

- ator:

  - cliente
- descrição:
  - ao efetuar o login o cliente terá acesso as funcionalidades do aplicativo.
- pré-condições:
  - ter realizado o cadastro no aplicativo.
- fluxo principal:
  - preencher as credenciais de *login*.
- pós-condições:
  - o cliente poderá ter acesso aos serviços oferecidos pelo aplicativo.

Acima foi possível observar o processo de realização de *login* no aplicativo. Para que o cliente possa efetuar chamados, ele deverá estar devidamente autenticado na plataforma, o que garante um maior nível de segurança.

7.3. ABRIR CHAMADO

O modelo de caso de uso para funcionalidade de cadastro de chamado está descrito logo abaixo:

a) ator:
   a. cliente
b) descrição:
   a. após a realização do login no aplicativo, o usuário terá acesso a uma listagem de ameaças. Ele poderá selecionar a ameaça que melhor se encaixa na situação;
   b. após selecionar a ameaça ele será redirecionado para uma tela de confir- mação contendo uma descrição da ameaça. Ele poderá prosseguir com o cadastro ou voltar para a listagem e selecionar outra opção melhor con- dizente com a situação;
   c. caso prossiga com o cadastro ele será redirecionado para o formulário de cadastro onde preencherá os dados para que o cadastro do chamado possa ser efetuado.
c) pré-condição:
   a. ter efetuado cadastro no aplicativo;
   b. estar autenticado no aplicativo.
d) fluxo principal:
   a. selecionar ameaça condizente com a situação encontrada;

b. confirmação de que o chamado que será aberto pertence a ameaça selecionada.
c. Preencher corretamente o formulário de cadastro para que o chamado possa ser cadastrado com sucesso.

e) pós-condições:
   a. o cliente será redirecionado para a listagem de ameaças, onde poderá re- alizar um novo cadastro de ameaça, ou sair do aplicativo.

O modelo apresentado descreve como o cliente poderá efetuar o cadastro de um chamado pelo aplicativo.

8. ARQUITETURA

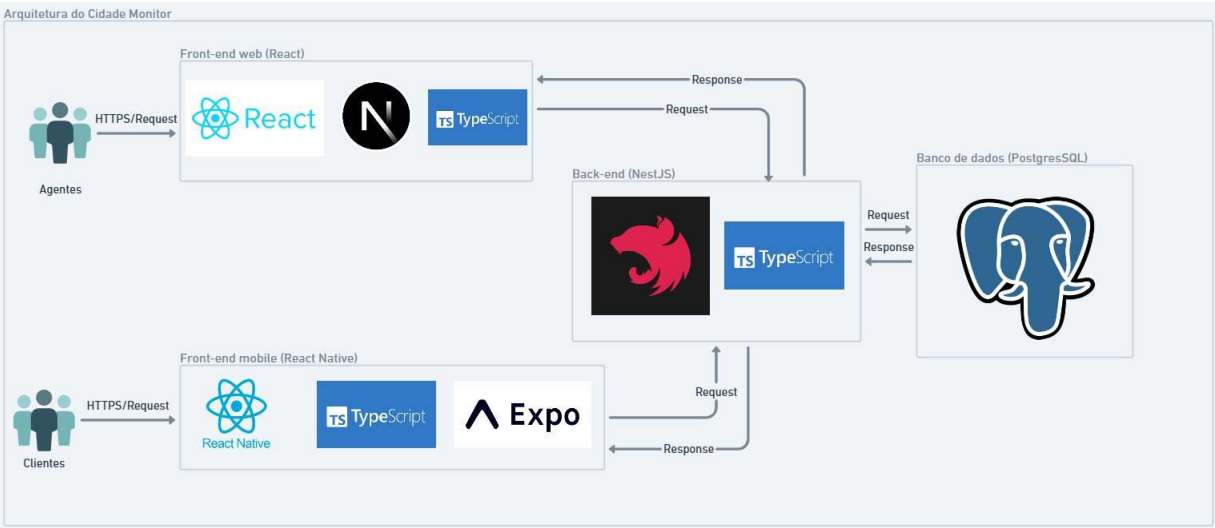
De acordo com Sommerville (2019), a arquitetura de *software* tem como objetivo com- preender e organizar adequadamente o sistema, sendo este o primeiro passo no processo de projeto. Também pode ser considerado o vínculo principal entre o processo de engenharia de requisitos e o projeto.

A arquitetura pode ser pontuada como uma maneira de estruturar o código da maneira mais adequada possível, visando atender não somente as especificações técnicas compreendidas durante a fase de elicitação de requisitos, mas também as nuances relacionadas à segurança e escalabilidade do código. As decisões arquiteturais são imprescindíveis para a construção de um sistema longevo.

Visando uma maior ênfase na construção do *software*, a arquitetura é responsável pela organização lógica do código, sua estrutura de classes, componentes e funções. Por meio dela é possível tomar decisões que influenciam no bom funcionamento do projeto e na simplificação do código, evitando assim que o código se torne legado antes mesmo da disponibilização do sistema para o público, desse modo pode-se dizer que a arquitetura enfatiza a criação de um sistema sustentável, escalável e adaptável, possibilitando tanto sua expansão quanto a sua ma- nutenção.

Dado o exposto, é possível observar que para que a plataforma funcione de maneira correta, é de suma importância uma arquitetura bem estruturada (Figura 4).

Figura 4 - Arquitetura da aplicação



Fonte: O autor (2024).

Conforme apresentado na Figura 4, a arquitetura do aplicativo Cidade Monitor é composta por uma variedade de ferramentas.

8.1. FRONT-END WEB

Para o desenvolvimento do *front-end*, optou-se pelo uso da biblioteca *React*. Ela permite a criação de interfaces baseadas em componentes, o que facilita a reutilização desses elementos conforme o contexto da página em tempo real. Esses componentes são desenvolvidos usando HTML (*Hypertext Markup Language*) e podem conter estados e funcionalidades específicas. Além do *React*, foi utilizada a *framework Next.js* com o objetivo de acelerar o processo de desenvolvimento. O *Next.js* já oferece uma estrutura pré-configurada para o projeto, incluindo funcionalidades como o SSR (*Server-Side Rendering*), que possibilita a renderização das pági- nas no servidor. Este aspecto torna-se benéfico, tendo em vista que a exigência de processa- mento de dados no lado do cliente será muito menor.

8.2. FRONT-END MOBILE

Para o *front-end mobile*, foi usada a biblioteca *React-Native* que, assim como o *React*, possibilita a criação de telas componentizadas com lógicas e estados próprios. Devido à complexidade do aplicativo, optou-se por utilizar o *framework Expo*, por seus *plugins* e facilidades com integração via *APIs*.

8.3. BACK-END

O *back-end* foi desenvolvido com *Node.js* e o *framework Nest.js*, que facilita a organização do código seguindo o padrão MVC (*Model-View-Controller*). O *Nest.js* foi projetado para resolver problemas de arquitetura em aplicações *server-side*, adotando princípios de Programa- ção Orientada a Objeto (POO) para garantir uma estrutura robusta, que simplifica o desenvol- vimento e melhora a escalabilidade e manutenção da aplicação.

8.4. ARMAZENAMENTO DE DADOS

Os dados cadastrados serão armazenados no *PostgresSql*, um Sistema Gerenciador de Bando de Dados (SGBD) objeto relacional. Devido a sua flexibilidade, o *Postgres* suporta tipos de dados tanto relacionais, quanto não relacionais, garantindo uma alta flexibilidade e confia- bilidade dos dados.

8.5. AUTENTICAÇÃO

O armazenamento de dados para autenticação do usuário fica a cargo do *Postgres*, sendo o *Nest.js* responsável por fornecer as rotas de autenticação. Já o *Postgres* é responsável pelo armazenamento dos dados de login e senha do usuário vinculado a cada município.

8.6. RESUMO DA ARQUITETURA

Em resumo, pode-se pontuar que o Cidade Monitor é composto por uma série de tecno- logias para a criação de um sistema robusto e escalável. Por meio delas foi possível desenvolver uma aplicação sofisticada com recursos que garantem a sua estabilidade. Através da utilização de padrões arquiteturais foi possível desenvolver uma aplicação que segue as boas práticas da produção de *software*. Por meio destas práticas foi possível desenvolver uma *API* consistente, utilizando o *Nest.js* para garantir a estrutura correta com base em orientação a objetos e uma estrutura dividida em camadas, garantindo assim o encapsulamento e o não acoplamento de código.

No lado do *front-end web* foi possível utilizar um framework pronto para a construção de código consistente e reutilizável, por meio da componentização, principal característica do *React*, foi possível desenvolver um *Single-page application (SPA)* responsivo por meio da fun- cionalidade de *Server-Side Rendered (SSR)* sendo possível desenvolver um *front* mais consis- tente com um menor nível de processamento do lado do cliente, possibilitando uma maior aces- sibilidade ao sistema, diminuindo as barreiras de uso.

9. MODELAGEM DO PROJETO

> "Conforme o modelo de projeto evolui, é definido um conjunto de classes de projeto que refina as classes de análise, fornecendo detalhe de projeto que permitirá que elas sejam implementadas e crie um novo conjunto de classes de projeto que implemente uma infraestrutura de *software* que suporte a solução de negócio". (PRESSMAN & MAXIM, 2021, p. 365).

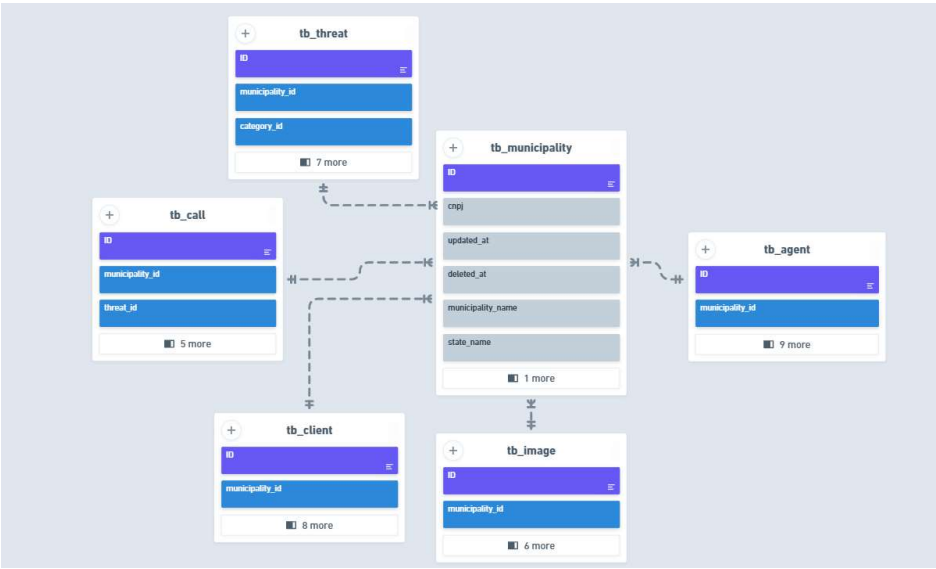
Conforme é pontuado por Pressman e Maxim (2021), na medida em que o projeto se desenvolve uma modelagem mais acurada se faz necessária para que exista um melhor entendimento de quais funcionalidades uma classe deverá realizar, essa análise mais aprofundada é comumente representada por *Unified Modeling Language* (UML) ou Entidade-Relacional (ER). Esse modelo de representação é fundamental para que exista uma maior compreensão da equipe de desenvolvimento. Além disso, a modelagem é responsável pela diminuição do nível de abstração de um software, garantindo a construção de classes bem definidas quanto a sua utilidade.

Pode-se observar também que a modelagem do projeto tem seu papel vital no quesito de dívida técnica, ou seja, o tempo de retrabalho que é gasto durante a produção de um *software* devido a soluções de baixa qualidade, implementadas durante as fazes iniciais do projeto.

Portanto, pode-se concluir que a modelagem possui significativa importância na criação de um *software*, diminuindo seus custos e prolongando seu bom funcionamento.

Segue abaixo a modelagem do projeto Cidade Monitor, disponível na Figura 5.

Figura 5 - Modelo entidade relacional



Fonte: O autor (2024).

Na figura acima foi possível observar a modelagem Entidade-Relacional do projeto Ci- dade Monitor. Nela é possível observar um conjunto de sete entidades: *tb_municipality*,

tb_agent, *tb_category*, *tb_threat, tb_call, tb_client* e *tb_image. E*ssas entidades serão responsá- veis por compor a lógica responsável pelo gerenciamento de ameaças.

A entidade *tb_municipality* possui cardinalidade de *muitos* para *um* (1, n) com todas as demais entidades, o que significa que cada uma das entidades deve ter seus dados associados a apena um município. Essa modelagem é fundamental para que o *software* possa ser utilizado por vários municípios, e isso se dá pelo fato de cada dado armazenado estará vinculado ao *id* específico de uma localidade. As seguintes entidades possuem o mesmo tipo de relacionamento:

- *tb_municipality e tb_agent*
- *tb_municipality e tb_threat*
- *tb_municipality e tb_call*
- *tb_municipality e tb_client*
- *tb_municipality e tb_image*

Entre as entidades *tb_threat* e *tb_call* existe um relacionamento de ***um*** para ***um*** (1, 1), ou seja, um chamado só pode estar associado a apenas um tipo de ameaça.

10. MODELO DE DADOS

Para a criação do banco de dados foi preferido o *PostgresSQL*, um banco de dados rela- cional. Neste tipo de banco, “os dados são percebidos pelos usuários como tabelas”. (Date, 2004, p.74). Nos bancos de dados relacionais cada registro possui um identificador único, cha- mado de chave primária. Esta chave é “escolhida pelo projetista de banco de dados como o principal meio de identificar tuplas dentro de uma relação”. (Silberschatz et al., 2012, p.29).

“Um banco de dados relacional consiste em uma coleção de tabelas, cada qual recebendo um nome exclusivo.” (Silberschatz et al., 2012, p. 25).

11. DIAGRAMA DE SEQUÊNCIA

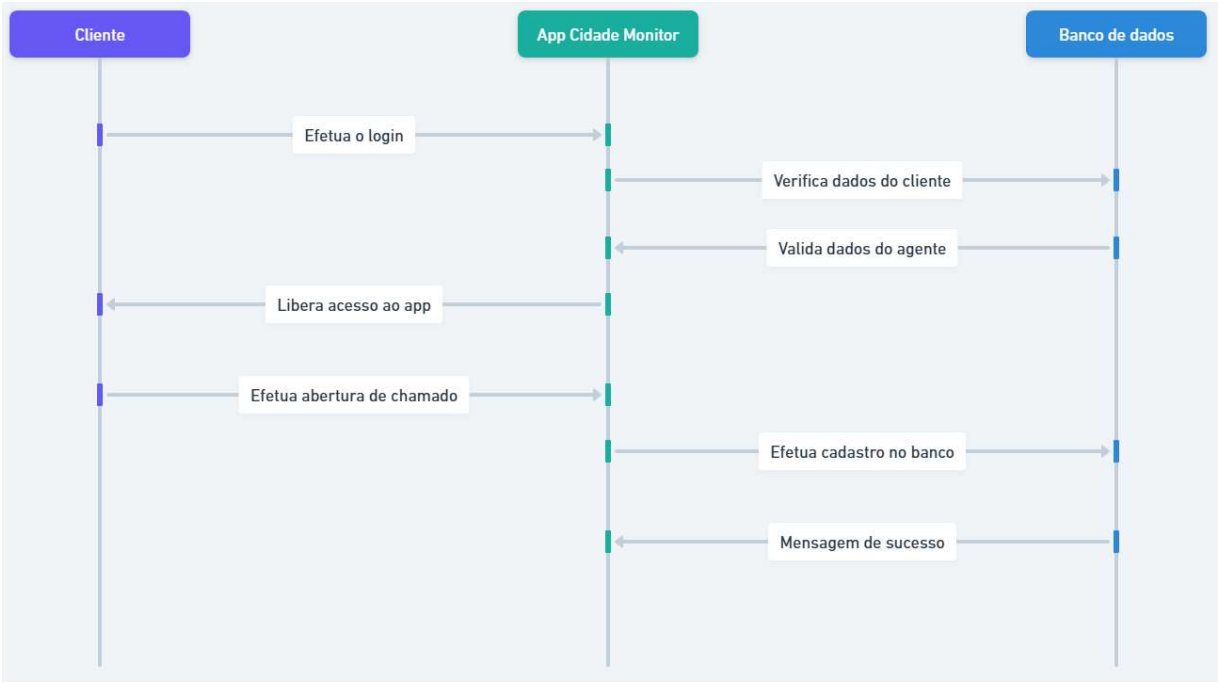
"O diagrama de sequência é um diagrama comportamental que se preocupa com a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos em um determinado processo" (Guedes, 2018, p. 41).

Como descrito acima, o diagrama de sequência é a representação temporal de uma sequência de passos necessários para a realização de uma tarefa dentro do *software*, tendo seu principal embasamento no diagrama de classes. Esse diagrama tem como objetivo identificar tanto o evento gerador do processo que é modelado, quanto o responsável por esse evento, e com isso determina como o processo deverá responder até o momento de sua conclusão.

11.1. CADASTRO DE MUNICÍPIO

A seguir é apresentado o diagrama de sequência (Figura 6) que representa o processo de cadastro de novo município.

Figura 6 - Cadastro na plataforma



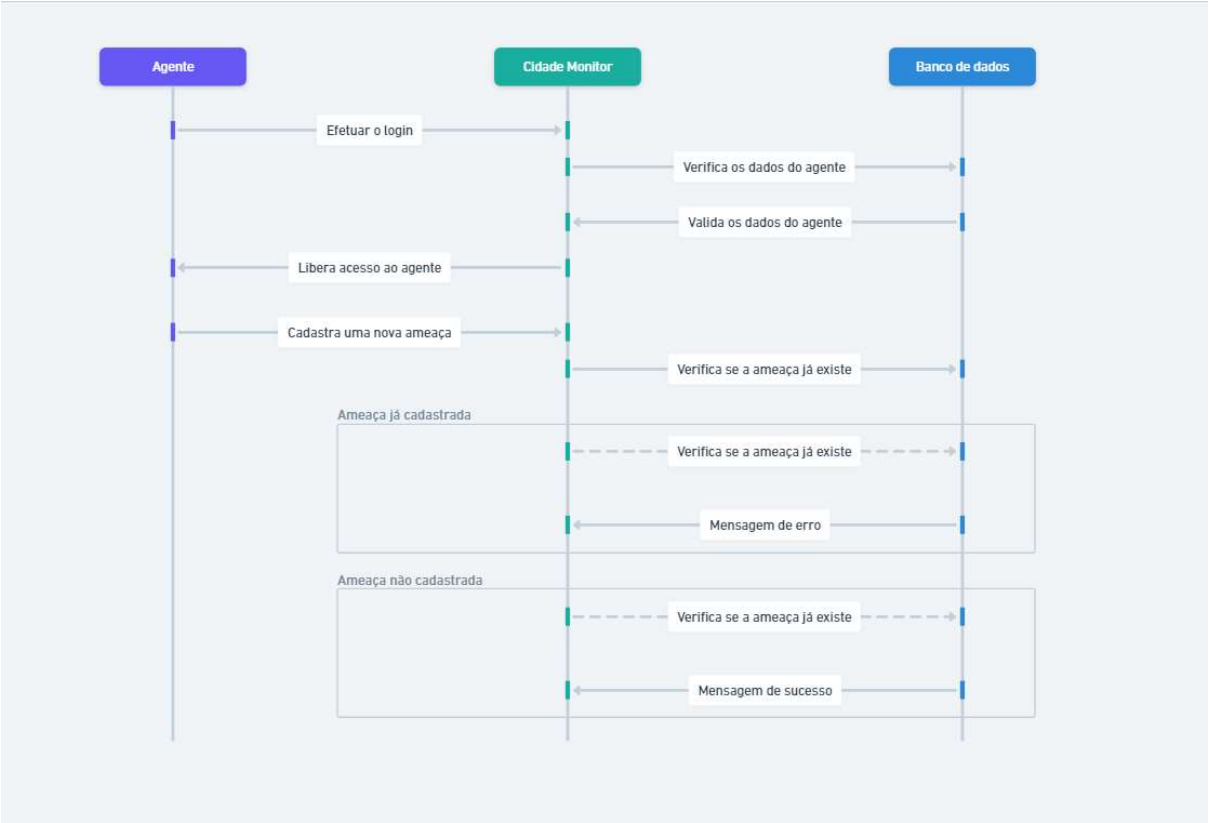
Fonte: O autor (2024)

Conforme exposto na Figura 6, quando o sistema recebe a solicitação de cadastro de um novo município, ele verifica a existência do mesmo no banco de dados, caso o município já esteja cadastrado, uma mensagem de erro é retornada ao usuário. Caso não exista, o novo mu- nicípio é cadastrado com êxito na plataforma.

11.2. CADASTRO DE AMEAÇAS

A seguir é apresentado o diagrama de sequência (Figura 7) que representa o processo de cadastro de uma nova ameaça.

Figura 7 - Cadastro de ameaça



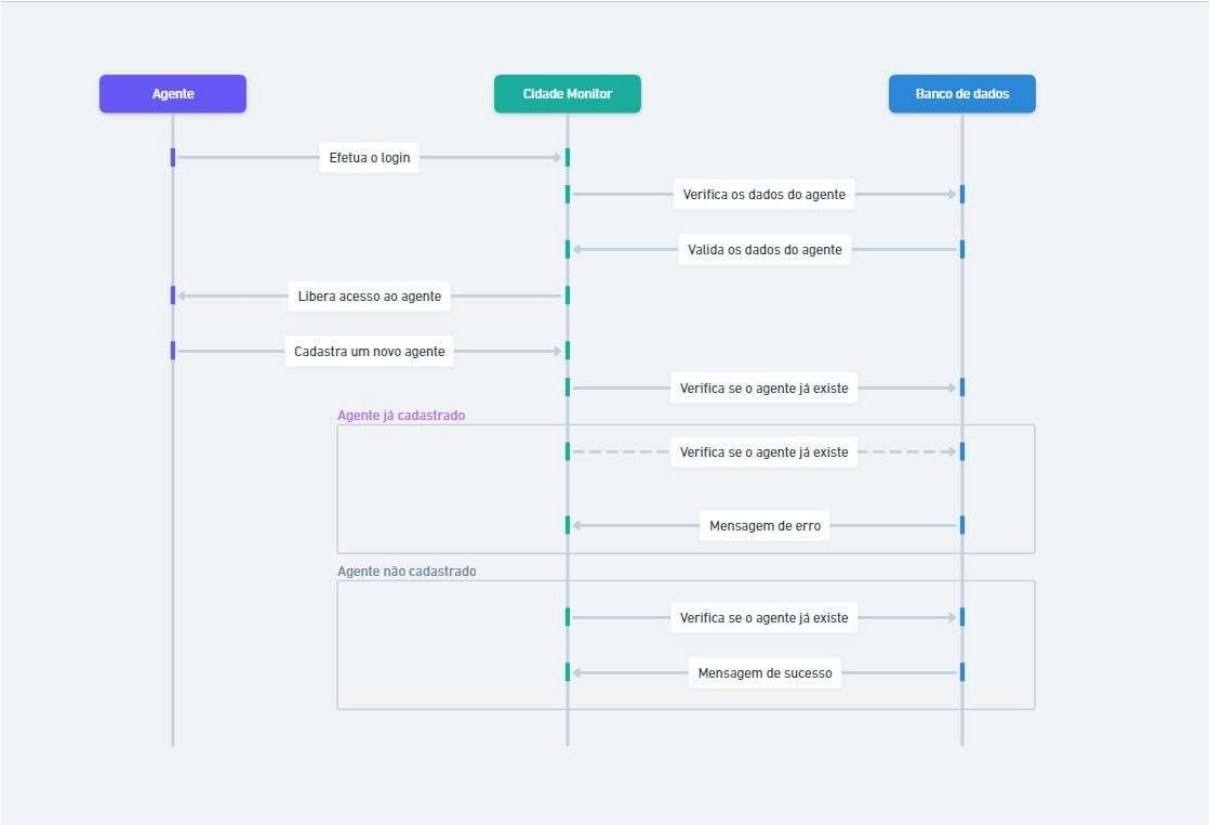
Fonte: O autor (2024)

Conforme verificado na Figura 7, ao tentar efetuar o cadastro de uma nova ameaça, uma verificação é realizada no banco de dados. Caso uma ameaça semelhante já tenha sido cadas- trada, o agente terá como resposta uma mensagem informando que aquela ameaça já se encontra cadastrada. Caso contrário, a ameaça será cadastrada com sucesso.

11.4. CADASTRO DE AGENTE

A seguir é apresentado o diagrama de sequência (Figura 8) que representa o processo de cadastro de um novo agente.

Figura 8 - Cadastro de agente



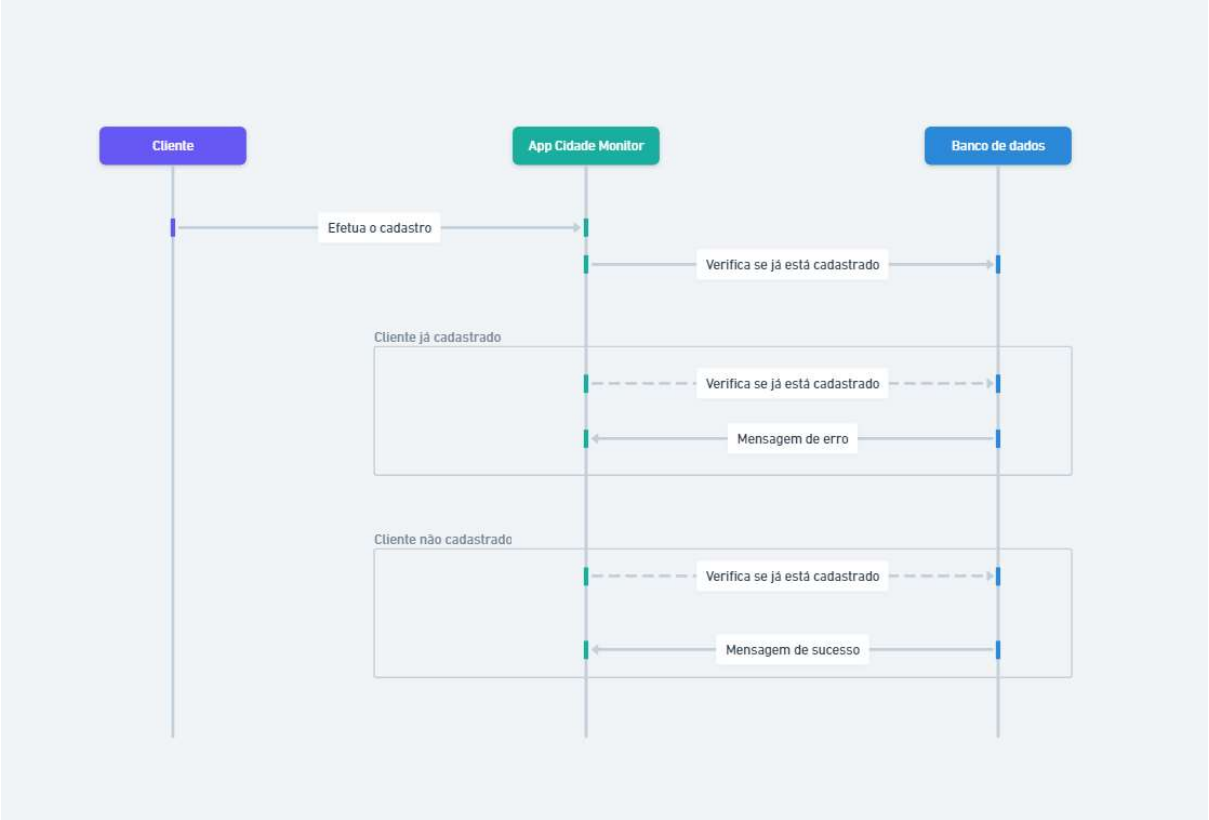
Fonte: O autor (2024)

Conforme é apresentado na Figura 8, ao tentar efetuar o cadastro de um novo agente, uma verificação é realizada no banco de dados, caso um agente de mesmo *e-mail* já exista, uma mensagem é retornada ao agente administrador, informando que o agente já está cadastrado na plataforma. Caso contrário, o cadastro é realizado e uma mensagem de sucesso é retornada ao agente.

11.5. CADASTRO DE CLIENTES

A seguir é apresentado o diagrama de sequência (Figura 9) que representa o processo de cadastro de um novo cliente.

Figura 9 - Cadastro de cliente



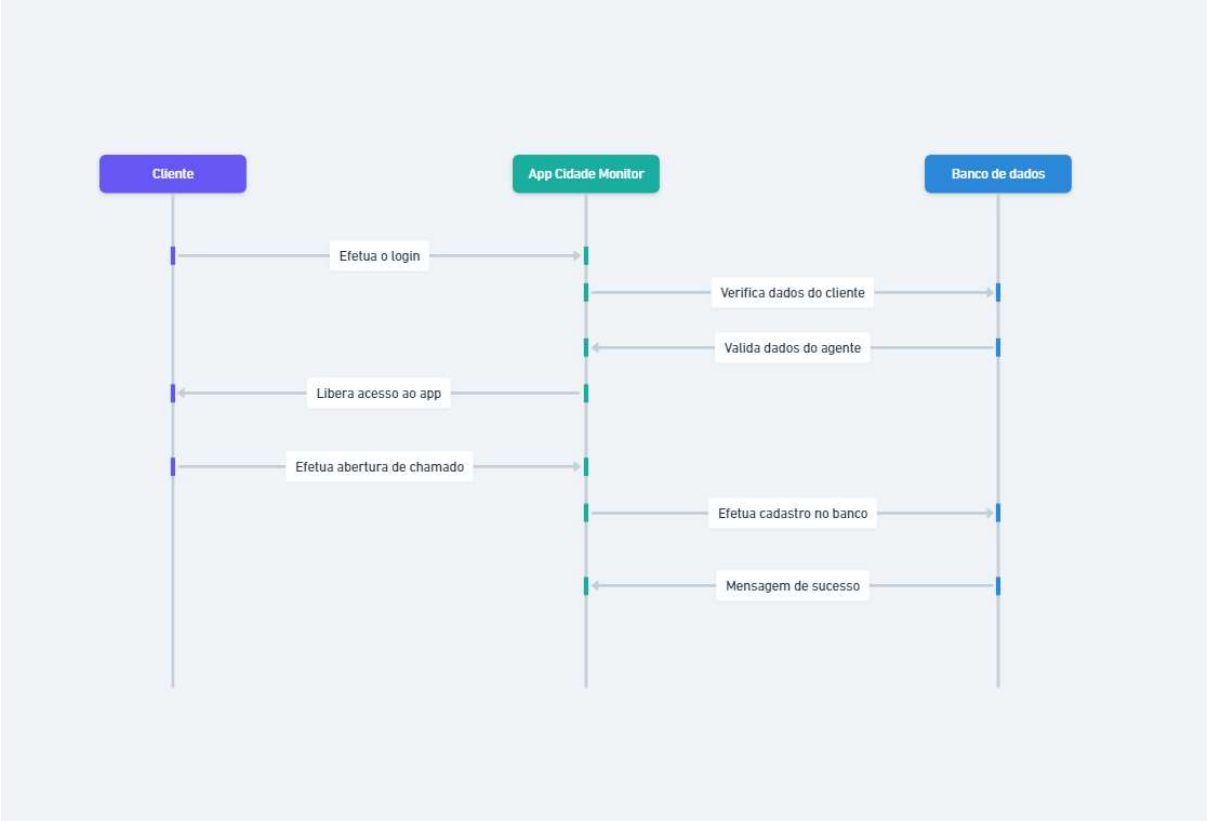
Fonte: O autor (2024)

Conforme é apresentado na Figura 9, ao cadastrar-se, é feita uma verificação do *e-mail* do cliente, caso esse *e-mail* conste no banco de dados, ele será redirecionado para a tela de recuperação de senha, onde poderá criar uma nova senha de acesso.

11.6. ABERTURA DE CHAMADOS

A seguir é apresentado o diagrama de sequência (Figura 10) que representa o processo de cadastro de um novo chamado.

Figura 10 - Abertura de chamado



Fonte: O autor (2024)

Conforme apresentado na Figura 10, caso ocorra algum erro durante o processo de cadastro de um chamado, é exibida ao cliente uma mensagem de erro. Caso contrário, é exibido uma mensagem indicando que o cadastro foi realizado com sucesso.

12. RELATÓRIO DE EXECUÇÃO DE TESTES

12.1. OBJETIVO GERAL DOS TESTES

Os testes de *software* têm como objetivo a longevidade do código, escalabilidade e a diminuição de erros durante o processo de desenvolvimento do sistema, a criação de código de testes é de suma importância durante a criação de um *software* aberto, para que eventuais mo- dificações não quebrem funcionalidades já consolidadas na regra de negócio do *software*. Sendo assim, os testes garantem que o sistema funcione conforme o esperado.

12.2. AMBIENTE DE TESTES

Para o desenvolvimento dos testes de *software*, foi criado um ambiente de testes desen- volvido e configurado de modo a atender os diversos cenários previstos nos requisitos inicial- mente coletados. Sendo assim as seguintes ferramentas foram escolhidas:

a) *framework* de testes:
    a. *Jest*
b) linguagem de programação
    a. *TypeScript*
c) ambiente de execução
    a. *Node.js 18.17.1.*
d) banco de dados
    a. *PostgresSql*
e) ferramentas adicionais
    a. Dados *mockados* para simulação de requisições

13. RESULTADOS DOS TESTES

Figura 11 - Resultado dos testes

Fonte: O autor (2024)

Como é possível observar na Figura 11, foi criado uma série de testes com o objetivo de assegurar a confiabilidade do *software*. Por questões práticas, serão expostos aqui somente os testes dos principais métodos utilizados pelo sistema, todos os demais testes não pontuados aqui, podem ser acessados no repositório do projeto e executados localmente.

13.1. TESTE DO MÉTODO *GETTHREAT*

A descrição e os resultados do teste *getThreat* são expostos a seguir:

a) descrição;

   a. testa se o método *getThreat*, ao receber o *id* do município, retorna os dados vinculados ao município específico.

b) resultado;

   a. sucesso. O método retorna corretamente um *array* contendo todas as ameaças relacionadas com o município em questão.

c) observações:

   a. nenhuma inconsistência observada

13.2. TESTE DO MÉTODO *GETTHREATBYID*

A descrição e os resultados do teste *getThreatById* são expostos a seguir:

a) descrição;

    a. testa se o método *getThreatById*, ao receber o *id* da ameaça, retorna os dados vinculados ao *id* especificado.

b) resultado;

    a. sucesso. O método retorna os dados vinculados ao id da ameaça.

c) observações;

    a. nenhuma inconsistência observada

13.3. TESTE DO MÉTODO *POSTTHREAT*

A descrição e os resultado do teste do método *postThreat* são expostos a seguir:

a) descrição;

    a. testa se o método *postThreat*, ao receber os dados corretamente, efetua o cadastro de uma nova ameaça

b) resultado;

    a. sucesso. O método realiza o cadastro de uma nova ameaça com sucesso.

c) observações;

    a. nenhuma inconsistência observada

13.4. TESTE DO MÉTODO *GETCALL*

A descrição e os resultados do teste do método *getCall* são expostos a seguir:

a) descrição;

    a. testa se o método *getCall*, ao receber o *id* do município, retorna os dados relacionados ao id especificado.

b) resultado;

    a. sucesso. O método *getCall* retorna um *array* contendo todos os chamados vinculados ao *id*.

c) observações;

    a. nenhuma inconsistência observada

13.5. TESTE DO MÉTODO *FINALIZECALL*

A descrição e os resultados do teste do método *finalizeCall* são expostos a seguir:

a) descrição;

    a. testa se o método *finalizeCall*, ao receber o *id* do chamado, finaliza-o corretamente.

b) resultado;

    a. sucesso. O método *finalizeCall* finaliza corretamente um chamado.

c) observações;

    a. nenhuma inconsistência observada

13.6. TESTE DO MÉTODO *GETAGENTS*

A descrição e os resultados do teste do método *getAgents* são expostos a seguir:

a) descrição;

    a. Testa se o método *getAgents*, ao receber o *id* do município, retorna os agentes vinculados aquele *id*.

b) resultado;

    a. sucesso. O método *getAgents* retorna um *array* contendo todos os agen- tes vinculados ao *id*.

c) observações;

    a. nenhuma inconsistência observada

13.7. TESTE DO MÉTDO *GETAGENTSBYID*

A descrição e os resultados do teste do método *getAgentsById* são expostos a seguir:

a) descrição;

    a. Testa se o método getAgentsById, ao receber o *id* do agente, retorna os dados vinculados ao *id*.

b) resultado;

    a. sucesso. O método *getAgentsById*, ao receber o *id* do agente, retorna os dados vinculados ao *id.*

c) observações;

    a. nenhuma inconsistência observada

13.8. TESTE DO MÉTODO *POSTAGENTS*

A descrição e os resultados do teste do método *postAgents* são expostos a seguir:

a) descrição;

   a. Testa se o método *postAgents*, ao receber os dados corretamente, deverá cadastrar um novo agente.

b) resultado;

   a. sucesso. O método postAgents cadastra um novo agente com sucesso.

c) observações;

   a. nenhuma inconsistência observada

13.9. TESTE DO MÉTODO *POSTMUNICIPALITY*

A descrição e os resultados do teste do método *postMunicipality* são expostos a seguir:

a) descrição;

   a. Testa se o método *postMunicipality*, ao receber os dados corretamente, deverá cadastrar um novo município.

b) resultado

   a. sucesso. O método *postMunicipality* cadastrou um município com sucesso.

c) observações;

   a. nenhuma inconsistência observada

13.10. TESTE DO MÉTODO *SIGNIN*

A descrição e os resultados do teste do método *signIn* são expostos a seguir:

a) descrição;

   a. Testa se o método *signIn*, ao receber os dados corretamente, retorna as informações relacionadas ao usuário.

b) resultado;

   a. sucesso. O método *signIn* retornou um *array* com os dados relacionados ao agente

c) observações;

a. nenhuma inconsistência observada

13.11. TESTE DO MÉTODO *POSTCLIENT*

A descrição e os resultados do teste do método *postClient* estão expostos a seguir:

a) descrição;

   a. Testa se o método postClient, ao receber os dados corretamente, cadas- tra com sucesso um novo cliente

b) resultado;

   a. sucesso. O método *postClient* cadastrou um novo cliente com sucesso.

c) observações;

   a. nenhuma inconsistência observada

13.12. TESTE DO MÉTODO *SIGNINCLIENT*

A descrição e os resultados do teste do método *signInClient* estão expostos a seguir:

a) descrição;

   a. Testa se o método *signInClient*, ao receber os dados corretamente, re- torna os dados relacionados ao cliente

b) resultado;

   a. sucesso. O método retornou os dados relacionados ao cliente com su- cesso.

c) observações;

   a. nenhuma inconsistência observada

14. CONCLUSÃO

O projeto Cidade Monitor demostra ser uma ferramenta inovadora e essencial para o gerenciamento de ameaças urbanas e ambientais, proporcionando à população e às autoridades uma plataforma eficaz de colaboração. A implementação do projeto a partir do uso de tecnolo- gias modernas e robustas, como *React, Nes.js* e *Postgres,* garantiu uma estrutura escalável, se- gura e eficiente, capaz de atender às necessidades tanto dos cidadãos quanto da administração pública.

Embora não tenha sido realizada uma avaliação detalhada de usabilidade com os usuá- rios finais, o desenvolvimento do sistema foi orientado por *feedbacks* de especialistas, assegu- rando sua relevância prática e técnica. A longo prazo, espera-se que o Cidade Monitor contribua significativamente para a melhoria da qualidade de vida nas cidades onde for implementado, fortalecendo a criação de políticas públicas focadas na prevenção e resolução de problemas urbanos e ambientais.

15. REFERÊNCIAS

BACCO, M. et al. Environmental Monitoring for Smart Cities. **IEEE Sensors Journal**, v. 17, n. 23, p. 7767–7774, 1 dez. 2017.

BROVINI, E. M. et al. Three-bestseller pesticides in Brazil: Freshwater concentrations and potential environmental risks. **Science of The Total Environment**, v. 771, p. 144754, jun. 2021.

CAMPOS, Rodrigo José de; BRANCO Priscila. Ocupação desordenada dos espaços urbanos e suas consequências socioambientais. **Revista Thêma et Scientia** – Vol. 11, no 2E, jul/dez 2021 – Edição Especial Arquitetura e Urbanismo.

CASTRO, C. M. de; PEIXOTO, M. N. de O; RIO, G. A. P. do. Riscos Ambientais e Geografia: Conceituações, Abordagens e Escalas. **Anuário do Instituto de Geociências** – UFRJ, v. 28. n 2, p. 11-30. 2005.

COLLÉTER, M. et al. Global overview of the applications of the Ecopath with Ecosim modeling approach using the EcoBase models repository. **Ecological Modelling**, v. 302, p. 42–53, abr. 2015.

COSTA B; PIRES, P. **Evaluating a Representational State Transfer (REST) Architecture.** Department of Computer Science – Federal University of Rio de Janeiro (UFRJ) 68.530 – Rio de Janeiro – RJ – Brazil. 2015.

DATE, C. J. **Introdução a sistemas de bancos de dados**. 8. ed. São Paulo: Pearson Addison Wesley, 2004.

DUARTE, A. C.; FARIAS FILHO, M. S. **Impactos da adequação de infraestrutura e problemas ambientais da Cidade Universitária Dom Delgado**, Universidade Federal do Maranhão, em São Luiz – MA. Geografia em Atos (Online), Presidente Prudente, v.2, n. 17, p. 80-99, 2020. DOI: 10.35416/geoatos. V2i17.6553. Disponível em: https://revista.fct.unesp.br/index.php/geografiaematos/article/view/6553. Acesso em: 1 ago. 2024.

FEDRA, K. Urban environmental management: monitoring, GIS, and modeling. **Computers, Environment and Urban Systems**, v. 23, n. 6, p. 443–457, nov. 1999.

FROES ASMUS, C. I. R. et al. A Systematic Review of Children's Environmental Health in Brazil. **Annals of Global Health**, v. 82, n. 1, p. 132, 17 jun. 2016.

GUEDES, Gilleanes T.A. **UML 2**: Uma Abordagem Prática**,** 3ª Edição, Editora Novatec, 2018. ISBN 9788575226445.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA (IBGE). **API de Dados Abertos**. Disponível em: https://servicodados.ibge.gov.br/api/docs. Acesso em: 26 jan. 2025.

JACOBSON**,** Ivar**. Object-oriented software engineering.** Boston: Addison-Wesley, 1992.

MAPBOX. **Mapbox API Documentation.** Disponível em: https://docs.mapbox.com/api/. Acesso em: 26 jan. 2025.

PENTEADO, J. O. et al. Health risk assessment in urban parks soils contaminated by metals, Rio Grande city (Brazil) case study. **Ecotoxicology and Environmental Safety**, v. 208, p. 111737, jan. 2021.

PIRES, C. Ângelo. O fenômeno da favelização no interior de Minas Gerais: O desafio das políticas públicas no direito à moradia. **Perspectivas em Políticas Públicas**, v. 9, n. 1, p. 146-167, 2016. Disponível em: https://revista.uemg.br/index.php/revistappp/article/view/1009.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software**. [s.l: s.n.].

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de banco de dados**. 6. ed. São Paulo: Pearson Addison Wesley, 2012.

SOMMERVILLE, Ian. **Engenharia de software,** 10ª Edição. Editora Pearson, 2019. ISBN 9788579361081.

ULLO, S. L.; SINHA, G. R. Advances in Smart Environment Monitoring Systems Using IoT and Sensors. **Sensors**, v. 20, n. 11, p. 3113, 1 jan. 2020.

VALENTE, Marco Túlio. **Engenharia de Software Moderna,** 1ª Edição. Editora Independente, 2022. ISBN 9786500019506.

VIEIRA, Pedro William de Oliveira; OLIVEIRA, Leonardo Silva de; SOARES, Rodrigo Ca- deira; CUNHA JÚNIOR, Humberto Beltrão de; AZEVÊDO, Evaldo de Lira; SANTOS, Suê- nia. AQUAMEAÇA: Aplicativo Android para identificação e monitoramento de ameaças
ecossistemas aquáticos. *In*: XXIII SIMPÓSIO BRASILEIRO DE RECURSOS HÍDRICOS, 2019. **Anais** [...]. Porto Alegre: ABRH, 2019. Disponível em: https://anais.abrhidro.org.br/works/6282. ISSN 2318-0358.

APÊNDICE A – MODELO DE DADOS DA TABELA *TB_MUNICIPALITY*

O modelo de dados da tabela *tb_municipality* está descrito no Quadro 1, onde será possível visualizar a nomenclatura das colunas, o tipo e sua obrigatoriedade.

Quadro 1 - Modelo de dados da tabela *tb_municipality*

Propriedade	Tipo	Obrigatoriedade
Id	*UUID*	Obrigatório
Cnpj	*String*	Obrigatório
municipality_name	*String*	Obrigatório
state_name	*String*	Obrigatório
created_at	*DateTime*	Obrigatório
updated_at	*DateTime*	Obrigatório
deleted_at	*DateTime*	Não obrigatório

Fonte: O autor (2024).

A explicação dos campos está descrita logo abaixo:

a) id:

   a. identificador único gerado automaticamente pelo *TypeORM* utilizado.

b) cnpj:

   a. identificador único utilizado para controle de municípios cadastrados.

c) municipality_name

   a. nome do município que será cadastrado.

d) state_name

a. nome do estado a qual o município pertence.

e) created_at

a. data de cadastro do município gerado automaticamente pelo *TypeORM* utilizado.

f) updated_at

a. data de atualização dos dados do município gerado automaticamente pelo *TypeORM* utilizado.

g) deleted_at

a. data de exclusão do município gerado automaticamente pelo *TypeORM* utilizado.

APÊNDICE B – MODELO DE DADOS DA TABELA *TB_AGENT*

O modelo de dados da tabela *tb_agent* está descrito no Quadro 2, onde será possível visualizar a nomenclatura das colunas, o tipo e sua obrigatoriedade.

Quadro 2 - Modelo de dados da tabela *tb_agent*

Propriedade	Tipo	Obrigatoriedade
Id	*UUID*	Obrigatório
municipality_id	*String*	Obrigatório
role_platform	*String*	Obrigatório
role_user	*String*	Obrigatório
name	*String*	Obrigatório
birth_date	*DateTime*	Obrigatório
password	*String*	Obrigatório
email	*String*	Obrigatório
created_at	*DateTime*	Obrigatório
updated_at	*DateTime*	Obrigatório
deleted_at	*DateTime*	Não obrigatório

Fonte: O autor (2024).

A explicação dos campos está descrita abaixo:

a) id

a. identificador único gerado automaticamente pelo *TypeORM* utilizado.

b) municipality_id

a. chave estrangeira referente ao município a qual o agente faz parte

c) role_platform

   a. identificador de qual plataforma será utilizada (*web/mobile*).

d) role_user

   a. identificador do nível de usuário da plataforma (Administrador/Não administra- dor).

e) name

   a. nome do agente que será cadastrado.

f) birth_date

   a. data de nascimento do agente.

g) password

   a. senha criada pelo agente durante o cadastro.

h) e-mail

   a. *e-mail* fornecido pelo agente durante o cadastro.

i) created_at

   a. data de cadastro do agente gerado automaticamente pelo *TypeORM.*

j) updated_at

   a. data de atualização dos dados do agente gerado automaticamente pelo *Type-ORM.*

k) deleted_at

   a. data de exclusão do agente gerado automaticamente pelo *TypeORM* utilizado.

APÊNDICE C – MODELO DE DADOS DA TABELA *TB_CLIENT*

O modelo de dados da tabela *tb_client* está descrito no Quadro 3, onde será possível visualizar a nomenclatura das colunas, o tipo de dados e sua obrigatoriedade.

Quadro 3 - Modelo de dados da tabela *tb_client*

Propriedade	Tipo	Obrigatoriedade
id	*UUID*	Obrigatório
municipality_id	*String*	Obrigatório
name	*String*	Obrigatório
gender	*String*	Obrigatório
email	*String*	Obrigatório
role_platform	*String*	Obrigatório
password	*String*	Obrigatório
created_at	*DateTime*	Obrigatório
updated_at	*DateTime*	Obrigatório
deleted_at	*DateTime*	Não obrigatório

Fonte: O autor (2024).

A explicação dos campos está descrita abaixo:

a) id

   a. identificador único gerado automaticamente pelo *TypeORM* utilizado.

b) municipality_id

   a. chave estrangeira referente ao município a qual o cliente pertence.

c) name

   a. nome do cliente fornecido durante o cadastro.

d) gender

   a. gênero do cliente informado durante o cadastro.

e) e-mail

   a. *e-mail* fornecido pelo cliente durante o cadastro.

f) role_platform

   a. tipo de plataforma (*web/mobile*)

g) password

   a. senha criada pelo usuário durante o cadastro.

h) created_at

   a. data de cadastro do cliente gerado automaticamente pelo *TypeORM* utilizado.

i) updated_at

   a. data de atualização dos dados do cliente gerado automaticamente pelo *TypeORM* utilizado.

j) deleted_at

   a. data de exclusão do cliente gerado automaticamente pelo *TypeORM* utilizado.

APÊNDICE D – MODELO DE DADOS DA TABELA *TB_THREAT*

O modelo de dados da tabela *tb_threat* está descrito no Quadro 4, onde será possível visualizar a nomenclatura das colunas, o tipo de dados e sua obrigatoriedade.

Quadro 4 - Modelo de dados da tabela *tb_threat.*

Propriedade	Tipo	Obrigatorieade
id	*UUID*	Obrigatório
municipality_id	*String*	Obrigatório
category_id	*String*	Obrigatório
image_id	*String*	Obrigatório
description	*String*	Obrigatório
threat_level	*Number*	Obrigatório
title	*String*	Obrigatório
created_at	*DateTime*	Obrigatório
updated_at	*DateTime*	Obrigatório
deleted_at	*DateTime*	Não obrigatório

Fonte: O autor (2024).

A explicação dos campos está descrita abaixo

a) id

   a. identificador único gerado automaticamente pelo *TypeORM* utilizado.

b) category_id

   a. chave estrangeira referente a categoria a qual a ameaça irá fazer parte

c) image_id

   a. chave estrangeira referente a imagem que será pertencente a ameaça cadastrada

d) description

   a. descrição da ameaça que será cadastrada.

e) threat_level

   a. nível da ameaça (será composta dos números de 1 a 3, sendo 1 como mais baixa e 3 como a mais alta).

f) title

   a. Título da ameaça que será cadastrada.

g) created_at

   a. data de cadastro da ameaça, gerado automaticamente pelo *TypeORM* utilizado.

h) updated_at

   a. data de atualização dos dados da ameaça, gerado automaticamente pelo *Type-ORM* utilizado.

i) deleted_at

   a. data de exclusão da ameaça, gerado automaticamente pelo *TypeORM* utilizado.

APÊNDICE E – MODELO DE DADOS DA TABELA *TB_CALL*

O modelo de dados da tabela *tb_call* está descrito no Quadro 5, onde será possível visualizar a nomenclatura das colunas, o tipo de dados e sua obrigatoriedade.

Quadro 5 - Modelo de dados da tabela *tb_call*

Propriedade	Tipo	Obrigatoriedade
id	*UUID*	Obrigatório
municipality_id	*String*	Obrigatório
threat_id	*String*	Obrigatório
latitude	*String*	Obrigatório
longitude	*String*	Obrigatório
created_at	*DateTime*	Obrigatório
updated_at	*DateTime*	Obrigatório
deleted_at	*DateTime*	Não Obrigatório

Fonte: O autor (2024)

A explicação dos campos está descrita abaixo:

a) id
   a. identificador único gerado automaticamente pelo *TypeORM* utilizado.

b) municipality_id
   a. chave estrangeira referente ao município a qual o cliente pertence.

c) threat_id
   a. chave estrangeira referente a ameaça que será cadastrada.

d) latitude
   a. dados de latitude do momento em que a ameaça é cadastrada.

e) longitude
   a. dados de longitude do momento em que o chamado é cadastrado.

f) created_at

a. data do cadastro da chamada, gerado automaticamente pelo *TypeORM* utilizado.

g) updated_at

a. data de atualização do chamado, gerado automaticamente pelo *TypeORM* utilizado.

h) deleted_at

a. data de exclusão do chamado, gerado automaticamente pelo *TypeORM* utilizado.

APÊNDICE F – MODELO DE DADOS DA TABELA *TB_IMAGE*

O modelo de dados da tabela *tb_image* está descrito no Quadro 6, onde será possível visualizar a nomenclatura das colunas, o tipo de dados e sua obrigatoriedade.

Quadro 6 - Modelo de dados da tabela *tb_image*

Propriedade	Tipo	Obrigatoriedade
Id	*UUID*	Obrigatório
municipality_id	*String*	Obrigatório
file_name	*String*	Obrigatório
content_length	*String*	Obrigatório
content_type	*String*	Obrigatório
url	*String*	Obrigatório
created_at	*DateTime*	Obrigatório
updated_at	*DateTime*	Obrigatório
deleted_at	*DateTime*	Não Obrigatório

Fonte: O autor (2024)

A explicação dos campos está descrita abaixo

a) id

a. identificador único gerado automaticamente pelo *TypeORM* utilizado.

b) file_name

a. nome do arquivo que será cadastrado.

c) content_lenght

a. tamanho do arquivo que foi cadastrado

d) content_type

a. tipo do arquivo que foi cadastrado

e) url

a. *url* do arquivo que foi cadastrado

f) created_at

a. data de cadastro da ameaça, gerado automaticamente pelo *TypeORM* utilizado

g) updated_at

a. data de atualização do arquivo, gerado automaticamente pelo *TypeORM* utilizado.

h) deleted_at

a. data de exclusão do arquivo, gerado automaticamente pelo *TypeORM*.

APÊNDICE G – PÁGINA DE LOGIN NA PLATAFORMA WEB

Segue a Figura 12 da página de login da plataforma *web*, a qual é visualizada assim que o *link* é acessado.

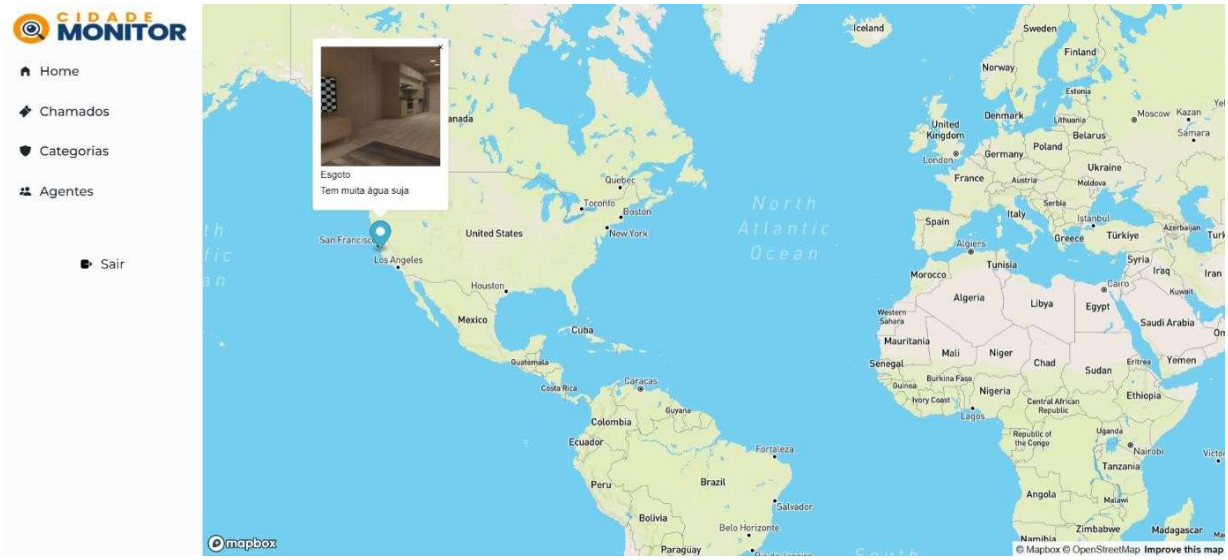
Figura 12 - Página de *login* na plataforma *web*

Fonte: O autor (2024)

APÊNDICE H – PÁGINA INICIAL DA PLATAFORMA

Segue a Figura 13 da página inicial da plataforma *web*, a qual é visualizada assim que o *login* é realizado.

Figura 13 - Tela inicial do Cidade Monitor

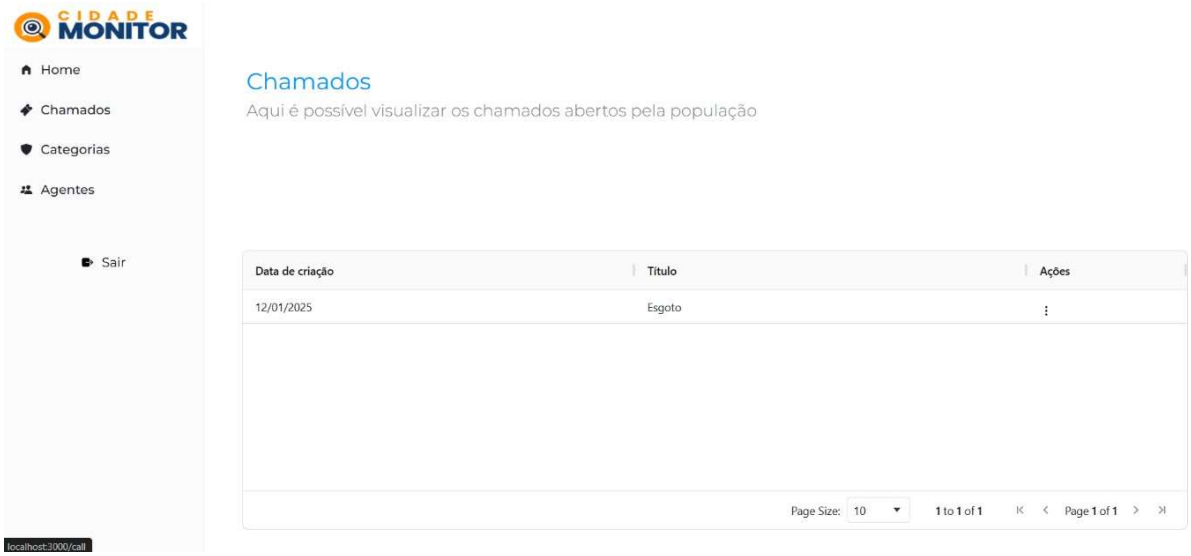


Fonte: O autor (2024)

APÊNDICE I – PÁGINA DE VISUALIZAÇÃO DE CHAMADOS ABERTOS

Segue a Figura 14 da página de visualização de chamados da plataforma *web*, a qual é visualizada acessando a opção de chamados no menu lateral.

Figura 14 - Página de visualização de chamados abertos



Fonte: O autor (2024)

APÊNDICE J – PÁGINA DE CADASTRO DE CATEGORIAS

Segue a Figura 15 da página de cadastro de categorias da plataforma *web*, a qual é visu- alizada acessando a opção de categorias no menu lateral.

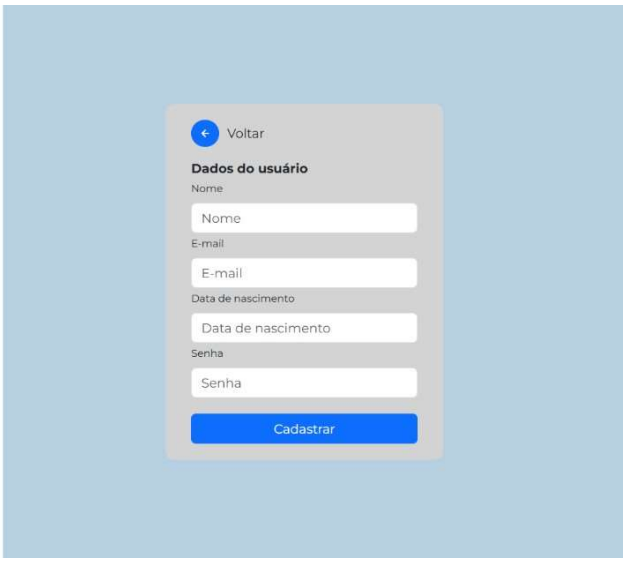
Figura 15 - Página de cadastro de categorias

Fonte: O autor (2024)

APÊNDICE K – PÁGINA DE CADASTRO DE AGENTES

Segue a Figura 16 da página de cadastro de agentes da plataforma *web*, a qual é visualizada acessando a opção de agentes no menu lateral.

Figura 16 - Página de cadastro de agentes



Fonte: O autor (2024)

APÊNDICE L – PÁGINA DE CADASTRO DE MUNICÍPIO

Segue a Figura 17 da página de cadastro de município da plataforma *web*, a qual é visu- alizada acessando a opção cadastre-se na página de *login*.

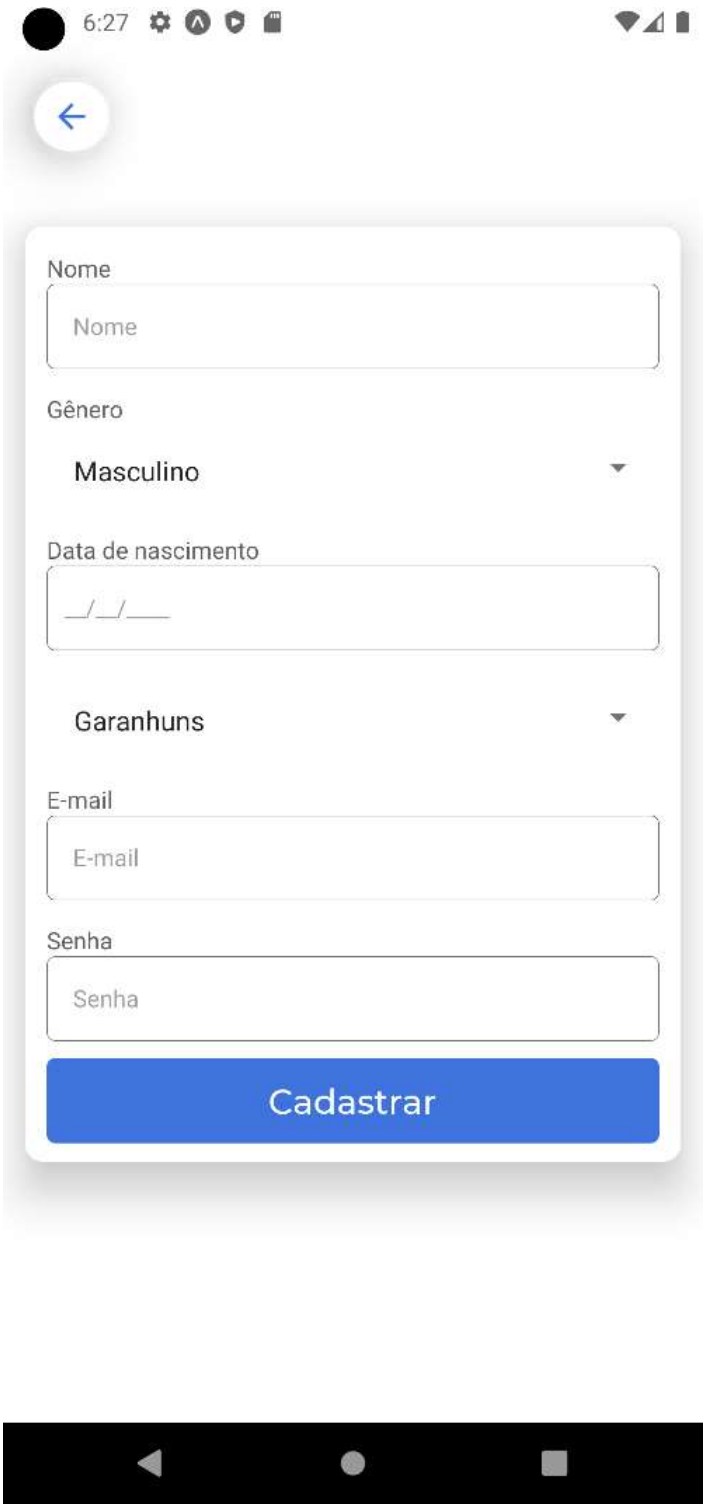
Figura 17 - Página de cadastro de município

Fonte: O autor (2024)

APÊNDICE M – PÁGINA DE CADASTRO DE CLIENTE

Segue a Figura 18 da página de cadastro de cliente do aplicativo *mobile*, o qual é visualizado acessando a opção cadastre-se na página de *login*.

Figura 18 - Página de cadastro de cliente



Fonte: O autor (2024)

APÊNDICE N – PÁGINA INICIAL DO APLICATIVO

Segue a Figura 19 da página inicial do aplicativo, a qual é visualizada após o *login* no aplicativo.

Figura 19 - Página inicial do aplicativo

Fonte: O autor (2024)

APÊNDICE O – PÁGINA DE DESCRIÇÃO DA AMEAÇA

Segue a Figura 20 da descrição de ameaças, a qual é visualizada após selecionar uma ameaça.

Figura 20 - Página de descrição de ameaça

Fonte: O autor (2024)

APÊNDICE P – PÁGINA DE ABERTURA DE CHAMADO

Segue a Figura 21 da página de abertura de chamado do aplicativo, a qual é visualizada após prosseguir a página de descrição.

Figura 21 - Página de abertura de chamado

Fonte: O autor (2024)

APÊNDICE Q – REPOSITÓRIO *FRONT-END WEB*

https://github.com/Cloves-Jose/trabalho-conclusao-frontend-web

APÊNDICE R – REPOSITÓRIO *FRONT-END MOBILE*

https://github.com/Cloves-Jose/trabalho-conclusao-frontend-mobile

APÊNDICE S – REPOSITÓRIO *BACK-END*

https://github.com/Cloves-Jose/trabalho-conclusao