

Universidade Federal Fluminense – UFF
Campus Universitário de Rio das Ostras
Instituto de Ciência e Tecnologia

Desenvolvimento de um aplicativo móvel para o Programa de Registro de Ocorrências em Defesa Civil (PRODEC)

Marcello de Paula Câmara

Rio das Ostras

2019

Ficha catalográfica automática - SDC/BRO
Gerada com informações fornecidas pelo autor

C172d Câmara, Marcello de Paula
Desenvolvimento de um aplicativo móvel para o Programa de Registro de Ocorrências em Defesa Civil (PRODEC) / Marcello de Paula Câmara ; Carlos Bazilio Martins, orientador.
Niterói, 2019.
113 p. : il.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação)-Universidade Federal Fluminense, Instituto de Ciência e Tecnologia, Rio das Ostras, 2019.

1. PRODEC. 2. Aplicativo. 3. Android. 4. Dispositivos móveis. 5. Produção intelectual. I. Bazilio Martins, Carlos, orientador. II. Universidade Federal Fluminense. Instituto de Ciência e Tecnologia. III. Título.

CDD -

Universidade Federal Fluminense – UFF
Campus Universitário de Rio das Ostras
Instituto de Ciência e Tecnologia

**Desenvolvimento de um aplicativo móvel para o
Programa de Registro de Ocorrências em Defesa Civil
(PRODEC)**

Trabalho de conclusão de curso submetido ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Fluminense, Campus de Rio das Ostras, para a obtenção do Grau de Bacharel em Ciência da Computação. Área de Concentração: Desenvolvimento de aplicativo móvel.

Marcello de Paula Câmara

Orientador: Prof. Dr. Carlos Bazilio Martins

Rio das Ostras

2019


Marcello de Paula Câmara

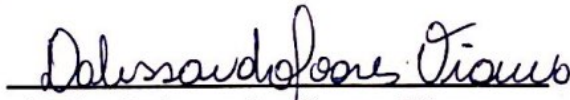
**Desenvolvimento de um aplicativo móvel para o
Programa de Registro de Ocorrências em Defesa Civil
(PRODEC)**

Trabalho de conclusão de curso submetido ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Fluminense, Campus de Rio das Ostras, para a obtenção do Grau de Bacharel em Ciência da Computação. Área de Concentração: Desenvolvimento de aplicativo móvel.

Trabalho aprovado. Rio das Ostras, 21 de julho de 2019.


D.Sc. Carlos Bazilio Martins
Orientador


D.Sc. Adriana Pereira de Medeiros
UFF


D.Sc. Dalessandro Soares Vianna
UFF

Rio das Ostras
2019

Agradecimentos

À Deus pela oportunidade de estudar na Universidade Federal Fluminense, me dando saúde e forças para superar os momentos difíceis. Dei o melhor de mim desde o início da faculdade, pois sei quantos gostariam de estar no meu lugar.

À minha mãe, Lilian Chaves, que sempre cuidou de mim, mesmo que distante, buscando melhores condições financeiras. Obrigado por ter sido uma mãe exemplar, inigualável, por mais que eu não o diga todos os dias. Ao meu pai, Juratan Câmara, por estar sempre bem-humorado apesar das dificuldades. Espero que cada dia cansativo de trabalho seja recompensado com minha formação. Agradeço por mostrar-se um bom pai, dando o seu melhor para mim. Obrigado meus pais, amo vocês!

Às minhas avós, Helly Terezinha e Maria Câmara (*in memoriam*), por terem me amado e mimado sempre que possível.

Ao meu padrasto, Edilson Mello, por cuidar de mim como se fosse um filho. Obrigado por fazer parte da minha família!

À minha tia Patrícia e meu tio Alexandre, que me ensinaram a olhar para longe, pensando no meu futuro profissional.

Aos meus tios, Jean e Francisco, com os quais convivi durante anos de minha vida e possuo muito carinho.

À minha professora particular, dos meus tempos de ensino fundamental e médio, “tia” Eliane Borges, que por mais levado que eu tenha sido na escola onde passei anos de minha vida, Externato Santa Ignez, não desistiu de mim e acreditou no meu potencial, torcendo para que eu concluísse meus estudos com louvor.

A todos os meus amigos que torcem por mim, João Marcos, Bernardo Scofano, Lucas Santiago, Bruno Levorato, Allan Almeida, e meu primo, Gabriel Rocha. Em especial os mais próximos, João Parise e sua família, pela forte amizade e incentivo; José Morista, pelos anos de companhia e estudos na faculdade; Larissa Paula, por todo o amor e lealdade nestes últimos anos; Matheus Lima, por ter sido meu primeiro amigo de verdade na faculdade e, a Ian Fabris e sua família, que por mais distante que estejam, torceram e torcem pelo meu sucesso.

E também a todos que não acreditaram no meu potencial. Estes sim, me deram ainda mais forças para provar que seria possível.

Aos citados acima, espero recompensá-los ainda mais no futuro, realizando ainda mais sonhos e atingindo mais sucessos.

*“O caminho de Deus é perfeito,
e a palavra do Senhor refinada;
e é o escudo de todos os que nEle confiam.
Por que, quem é Deus, senão o Senhor?
E quem é rochedo, senão o nosso Deus?
Deus é a minha fortaleza e a minha força,
e Ele perfeitamente desembaraça o meu caminho.”
(Bíblia Sagrada, 2 Samuel 22:31-33)*

Resumo

Com a evolução das tecnologias da informação e comunicação, os Órgãos Públicos vêm a necessidade de ampliar a gestão interna dos setores públicos, mudando a relação entre os servidores e a sociedade, visando uma melhor eficiência no atendimento e na prestação de serviços aos cidadãos. O PRODEC é um sistema que fornece aos Órgãos de Defesa Civil Municipais uma ferramenta de ambiente virtual para registro de suas ocorrências diárias. O objetivo deste trabalho é desenvolver um aplicativo móvel para que o sistema do PRODEC seja utilizado em qualquer lugar, desde que haja internet. A vantagem da ferramenta proposta é melhorar a coleta e o envio dos dados para o sistema, sendo feita em tempo real, no momento do incidente. A aplicação foi implementada para dispositivos móveis, com sistema operacional Android, utilizando a linguagem de programação Java e o ambiente de desenvolvimento integrado Android Studio.

Palavras-chave: PRODEC. Aplicativo. Android. Dispositivos móveis. Java.

Lista de ilustrações

Figura 1 – Aplicativo 1746 - Tela inicial	22
Figura 2 – Aplicativo 1746 - Histórico de solicitações	22
Figura 3 – Aplicativo 1746 - Informar dados	23
Figura 4 – Aplicativo 1746 - Cadastro	23
Figura 5 – Aplicativo 1746 - Nova solicitação	24
Figura 6 – Aplicativo Delegacia Online PCERJ - Tela inicial	25
Figura 7 – Aplicativo Delegacia Online PCERJ - Denúncia de bairro	26
Figura 8 – Aplicativo Delegacia Online PCERJ - Comunicar ocorrência	27
Figura 9 – Aplicativo Delegacia Online PCERJ - Localidade do delito	27
Figura 10 – Aplicativo Delegacia Online PCERJ - Tipo de ocorrência	28
Figura 11 – Aplicativo Delegacia Online PCERJ - Informar dados pessoais	29
Figura 12 – Diagrama de Casos de Uso	42
Figura 13 – Bibliotecas : SmartTabLayout - Exemplo de uso	45
Figura 14 – Bibliotecas : JustifiedTextView - Exemplo de uso	46
Figura 15 – Bibliotecas : Toasty - Exemplo de uso	47
Figura 16 – Bibliotecas : StateProgressBar - Exemplo de uso	48
Figura 17 – Modelo do Banco de Dados	50
Figura 18 – Aplicativo PRODEC - SplashScreen	53
Figura 19 – Aplicativo PRODEC - Tela inicial	55
Figura 20 – Aplicativo PRODEC - Cadastro de Adesão	56
Figura 21 – Aplicativo PRODEC - Login	57
Figura 22 – Aplicativo PRODEC - Fale Conosco	61
Figura 23 – Aplicativo PRODEC - Menu de navegação	63
Figura 24 – Aplicativo PRODEC - Tela Inicial do Sistema	64
Figura 25 – Aplicativo PRODEC - Iniciar Boletim de Ocorrência	65
Figura 26 – Aplicativo PRODEC - Minha Conta	66
Figura 27 – Aplicativo PRODEC - Alterar Senha	67
Figura 28 – Aplicativo PRODEC - Documentos	68
Figura 29 – Aplicativo PRODEC - Identificação do Boletim de Ocorrência	70
Figura 30 – Aplicativo PRODEC - Dados da Vistoria	71
Figura 31 – Aplicativo PRODEC - Danos Humanos	72
Figura 32 – Aplicativo PRODEC - Relatório Fotográfico	73
Figura 33 – Plataforma do PRODEC - Home	99
Figura 34 – Plataforma do PRODEC - Cadastro de Adesão	100
Figura 35 – Plataforma do PRODEC - Login	101
Figura 36 – Plataforma do PRODEC - Fale conosco	102

Figura 37 – Plataforma do PRODEC - Tela inicial do sistema	103
Figura 38 – Plataforma do PRODEC - Iniciar Boletim de Ocorrência	104
Figura 39 – Plataforma do PRODEC - Minha conta	105
Figura 40 – Plataforma do PRODEC - Alterar senha	105
Figura 41 – Plataforma do PRODEC - Documentos	106
Figura 42 – Plataforma do PRODEC - Identificação do Boletim de Ocorrência . . .	106
Figura 43 – Plataforma do PRODEC - Dados da Vistoria	107
Figura 44 – Plataforma do PRODEC - Danos Humanos	107
Figura 45 – Plataforma do PRODEC - Relatório Fotográfico	108

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i>
Cel BM	Coronel Bombeiro Militar
CNPJ	Cadastro Nacional da Pessoa Jurídica
CPF	Cadastro de Pessoa Física
DGDEC	Departamento Geral de Defesa Civil
IDE	<i>Integrated Development Environment</i>
JDK	<i>Java Development Kit</i>
JVM	<i>Java Virtual Machine</i>
PCERJ	Polícia Civil do Estado do Rio de Janeiro
PRODEC	Programa de Registro de Ocorrências em Defesa Civil
SQL	<i>Structured Query Language</i>

Sumário

1	INTRODUÇÃO	17
1.1	Introdução Geral	17
1.2	Motivação	17
1.3	Objetivos	18
1.4	Estrutura do trabalho	19
2	TRABALHOS RELACIONADOS	21
2.1	1746 Rio	21
2.2	Delegacia Online PCERJ	24
3	MODELAGEM	31
3.1	Requisitos	31
3.1.1	Requisitos funcionais	31
3.1.2	Requisitos não funcionais	32
3.2	Casos de uso	33
3.2.1	Descrição dos Casos de Uso	34
3.2.2	Diagrama de casos de uso	42
4	TECNOLOGIAS UTILIZADAS	43
4.1	Ambiente de desenvolvimento	43
4.2	Controle de versão	44
4.3	Bibliotecas	45
4.3.1	SmartTabLayout	45
4.3.2	JustifiedTextView	46
4.3.3	Volley	46
4.3.4	Toasty	47
4.3.5	GmailBackground	47
4.3.6	StateProgressBar	48
5	DESENVOLVIMENTO	49
5.1	Banco de dados	49
5.1.1	Modelo do banco de dados	49
5.1.2	API	51
5.2	Implementação	53
5.2.1	Tela de Boas-vindas	53
5.2.2	Tela Inicial	54

5.2.3	Cadastro de Adesão	56
5.2.4	Login	57
5.2.5	Fale Conosco	60
5.2.6	Menu de navegação	61
5.2.7	Tela Inicial do Sistema	63
5.2.8	Iniciar Boletim de Ocorrência	64
5.2.9	Minha Conta	65
5.2.10	Alterar Senha	66
5.2.11	Documentos	67
5.2.12	Identificação do Boletim de Ocorrência	69
5.2.13	Dados da Vistoria	70
5.2.14	Danos Humanos	71
5.2.15	Relatório Fotográfico	72
6	CONCLUSÃO	75
6.1	Trabalhos futuros	75
	REFERÊNCIAS	77
	APÊNDICES	79
	APÊNDICE A – API DESENVOLVIDA	81
A.1	Conexão	81
A.2	Login	81
A.3	Minha conta	83
A.4	Trocar senha	84
A.5	Criar Boletim de Ocorrência	85
A.6	Consultar Boletins de Ocorrência	86
A.7	Consultar Resumo do Boletim de Ocorrência	87
A.8	Alterar Boletim de Ocorrência	88
A.9	Criar Dados da Vistoria	90
A.10	Consultar Dados da Vistoria	91
A.11	Criar Danos Humanos	92
A.12	Consultar Danos Humanos	94
A.13	Criar Relatório Fotográfico	95

ANEXOS	97
ANEXO A – PLATAFORMA DO PRODEC	99
ANEXO B – DOCUMENTO DE BOLETIM DE OCORRÊNCIA . .	109
ANEXO C – DOCUMENTO DE DADOS DA VISTORIA E DA- NOS HUMANOS	111

1 Introdução

1.1 Introdução Geral

O Agente de Defesa Civil é responsável por realizar ações preventivas, dentre elas a vistoria, com o objetivo de minimizar os desastres naturais, evitando riscos que comprometam a segurança de pessoas e bens. A vistoria, de modo geral, é uma forma de avaliar o nível de periculosidade de um cenário, como por exemplo imóveis, vias públicas, árvores, entre outros.

O Agente, responsável pela vistoria, descreve o Boletim de Ocorrência, e um engenheiro faz o laudo e o assina. O requerente recebe uma cópia do documento para saber que providências foram solicitadas pelo engenheiro. Sendo assim, o Agente pode interditar parcialmente, totalmente ou administrativamente o cenário (local vistoriado), até que os possíveis problemas sejam resolvidos, em caso de ameaças ou riscos.

A plataforma do PRODEC, é uma iniciativa do Major Jorge Carvalho, de otimizar e padronizar os serviços de vistoria, prestados pelos agentes de Defesa Civil do Estado do Rio de Janeiro. O sistema é destinado ao registro de ocorrências em Defesa Civil, com todas as funções necessárias para a gestão operacional do órgão de Defesa Civil.

O sistema, ainda em fase de testes, fornece condições aos Municípios de gerarem uma série de ocorrências na área de Defesa Civil, tornando possível um planejamento aprimorado de ações por parte dos Órgãos de Defesa Civil em nível estadual. Para sua utilização, é necessário pertencer à algum órgão de Defesa Civil Estadual do Rio de Janeiro e solicitar o cadastro. Após a confirmação, os analistas entrarão em contato com o Gestor do Órgão para transmitir informações adicionais para utilização do sistema.

Algumas imagens das telas do sistema do PRODEC foram adicionadas ao anexo [Plataforma do PRODEC](#) para visualização da plataforma desenvolvida pelo Major Jorge Carvalho. A aplicação proposta neste trabalho deverá seguir o modelo do sistema desenvolvido para que o usuário sinta-se familiarizado com o ambiente.

1.2 Motivação

A principal motivação surgiu de uma disciplina que o autor deste trabalho cursou durante o tempo para a graduação, Introdução à Computação Móvel, ministrada pelo professor Alessandro Copetti, na qual os alunos inscritos deveriam desenvolver aplicativos móveis, utilizando o *Android Studio*¹, com temas voltados para a saúde. No desenvolvimento

¹ Ambiente de Desenvolvimento Integrado (IDE) oficial para o desenvolvimento de aplicativos Android

do tema escolhido, Monitoramento da Frequência Cardíaca, o grupo composto pelos alunos José Morista, Lais Aguiar e o autor deste trabalho, realizou um estudo superficial (pouco profundo) da plataforma *Android Studio*, visto que a duração da disciplina contou com apenas 4 (quatro) meses do primeiro período letivo do ano de 2016.

Uma outra motivação deu-se durante a participação de apresentações de TCC (Trabalho de Conclusão de Curso) em sua Universidade, durante os quais, ao seu ver, muitos alunos desenvolviam ferramentas somente com o objetivo de serem aprovados, sem a intenção de utilizá-las em alguma situação e/ou cenário reais. Então, afim de buscar maiores experiências no desenvolvimento de aplicativos móveis, surgiu a ideia de realizar um estudo bem mais profundo de aplicativos Android.

Como sugestão de seu pai, Juratan Câmara, funcionário da Defesa Civil Municipal da cidade do Rio de Janeiro, o autor resolveu procurar o órgão público para estudar a viabilidade da ferramenta desenvolvida neste trabalho. Ao visitar o DGDEC (Departamento Geral de Defesa Civil), localizado na rua Elpídio Boamorte, no bairro carioca Praça da Bandeira, foi recebido de braços abertos pelo Diretor-Geral, Cel BM (Coronel Bombeiro Militar) Marcello Silva da Costa, que se demonstrou interessado com a proposta do trabalho voluntário oferecido pelo autor deste trabalho.

1.3 Objetivos

O PRODEC (Programa de Registro de Ocorrências em Defesa Civil) é um sistema inteiramente voltado para *desktop* (meio mais tradicional de navegação pela *web*), sendo uma tecnologia que a maior parte dos usuários já tem experiência em sua utilização. Com a diversidade de funcionalidades que a plataforma apresenta, o sistema não foi desenvolvido com responsividade. Sendo assim, ele não mantém o visual semelhante e agradável de se utilizar, em quaisquer tamanhos e qualidades de telas que sejam abertos. Desta maneira, ao utilizar o sistema do PRODEC em dispositivos móveis, a interface apresenta alguns erros como sobreposição de elementos textuais, imagens e campos de digitação.

Da maneira como o sistema PRODEC está desenvolvido, o Agente de Defesa Civil, responsável por ir até o local de registro de ocorrência, necessita preencher um formulário impresso em papel A4 para realizar o registro de ocorrência, vide [Documento de Boletim de Ocorrência](#) e [Documento de Dados da Vistoria e Danos Humanos](#). Após a coleta dos dados da ocorrência, o agente ainda precisará acessar o sistema do PRODEC e enviar esse formulário, preenchido em papel A4 impresso, através de um computador, por meio de uma foto ou escaneamento. Desta maneira, o trabalho torna-se redundante e há a possibilidade do agente deixar esta tarefa para ser feita depois e/ou até mesmo esquecer de fazê-la.

O objetivo deste trabalho é filtrar as funcionalidades que são de maior importância no dia a dia do Agente de Defesa Civil, que será o usuário principal, utilizando o aplicativo

para realizar uma tarefa de maneira ágil, prática e agradável, visando a eficiência na coleta dos dados e envio imediato do registro de ocorrência, através do aplicativo desenvolvido.

1.4 Estrutura do trabalho

O trabalho se encontra dividido em seis capítulos.

No primeiro capítulo, apresenta-se a Introdução, que é composta por: Introdução geral, que conta a história dos aplicativos móveis com a necessidade de seu uso na sociedade; Motivação para realização deste trabalho; Objetivos; e Estrutura do trabalho.

O segundo capítulo conta com a discussão de 2 (dois) trabalhos relacionados, 1746 Rio e Delegacia Online PCERJ (Polícia Civil do Estado do Rio de Janeiro), abordando suas respectivas relevâncias para o atual trabalho.

No terceiro capítulo, se encontra a modelagem do sistema desenvolvido: requisitos, funcionais e não funcionais, onde se pode observar como serão todas as funcionalidades do aplicativo deste trabalho, descrições dos casos de uso e o diagrama de casos de uso.

O quarto capítulo aborda as tecnologias utilizadas, mostrando as ferramentas necessárias para configuração do ambiente de desenvolvimento e organização do código.

No quinto capítulo, é apresentado o desenvolvimento da ferramenta, abordando as bibliotecas utilizadas, os aspectos de implementação e as interfaces gráficas desenvolvidas.

No sexto e último capítulo, é apresentada uma conclusão contendo a avaliação do trabalho desenvolvido e sugestões de trabalhos futuros.

2 Trabalhos relacionados

Nesta seção, serão apresentados dois trabalhos de aplicativos móveis que contêm as funcionalidades de solicitações ou registros de ocorrências. Outras funcionalidades dos trabalhos não serão abordadas por não serem relevantes para o atual trabalho. Serão citados neste projeto os aplicativos 1746 Rio e Delegacia Online PCERJ. Existem vários outros trabalhos, porém, os projetos apresentados a seguir possuem maior semelhança com o presente projeto.

2.1 1746 Rio

O telefone 1746 é um canal de atendimento da Prefeitura do Rio de Janeiro, que permite ao cidadão carioca conferir notícias sobre a cidade. Ao interagir com a Central 1746, o cidadão pode fazer solicitações de serviços municipais como retirada de entulho, reparos de iluminação pública e semáforos, reclamações de estacionamento irregular, limpeza de rua e controle de dengue e roedores.

De acordo com (NENO, 2011), a Central 1746 recebe solicitações de 13 secretarias e órgãos cadastrados: Defesa Civil, Disque-Dívida Ativa, Disque-IPTU, Disque-Luz, Disque-Ordem, Disque-Patrolha, Disque-Sinal, Disque-Transportes, Nota Carioca, TeleBuraco, TeleComlurb, TeleOrdem e TeleSaúde.

Segundo (PREFEITURA DO RIO, 2011), o modelo foi inspirado no “311” de Nova Iorque. A diferença é que no Rio de Janeiro, o cidadão recebe por e-mail ou SMS¹ (*Short Message Service*), um prazo para a solicitação ser atendida. Além do telefone, a Central 1746 também pode ser acessada via plataforma *web* (<<http://www.1746.rio/>>) ou pelo aplicativo móvel, que será analisado a seguir, com funcionamento de 24 horas por dia, em qualquer dia da semana, inclusive domingos e feriados.

Na [Figura 1](#), pode-se observar a estrutura da tela principal do aplicativo móvel. De início, o usuário já encontra um menu circular com opções para abrir um chamado para cada um dos serviços listados e acompanhar sua solicitação através de um número de protocolo fornecido, pelo próprio aplicativo (também gerado via plataforma *web* ou pelo telefone). Os serviços listados na tela inicial são: controle de Aedes Aegypti, estacionamento irregular, limpeza de rua, iluminação pública, ocupação irregular, sinais de trânsito com defeito, remoção de entulho, buracos, fiscalização de táxis, ônibus e vans e, caso o serviço não seja nenhuma das opções anteriores, outros. Também há um botão, “Ouvidoria”, para o usuário enviar uma mensagem com assuntos de sugestão, elogio, reclamação, denúncia

¹ Serviço de telefones celulares digitais que permite o envio de mensagens curtas.

ou crítica diretamente para a central de atendimento.

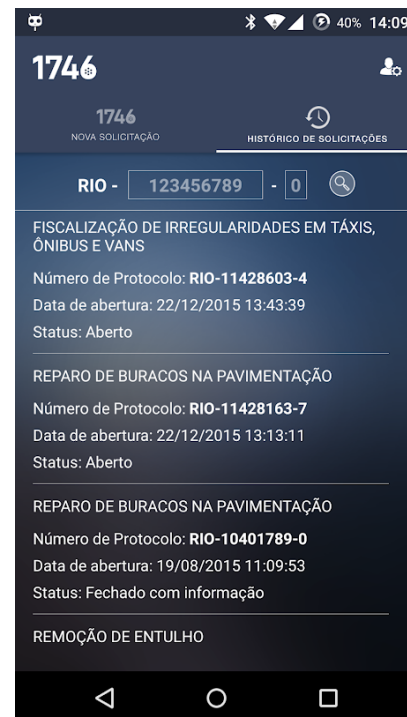
Caso o usuário já possua uma conta e esteja logado² no aplicativo, como na [Figura 2](#), pode-se observar o histórico de solicitações do usuário, que são os registros de ocorrências, contendo: o título da solicitação, o número do protocolo, a data e a hora de abertura da solicitação e o seu atual status. Também há uma funcionalidade de pesquisa para o usuário encontrar uma solicitação que ele enviou pelo número do protocolo gerado.

Figura 1 – Aplicativo 1746 - Tela inicial



Fonte: 1746 Rio

Figura 2 – Aplicativo 1746 - Histórico de solicitações



Fonte: 1746 Rio

Caso o usuário não possua uma conta e não esteja logado, o aplicativo dirá ao usuário que estão faltando algumas informações para continuar, como mostrado na [Figura 3](#). Ao selecionar a opção do botão “Informar dados”, o usuário é levado para uma nova tela, vide [Figura 4](#), para realizar seu cadastro no sistema. As informações que o usuário necessita preencher são: nome e sobrenome, telefone de contato e seu *e-mail*. Nesta tela, ainda há uma opção de entrar em contato com a Central 1746. Basta selecionar o botão “Ligar para a central 1746” e será aberto a tela de discagem do celular com o número 1746 para continuar a ligação.

Analisando a funcionalidade de abrir uma nova solicitação, como é o exemplo de uma reclamação da falta ou problema de iluminação pública na [Figura 5](#), o aplicativo leva o usuário à uma tela a qual é necessário o preenchimento de alguns dados para que a solicitação seja aberta:

² Acedeu a um sistema informático, mediante uma identificação e uma respectiva senha.

- Local da solicitação

Endereço físico onde a falta de iluminação pública se encontra, onde deve-se ativar o GPS do *smartphone* e utilizá-lo juntamente com o sinal de internet para obter a aproximação da localização do usuário.

- Complemento do endereço

Espaço onde o usuário pode fornecer alguma informação de referência para encontrar a localidade da solicitação.

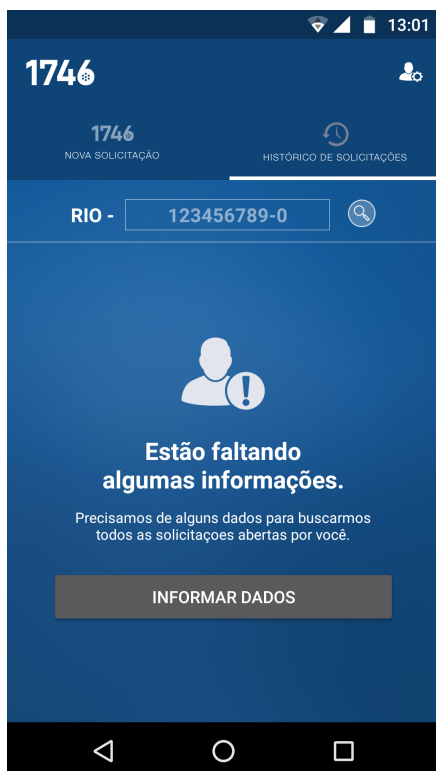
- Descrição do problema

Campo onde o usuário descreve o problema encontrado na iluminação pública, como por exemplo o número de postes, se são sequenciais ou intercalados e se são postes altos ou baixos.

- Incluir foto

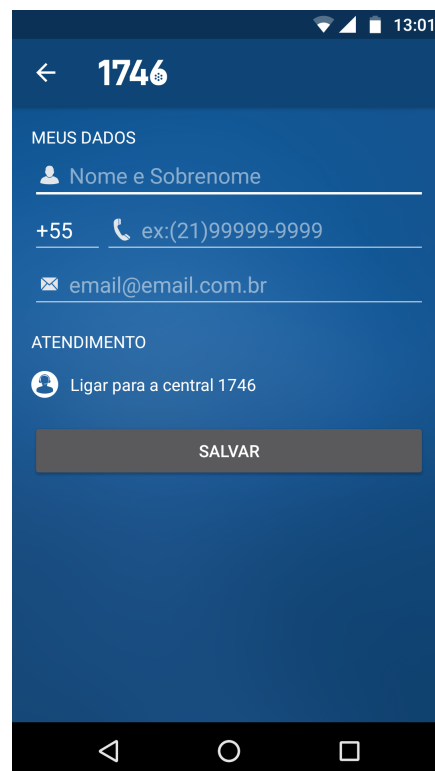
Opção que permite o usuário utilizar a câmera do *smartphone* para enviar uma foto de onde há a falta e/ou problema de iluminação pública, comprovando sua problemática.

Figura 3 – Aplicativo 1746 - Informar dados



Fonte: 1746 Rio

Figura 4 – Aplicativo 1746 - Cadastro



Fonte: 1746 Rio

Figura 5 – Aplicativo 1746 - Nova solicitação



Fonte: 1746 Rio

Segundo (MAGNI, 2013) e (JORNAL DO BRASIL, 2017), a Central 1746 foi reconhecida mundialmente em 2013 como uma das ações inovadoras que melhoraram a vida do cidadão carioca. O sistema foi um dos projetos que levaram a cidade do Rio de Janeiro a conquistar o prêmio *World Smart City*, em Barcelona. A cerimônia de premiação reuniu representantes de cidades, universidades, empresas privadas, gestores públicos e especialistas que debateram soluções para melhorar a vida nas metrópoles. A cidade do Rio de Janeiro foi eleita a mais inteligente do mundo, superando mais de 200 candidaturas de 35 países.

2.2 Delegacia Online PCERJ

O atendimento ao cidadão em delegacias cíveis nem sempre é ágil e prático. Para realizar um registro de ocorrência, o cidadão necessita informar uma série de detalhes do ocorrido e fornecer dados pessoais, que são repassados à um Agente de Polícia Civil (geralmente um Escrivão³) encarregado de transcrever o registro da ocorrência. Muitas pessoas eventualmente não comunicam um crime sofrido, seja por não encontrarem uma delegacia próxima, não quererem enfrentar filas, não se exporem indo à delegacia, entre

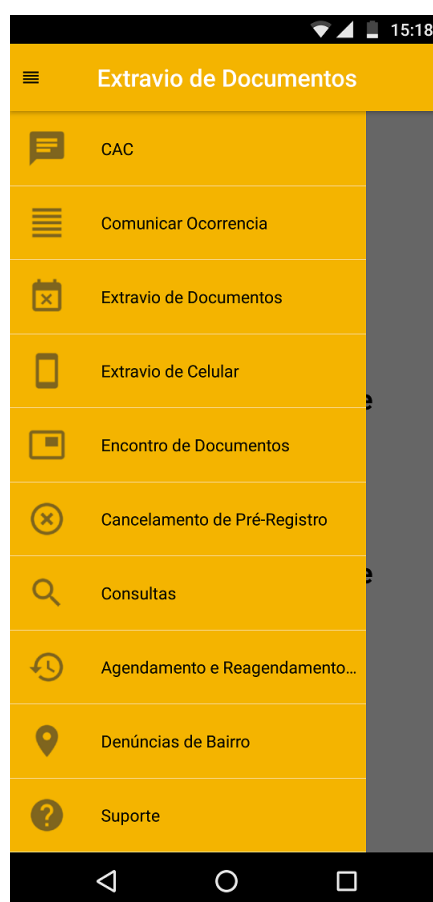
³ Funcionário responsável por conferir legitimidade às atribuições de polícia judiciária no esclarecimento de crimes e demais ocorrências.

outros motivos.

Segundo (EXTRA, 2016), a necessidade de ampliar a comunicação entre a Instituição e os cidadãos do estado do Rio de Janeiro, tornando o atendimento mais atrativo e simplificado, levou o DGTIT (Departamento Geral de Tecnologia da Informação e Telecomunicações) da Instituição a criar a Delegacia Online, que hoje conta com a plataforma *web* (<<https://dedic.pcivil.rj.gov.br/>>) e o aplicativo (Delegacia Online PCERJ) para maior prestação de serviços.

Na Figura 6, pode-se observar a tela inicial do aplicativo Delegacia Online PCERJ. O menu localizado na lateral à esquerda, comumente conhecido como *Navigation Drawer*⁴, dá ao usuário as seguintes opções: CAC (Central de Atendimento ao Cidadão), comunicar ocorrência, extravio de documentos, extravio de celular, encontro de documentos, cancelamento de pré-registro, consultas, agendamento e reagendamento, denúncias de bairro e suporte.

Figura 6 – Aplicativo Delegacia Online PCERJ - Tela inicial



Fonte: Delegacia Online PCERJ

Acessando a funcionalidade de denúncias de bairro, conforme Figura 7, o usuário

⁴ Menu de navegação, com ícone de três linhas horizontais, que surge da esquerda para a direita, cobrindo o conteúdo do aplicativo.

necessita informar as seguintes informações:

- Local

Menu de seleção (também conhecido como *Dropdown Menu*), onde o usuário seleciona o município referente a localidade de denúncia.

- Bairro

Outro menu de seleção, onde o usuário especifica o bairro da denúncia.

- Assunto

Último menu de seleção, onde o usuário seleciona o tipo do assunto da denúncia. Os tipos de assunto à serem selecionados são os seguintes: depósito de armas, elogios, gangues ou milícia, homicídio, homofobia, informação, localização de desaparecido, outros, perturbação da ordem social, reclamações, roubo furto, roubo seguido de morte (latrocínio), sugestão, tráfico de drogas e violência contra mulher.

- Denúncia

Campo onde o usuário descreve ou comenta o tipo de denúncia a ser realizada.

Figura 7 – Aplicativo Delegacia Online PCERJ - Denúncia de bairro

A imagem mostra a interface de usuário do aplicativo 'Delegacia Online' da Polícia Civil de Rio de Janeiro. O título da tela é 'Denúncia de Bairro'. O aplicativo apresenta o seguinte formulário:

- Local:** Dropdown menu com a opção selecionada 'RIO DAS OSTRAS'.
- Bairro:** Dropdown menu com a opção selecionada 'JARDIM MARILEA'.
- Assunto:** Dropdown menu com a opção selecionada 'Selecione...'.
- Denúncia:** Campo de texto para a descrição da denúncia, com o placeholder 'Denúncia ou comentario'.

Na base do formulário, há dois botões: 'Confirmar' e 'Limpar'. O aplicativo também exibe uma barra de status no topo com o horário 15:35 e ícones de conexão e bateria.

Fonte: Delegacia Online PCERJ

Ao fazer uma denúncia de bairro, o indivíduo colabora com a segurança do seu bairro, seja informando denúncias anônimas ou mesmo buscando orientações da CAC. De acordo com (CONSTANCIO, 2016), utilizando a Delegacia Online, seja pelo sistema *web* ou através do aplicativo móvel, a população pode tirar dúvidas, fazer elogios ou reclamações, contribuir com informações para investigações e auxiliar a Polícia Civil na localização de pessoas foragidas.

Já na funcionalidade de comunicação de ocorrência, demonstrado na Figura 8, o aplicativo levará o usuário à algumas telas de perguntas para coleta de dados. Nesta primeira tela, o usuário necessita selecionar um dos botões de opção (ou botões de rádio), também conhecidos como *Radio Buttons*. As duas opções disponíveis são as seguintes: Usuário residente da cidade a qual irá comunicar a ocorrência ou é turista nacional; Usuário é um turista estrangeiro.

Ao prosseguir para o segundo passo, uma nova tela surge para o usuário informar onde a comunicação de ocorrência está sendo realizada, como pode-se observar na Figura 9. Para prosseguir ao próximo passo, é necessário informar o município, o bairro e a rua onde ocorreu o delito, através dos 3 (três) menus de seleção disponíveis.

Figura 8 – Aplicativo Delegacia Online PCERJ - Comunicar ocorrência



Comunicar Ocorrência

Delegacia Online
Polícia Civil - Em defesa de quem precisar

Bem-vindo à Delegacia Online

Comunicação de Ocorrência

Você é residente ou turista?

Residente / Turista Nacional

Turista Estrangeiro

Prosseguir »

Atendimento

Fonte: Delegacia Online PCERJ

Figura 9 – Aplicativo Delegacia Online PCERJ - Localidade do delito



Comunicar Ocorrência

Delegacia Online
Polícia Civil - Em defesa de quem precisar

Bem-vindo à Delegacia Online

Comunicação de Ocorrência

Informe o município onde ocorreu o delito:
RIO DAS OSTRAS

Informe o bairro onde ocorreu o delito:
JARDIM MARILEA

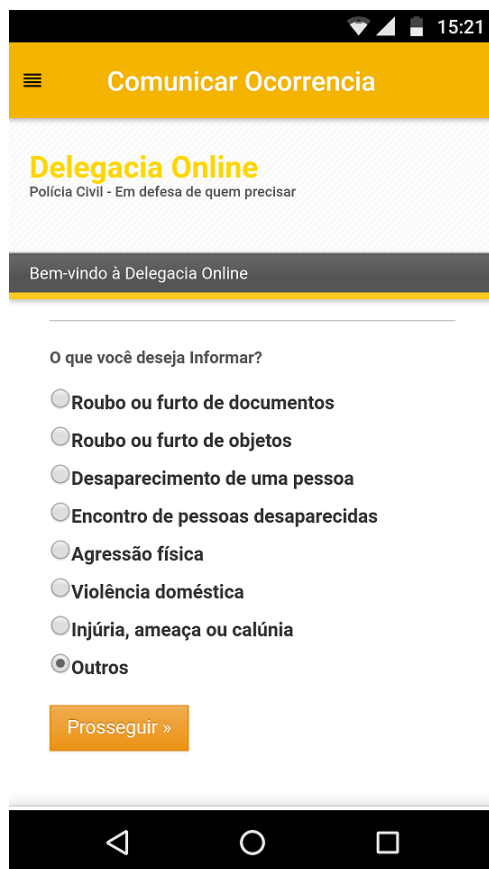
Informe a rua onde ocorreu o delito:
VALENÇA, RUA

Prosseguir »

Fonte: Delegacia Online PCERJ

Como terceiro passo, o aplicativo leva o usuário à uma nova tela, mostrado na Figura 10. Nesta tela, o usuário necessita selecionar o tipo de ocorrência que ele deseja comunicar, selecionando uma opção através do menu de opções. Os tipos de ocorrência disponíveis para seleção são os seguintes: Roubo ou furto de documentos, a qual necessita ser especificado se na subtração do documento foi usado arma de fogo (revólver, pistola) ou arma branca, se alguém foi ameaçado ou se houve agressão física; Roubo ou furto de objetos, o qual também necessita da mesma especificação do item anterior; Desaparecimento de uma pessoa; Encontro de pessoas desaparecidas; Agressão física; Violência doméstica; Injúria, ameaça ou calúnia; E outros.

Figura 10 – Aplicativo Delegacia Online PCERJ - Tipo de ocorrência



Fonte: Delegacia Online PCERJ

Ao prosseguir para o quarto passo, o usuário é levado à mais uma nova tela a qual ele precisará preencher, com seus dados pessoais, alguns campos, demonstrado na Figura 11, para criar a comunicação de ocorrência.

De acordo com (MOURA, 2016), quando o usuário faz uma denúncia pelo aplicativo, é gerado um registro de ocorrência que será validado e assinado pelo delegado (da mesma maneira como é feito presencialmente nas delegacias) que pode ser acessado e impresso

em casa. Além disso, durante o processo de denúncia, o usuário escolhe a delegacia mais próxima onde deseja ser atendido, caso haja a necessidade. Quanto melhor a qualidade e quantidade de informações descritas pelo usuário no aplicativo, a investigação se dará a partir daquele ponto descrito. Dependendo da complexidade do caso, ou se houver a necessidade da representação da pessoa, o delegado comunica à vítima (usuário), que poderá escolher pelo aplicativo a data e hora, para ir à delegacia.

Figura 11 – Aplicativo Delegacia Online PCERJ - Informar dados pessoais

Comunicar Ocorrência

Delegacia Online
Polícia Civil - Em defesa de quem precisar

Bem-vindo à Delegacia Online

INFORME SEUS DADOS PESSOAIS

Envolvimento:
Vítima

CPF:
* Obrigatório.

Nome:
* Obrigatório.

Pai:
* Obrigatório.

Mãe:
* Obrigatório.

Nascimento:
DD/MM/YYYY
* Obrigatório.

Cor da Pele:
Selecione...
* Obrigatório.

Sexo:
Selecione...
* Obrigatório.

Tipo Doc.:
Selecione...
* Obrigatório.

Orgão Exp.:
Selecione...
* Obrigatório.

N° DOC:
* Obrigatório.

Profissão:
Selecione...
* Obrigatório.

Naturalidade:
Selecione...
* Obrigatório.

CEP:
Buscar

Logradouro:
Selecione...
* Obrigatório.

Rua:
* Obrigatório.

Número:
* Obrigatório.

Complemento:

UF do Endereço:
Selecione...
* Obrigatório.

Município:
* Obrigatório.

Bairro:
* Obrigatório.

Telefone:
* Obrigatório.

Celular:
* Obrigatório.

Fax:

E-mail:
* Obrigatório.

Próximo Passo »

Fonte: Delegacia Online PCERJ

3 Modelagem

Nesta seção, será abordada a modelagem do aplicativo desenvolvido, através dos requisitos, funcionais e não funcionais, e o diagrama de casos de uso. Com a realização da modelagem, pode-se perceber a minimização dos riscos de fracasso de um projeto, garantindo a qualidade do mesmo.

3.1 Requisitos

De acordo com (SPÍNOLA, 2008), a engenharia de requisitos é uma atividade fundamental para o desenvolvimento de um *software*. Esta atividade consiste em analisar, identificar e documentar todas as funcionalidades e restrições do sistema em questão. Um requisito é uma característica, ou descrição, de algo que o sistema é capaz de realizar para atingir o seu objetivo.

3.1.1 Requisitos funcionais

Segundo (SOMMERVILLE, 2011), os requisitos funcionais são explicitamente as funcionalidades e serviços que o sistema deve fornecer. Eles descrevem as funções que o *software* deve executar, como o sistema deve reagir e como ele deve se comportar em determinadas situações.

Na Tabela 1, serão apresentados os requisitos funcionais do trabalho em questão. Os requisitos funcionais estão representados por um identificador (ID), à esquerda, a descrição do requisito funcional, ao centro, e sua determinada prioridade para o funcionamento da aplicação, à direita.

Tabela 1 – Requisitos funcionais

ID	Descrição	Prioridade
RF01	O sistema deve apresentar um resumo da plataforma e seus termos e condições de uso	Baixa
RF02	O sistema deve permitir que o usuário faça o cadastro na aplicação	Alta
RF03	O sistema deve permitir que o usuário envie uma mensagem de contato, preenchendo as seguintes informações: nome, <i>e-mail</i> , assunto e mensagem	Baixa

ID	Descrição	Prioridade
RF04	O sistema deve permitir que o usuário faça o <i>login</i> na aplicação utilizando seu <i>e-mail</i> e senha	Alta
RF05	O sistema deve permitir que o usuário possa continuar conectado na aplicação e também finalizar sua sessão	Baixa
RF06	O sistema deve apresentar um menu para navegar entre as funcionalidades do sistema: página inicial, iniciar Boletim de Ocorrência, minha conta e trocar senha.	Alta
RF07	O sistema deve permitir ao usuário alterar sua senha	Baixa
RF08	O sistema deve permitir ao usuário alterar seus dados cadastrais	Média
RF09	O sistema deve permitir ao usuário criar um Boletim de Ocorrência	Alta
RF10	O sistema deve lista os documentos recebidos e os documentos emitidos, separando-os em documentos resolvidos, não resolvidos com prazo não expirado e não resolvidos com prazo expirado	Alta
RF11	O sistema deve listar cada documento, informando os seguintes campos: entrada/emissão, documento, origem/destino, assunto, síntese, número, protocolo e prazo	Alta
RF12	O sistema deve exibir a identificação do Boletim de Ocorrência ao interagir com algum documento	Alta
RF13	O sistema deve permitir que o usuário crie os dados da vistoria	Média
RF14	O sistema deve permitir que o usuário edite um Boletim de Ocorrência existente	Média
RF15	O sistema deve permitir que o usuário envie o relatório fotográfico da ocorrência	Média
RF16	O sistema deve exibir ao usuário uma mensagem caso a conexão com a internet comprometa a tarefa em questão	Baixa

Fonte: Próprio autor

3.1.2 Requisitos não funcionais

Ainda de acordo com (SOMMERVILLE, 2011), os requisitos não funcionais definem propriedades e restrições aos serviços ou funcionalidades oferecidos pelo sistema. Os requisitos não funcionais especificam o comportamento do sistema.

Na Tabela 2, serão apresentados os requisitos não funcionais do trabalho em questão. Os requisitos não funcionais também estão representados por um identificador (ID), à esquerda, a descrição do requisito não funcional, ao centro, e sua determinada prioridade para o funcionamento da aplicação, à direita.

Tabela 2 – Requisitos não funcionais

ID	Descrição	Prioridade
RNF01	O sistema deve ser desenvolvido para dispositivos com sistema operacional <i>Android</i> utilizando a linguagem de programação Java	Alta
RNF02	O sistema deve ser desenvolvido para a versão mínima do <i>Android</i> 4.0.3 ¹	Média
RNF03	O sistema deverá se comunicar com o banco MySQL ²	Alta
RNF04	O sistema não deverá permitir o acesso à aplicação sem estar logado	Alta
RNF05	O sistema requer conexão com a internet para carregar os dados	Alta
RNF06	O sistema não deve permitir a criação de documentos com campos em branco	Baixa

Fonte: Próprio autor

3.2 Casos de uso

Segundo (LEITE, 2000), após definir as tarefas associadas a cada papel do usuário, o próximo passo é elaborar os casos de uso (*use cases*). Os casos de uso permitem definir funções da aplicação que o sistema deverá oferecer para o usuário. Vale ressaltar que os casos de uso podem ser utilizadas durante a análise e levantamento dos requisitos para descrever a funcionalidade do sistema.

Ainda de acordo com (LEITE, 2000), o caso de uso especifica o comportamento do sistema, não descrevendo como o comportamento será implementado. Um caso de uso representa o que o sistema faz e não como o sistema faz, proporcionando uma visão externa do sistema.

Cada caso de uso define um requisito funcional do sistema. O caso de uso descreve um conjunto de ações que o sistema desempenha. Cada sequência representa a interação de entidades externas e o sistema. Estas entidades são chamadas de atores, que podem ser usuários ou outros sistemas. No caso do usuário, o ator representa uma função do usuário.

Para realizar o Diagrama de Casos de Uso, utilizou-se do *software* Astah. O Astah é uma ferramenta de modelagem UML (*Unified Modeling Language*), em sua tradução Linguagem de Modelagem Unificada, que ajuda na tarefa de modelar e documentar os sistemas orientados a objetos.

¹ Versão da API (*Application Programming Interface* - Interface de Programação de Aplicativos) que abrange 100% dos dispositivos.

² Sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (*Structured Query Language*) como interface.

3.2.1 Descrição dos Casos de Uso

Nas tabelas a seguir, serão apresentadas as seguintes descrições de caso de uso:

- Tabela 3) Realizar cadastro (CSU01)
- Tabela 4) Enviar mensagem (CSU02)
- Tabela 5) Alterar dados cadastrais (CSU03)
- Tabela 6) Alterar senha (CSU04)
- Tabela 7) Criar Boletim de Ocorrência (CSU05)
- Tabela 8) Listar Boletim de Ocorrência (CSU06)
- Tabela 9) Listar Resumo do Boletim de Ocorrência (CSU07)
- Tabela 10) Criar Dados da Vistoria (CSU08)
- Tabela 11) Criar Danos Humanos (CSU09)
- Tabela 12) Criar Relatório Fotográfico (CSU10)
- Tabela 13) Listar Dados da Vistoria (CSU11)
- Tabela 14) Listar Danos Humanos (CSU12)
- Tabela 15) Editar Boletim de Ocorrência (CSU13)
- Tabela 16) Editar Dados da Vistoria (CSU14)
- Tabela 17) Editar Danos Humanos (CSU15)

Tabela 3 – Casos de uso - Realizar cadastro (CSU01)

Realizar cadastro (CSU01)
Ator primário: Usuário
Descrição: O usuário fornece os dados cadastrais e o sistema envia para o banco de dados
Pré-condições: O sistema deve ter acesso à internet
Fluxo principal: <ol style="list-style-type: none"> 1. O sistema solicita que o usuário informe os seus dados 2. O usuário informa seus dados 3. O usuário aceita o termo e condições de uso 4. O sistema libera o botão "Enviar" para que o usuário envie os dados 5. O usuário solicita o envio dos dados 6. O sistema valida os dados preenchidos 7. O sistema envia os dados para o banco de dados 8. O sistema imprime uma mensagem de sucesso e o caso de uso termina
Fluxo de exceção (Passo 3): O usuário não aceita o termo e condições de uso <ol style="list-style-type: none"> 1. O sistema não libera o botão para enviar os dados e o caso de uso continua a partir do passo 3
Fluxo de exceção (Passo 6): Campos em branco <ol style="list-style-type: none"> 1. O sistema deve informar ao usuário que os dados estão em branco e requisitar que este preencha-os corretamente e o caso de uso continua a partir do passo 2

Fonte: Próprio autor

Tabela 4 – Casos de uso - Enviar mensagem (CSU02)

Enviar mensagem (CSU02)
Ator primário: Usuário
Descrição: O usuário envia uma mensagem para o sistema e o sistema armazena a mensagem
Pré-condições: O sistema deve ter acesso à internet
Fluxo principal: 1. O sistema solicita que o usuário informe os dados (nome, e-mail, assunto e mensagem) 2. O usuário informa os dados 3. O usuário solicita o envio dos dados 4. O sistema valida os dados preenchidos 5. O sistema envia os dados para o banco de dados 6. O sistema imprime uma mensagem de sucesso e o caso de uso termina
Fluxo de exceção (Passo 4): Campos em branco 1. O sistema deve informar ao usuário que os dados estão branco e requisitar que este preencha-os corretamente e o caso de uso continua a partir do passo 2

Fonte: Próprio autor

Tabela 5 – Casos de uso - Alterar dados cadastrais (CSU03)

Alterar dados cadastrais (CSU03)
Ator primário: Usuário
Descrição: O usuário altera os dados cadastrais da sua conta
Pré-condições: O usuário deve estar logado e o sistema deve ter acesso à internet
Fluxo principal: 1. O sistema solicita que o usuário altere os dados (nome, matrícula, CPF, e-mail, função e Órgão) 2. O usuário altera os dados 3. O usuário solicita o envio dos dados 4. O sistema valida os dados preenchidos 5. O sistema envia os dados para o banco de dados e o caso de uso termina
Fluxo de exceção (Passo 4): Campos em branco 1. O sistema deve informar ao usuário que os dados estão incompletos e requisitar que este preencha-os corretamente e o caso de uso continua a partir do passo 2
Fluxo de exceção (Passo 4): Nenhum dado alterado 1. O sistema deve informar ao usuário que os dados não foram alterados e requisitar que este altere-os corretamente e o caso de uso continua a partir do passo 2

Fonte: Próprio autor

Tabela 6 – Casos de uso - Alterar senha (CSU04)

Alterar senha (CSU04)
Ator primário: Usuário
Descrição: O usuário altera sua senha de acesso ao sistema
Pré-condições: O usuário deve estar logado e o sistema deve ter acesso à internet
Fluxo principal: 1. O sistema solicita que o usuário informe os dados (senha atual, nova senha e repetir nova senha) 2. O usuário informa os dados 3. O usuário solicita o envio dos dados 4. O sistema valida os dados preenchidos 5. O sistema envia os dados para o banco de dados e o caso de uso termina
Fluxo de exceção (Passo 4): Campos em branco 1. O sistema deve informar ao usuário que os dados estão incompletos e requisitar que este preencha-os corretamente e o caso de uso continua a partir do passo 2
Fluxo de exceção (Passo 4): Nova senha não confere com repetir nova senha 1. O sistema deve informar ao usuário que os dados estão incorretos e requisitar que este preencha-os corretamente e o caso de uso continua a partir do passo 2
Fluxo de exceção (Passo 4): Senha atual não confere 1. O sistema deve informar ao usuário que sua senha está incorreta e requisitar que este preencha-a corretamente e o caso de uso continua a partir do passo 2

Fonte: Próprio autor

Tabela 7 – Casos de uso - Criar Boletim de Ocorrência (CSU05)

Criar Documento : Boletim de Ocorrência (CSU05)
Ator primário: Usuário
Descrição: O usuário cria um Boletim de Ocorrência e o sistema envia para o banco de dados
Pré-condições: O usuário deve estar logado e o sistema deve ter acesso à internet
Fluxo principal: <ol style="list-style-type: none"> 1. O sistema solicita que o usuário informe os dados (data, hora, condição meteorológica, ocorrência em consequência de chuva, solicitante, telefone, endereço, número, complemento, bairro, ponto de referência, DIV ADM, responsável, situação do solicitante, tipo de solicitação e ocorrência) 2. O usuário informa os dados 3. O usuário solicita o envio dos dados 4. O sistema valida os dados preenchidos 5. O sistema envia os dados para o banco de dados e o caso de uso termina
Fluxo de exceção (Passo 4): Campos em branco <ol style="list-style-type: none"> 1. O sistema deve informar ao usuário que os dados estão em branco e requisitar que este preencha-os corretamente e o caso de uso continua a partir do passo 2
Pós-condições: Os dados são inseridos no banco de dados e o sistema exibe uma nova tela realizando o (CSU07)

Fonte: Próprio autor

Tabela 8 – Casos de uso - Listar Boletim de Ocorrência (CSU06)

Listar Boletim de Ocorrência (CSU06)
Ator primário: Usuário
Descrição: O usuário solicita que o documento Boletim de Ocorrência seja listado pelo sistema e o sistema exibe os dados para o usuário
Pré-condições: O usuário deve estar logado, o sistema deve ter acesso à internet e o documento de Boletim de Ocorrência deve ter sido criado (CSU05)
Fluxo principal: <ol style="list-style-type: none"> 1. O usuário solicita os dados do documento Boletim de Ocorrência ao sistema 2. O sistema lista os dados do documento Boletim de Ocorrência e o caso de uso termina

Fonte: Próprio autor

Tabela 9 – Casos de uso - Listar Resumo do Boletim de Ocorrência (CSU07)

Listar Resumo do Boletim de Ocorrência (CSU07)
Ator primário: Usuário
Descrição: O usuário solicita que o sistema exiba o resumo do boletim de ocorrência
Pré-condições: O usuário deve estar logado, o sistema deve ter acesso à internet e o documento Boletim de Ocorrência deve ter sido listado (CSU06)
Fluxo principal: 1. O sistema exibe os dados do resumo do Boletim de Ocorrência obtidos no (CSU06) e o caso de uso termina

Fonte: Próprio autor

Tabela 10 – Casos de uso - Criar Dados da Vistoria (CSU08)

Criar Dados da Vistoria (CSU08)
Ator primário: Usuário
Descrição: O usuário cria um documento de Dados da Vistoria e o sistema envia para o banco de dados
Pré-condições: O usuário deve estar logado, o sistema deve ter acesso à internet e o Resumo do Boletim de Ocorrência associado deve ter sido listado (CSU07)
Fluxo principal: 1. O sistema solicita que o usuário informe os dados (identificação, classificação da edificação, danos da edificação, área/propriedade e ocupação) 2. O usuário informa os dados 3. O usuário solicita o envio dos dados 4. O sistema valida os dados preenchidos 5. O sistema envia os dados para o banco de dados e o caso de uso termina
Fluxo de exceção (Passo 4): Campos em branco 1. O sistema deve informar ao usuário que os dados estão incompletos e requisitar que este preencha-os corretamente e o caso de uso continua a partir do passo 2

Fonte: Próprio autor

Tabela 11 – Casos de uso - Criar Danos Humanos (CSU09)

Criar Danos Humanos (CSU09)
Ator primário: Usuário
Descrição: O usuário cria um documento de Danos Humanos e o sistema envia para o banco de dados
Pré-condições: O usuário deve estar logado, o sistema deve ter acesso à internet e o Resumo do Boletim de Ocorrência associado deve ter sido listado (CSU07)
Fluxo principal: <ol style="list-style-type: none"> 1. O sistema solicita que o usuário informe os dados 2. O usuário informa os dados 3. O usuário solicita o envio dos dados 4. O sistema envia os dados para o banco de dados e o caso de uso termina

Fonte: Próprio autor

Tabela 12 – Casos de uso - Criar Relatório Fotográfico (CSU10)

Criar Relatório Fotográfico (CSU10)
Ator primário: Usuário
Descrição: O usuário realiza o <i>upload</i> de uma imagem para criar o Relatório Fotográfico e o sistema a envia para o banco de dados
Pré-condições: O usuário deve estar logado, o sistema deve ter acesso à internet e o Resumo do Boletim de Ocorrência associado deve ter sido listado (CSU07)
Fluxo principal: <ol style="list-style-type: none"> 1. O sistema solicita que o usuário faça o <i>upload</i> da imagem 2. O usuário realiza o <i>upload</i> da imagem 3. O usuário solicita ao sistema o envio da imagem 4. O sistema envia a imagem para o banco de dados e o caso de uso termina

Fonte: Próprio autor

Tabela 13 – Casos de uso - Listar Dados da Vistoria (CSU11)

Listar Dados da Vistoria (CSU11)
Ator primário: Usuário
Descrição: O usuário solicita que o documento Dados da Vistoria seja listado pelo sistema e o sistema exibe os dados para o usuário
Pré-condições: O usuário deve estar logado, o sistema deve ter acesso à internet e o documento Dados da Vistoria deve ter sido criado (CSU08)
Fluxo principal: <ol style="list-style-type: none"> 1. O usuário solicita os dados do documento Dados da Vistoria ao sistema 2. O sistema lista os dados do documento Dados da Vistoria e o caso de uso termina

Fonte: Próprio autor

Tabela 14 – Casos de uso - Listar Danos Humanos (CSU12)

Listar Danos Humanos (CSU12)
Ator primário: Usuário
Descrição: O usuário solicita que o documento Danos Humanos seja listado pelo sistema e o sistema exibe os dados para o usuário
Pré-condições: O usuário deve estar logado, o sistema deve ter acesso à internet e o documento Dados da Vistoria deve ter sido criado (CSU09)
Fluxo principal: 1. O usuário solicita os dados do documento Danos Humanos ao sistema 2. O sistema lista os dados do documento Danos Humanos e o caso de uso termina

Fonte: Próprio autor

Tabela 15 – Casos de uso - Editar Boletim de Ocorrência (CSU13)

Editar Boletim de Ocorrência (CSU13)
Ator primário: Usuário
Descrição: O usuário edita o documento Boletim de Ocorrência e o sistema envia para o banco de dados
Pré-condições: O usuário deve estar logado, o sistema deve ter acesso à internet e o documento Boletim de Ocorrência deve ter sido listado (CSU06)
Fluxo principal: 1. O sistema solicita que o usuário modifique os dados a serem alterados 2. O usuário realiza as alterações dos dados 3. O usuário solicita o envio dos dados 4. O sistema valida os dados preenchidos 5. O sistema envia os dados para o banco de dados e o caso de uso termina
Fluxo de exceção (Passo 4): Campos em branco 1. O sistema deve informar ao usuário que os dados estão incompletos e requisitar que este preencha-os corretamente e o caso de uso continua a partir do passo 2

Fonte: Próprio autor

Tabela 16 – Casos de uso - Editar Dados da Vistoria (CSU14)

Editar Dados da Vistoria (CSU14)
Ator primário: Usuário
Descrição: O usuário edita o documento Dados da Vistoria e o sistema envia para o banco de dados
Pré-condições: O usuário deve estar logado, o sistema deve ter acesso à internet e o documento Dados da Vistoria deve ter sido listado (CSU11)
Fluxo principal: <ol style="list-style-type: none"> 1. O sistema solicita que o usuário modifique os dados a serem alterados 2. O usuário realiza as alterações dos dados 3. O usuário solicita o envio dos dados 4. O sistema valida os dados preenchidos 5. O sistema envia os dados para o banco de dados e o caso de uso termina
Fluxo de exceção (Passo 4): Campos em branco <ol style="list-style-type: none"> 1. O sistema deve informar ao usuário que os dados estão incompletos e requisitar que este preencha-os corretamente e o caso de uso continua a partir do passo 2

Fonte: Próprio autor

Tabela 17 – Casos de uso - Editar Danos Humanos (CSU15)

Editar Danos Humanos (CSU15)
Ator primário: Usuário
Descrição: O usuário edita o documento Danos Humanos e o sistema envia para o banco de dados
Pré-condições: O usuário deve estar logado, o sistema deve ter acesso à internet e o documento Boletim de Ocorrência deve ter sido listado (CSU12)
Fluxo principal: <ol style="list-style-type: none"> 1. O sistema solicita que o usuário modifique os dados a serem alterados 2. O usuário realiza as alterações dos dados 3. O usuário solicita o envio dos dados 4. O sistema envia os dados para o banco de dados e o caso de uso termina

Fonte: Próprio autor

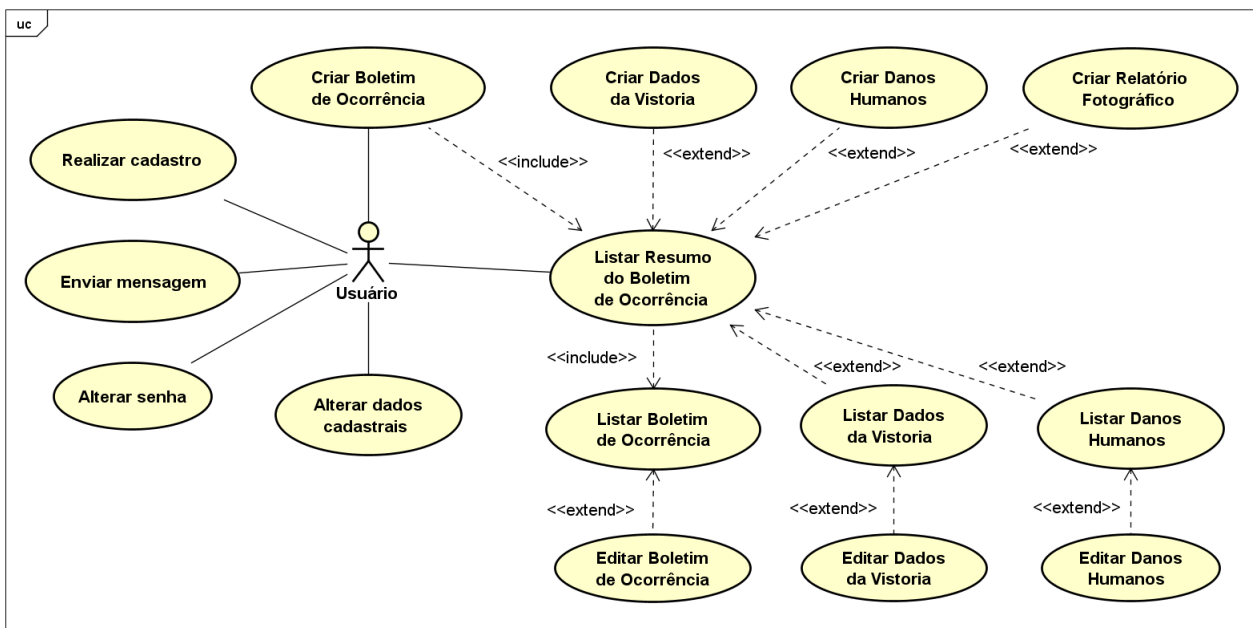
3.2.2 Diagrama de casos de uso

Segundo (RIBEIRO, 2012), o diagrama de casos de uso documenta o que o sistema faz do ponto de vista do usuário, descrevendo as principais funcionalidades e interações do sistema. Também pode-se afirmar que o diagrama de casos de uso é um documento narrativo que descreve a sequência de eventos de um ator, que usa um sistema, para completar um processo.

O diagrama de casos de uso é composto por três componentes:

- Ator: usuário (ou sistema) que interage com o sistema.
- Caso de uso: tarefa ou funcionalidade realizada pelo ator (ou sistema).
- Relacionamento: ligação de um ator, sistema, ou caso de uso com um caso de uso.

Figura 12 – Diagrama de Casos de Uso



Fonte: Próprio autor

4 Tecnologias Utilizadas

Nesta seção, será abordada a configuração do ambiente, todas as ferramentas de *softwares* necessárias para o iniciar o desenvolvimento e as tecnologias utilizadas pelo autor para organização do código-fonte do aplicativo proposto.

4.1 Ambiente de desenvolvimento

O trabalho foi desenvolvido no computador pessoal do autor. Porém, para uso de todas as ferramentas descritas, as configurações de *hardware* e *software* devem seguir as especificações recomendadas:

Sistema operacional: Microsoft Windows 10 / Mac OS X 10.10 / Ubuntu 14.04

Memória RAM: 8 GB

Armazenamento: 10 GB

Resolução da tela: 1280 x 800 pixels

Para configurar o ambiente de desenvolvimento, foi necessário instalar alguns *softwares*, contendo pacotes com ferramentas voltadas para a criação de aplicativos:

- JDK

Como primeiro passo, foi instalado o JDK (*Java Development Kit*) – versão 1.8, que são pacotes de ferramentas básicas, contendo os recursos necessários, como o compilador e bibliotecas, para criação e execução de programas em Java.

- Android Studio

Após a instalação do JDK, o próximo passo foi instalar o Android Studio – versão 3.1, que é a ferramenta oficial da Google para desenvolvimento de aplicativos Android. Por ser uma IDE, ela fornece uma interface gráfica permitindo ao desenvolvedor executar tarefas de forma prática.

- Android SDK

Instalando o Android Studio, que por padrão instala juntamente o Android *Software Development Kit* (Kit de Desenvolvimento de Software para Android), o qual inclui ferramentas de desenvolvimento, projetos exemplos com código-fonte, entre outros.

- AVD

Também na instalação do Android Studio, há a possibilidade de se instalar o AVD – *Android Virtual Device* (Dispositivo Android Virtual), que é o emulador utilizado para debugar e testar o aplicativo em um ambiente de execução Android.

- Git

O software Git – versão 2.17, que será abordado na próxima seção, foi utilizado para versionamento. O próprio Android Studio tem ferramentas de integração com o Git para que o desenvolvedor faça o controle de versão do código.

4.2 Controle de versão

Segundo (SCHMITZ, 2015), o versionamento, também conhecido como VCS (*Version Control System*), é um sistema de controle de versão de arquivos, ou um conjunto de arquivos. Isso permite ao desenvolvedor recuperar versões específicas de códigos já realizados em passos anteriores. Desta maneira, há maior gerenciamento do código, com a possibilidade de serem criadas várias versões para pôr em prática testes sem modificar a versão anterior de um conjunto de arquivos ou códigos.

Pelo fato do ambiente de desenvolvimento integrado escolhido para este trabalho possuir o *plugin*¹ para o Git, há maior acessibilidade e facilidade em utilizar o versionamento, uma vez que há uma interface dedicada com botões e notificações para se realizar toda a tarefa de controle de versão do código.

De acordo com (CUNHA, 2018), o Git serve para controle de versão, mas para isso, deve-se ter repositórios para serem gerenciados. O GitHub (<<https://github.com/>>) e o GitLab (<<https://gitlab.com/>>) são as duas plataformas de hospedagem baseadas no sistema de controle de versão Git mais utilizadas. Também são conhecidas como as redes sociais dos desenvolvedores, onde podem ser exibidos seus projetos. Para este trabalho, foi utilizado a plataforma do GitLab pela possibilidade de se criar um repositório privado, diferentemente do GitHub que só permite criar um repositório privado caso seja pago um valor mensal pela sua utilização.

Então, como o Android Studio possui suporte para utilizar o Git para versionamento, basta que o desenvolvedor tenha instalado o *software* do Git e logue sua conta do GitHub ou GitLab pelo Android Studio. Dessa forma, pode-se criar um repositório e armazená-lo em sua plataforma de hospedagem de escolha.

¹ Programa, ferramenta ou extensão encaixada a outro programa para adicionar mais funções e recursos.

4.3 Bibliotecas

Para utilizar bibliotecas em um projeto do Android Studio, é fundamental conhecer os conceitos do Gradle. Segundo (DUARTE, 2017), o Gradle é um *build system* (sistema de build responsável por construir seus projetos) moderno, que junta as melhores características de outros sistemas de *build* em um só. O Gradle roda sobre a JVM, permitindo um código ser escrito em Java para executar *scripts*² durante o *build*, sendo vantajoso para programadores Java (não obrigando-os a aprender uma nova linguagem). Ele funciona à base de *plugins*, então o desenvolver pode criar diversos *scripts* para que, durante a compilação, outras tarefas sejam executadas.

Segundo (CORDEIRO, 2015), todo projeto criado no Android Studio é estruturado para usar o Gradle, sendo um arquivo de configuração para o projeto principal e um para cada módulo. Os arquivos de configuração são chamados de *build.gradle*.

Para este trabalho, será apenas adicionado configurações de dependências no arquivo *build.gradle* do módulo para que as bibliotecas sejam importadas. Todas as bibliotecas utilizadas são de código aberto e estão disponíveis na plataforma GitHub, com licença para uso.

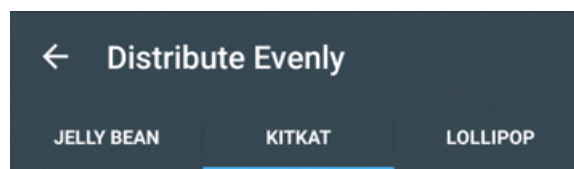
4.3.1 SmartTabLayout

A biblioteca SmartTabLayout, criada por ogaclejapan, em 2015, disponível em <<https://github.com/ogaclejapan/SmartTabLayout>>, permite ao desenvolvedor a criação de um *widget*³ para atuar como um menu de títulos customizado, fornecendo feedback contínuo ao usuário durante a seleção dos itens.

Para utilizar a biblioteca SmartTabLayout, bastou adicionar ao *build.gradle* as dependências:

```
1 implementation 'com.ogaclejapan.smarttablayout:library:1.6.1@aar'
2 implementation 'com.ogaclejapan.smarttablayout:utils-v4:1.6.1@aar'
```

Figura 13 – Bibliotecas : SmartTabLayout - Exemplo de uso



Fonte: <<https://github.com/ogaclejapan/SmartTabLayout>>

² Códigos que automatizam a execução de tarefas.

³ Componente de interface gráfica do usuário que inclui menus, botões, ícones, barras de rolagem, entre outros.

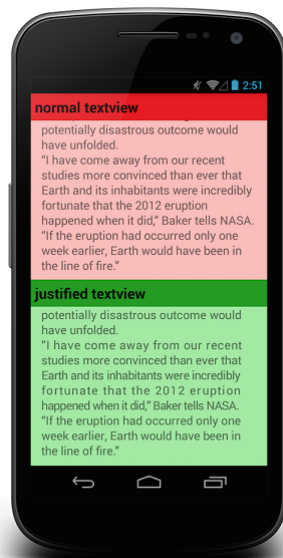
4.3.2 JustifiedTextView

A justificação de textos requer, no mínimo, API 26 (Android 8.0, *Oreo*). Como o aplicativo proposto utiliza a API 14, (Android 4.0, Ice Cream Sandwich) para abranger *smartphones* com sistemas ainda antigos, o uso da biblioteca `JustifiedTextView`, criada por `ufo22940268`, em 2014, disponível em <https://github.com/ufo22940268/android-justifiedtextview>, tornou-se fundamental no uso de textos, preenchendo a largura da tela sem espaços extras no final de cada linha.

Para utilizar a biblioteca `JustifiedTextView`, bastou adicionar ao `build.gradle` a dependência:

```
1 implementation 'me.biubiubiu.justifytext:library:1.1'
```

Figura 14 – Bibliotecas : `JustifiedTextView` - Exemplo de uso



Fonte: <https://github.com/ufo22940268/android-justifiedtextview>

4.3.3 Volley

Desenvolvida pela Google, disponível em <https://github.com/google/volley>, a `Volley` é uma biblioteca HTTP (*Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto) utilizada para fazer conexões de forma simples e rápida, sem a necessidade de criar várias *threads*⁴ secundárias. Integra-se facilmente a qualquer protocolo, dando suporte para imagens, strings e JSON⁵.

Para utilizar a biblioteca `Volley`, bastou adicionar ao `build.gradle` a dependência:

```
1 implementation 'com.android.volley:volley:1.1.1'
```

⁴ Forma de um processo se auto dividir em duas ou mais tarefas.

⁵ *JavaScript Object Notation* - Notação de Objetos JavaScript

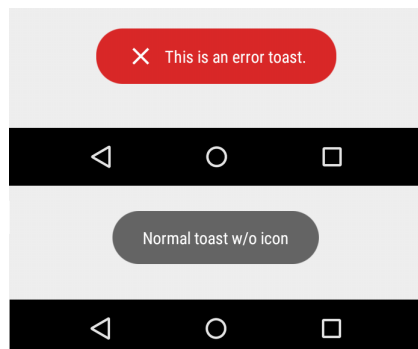
4.3.4 Toasty

Toast é um recurso do Android que exibe mensagens rápidas, e temporais, em uma pequena janela rodapé do aparelho, sendo útil para enviar uma informação ao usuário. A biblioteca Toasty, criada por Daniel Morales, em 2016, disponível em <<https://github.com/GreenderG/Toasty>>, adiciona maior facilidade de uso de um Toast, podendo customizar suas cores e adicionar um ícone à mensagem.

Para utilizar a biblioteca Toasty, foi necessário adicionar ao *build.gradle* a dependência:

```
1 implementation 'com.github.GreenderG:Toasty:1.3.0'
```

Figura 15 – Bibliotecas : Toasty - Exemplo de uso



Fonte: <<https://github.com/GreenderG/Toasty>>

4.3.5 GmailBackground

Inicialmente desenvolvida por Yesid, em 2015, disponível em <<https://github.com/yesidlazaro/GmailBackground>>, tendo como propósito o envio de e-mails, em segundo plano, sem a interação do usuário. Dessa maneira, utilizando a biblioteca, o usuário não é redirecionado para um aplicativo de e-mails dedicado, no seu dispositivo móvel, para enviar sua mensagem de e-mail, como é de forma padrão nos aplicativos. Porém, a biblioteca foi descontinuada, deixando de receber *updates*⁶. Então, Luong Vo decide continuar o projeto de Yesid, deixando a biblioteca, disponível em <<https://github.com/luongvo/GmailBackground>>, atualizada até os dias atuais.

Para utilizar a biblioteca GmailBackground, bastou adicionar ao *build.gradle* a dependência:

```
1 implementation 'com.github.luongvo:GmailBackground:2.1.1'
```

⁶ Atualização de um dado por algo mais recente e sofisticado.

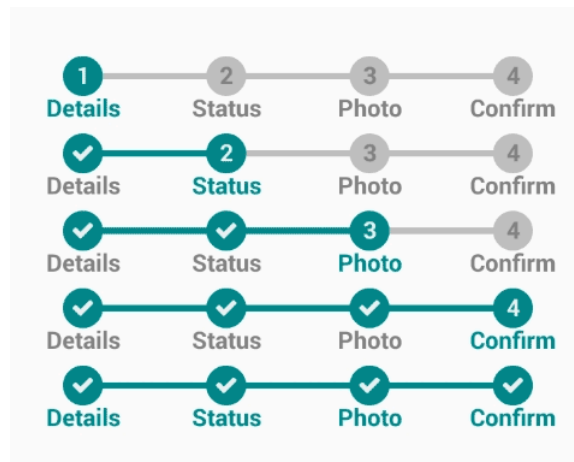
4.3.6 StateProgressBar

A biblioteca StateProgressBar, criada por Kofi Gyan, em 2016, disponível em <https://github.com/kofigyan/StateProgressBar>, permite separar e executar os vários estados de transições em uma ProgressBar. Desta maneira, uma tela que contém diversas informações, pode ser dividida em até 5 (cinco) telas, que serão etapas a serem concluídas, tornando a tarefa de percorrê-las uma a uma atrativa e menos cansativa, informando ao usuário quantos passos ainda faltam para conclusão.

Para utilizar a biblioteca StateProgressBar, bastou adicionar ao *build.gradle* a dependência:

```
1 implementation 'com.kofigyan.stateprogressbar:stateprogressbar:1.0.0'
```

Figura 16 – Bibliotecas : StateProgressBar - Exemplo de uso



Fonte: <https://github.com/kofigyan/StateProgressBar>

5 Desenvolvimento

Neste capítulo, serão abordadas as etapas do desenvolvimento do aplicativo, desde a criação do banco de dados, apresentação de imagens das telas desenvolvidas e partes do código para que algumas funcionalidades estejam disponíveis na aplicação.

5.1 Banco de dados

Como o PRODEC é um sistema web, que utiliza o SQL (*Structured Query Language* - Linguagem de Consulta Estruturada) e o SGBD (Sistema de Gerenciamento de Banco de Dados) que utiliza a linguagem SQL (o MySQL), a aplicação desenvolvida deve ter uma conexão com um banco de dados externo.

Por meio de encontros na Defesa Civil, foi proposto que o autor deste trabalho utilizasse um banco de dados fictício para simular a integração da aplicação com o banco de dados. Para isso, foi escolhido o 000webhost (<<https://br.000webhost.com/>>) que é um serviço de hospedagem gratuita de sites, dando suporte à PHP e MySQL.

5.1.1 Modelo do banco de dados

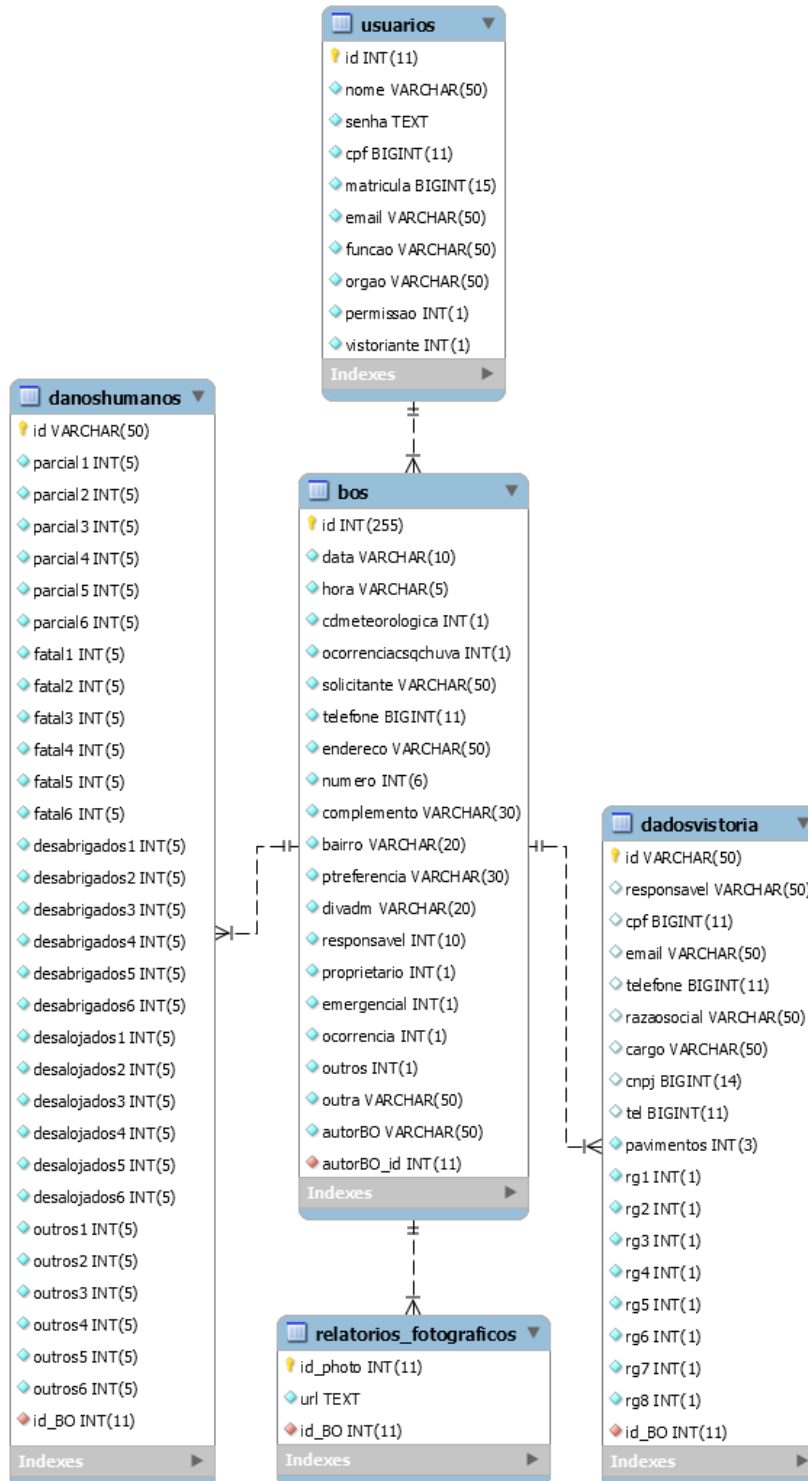
Segundo (ADAIL, 2011), o conceito de modelo relacional vem da teoria de conjuntos (álgebra relacional) atrelado a ideia de que não é relevante ao usuário saber onde os dados estão ou como eles se encontram. O modelo é representado por uma coleção de tabelas (entidade e relacionamento) e um conjunto de linhas (tuplas), que é uma lista de valores de atributos.

Cinco tabelas foram criadas para controle dos dados do sistema:

- usuarios : Armazena os usuários cadastrados no sistema.
- bos : Armazena os documentos de Boletins de Ocorrência registrados pelo usuário no sistema.
- dadosvistoria : Armazena os documentos de Dados da Vistoria dos Boletins de Ocorrência registrados pelo usuário no sistema.
- danoshumanos : Armazena os documentos de Danos Humanos dos Boletins de Ocorrência registrados pelo usuário do sistema.
- relatoriosfotograficos : Armazena os Relatórios Fotográficos dos Boletins de Ocorrência registrados pelo usuário do sistema.

A figura 17, representa o modelo do banco de dados criado a partir do software *MySQL Workbench*.

Figura 17 – Modelo do Banco de Dados



Fonte: Próprio autor

5.1.2 API

Segundo (FERNANDES, 2018), as APIs (*Application Programming Interface* - Interface de Programação de Aplicativos) são uma forma de integrar sistemas, proporcionando segurança dos dados e facilidade no intercâmbio entre sistemas que possuem linguagem totalmente distintas, de maneira ágil.

De acordo com (LACERDA, 2017), a linguagem de programação *PHP* (um acrônimo recursivo para *PHP: Hypertext Preprocessor*) é amplamente utilizada por desenvolvedores para construção de aplicações, como por exemplo *websites* dinâmicos, pois permite a interação com o usuário por meio de links, formulários e parâmetros de *URL*.

Por meio de um *web service*, pode-se disponibilizar funções ou mensagens para qualquer aplicação na internet. Nesse contexto, o *PHP* fornece funções para trabalhar com *web services* de forma simples. Um *web service*, basicamente processa as solicitações recebidas de uma fonte externa e retorna um resultado. *Web services REST* utilizam recursos do protocolo *HTTP* (por exemplo *POST*) para comunicação entre a aplicação e o servidor.

A API desenvolvida para este trabalho está disponível nas seguintes seções de Apêndices:

- 1) seção A.1 - Conexão - Responsável pela conexão com o banco de dados
- 2) seção A.2 - Login - Responsável pela autenticação do usuário
- 3) seção A.3 - Minha Conta - Responsável pela alteração dos dados do usuário
- 4) seção A.4 - Trocar Senha - Responsável pela alteração da senha do usuário
- 5) seção A.5 - Criar Boletim de Ocorrência - Responsável pela criação de um documento de Boletim de Ocorrência
- 6) seção A.6 - Consultar Boletins de Ocorrência - Responsável pela busca de documentos de Boletins de Ocorrência
- 7) seção A.7 - Consultar Resumo do Boletim de Ocorrência - Responsável pela busca de um documento de Boletim de Ocorrência
- 8) seção A.8 - Alterar Boletim de Ocorrência - Responsável pela alteração de um documento de Boletim de Ocorrência
- 9) seção A.9 - Criar Dados da Vistoria - Responsável pela criação de um documento de Dados da Vistoria
- 10) seção A.10 - Consultar Dados da Vistoria - Responsável pela busca de um documento de Dados da Vistoria
- 11) seção A.11 - Criar Danos Humanos - Responsável pela criação de um documento

de Danos Humanos

12) seção A.12 - Consultar Danos Humanos - Responsável pela busca de um documento de Danos Humanos

13) seção A.13 - Criar Relatório Fotográfico - Responsável pelo *upload* de um arquivo de imagem do Relatório Fotográfico

Para melhor organização e manutenção do projeto, algumas **variáveis de acesso ao banco de dados** foram criadas em uma classe, no pacote da aplicação, com todos os endereços de comunicação com a API desenvolvida.

Cada classe é responsável por controlar seus atributos, sendo eles públicos, privados ou protegidos (modificadores de acesso que dão visibilidade de acessos às classes). Ao declarar um atributo como público estático final, o atributo ficará visível à todas as classes do projeto e têm valor único, sendo impossível alterá-los.

```
1 private static final String SERVER_IP = "http://marcellocamara
    .000webhostapp.com/";
2 private static final String DATABASE_NAME_FOLDER = "TCC/";
3 public static final String URL = SERVER_IP + DATABASE_NAME_FOLDER
    ;
4
5 public static final String LOGIN = URL+"login.php";
6 public static final String READ_BOS = URL+"consultarBOS.php";
7 public static final String EDIT_PROFILE=URL+"minhaConta.php";
8 public static final String EDIT_PASSWORD = URL + "trocarSenha.php
    ";
9 public static final String NEW_BO = URL+"criarBO.php";
10 public static final String PULL_BO = URL+"resumoBO.php";
11 public static final String UPDATE_BO = URL+"updateBO.php";
12 public static final String UPLOAD_PHOTO = URL + "uploadFotos.php"
    ;
13 public static final String READ_DADOS_VISTORIA = URL + "
    consultarDadosVistoria.php";
14 public static final String NEW_DADOS_VISTORIA = URL + "
    criarDadosVistoria.php";
15 public static final String READ_DANOS_HUMANOS = URL + "
    consultarDanosHumanos.php";
16 public static final String NEW_DANOS_HUMANOS = URL + "
    criarDanosHumanos.php";
```

Variáveis de acesso ao banco de dados

5.2 Implementação

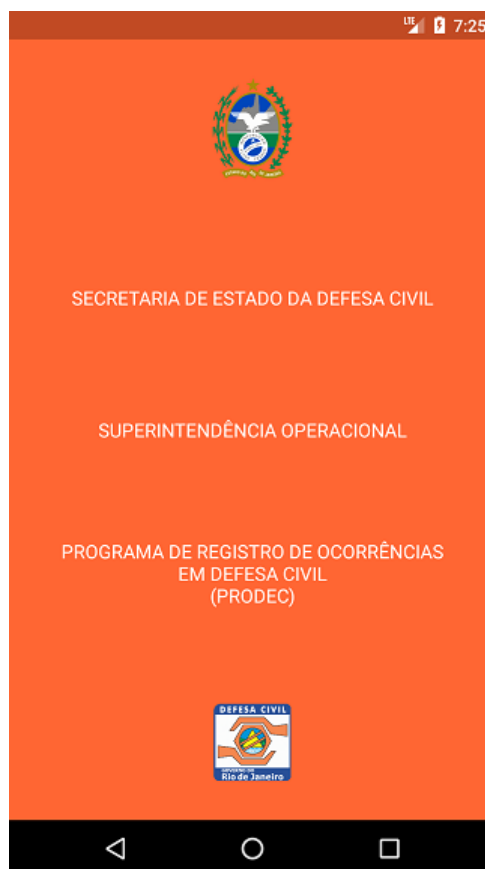
Nesta seção serão abordadas as telas do aplicativo desenvolvido e suas principais funcionalidades. Também serão abordados trechos de códigos, quando necessário para explicação de alguma funcionalidade desenvolvida.

5.2.1 Tela de Boas-vindas

Também conhecida como *splash screen*, a tela de boas-vindas é uma tela de apresentação ao usuário no primeiro instante em que o aplicativo é executado. Muitas empresas utilizam, em seus aplicativos, esta tela com intuito de apresentar uma logotipo¹ ou realizar algum pré-processamento (carregar algum dado).

Para o presente projeto, a *splash screen* foi utilizada apenas para apresentação de uma aplicação da Defesa Civil Estadual do Rio de Janeiro, pois não há nenhum tipo de dado para ser pré-processado.

Figura 18 – Aplicativo PRODEC - SplashScreen



Fonte: Próprio autor

¹ Peça de design que identifica ou representa uma entidade (marca de produto ou serviço).

O próprio Android Studio oferece a classe *Handler*, que trabalha com *threads*². Para executar uma *thread* com *delay*³ no *Handler*, utilizou-se o método *postDelayed()*, localizado na linha 7 da [tela de boas-vindas](#). Este método recebe uma interface *Runnable*, que é a *thread* que será executada após um determinado tempo (*delay*), em milissegundos.

```

1 public class SplashScreen extends AppCompatActivity {
2
3     @Override
4     protected void onCreate(Bundle savedInstanceState) {
5
6         super.onCreate(savedInstanceState);
7         setContentView(R.layout.activity_splash_screen);
8         Handler handle = new Handler();
9         handle.postDelayed(new Runnable() {
10            @Override
11            public void run() {
12                //Executado quando o delay de 3500ms acabar
13                Intent prodec = new Intent(SplashScreen.this,
14                    ProdecMain.class);
15                startActivity(prodec);
16                finish();
17            }
18        }, 3500);
19    }
20
21 }

```

Tela de boas-vindas

5.2.2 Tela Inicial

Para implementação da [Tela Inicial](#), foi utilizada a biblioteca [SmartTabLayout](#), também abordada anteriormente. Esta biblioteca, permite que um componente do tipo *ViewPager* possa exibir e alternar entre, no máximo, 3 (três) *layouts*. Desta maneira, as telas [Tela Inicial](#), [Cadastro de Adesão](#) e [Login](#) são 3 (três) *Fragments* que são exibidas dentro do *ViewPager* de uma única *Activity*, a *MainActivity*.

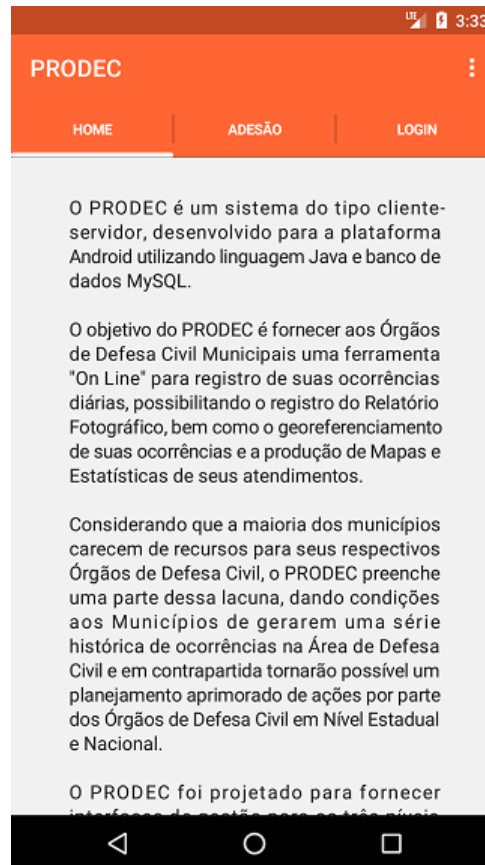
Um fato curioso é que o Android Studio não conta com suporte para justificar textos. O SDK conta com as ferramentas de alinhamento do texto à esquerda, ao centro e

² Forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente

³ Termo usado para designar o tempo de atraso de qualquer coisa.

à direita. Então, para justificar o texto de apresentação do aplicativo, da [Tela Inicial](#), foi utilizada a biblioteca [JustifiedTextView](#).

Figura 19 – Aplicativo PRODEC - Tela inicial



Fonte: Próprio autor

Na linha 9, da [configuração do Adapter para o ViewPager](#) pode-se observar a criação dos itens de tela, do tipo *Fragment*, que serão utilizados no *ViewPager*, responsável pela exibição das telas e do menu ao topo para que se possa alternar entre as telas.

```

1 viewPager = findViewById(R.id.viewPager);
2 smartTabLayout = findViewById(R.id.smartTabLayout);
3 //Configurando o adapter e exibindo os fragments
4 FragmentPagerAdapter adapter = new FragmentPagerAdapter(
5     getSupportFragmentManager(), FragmentPagerAdapter.with(this
6     )
7     .add("Home", HomeFragment.class)
8     .add("Adesao", CadastroFragment.class)
9     .create()
10 );
11 viewPager.setAdapter(adapter);

```

```
12 smartTabLayout.setViewPager(viewPager);
13 //Exibe o primeiro Fragment criado (Home)
14 viewPager.setCurrentItem(0);
15 }
```

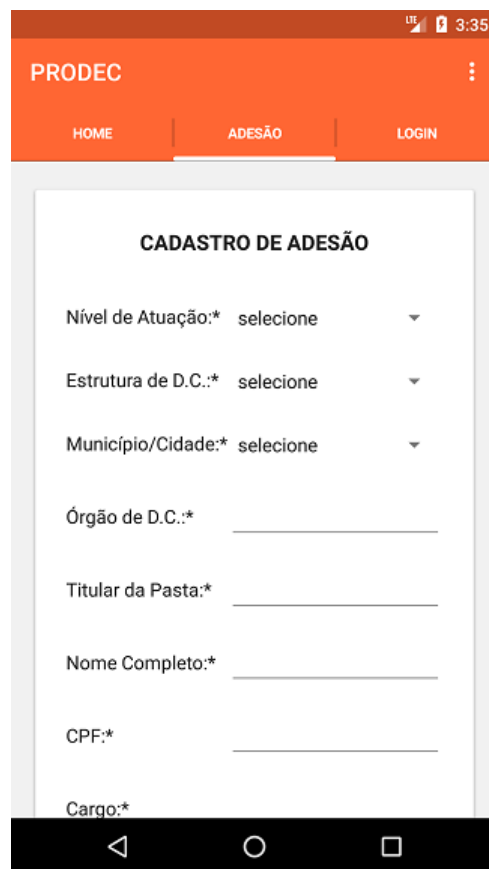
Configuração do *adapter* para o *ViewPager*

5.2.3 Cadastro de Adesão

Para a tela de [Cadastro de Adesão](#) e outras adiante, utilizou-se um componente gráfico de design, o *CardView*, que é, por padrão, um cartão branco com objetivo de sobrepor e deixar os *widgets*, como textos, figuras, botões, entre outros, com “visual amigável” para apresentação ao usuário.

Esta tela, e também outras que contêm formulários adiante, contam com validação dos dados preenchidos, para verificação de campos vazios e seleção de itens.

Figura 20 – Aplicativo PRODEC - Cadastro de Adesão



A imagem mostra a interface de usuário do aplicativo PRODEC, especificamente a tela de "CADASTRO DE ADESÃO". No topo, há uma barra de navegação laranja com o nome do aplicativo "PRODEC" e ícones para "HOME", "ADESÃO" (ativo) e "LOGIN". O formulário principal é branco e contém os seguintes campos:

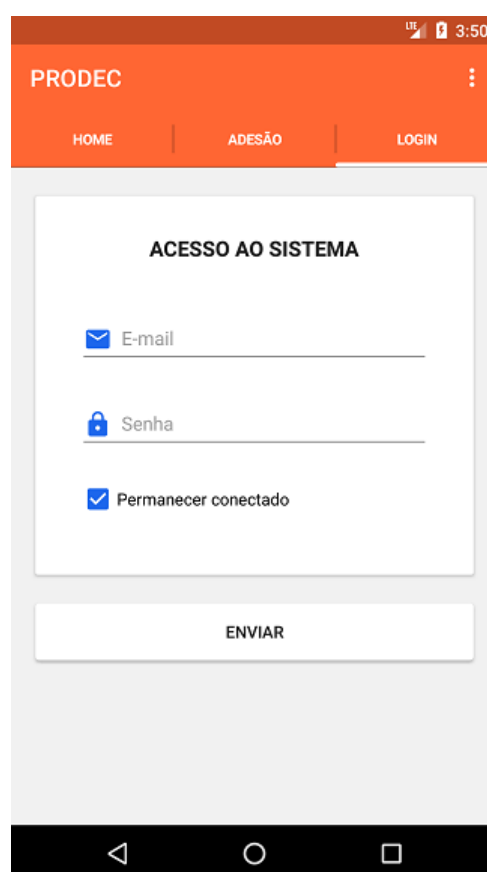
- Nível de Atuação:* seleccione (menu suspenso)
- Estrutura de D.C.:* seleccione (menu suspenso)
- Município/Cidade:* seleccione (menu suspenso)
- Órgão de D.C.:* (campo de texto)
- Titular da Pasta:* (campo de texto)
- Nome Completo:* (campo de texto)
- CPF:* (campo de texto)
- Cargo:* (campo de texto)

Fonte: Próprio autor

5.2.4 Login

Todo processo de comunicação com a API no aplicativo, através de um *web service*, utiliza a biblioteca [Volley](#), abordada anteriormente. Nesta subseção, será abordado como é realizada esta requisição, sendo a mesma para todas as outras que necessitam criar ou alterar dados. A diferença entre as demais requisições são a quantidade de parâmetros passadas e a forma de leitura do arquivo JSON.

Figura 21 – Aplicativo PRODEC - Login



Fonte: Próprio autor

Para entendimento das explicações a seguir, a [requisição HTTP](#) será utilizada como exemplo de utilização da biblioteca citada acima. Como o trecho de código é extenso, este será o único exemplo de *web service*.

Ao realizar requisições *web services* deve-se utilizar a classe `RequestQueue`, que utiliza uma nova *thread* responsável pelo gerenciamento das requisições. Uma instância da classe `RequestQueue` deve ser criada, como mostrado na linha 73, através da classe `Volley`, utilizando o método `newRequestQueue`. Após criar a instância da classe `RequestQueue`, deve-se utilizar um objeto do tipo `Request` como parâmetro para o método `add` da classe `RequestQueue`, demonstrado na linha 74.

Um `ProgressDialog`, que é um elemento de interação com o usuário, é exibido

para indicar que há algo sendo processado, demonstrado na linha 5. Esta é uma boa estratégia para impedir que o usuário realize outras operações enquanto os dados estão sendo processados.

```
1 final ProgressDialog progressDialog = new ProgressDialog(this);
2 progressDialog.setTitle("Login");
3 progressDialog.setMessage("Carregando...");
4 progressDialog.setCancelable(false);
5 progressDialog.show();
6
7 StringRequest stringRequest = new StringRequest(Request.Method.
    POST, DatabaseAddresses.LOGIN,
8     new Response.Listener<String>() {
9         @Override
10            public void onResponse(String response) {
11
12                progressDialog.dismiss();
13                try {
14
15                    JSONObject jsonObject = new JSONObject(
16                        response);
17                    if(jsonObject.getString("sucesso").equals("1"
18                        )){
19                        //Lê o JSON retornado
20                        JSONObject object = jsonArray.
21                            getJSONObject(0);
22                        //Obtém os dados do Banco de dados
23                        //passados pelo JSON
24                        String bd_id = object.getString("id")
25                            .trim();
26                        String bd_nome = object.getString("
27                            nome").trim();
28                        String bd_cpf = object.getString("cpf
29                            ").trim();
30                        String bd_matricula = object.
31                            getString("matricula").trim();
32                        String bd_email = object.getString("
33                            email").trim();
34                        String bd_funcao = object.getString("
35                            funcao").trim();
36                        String bd_orgao = object.getString("
37                            orgao").trim();
```

```
27         }
28
29         //Cria uma sessão do login realizado na
30         //cache, utilizando os dados obtidos do
31         //JSON
32         SessionManager sessionManager = new
33         SessionManager(this);
34         sessionManager.criarSessao(bd_id, bd_nome
35         , bd_cpf, bd_matricula, bd_email,
36         bd_funcao, bd_orgao);
37
38         Intent PRODEC = new Intent(this,
39         ProdecMain.class);
40
41         if(!checkBox.isChecked()){
42             //Variável que controla se há
43             //necessidade de realizar logout da
44             //sessão ao fechar o app na próxima
45             //tela
46             PRODEC.putExtra("logout", true);
47         }
48         startActivity(PRODEC);
49         finish();
50     } else {
51         //Caso a resposta não seja "sucesso" =
52         // "1"
53         Toasty.warning(this, "Login ou senha
54         incorretos.", Toast.LENGTH_SHORT, true)
55         .show();
56     }
57
58     } catch (JSONException e) {
59         e.printStackTrace();
60         Toasty.error(this, "Login ou senha incorretos
61         .", Toast.LENGTH_LONG, true).show();
62     }
63 }
64 },
65
66 new Response.ErrorListener() {
67     @Override
68     public void onErrorResponse(VolleyError error) {
```

```

56         //Sem conexão com a URL fornecida na variável
           URL_LOGIN
57         progressDialog.dismiss();
58         Toasty.error(this, "Erro: sem conexão com o banco
           de dados.", Toast.LENGTH_LONG, true).show();
59     }
60     })
61 {
62     @Override
63     protected Map<String, String> getParams() {
64         //Variáveis passadas por parâmetro via POST
65         Map<String, String> params = new HashMap<>();
66         params.put("email", email);
67         params.put("senha", senha);
68         return params;
69     }
70
71 };
72
73 RequestQueue requestQueue = Volley.newRequestQueue(this);
74 requestQueue.add(stringRequest);
75
76 }

```

Requisição HTTP

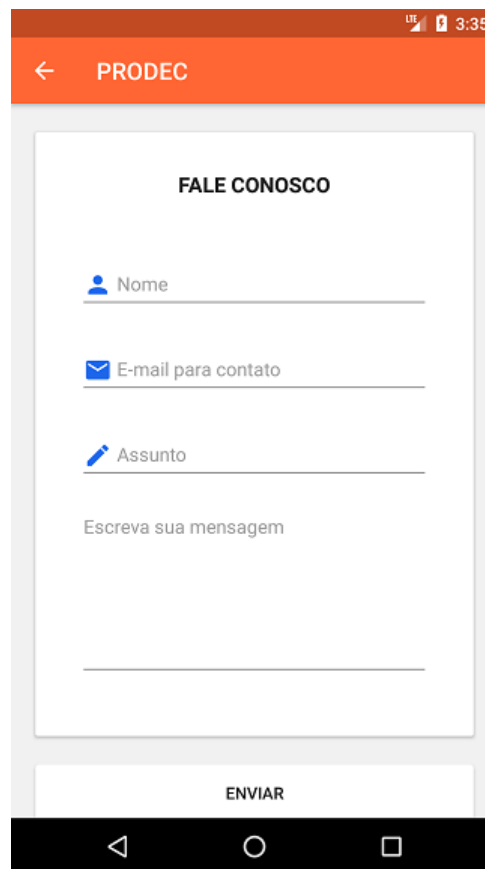
5.2.5 Fale Conosco

Esta é uma tela na qual o usuário pode interagir com o administrador do sistema, enviando uma mensagem por escrito. Para sua implementação, utilizou-se a biblioteca [GmailBackground](#), abordada anteriormente.

A biblioteca permite o envio de mensagens de e-mail direto pelo aplicativo desenvolvido, sem que haja o redirecionamento para um aplicativo de envio de e-mails no *smartphone* do usuário, através de configurações SMTP (*Simple Mail Transfer Protocol* - Protocolo Simples de Transferência de Correio) do *Gmail*⁴. Para isso, foi necessário deixar um e-mail do *Gmail*, com login e senha no código, disponível para o envio das mensagens.

⁴ Serviço de e-mails da Google. É comum ver *smartphones* com sistema operacional Android possuir este aplicativo pré-instalado.

Figura 22 – Aplicativo PRODEC - Fale Conosco



Fonte: Próprio autor

5.2.6 Menu de navegação

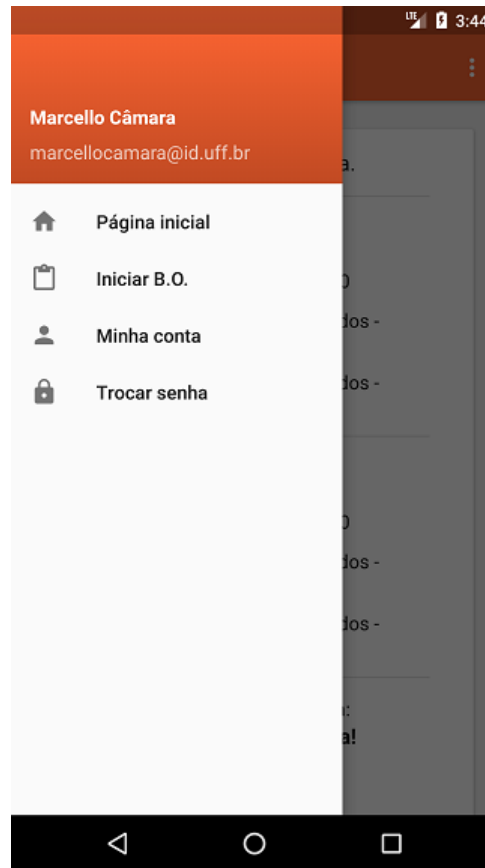
O *navigation drawer* permite a criação de um menu de navegação para a aplicação, através do uso da classe *Drawer Layout*.

Para alternar entre as opções do menu de navegação, utilizou-se o método *onNavigationItemSelectedListener* demonstrado na linha 2 do [menu de navegação](#), onde cada evento de clique nas opções do menu são tratadas.

```
1 @Override
2 public boolean onNavigationItemSelectedListener(MenuItem item) {
3     //Tratamento de cada um dos itens do menu
4     switch (item.getItemId()){
5         case R.id.nav_inicial : {
6             ProdecHomeFragment prodecHomeFragment = new
7                 ProdecHomeFragment();
8             FragmentTransaction fragmentTransaction =
9                 getSupportFragmentManager().beginTransaction();
10            fragmentTransaction.replace(R.id.frameLayout,
11                prodecHomeFragment);
```

```
9         fragmentTransaction.commit();
10         break;
11     }
12     case R.id.nav_B0 : {
13         ProdecIniciarBOFragment prodecIniciarBOFragment = new
14             ProdecIniciarBOFragment();
15         FragmentTransaction fragmentTransaction =
16             getSupportFragmentManager().beginTransaction();
17         fragmentTransaction.replace(R.id.frameLayout,
18             prodecIniciarBOFragment);
19         fragmentTransaction.commit();
20         break;
21     }
22     case R.id.nav_conta : {
23         ProdecMinhaContaFragment prodecMinhaContaFragment =
24             new ProdecMinhaContaFragment();
25         FragmentTransaction fragmentTransaction =
26             getSupportFragmentManager().beginTransaction();
27         fragmentTransaction.replace(R.id.frameLayout,
28             prodecMinhaContaFragment);
29         fragmentTransaction.commit();
30         break;
31     }
32     case R.id.nav_senha : {
33         ProdecTrocarSenhaFragment prodecTrocarSenhaFragment =
34             new ProdecTrocarSenhaFragment();
35         FragmentTransaction fragmentTransaction =
36             getSupportFragmentManager().beginTransaction();
37         fragmentTransaction.replace(R.id.frameLayout,
38             prodecTrocarSenhaFragment);
39         fragmentTransaction.commit();
40         break;
41     }
42 }
43
44 DrawerLayout drawer = findViewById(R.id.drawer_layout);
45 drawer.closeDrawer(GravityCompat.START);
46 return true;
47
48 }
49 }
```


Figura 23 – Aplicativo PRODEC - Menu de navegação



Fonte: Próprio autor

5.2.7 Tela Inicial do Sistema

Para construir esta tela, idêntica a da plataforma *web*, foi necessário implementar *drawables*, que são recursos de imagens da aplicação, para que os ícones circulares ficassem piscando, parecendo um semáforo.

Para isso, foram criados 3 (três) diferentes *drawables* para simular o efeito de piscar das cores verde, amarelo e vermelho. A [animação da circunferência](#) exemplifica a criação de um *drawable* na cor verde, que altera para a cor branca de 500 em 500 milissegundos, demonstrados na linha 4 e 7.

```
1 <animation-list xmlns:android="http://schemas.android.com/apk/res
  /android">
2
3   <item android:drawable="@drawable/ic_circle_green"
4       android:duration="500"/>
5
6   <item android:drawable="@drawable/ic_circle_uncolored"
7       android:duration="500"/>
```

```
8 </animation-list>
```

Animação da circunferência

Figura 24 – Aplicativo PRODEC - Tela Inicial do Sistema



Fonte: Próprio autor

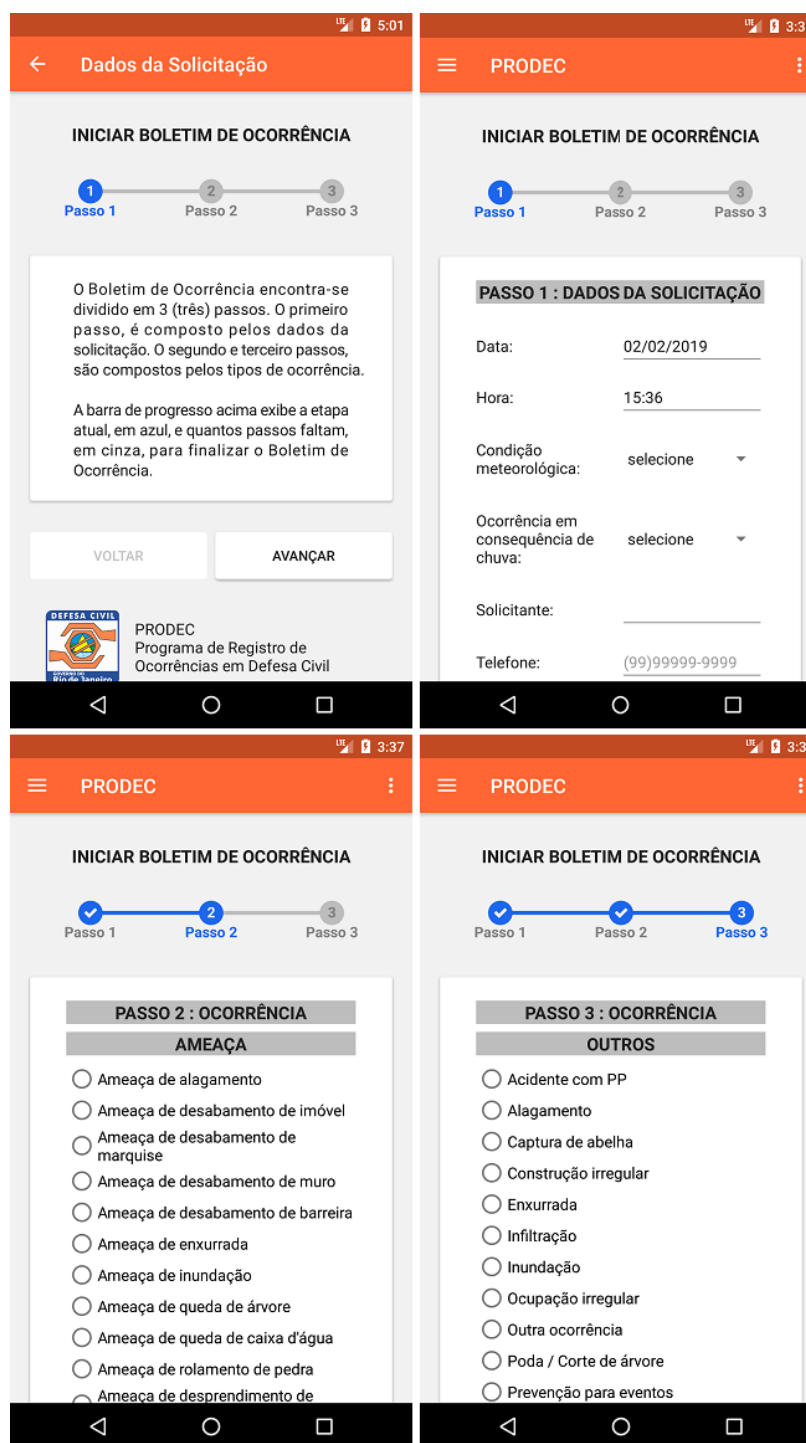
5.2.8 Iniciar Boletim de Ocorrência

O documento de Boletim de Ocorrência foi dividido em 3 (três) passos, devido à sua extensão para preenchimento dos dados, demonstrado no [Apêndice B](#).

A [Figura 25](#) demonstra as 4 (quatro) telas desenvolvidas para criação de um documento de Boletim de Ocorrência, sendo a primeira tela introdutória, a segunda tela para preenchimento dos dados da solicitação e do solicitante, a terceira e quarta para marcação do tipo de ocorrência.

Ao gerar um novo documento de Boletim de Ocorrência, o usuário é redirecionado para a tela da [Figura 29](#), de identificação do Boletim de Ocorrência, sendo possível aderir outros documentos.

Figura 25 – Aplicativo PRODEC - Iniciar Boletim de Ocorrência



Fonte: Próprio autor

5.2.9 Minha Conta

Nesta tela, é possível que o usuário altere seus dados cadastrais no sistema do PRODEC. Não foi implementada uma validação do número do CPF, pois o sistema já realiza esta verificação.

Para esta tela, assim como qualquer outra que requer o preenchimento de um endereço de e-mail, há uma validação do campo preenchido. O método `isValidEmail()`, do trecho de código de [validação de e-mail](#), exemplifica este processo descrito acima.

```
1 public static boolean isValidEmail(String email) {  
2  
3     Pattern pattern = Patterns.EMAIL_ADDRESS;  
4     boolean result = pattern.matcher(email).matches();  
5     return result;  
6  
7 }
```

Validação de *e-mail*

Figura 26 – Aplicativo PRODEC - Minha Conta



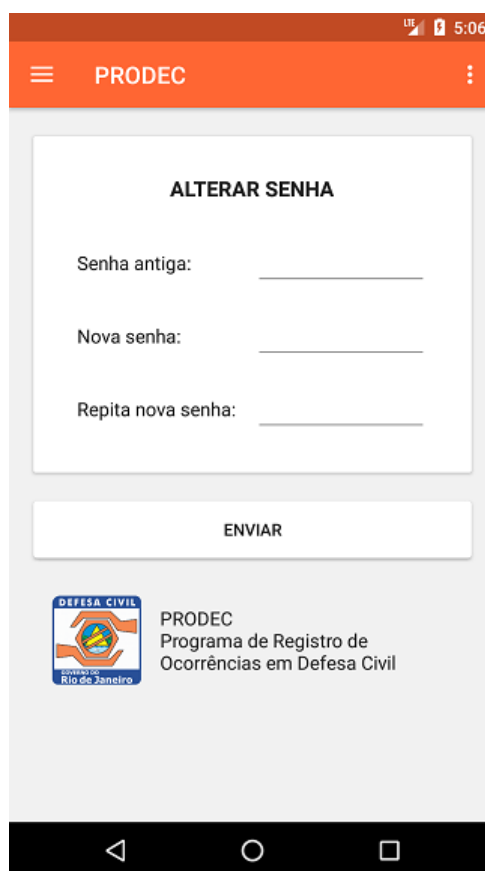
A imagem mostra a interface de usuário do aplicativo PRODEC, especificamente a tela 'MINHA CONTA'. No topo, há uma barra de navegação laranja com o ícone de menu, o texto 'PRODEC' e o ícone de opções. Abaixo, o título 'MINHA CONTA' está centralizado. O formulário contém campos de texto para: Nome (Marcello Câmara), Matrícula (213060089), CPF (1234567890), E-mail (marcellocamara@id.uff.br), Função (Desenvolvedor) e Órgão (DGDEC). Um botão 'ENVIAR' está localizado na base do formulário. O rodapé do aplicativo mostra o texto 'NECESSA AJUDA?' e o sistema operacional Android com seus ícones de navegação.

Fonte: Próprio autor

5.2.10 Alterar Senha

Como as senhas geradas para o primeiro acesso são randômicas, esta tela foi desenvolvida para que o usuário possa alterá-la.

Figura 27 – Aplicativo PRODEC - Alterar Senha

A imagem mostra a interface de usuário do aplicativo PRODEC para alterar a senha. No topo, há uma barra de status com o nome do aplicativo 'PRODEC' e ícones de menu e opções. Abaixo, o título 'ALTERAR SENHA' está centralizado. O formulário contém três campos de entrada: 'Senha antiga:', 'Nova senha:' e 'Repita nova senha:'. Abaixo dos campos, há um botão 'ENVIAR'. Na base da tela, há um logotipo da Defesa Civil do Rio de Janeiro e o texto 'PRODEC Programa de Registro de Ocorrências em Defesa Civil'. A barra de navegação do Android está visível na base.

Fonte: Próprio autor

Uma simples validação de campos foi implementada para esta tela, permitindo que a senha só seja alterada caso a nova senha seja diferente da senha antiga e que a nova senha seja a mesma preenchida nos campos 'Nova senha' e 'Repita nova senha'.

5.2.11 Documentos

É possível navegar até esta tela pressionando uma das circunferências da tela inicial do sistema, em verde, amarelo ou vermelho (se não houver nenhum documento para ser exibido, o clique não abre a tela de documentos).

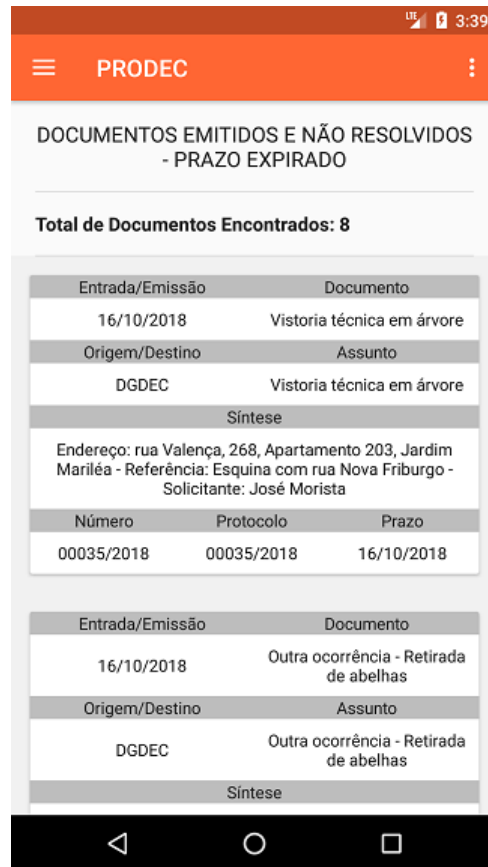
Para implementação desta tela, foi utilizado o *widget RecyclerView*, responsável por carregar cada um dos itens de uma lista, para exibir os documentos de Boletim de Ocorrência.

O trecho de código da [configuração do Adapter para o RecyclerView](#) demonstra a utilização de um *adapter* que será utilizado para criação do *RecyclerView*. O *adapter* é uma classe responsável por associar a lista de conteúdo à tela correspondente, onde cada objeto da lista será um item na lista.

O *ViewHolder*, mostrado na linha 36, é a parte visual de cada item da lista, que

será replicada para todos elementos (ficando dentro do *adapter*, na estrutura acima).

Figura 28 – Aplicativo PRODEC - Documentos



Fonte: Próprio autor

```

1 public class ListBOAdapter extends RecyclerView.Adapter<
    ListBOAdapter.ViewHolder> {
2
3     private List<BoletimOcorrencia> listaBOS;
4     private Context context;
5
6     public ListBOAdapter(List<BoletimOcorrencia> listaBOS ,
        Context context) {
7         this.listaBOS = listaBOS;
8         this.context = context;
9     }
10
11     @NonNull
12     @Override
13     public ListBOAdapter.ViewHolder onCreateViewHolder(@NonNull
        ViewGroup parent, int viewType) {

```

```
14     View v = LayoutInflater.from(parent.getContext()).inflate
15         (R.layout.list_bo_item, parent, false);
16     final ViewHolder viewHolder = new ViewHolder(v);
17
18     viewHolder.cardView.setOnClickListener(new View.
19         OnClickListener() {
20         @Override
21         public void onClick(View v) {
22             //Tratamento de clique de um item da lista
23         }
24     });
25     return viewHolder;
26
27     @Override
28     public void onBindViewHolder(@NonNull ListBOAdapter.
29         ViewHolder holder, int position) {
30         //Exibe de cada item da lista
31     }
32
33     @Override
34     public int getItemCount() {
35         return listaBOS.size();
36     }
37
38     public class ViewHolder extends RecyclerView.ViewHolder{
39
40         //Cria elementos que compõe a tela
41         public ViewHolder(View itemView) {
42             super(itemView);
43             //Referência de cada um dos elementos criados acima
44         }
45     }
```

Configuração do *Adapter* para o *RecyclerView*

5.2.12 Identificação do Boletim de Ocorrência

As duas maneiras de se chegar até esta tela são as seguintes: clicar em um documento na tela de documentos, vide [Figura 28](#), ou após a criação de um documento de Boletim de Ocorrência, vide [Figura 25](#).

Ambas as ações descritas acima, irão redirecionar o usuário à esta tela, com o resumo dos dados do Boletim de Ocorrência em questão.

Nesta tela, também é possível editar o documento de Boletim de Ocorrência, no botão ‘Solicitação’, criar um documento de Dados da Vistoria, no botão ‘Dados da Vistoria’, criar um documento de Danos Humanos, no botão ‘Danos Humanos’ e enviar arquivos de imagem, no botão ‘Relatório Fotográfico’.

Figura 29 – Aplicativo PRODEC - Identificação do Boletim de Ocorrência



Fonte: Próprio autor

5.2.13 Dados da Vistoria

É possível criar ou editar um documento de Dados da Vistoria nesta tela. Caso o documento de Dados da Vistoria já exista, a aplicação buscará os dados no banco de dados e preencherá os campos automaticamente. Caso contrário, os campos estarão em branco para que se crie um novo documento de Dados da Vistoria.

Para implementação desta tela, uma validação da escolha do item ‘Responsável ausente’ foi realizada, fazendo com que os outros campos de preenchimento com os dados do responsável se ocultem da tela.

Figura 30 – Aplicativo PRODEC - Dados da Vistoria



A imagem mostra a interface de usuário do aplicativo PRODEC na tela 'Dados da Vistoria'. O cabeçalho é laranja com um ícone de seta para trás e o texto 'Dados da Vistoria'. Abaixo, há um formulário branco com o título 'DADOS DA VISTORIA' e um subtítulo 'IDENTIFICAÇÃO'. O formulário contém os seguintes campos e opções:

- Responsável ausente:
 - Sim
 - Não
- Responsável: _____
- CPF: _____
- E-mail: _____
- Telefone: _____
- Relacionamento:
 - Proprietário
 - Inquilino
 - Funcionário
 - Parente

Fonte: Próprio autor

Também há uma verificação a qual, o usuário, só pode prosseguir com o documento de Dados da Vistoria, clicando no botão 'Salvar', caso todos os campos do formulário estejam devidamente preenchidos e marcados.

5.2.14 Danos Humanos

Nesta tela, é possível criar ou editar um documento de Danos Humanos. Assim como abordado na subseção acima, caso o documento de Danos Humanos já exista, a aplicação buscará os dados no banco de dados e preencherá os campos da tabela automaticamente. Caso contrário, os campos da tabela estarão com valor 0 (zero) para que se crie um novo documento de Danos Humanos.

Como esta é uma tela que contém uma tabela, que ocupa uma largura maior do que a disponível para visualização, foi utilizado o *widget HorizontalScrollView*, que permite a rolagem do conteúdo da tela, na horizontal.

Para que não haja erro ao enviar os dados da tabela, foi pensada uma solução para os campos vazios. O trecho de código do [método para retornar um número inteiro](#) exemplifica o trecho de código que retorna o valor 0 (zero) caso o campo de texto (representado pelo

widget de um *EditText*, que é um campo de entrada de texto) esteja vazio.

Figura 31 – Aplicativo PRODEC - Danos Humanos

	PARCIAL	FAT.
LACTENTES	0	0
CRIANÇAS/ADOLESCENTES	0	0
ADULTOS	0	0
IDOSOS	0	0
GESTANTES	0	0
PORT. DE NEC. ESPECIAIS	0	0
TOTAL	0	0

Fonte: Próprio autor

```

1 private int getNumber(EditText editText){
2
3     String num = editText.getText().toString();
4
5     if (num.isEmpty()){
6         return 0;
7     }
8     else {
9         return Integer.parseInt(num);
10    }
11 }

```

Método para retornar um número inteiro

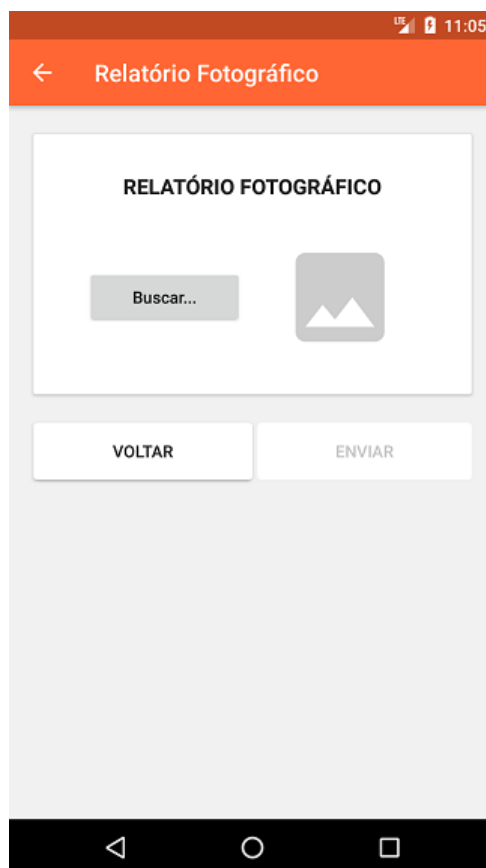
5.2.15 Relatório Fotográfico

Nesta tela é possível fazer o *upload* de um arquivo de imagem, seja da câmera ou da galeria do *smartphone*, para criação de um Relatório Fotográfico. O arquivo de imagem

é processado e enviado para o banco de dados.

Para minimizar a possibilidade de falha no envio, foi pensada uma solução que permite enviar apenas uma imagem por vez, pelo fato de arquivos de imagem pesarem, em média, 5 megabytes, dependendo da resolução da imagem. Sendo assim, o processo só será bem-sucedido caso a conexão com a internet não seja interrompida.

Figura 32 – Aplicativo PRODEC - Relatório Fotográfico



Fonte: Próprio autor

Diferente dos *webservices* que são realizados em telas anteriores, este necessita que a imagem seja compactada em *bytes* e convertida para o formato *String*, utilizando o *Base64* como criptografia. Desta maneira, a imagem pode ser enviada para o banco de dados.

O trecho de código de [codificação de uma imagem no formato Bitmap](#) demonstra esta mudança no *webservice* para o processo descrito acima.

```
1 ByteArrayOutputStream byteArrayOutputStream = new
    ByteArrayOutputStream();
2 bitmap.compress(Bitmap.CompressFormat.JPEG, 100,
    byteArrayOutputStream);
3 byte[] imageByteArray = byteArrayOutputStream.toByteArray();
```

```
4 final String encoded_image = Base64.encodeToString(imageByteArray  
    , Base64.DEFAULT);
```

Codificação de uma imagem no formato Bitmap

6 Conclusão

Este trabalho apresentou uma solução para a utilização da plataforma PRODEC em dispositivos móveis, através do desenvolvimento de uma aplicação para sistemas Android, facilitando a criação de novos documentos e relatórios para a Defesa Civil do Estado do Rio de Janeiro.

Para início do trabalho, foram feitos alguns encontros no DGDEC, diretamente com o Major Jorge Carvalho, criador do sistema do PRODEC, para orientação do fluxo do sistema e das funcionalidades que a aplicação deveria contemplar. Durante este processo, foi realizada toda a modelagem do sistema desenvolvido, através dos requisitos, casos de uso e diagramas, com o objetivo de documentar e identificar falhas no trabalho proposto, antes da sua devida implementação.

Por fim, a aplicação desenvolvida foi apresentada e entregue para o Major Jorge Carvalho, que por sua vez analisou e aprovou o trabalho, dizendo que gostaria de ver a aplicação inserida em seu sistema.

6.1 Trabalhos futuros

Existem muitas outras funcionalidades para dar continuidade à aplicação desenvolvida neste trabalho, para que se torne uma ferramenta mais robusta, facilitando ainda mais a criação de documentos e relatórios do cotidiano da Defesa Civil. Porém, antes de prosseguir com novas funcionalidades, devem ser seguidos dois passos cruciais.

Por se trata de uma aplicação desenvolvida para um órgão estadual, o primeiro passo para seguir com trabalhos futuros deve ser adicionar a aplicação aos bancos de dados do PRODEC. Dessa maneira, a plataforma e o aplicativo trabalharão, ambos, com a mesma base de dados.

Para que a aplicação desenvolvida seja utilizada, de fato, nos dispositivos Android, deve-se realizar, como segundo passo, uma bateria de testes em diversos *smartphones*, com diferentes versões de Android instalados, diferentes resolução de tela e os testes nas ruas, simulando os diferentes tipos de cenários do cotidiano, colhendo os dados, preenchendo-os no aplicativo, para que sejam enviados aos bancos de dados do PRODEC. Dessa forma, os usuários avaliariam a usabilidade e a aplicabilidade da solução proposta neste trabalho.

Seguindo estes dois passos, a aplicação terá uma forte base para que seja posta em prática, com grande possibilidade de atingir o sucesso.

Referências

- ADAIL. *Modelagem Relacional*. 2011. Disponível em: <<https://www.devmedia.com.br/modelagem-relacional/19614>>. Acesso em: 27 mar. 2019. Citado na página 49.
- CONSTANCIO, T. *Polícia Civil lança aplicativo que facilita registro de ocorrências*. 2016. Disponível em: <<http://www.rj.gov.br/web/seseg/exibeconteudo?article-id=2926152>>. Acesso em: 17 mar. 2018. Citado na página 27.
- CORDEIRO, F. *Gradle para Android*. 2015. Disponível em: <<https://www.androidpro.com.br/blog/android-studio/gradle/>>. Acesso em: 22 out. 2018. Citado na página 45.
- CUNHA, L. G. *Tema GitHub vs GitLab – Comparativo entre as duas plataformas*. 2018. Disponível em: <https://tableless.com.br/github_vs_gitlab/>. Acesso em: 30 ago. 2018. Citado na página 44.
- DUARTE, L. *Tudo sobre o Gradle – Android Studio*. 2017. Disponível em: <<http://www.luiztools.com.br/post/tudo-sobre-o-gradle-android-studio/>>. Acesso em: 22 out. 2018. Citado na página 45.
- EXTRA. *Polícia Civil lança aplicativo para registro de ocorrências via celular*. 2016. Disponível em: <<https://extra.globo.com/noticias/rio/policia-civil-lanca-aplicativo-para-registro-de-ocorrencias-via-celular-20001861.html>>. Acesso em: 17 mar. 2018. Citado na página 25.
- FERNANDES, A. *O que é API? Entenda de uma maneira simples*. 2018. Disponível em: <<https://vertigo.com.br/o-que-e-api-entenda-de-uma-maneira-simples/>>. Acesso em: 27 mar. 2019. Citado na página 51.
- JORNAL DO BRASIL. *Central 1746 completa 1 milhão de chamados com 75% de casos resolvidos*. 2017. Disponível em: <<http://www.jb.com.br/rio/noticias/2017/09/30/central-1746-completa-1-milhao-de-chamados-com-75-de-casos-resolvidos/>>. Acesso em: 05 jul. 2018. Citado na página 24.
- LACERDA, B. *O que é PHP? Para que serve?* 2017. Disponível em: <<http://phpdozeroaprofissional.net.br/o-que-e-php-para-que-serve/>>. Acesso em: 28 mar. 2019. Citado na página 51.
- LEITE, J. C. *Notas de aula de Engenharia de Software: 4. análise e especificação de requisitos*. 2000. Disponível em: <<https://www.dimap.ufrn.br/~jair/ES/c4.html>>. Acesso em: 09 fev. 2019. Citado na página 33.
- MAGNI, E. *Rio de Janeiro ganha prêmio de cidade inteligente do ano*. 2013. Disponível em: <<https://oglobo.globo.com/rio/rio-de-janeiro-ganha-premio-de-cidade-inteligente-do-ano-10843951>>. Acesso em: 09 fev. 2018. Citado na página 24.
- MOURA, M. *Polícia Civil do Rio de Janeiro lança aplicativo que facilita registro de ocorrências*. 2016. Disponível em: <<http://techapple.com.br/2016/08/policia-civil-do-rio-de-janeiro-que-facilita-registro-de-ocorrencias/>>. Acesso em: 21 fev. 2018. Citado na página 28.

- NENO, M. *Aplicativo permite enviar reclamações em tempo real para Prefeitura do Rio*. 2011. Disponível em: <<http://g1.globo.com/rio-de-janeiro/noticia/2011/09/aplicativo-permite-enviar-reclamacoes-em-tempo-real-para-prefeitura-do-rio.html>>. Acesso em: 13 mai. 2018. Citado na página 21.
- PREFEITURA DO RIO. *Cariocas aprovam Central de Atendimento ao Cidadão (1746)*. 2011. Disponível em: <<http://www.rio.rj.gov.br/web/cvl/exibeconteudo?id=2238562>>. Acesso em: 11 mar. 2018. Citado na página 21.
- RIBEIRO, L. *O que é UML e Diagramas de Caso de Uso: Introdução prática à uml*. 2012. Disponível em: <<https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>>. Acesso em: 25 mar. 2019. Citado na página 42.
- SCHMITZ, D. *Tudo que você queria saber sobre Git e GitHub, mas tinha vergonha de perguntar*. 2015. Disponível em: <<https://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar/>>. Acesso em: 29 ago. 2018. Citado na página 44.
- SOMMERVILLE, I. *Engenharia De Software - 9ª Ed. Cap. 4 (Seção 4.1)*. [S.l.]: Pearson, 2011. Citado 2 vezes nas páginas 31 e 32.
- SPÍNOLA, R. *Artigo Engenharia de Software - Introdução à Engenharia de Requisitos*. 2008. Disponível em: <<https://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-engenharia-de-requisitos/8034>>. Acesso em: 21 jan. 2019. Citado na página 31.

Apêndices

APÊNDICE A – API desenvolvida

A.1 Conexão

connect.php

```
1 <?php
2
3     $conn = mysqli_connect("localhost", "id6786933_marcellocamara", "prodec2019", "id6786933_prodec");
4
5     $conn->set_charset("utf8");
6
7 ?>
```

Fonte: Próprio autor

A.2 Login

login.php

```
1 <?php
2
3     if ($_SERVER['REQUEST_METHOD'] == 'POST') {
4
5         $email = $_POST['email'];
6         $senha = $_POST['senha'];
7
8         require_once 'connect.php';
9
10        $sql = "SELECT * FROM usuarios WHERE email='$email' ";
11
12        $response = mysqli_query($conn, $sql);
13
14        $result = array();
15
16        if ( mysqli_num_rows($response) === 1 ) {
17
18            //E-mail encontrado
19
20            $row = mysqli_fetch_assoc($response);
21
```

```
22     if ( password_verify($senha, $row['senha']) ) {
23
24         //E-mail encontrado e senha correta
25
26         $result['login'] = array();
27
28         $index['id'] = $row['id'];
29         $index['nome'] = $row['nome'];
30         $index['cpf'] = $row['cpf'];
31         $index['matricula'] = $row['matricula'];
32         $index['email'] = $row['email'];
33         $index['funcao'] = $row['funcao'];
34         $index['orgao'] = $row['orgao'];
35         $index['permissao'] = $row['permissao'];
36         $index['vistoriante'] = $row['vistoriante'];
37
38         array_push($result['login'], $index);
39
40         $result['sucesso'] = "1";
41         echo json_encode($result);
42
43     } else {
44
45         //E-mail encontrado mas senha incorreta
46
47         $result['sucesso'] = "0";
48         echo json_encode($result);
49
50     }
51 }
52 else{
53     //E-mail incorreto / inexistente
54
55     $result['sucesso'] = "0";
56     echo json_encode($result);
57 }
58 mysqli_close($conn);
59 }
60
61 ?>
```

A.3 Minha conta

minhaConta.php

```
1 <?php
2
3  if($_SERVER['REQUEST_METHOD'] == 'POST'){
4
5      $id = $_POST['id'];
6      $nome = $_POST['nome'];
7      $cpf = $_POST['cpf'];
8      $matricula = $_POST['matricula'];
9      $email = $_POST['email'];
10     $funcao = $_POST['funcao'];
11     $orgao = $_POST['orgao'];
12     $permissao = $_POST['permissao'];
13     $vistoriante = $_POST['vistoriante'];
14
15     require_once 'connect.php';
16
17     $sql = "UPDATE usuarios SET nome='$nome', cpf='$cpf',
18           matricula='$matricula', email='$email', funcao='$funcao',
19           orgao='$orgao', permissao='$permissao', vistoriante='
20           $vistoriante' WHERE id='$id' ";
21
22     if(mysqli_query($conn, $sql)) {
23
24         $result["sucesso"] = "1";
25         echo json_encode($result);
26
27     }else{
28         $result["sucesso"] = "0";
29         echo json_encode($result);
30     }
31     mysqli_close($conn);
32 }
33 else{
34     $result["sucesso"] = "0";
35     echo json_encode($result);
36 }
37 ?>
```

Fonte: Próprio autor

A.4 Trocar senha

trocarSenha.php

```
1 <?php
2
3     if ( $_SERVER['REQUEST_METHOD']=='POST' ) {
4
5         $id = $_POST['id'];
6         $senhaAntiga = $_POST['senhaAntiga'];
7         $senhaNova = $_POST['senhaNova'];
8
9         require_once 'connect.php';
10
11        $sql = "SELECT * FROM usuarios WHERE id='$id' ";
12
13        $response = mysqli_query($conn, $sql);
14
15        if ( mysqli_num_rows($response) === 1 ) {
16            //Encontrou o id do usuário
17            $row = mysqli_fetch_assoc($response);
18            if ( strcmp($senhaAntiga, $row['senha']) == 0 ) {
19                //Encontrou id e a senha confere
20                $sql = "UPDATE usuarios SET senha='$senhaNova'
21                    WHERE id='$id' ";
22                if(mysqli_query($conn, $sql)){
23                    $result["sucesso"] = "1";
24                    echo json_encode($result);
25                }
26                else{
27                    $result["sucesso"] = "0";
28                    echo json_encode($result);
29                }
30            } else {
31                //Encontrou o e-mail mas a senha não confere
32                $result['sucesso'] = "0";
33                echo json_encode($result);
34            }
35        }
36        mysqli_close($conn);
37    }
38 ?>
```

Fonte: Próprio autor

A.5 Criar Boletim de Ocorrência

criarBO.php

```
1 <?php
2
3     if($_SERVER['REQUEST_METHOD'] == 'POST'){
4
5         $data = $_POST['data'];
6         $hora = $_POST['hora'];
7         $solicitante = $_POST['solicitante'];
8         $telefone = $_POST['telefone'];
9         $endereco = $_POST['endereco'];
10        $numero = $_POST['numero'];
11        $complemento = $_POST['complemento'];
12        $bairro = $_POST['bairro'];
13        $ptreferencia = $_POST['ptreferencia'];
14        $divadm = $_POST['divadm'];
15        $responsavel = $_POST['responsavel'];
16        $outra = $_POST['outra'];
17        $autorBO = $_POST['autorBO'];
18        $autorBO_id = $_POST['autorBO_id'];
19
20        $cdmeteorologica = $_POST['cdmeteorologica'];
21        $ocorrenciacsqchuva = $_POST['ocorrenciacsqchuva'];
22        $proprietario = $_POST['proprietario'];
23        $emergencial = $_POST['emergencial'];
24        $ocorrencia = $_POST['ocorrencia'];
25        $outros = $_POST['outros'];
26
27        require_once 'connect.php';
28
29        $sql = "INSERT INTO bos (data, hora, cdmeteorologica,
            ocorrenciacsqchuva, solicitante, telefone, endereco,
            numero, complemento, bairro, ptreferencia, divadm,
            responsavel, proprietario, emergencial, ocorrencia,
            outros, outra, autorBO, autorBO_id) VALUES ('$data', '
            $hora', '$cdmeteorologica', '$ocorrenciacsqchuva', '
            $solicitante', '$telefone', '$endereco', '$numero', '
            $complemento', '$bairro', '$ptreferencia', '$divadm', '
            $responsavel', '$proprietario', '$emergencial', '
            $ocorrencia', '$outros', '$outra', '$autorBO', '
            $autorBO_id' ) ";
```

```
30     if(mysqli_query($conn, $sql)){
31
32         $result = array();
33         $result["sucesso"] = "1";
34         $result["idBO"] = mysqli_insert_id($conn);
35         echo json_encode($result);
36
37     }
38     else{
39
40         $result["sucesso"] = "0";
41         echo json_encode($result);
42
43     }
44     mysqli_close($conn);
45 }
46
47 ?>
```

Fonte: Próprio autor

A.6 Consultar Boletins de Ocorrência

consultarBOs.php

```
1 <?php
2
3     if ($_SERVER['REQUEST_METHOD'] == 'POST') {
4
5         require_once 'connect.php';
6
7         $sql = "SELECT * FROM bos";
8
9         $response = mysqli_query($conn, $sql);
10
11        $result = array();
12
13        if ( mysqli_num_rows($response) > 0 ) {
14
15            $result["sucesso"] = "1";
16            $result["bos"] = array();
17
18            while ($row = mysqli_fetch_assoc($response)) {
19
```



```
20         array_push($result["bos"], $row);
21
22     }
23
24     echo json_encode($result);
25
26 }else{
27
28     $result["sucesso"] = "0";
29     echo json_encode($result);
30
31 }
32
33 mysqli_close($conn);
34
35 }
36
37 ?>
```

Fonte: Próprio autor

A.7 Consultar Resumo do Boletim de Ocorrência

resumoBO.php

```
1 <?php
2
3     if ($_SERVER['REQUEST_METHOD'] == 'POST') {
4
5         $id = $_POST['id'];
6
7         require_once 'connect.php';
8
9         $sql = "SELECT * FROM bos WHERE id='$id' ";
10
11        $response = mysqli_query($conn, $sql);
12
13        $result = array();
14
15        if ( mysqli_num_rows($response) > 0 ) {
16
17            $result["sucesso"] = "1";
18            $result["bo"] = array();
19
```

```
20     while ($row = mysqli_fetch_assoc($response)) {
21
22         array_push($result["bo"], $row);
23
24     }
25
26     echo json_encode($result);
27
28 }else{
29
30     $result["sucesso"] = "0";
31     echo json_encode($result);
32
33 }
34
35 mysqli_close($conn);
36 }
37
38 ?>
```

Fonte: Próprio autor

A.8 Alterar Boletim de Ocorrência

updateBO.php

```
1 <?php
2
3     if($_SERVER['REQUEST_METHOD'] == 'POST'){
4
5         $id = $_POST['id'];
6         $data = $_POST['data'];
7         $hora = $_POST['hora'];
8         $solicitante = $_POST['solicitante'];
9         $telefone = $_POST['telefone'];
10        $endereco = $_POST['endereco'];
11        $numero = $_POST['numero'];
12        $complemento = $_POST['complemento'];
13        $bairro = $_POST['bairro'];
14        $ptreferencia = $_POST['ptreferencia'];
15        $divadm = $_POST['divadm'];
16        $responsavel = $_POST['responsavel'];
17        $outra = $_POST['outra'];
18        $autorBO = $_POST['autorBO'];
```

```
19     $autorBO_id = $_POST['autorBO_id'];
20
21     $cdmeteorologica = $_POST['cdmeteorologica'];
22     $ocorrenciacsqchuva = $_POST['ocorrenciacsqchuva'];
23     $proprietario = $_POST['proprietario'];
24     $emergencial = $_POST['emergencial'];
25     $ocorrencia = $_POST['ocorrencia'];
26     $outros = $_POST['outros'];
27
28     require_once 'connect.php';
29
30     $sql = "UPDATE bos SET data = '$data', hora = '$hora',
           cdmeteorologica = '$cdmeteorologica',
           ocorrenciacsqchuva = '$ocorrenciacsqchuva', solicitante
           = '$solicitante', telefone = '$telefone', endereco = '
           $endereco', numero = '$numero', complemento = '
           $complemento', bairro = '$bairro', ptreferencia = '
           $ptreferencia', divadm = '$divadm', responsavel = '
           $responsavel', proprietario = '$proprietario',
           emergencial = '$emergencial', ocorrencia = '$ocorrencia
           ', outros = '$outros', outra = '$outra', autorBO = '
           $autorBO', autorBO_id = '$autorBO_id' WHERE id='$id' ";
31
32     if(mysqli_query($conn, $sql)){
33
34         $result = array();
35         $result["sucesso"] = "1";
36         echo json_encode($result);
37
38     }
39     else{
40
41         $result["sucesso"] = "0";
42         echo json_encode($result);
43
44     }
45     mysqli_close($conn);
46 }
47
48 ?>
```

A.9 Criar Dados da Vistoria

criarDadosVistoria.php

```
1 <?php
2
3     if($_SERVER['REQUEST_METHOD'] == 'POST'){
4
5         $id = $_POST['id'];
6         $responsavel = $_POST['responsavel'];
7         $cpf = $_POST['cpf'];
8         $email = $_POST['email'];
9         $telefone = $_POST['telefone'];
10        $razaosocial = $_POST['razaosocial'];
11        $cargo = $_POST['cargo'];
12        $cnpj = $_POST['cnpj'];
13        $tel = $_POST['tel'];
14        $pavimentos = $_POST['pavimentos'];
15        $rg1 = $_POST['rg1'];
16        $rg2 = $_POST['rg2'];
17        $rg3 = $_POST['rg3'];
18        $rg4 = $_POST['rg4'];
19        $rg5 = $_POST['rg5'];
20        $rg6 = $_POST['rg6'];
21        $rg7 = $_POST['rg7'];
22        $rg8 = $_POST['rg8'];
23        $idbo = $_POST['idbo'];
24
25        require_once 'connect.php';
26
27        $sql = "INSERT INTO dadosvistoria (id, responsavel, cpf,
                email, telefone, razaosocial, cargo, cnpj, tel,
                pavimentos, rg1, rg2, rg3, rg4, rg5, rg6, rg7, rg8, id_BO
                ) VALUES ('$id', '$responsavel', '$cpf', '$email', '
                $telefone', '$razaosocial', '$cargo', '$cnpj', '$tel', '
                $pavimentos', '$rg1', '$rg2', '$rg3', '$rg4', '$rg5', '
                $rg6', '$rg7', '$rg8', '$idbo' ) ON DUPLICATE KEY UPDATE
                responsavel = '$responsavel', cpf = '$cpf', email = '
                $email', telefone = '$telefone', razaosocial = '
                $razaosocial', cargo = '$cargo', cnpj = '$cnpj', tel = '
                $tel', pavimentos = '$pavimentos', rg1 = '$rg1', rg2 = '
                $rg2', rg3 = '$rg3', rg4 = '$rg4', rg5 = '$rg5', rg6 = '
                $rg6', rg7 = '$rg7', rg8 = '$rg8', id_BO = '$idbo' ";
```

```
28     if(mysqli_query($conn, $sql)){
29
30         $result = array();
31         $result["sucesso"] = "1";
32         echo json_encode($result);
33
34     }
35     else{
36
37         $result["sucesso"] = "0";
38         echo json_encode($result);
39     }
40     mysqli_close($conn);
41 }
42
43 ?>
```

Fonte: Próprio autor

A.10 Consultar Dados da Vistoria

consultarDadosVistoria.php

```
1 <?php
2
3     if ($_SERVER['REQUEST_METHOD'] == 'POST') {
4
5         $id = $_POST['id'];
6
7         require_once 'connect.php';
8
9         $sql = "SELECT * FROM dadosvistoria WHERE id='$id' ";
10
11        $response = mysqli_query($conn, $sql);
12        $result = array();
13
14        if ( mysqli_num_rows($response) > 0 ) {
15
16            $result["sucesso"] = "1";
17            $result["dadosvistoria"] = array();
18
19            while ($row = mysqli_fetch_assoc($response)) {
20                array_push($result["dadosvistoria"], $row);
21            }
22        }
23    }
```

```
22         echo json_encode($result);
23
24     }else{
25
26         $result["sucesso"] = "0";
27         echo json_encode($result);
28
29     }
30     mysqli_close($conn);
31 }
32
33 ?>
```

Fonte: Próprio autor

A.11 Criar Danos Humanos

criarDanosHumanos.php

```
1 <?php
2
3     if($_SERVER['REQUEST_METHOD'] == 'POST'){
4
5         $id = $_POST['id'];
6         $parcial1 = $_POST['parcial1'];
7         $parcial2 = $_POST['parcial2'];
8         $parcial3 = $_POST['parcial3'];
9         $parcial4 = $_POST['parcial4'];
10        $parcial5 = $_POST['parcial5'];
11        $parcial6 = $_POST['parcial6'];
12        $fatal1 = $_POST['fatal1'];
13        $fatal2 = $_POST['fatal2'];
14        $fatal3 = $_POST['fatal3'];
15        $fatal4 = $_POST['fatal4'];
16        $fatal5 = $_POST['fatal5'];
17        $fatal6 = $_POST['fatal6'];
18        $desabrigados1 = $_POST['desabrigados1'];
19        $desabrigados2 = $_POST['desabrigados2'];
20        $desabrigados3 = $_POST['desabrigados3'];
21        $desabrigados4 = $_POST['desabrigados4'];
22        $desabrigados5 = $_POST['desabrigados5'];
23        $desabrigados6 = $_POST['desabrigados6'];
24        $desalojados1 = $_POST['desalojados1'];
25        $desalojados2 = $_POST['desalojados2'];
```

```
26     $desalojados3 = $_POST['desalojados3'];
27     $desalojados4 = $_POST['desalojados4'];
28     $desalojados5 = $_POST['desalojados5'];
29     $desalojados6 = $_POST['desalojados6'];
30     $outros1 = $_POST['outros1'];
31     $outros2 = $_POST['outros2'];
32     $outros3 = $_POST['outros3'];
33     $outros4 = $_POST['outros4'];
34     $outros5 = $_POST['outros5'];
35     $outros6 = $_POST['outros6'];
36     $idbo = $_POST['idbo'];
37
38     require_once 'connect.php';
39
40     $sql = "INSERT INTO danoshumanos (id, parcial1, parcial2,
        parcial3, parcial4, parcial5, parcial6, fatal1, fatal2,
        fatal3, fatal4, fatal5, fatal6, desabrigados1,
        desabrigados2, desabrigados3, desabrigados4,
        desabrigados5, desabrigados6, desalojados1, desalojados2,
        desalojados3, desalojados4, desalojados5, desalojados6,
        outros1, outros2, outros3, outros4, outros5, outros6,
        id_BO)
41     VALUES ('$id', '$parcial1', '$parcial2', '$parcial3', '
        $parcial4', '$parcial5', '$parcial6', '$fatal1', '$fatal2
        ', '$fatal3', '$fatal4', '$fatal5', '$fatal6', '
        $desabrigados1', '$desabrigados2', '$desabrigados3', '
        $desabrigados4', '$desabrigados5', '$desabrigados6', '
        $desalojados1', '$desalojados2', '$desalojados3', '
        $desalojados4', '$desalojados5', '$desalojados6', '
        $outros1', '$outros2', '$outros3', '$outros4', '$outros5
        ', '$outros6', '$idbo' )
42     ON DUPLICATE KEY UPDATE parcial1 = '$parcial1', parcial2 =
        '$parcial2', parcial3 = '$parcial3', parcial4 = '
        $parcial4', parcial5 = '$parcial5', parcial6 = '$parcial6
        ', fatal1 = '$fatal1', fatal2 = '$fatal2', fatal3 = '
        $fatal3', fatal4 = '$fatal4', fatal5 = '$fatal5', fatal6
        = '$fatal6', desabrigados1 = '$desabrigados1',
        desabrigados2 = '$desabrigados2', desabrigados3 = '
        $desabrigados3', desabrigados4 = '$desabrigados4',
        desabrigados5 = '$desabrigados5', desabrigados6 = '
        $desabrigados6', desalojados1 = '$desalojados1',
        desalojados2 = '$desalojados2', desalojados3 = '
```

```
43     $desalojados3', desalojados4 = '$desalojados4',
44     desalojados5 = '$desalojados5', desalojados6 = '
45     $desalojados6', outros1 = '$outros1', outros2 = '$outros2
46     ', outros3 = '$outros3', outros4 = '$outros4', outros5 =
47     '$outros5', outros6 = '$outros6', id_BO = '$idbo'
48 ";
49
50     if(mysqli_query($conn, $sql)){
51         $result = array();
52         $result["sucesso"] = "1";
53         echo json_encode($result);
54     }
55     else{
56         $result["sucesso"] = "0";
57         echo json_encode($result);
58     }
59     mysqli_close($conn);
60 }
61 }>
```

Fonte: Próprio autor

A.12 Consultar Danos Humanos

consultarDanosHumanos.php

```
1 <?php
2
3     $id = $_POST['id'];
4
5     if ($_SERVER['REQUEST_METHOD'] == 'POST') {
6
7         require_once 'connect.php';
8
9         $sql = "SELECT * FROM danoshumanos WHERE id='$id' ";
10
11         $response = mysqli_query($conn, $sql);
```



```
12
13     $result = array();
14
15     if ( mysqli_num_rows($response) > 0 ) {
16
17         $result["sucesso"] = "1";
18         $result["danoshumanos"] = array();
19
20         while ($row = mysqli_fetch_assoc($response)) {
21
22             array_push($result["danoshumanos"], $row);
23
24         }
25
26         echo json_encode($result);
27
28     }else{
29
30         $result["sucesso"] = "0";
31         echo json_encode($result);
32
33     }
34
35     mysqli_close($conn);
36
37 }
38
39 ?>
```

Fonte: Próprio autor

A.13 Criar Relatório Fotográfico

uploadFotos.php

```
1 <?php
2
3     if ($_SERVER['REQUEST_METHOD'] == 'POST') {
4
5         $id = $_POST['id'];
6         $photo = $_POST['photo'];
7
8         require_once 'connect.php';
9
```

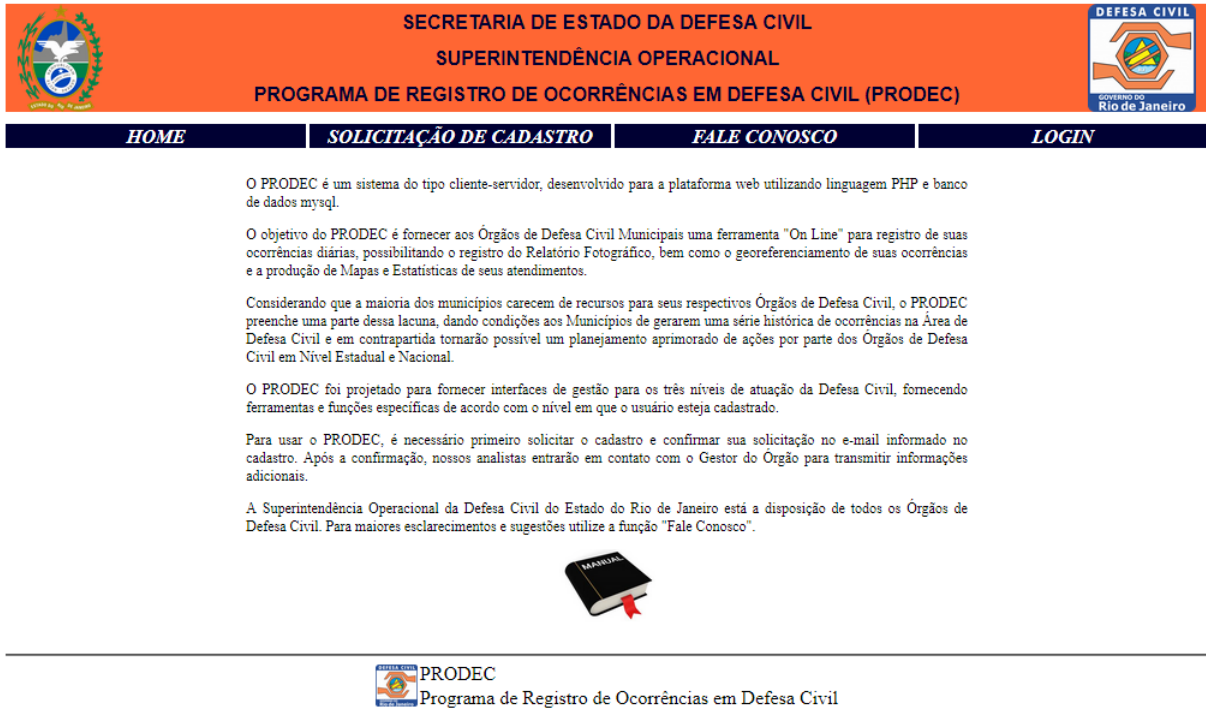
```
10     $path = "relatorios_fotograficos/$id.jpeg";
11
12     $finalPath = "http://192.168.31.90/prodec/".$path;
13
14     $sql = "INSERT INTO relatorios_fotograficos (id, url)
15           VALUES ('$id', '$finalPath') ";
16
17     $result = array();
18
19     if (mysqli_query($conn, $sql)) {
20
21         if (file_put_contents($path, base64_decode($photo))){
22
23             $result['sucesso'] = "1";
24             echo json_encode($result);
25
26         }else{
27
28             $result['sucesso'] = "0";
29             echo json_encode($result);
30         }
31
32     }else{
33
34         $result['sucesso'] = "-1";
35         echo json_encode($result);
36
37     }
38
39     mysqli_close($conn);
40
41 }
42
43 ?>
```

Fonte: Próprio autor

Anexos

ANEXO A – Plataforma do PRODEC

Figura 33 – Plataforma do PRODEC - Home



SECRETARIA DE ESTADO DA DEFESA CIVIL
SUPERINTENDÊNCIA OPERACIONAL
PROGRAMA DE REGISTRO DE OCORRÊNCIAS EM DEFESA CIVIL (PRODEC)

HOME **SOLICITAÇÃO DE CADASTRO** **FALE CONOSCO** **LOGIN**

O PRODEC é um sistema do tipo cliente-servidor, desenvolvido para a plataforma web utilizando linguagem PHP e banco de dados mysql.

O objetivo do PRODEC é fornecer aos Órgãos de Defesa Civil Municipais uma ferramenta "On Line" para registro de suas ocorrências diárias, possibilitando o registro do Relatório Fotográfico, bem como o georeferenciamento de suas ocorrências e a produção de Mapas e Estatísticas de seus atendimentos.


Considerando que a maioria dos municípios carecem de recursos para seus respectivos Órgãos de Defesa Civil, o PRODEC preenche uma parte dessa lacuna, dando condições aos Municípios de gerarem uma série histórica de ocorrências na Área de Defesa Civil e em contrapartida tornarão possível um planejamento aprimorado de ações por parte dos Órgãos de Defesa Civil em Nivel Estadual e Nacional.

O PRODEC foi projetado para fornecer interfaces de gestão para os três níveis de atuação da Defesa Civil, fornecendo ferramentas e funções específicas de acordo com o nível em que o usuário esteja cadastrado.

Para usar o PRODEC, é necessário primeiro solicitar o cadastro e confirmar sua solicitação no e-mail informado no cadastro. Após a confirmação, nossos analistas entrarão em contato com o Gestor do Órgão para transmitir informações adicionais.

A Superintendência Operacional da Defesa Civil do Estado do Rio de Janeiro está a disposição de todos os Órgãos de Defesa Civil. Para maiores esclarecimentos e sugestões utilize a função "Fale Conosco".

MANUAL

 **PRODEC**
 Programa de Registro de Ocorrências em Defesa Civil

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

Figura 34 – Plataforma do PRODEC - Cadastro de Adesão

The image shows a web interface for the PRODEC registration process. At the top, there is a header with the logo of the Secretariat of State for Civil Defense on the left and the 'DEFESA CIVIL' logo on the right. The central text reads: 'SECRETARIA DE ESTADO DA DEFESA CIVIL', 'SUPERINTENDÊNCIA OPERACIONAL', and 'PROGRAMA DE REGISTRO DE OCORRÊNCIAS EM DEFESA CIVIL (PRODEC)'. Below the header is a navigation bar with four buttons: 'HOME', 'SOLICITAÇÃO DE CADASTRO', 'FALE CONOSCO', and 'LOGIN'. The main content area is titled 'CADASTRO DE ADESÃO AO PRODEC'. It starts with a checkbox for 'Concordo com o termo e condições de uso.' followed by a link. Below this are several form fields: 'Nível de Atuação:*' (dropdown), 'Estrutura de Defesa Civil:*' (dropdown), 'UF:*' (dropdown), 'Município / Cidade:*' (dropdown with 'Selecione Primeiro a UF' text), 'Órgão de Defesa Civil:*' (text input), 'Titular da Pasta:*' (text input), 'Nome Completo:*' (text input), 'CPF:*' (text input), 'Cargo:*' (text input), 'E-mail:*' (text input), 'Telefone Celular:*' (text input with '(99) 99997-9999' placeholder), 'Telefone Fixo:*' (text input with '(99) 9999-9999' placeholder), and 'Digite o Código:*' (text input). A CAPTCHA image with the code '867c4e' is displayed below the code field. At the bottom of the form is an 'Enviar' button.

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

Figura 35 – Plataforma do PRODEC - Login

SECRETARIA DE ESTADO DA DEFESA CIVIL
SUPERINTENDÊNCIA OPERACIONAL
PROGRAMA DE REGISTRO DE OCORRÊNCIAS EM DEFESA CIVIL (PRODEC)

DEFESA CIVIL
GOVERNO DO
Rio de Janeiro

HOME SOLICITAÇÃO DE CADASTRO FALE CONOSCO LOGIN

ACESSO AO SISTEMA


Login:*

Senha:*

87d5b3

Digite o Código:*


[Recuperar senha](#)

 PRODEC
Programa de Registro de Ocorrências em Defesa Civil

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

Figura 37 – Plataforma do PRODEC - Tela inicial do sistema

BEM VINDO! MARCELLO DE PAULA CÂMARA

Página Inicial	Documentos Recebidos	Cemaden - RJ informa: Estágio de Vigilância!
Alterar Senha		
Cadastrar	 Documentos Resolvidos: 0	
Configurar	 Documentos Não Resolvidos - Prazo Não Expirado: 0	
Estatísticas	 Documentos Não Resolvidos - Prazo Expirado: 1	
Exibir Usuários	<hr/>	
Incluir	Documentos Emitidos	
Iniciar B.O.		
Mapa de	 Documentos Resolvidos: 0	
Minha Conta	 Documentos Não Resolvidos - Prazo Não Expirado: 0	
Nível de Alerta	 Documentos Não Resolvidos - Prazo Expirado: 1	
Organograma		
Pesquisar		
Relatórios		
Sair		

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

Figura 38 – Plataforma do PRODEC - Iniciar Boletim de Ocorrência

BEM VINDO! MARCELLO DE PAULA CÂMARA

DADOS DE SOLICITAÇÃO	
Página Inicial	DATA: 31/01/2019
Alterar Senha	HORA: 15:21
Cadastrar	CONDIÇÃO METEOROLÓGICA: <input type="text"/>
Configurar	OCORRÊNCIA EM CONSEQUENCIA DE CHUVA: Não <input type="text"/>
Estatísticas	SOLICITANTE: <input type="text"/>
Exibir Usuários	TELEFONE: (99) 99997-9999
Incluir Ocorrência	ENDEREÇO: <input type="text"/>
Iniciar B.O.	Nº <input type="text"/>
Mapa de	BAIRRO: <input type="text"/>
Minha Conta	COMPLEMENTO: <input type="text"/>
Nível de Alerta	PONTO DE REFERÊNCIA: <input type="text"/>
Organograma	DIV ADM: Distrito AP URG etc
Pesquisar	RESPONSÁVEL: <input type="text"/>
Relatórios	
Sair	
<input type="radio"/> PROPRIETÁRIO <input type="radio"/> INQUILINO <input type="radio"/> VIZINHO <input type="radio"/> PARENTE <input type="radio"/> OUTROS <input type="radio"/>	
<input type="radio"/> EMERGENCIAL <input type="radio"/> PREVENTIVA <input type="radio"/>	
OCORRÊNCIAS	
AMEAÇA	
AMEAÇA DE ALAGAMENTO	<input type="radio"/>
AMEAÇA DE DESABAMENTO DE MARQUISE	<input type="radio"/>
AMEAÇA DE DESABAMENTO DE MURO	<input type="radio"/>
AMEAÇA DE DESLIZAMENTO DE BARREIRA	<input type="radio"/>
AMEAÇA DE ENXURRADA	<input type="radio"/>
AMEAÇA DE INUNDAÇÃO	<input type="radio"/>
AMEAÇA DE QUEDA DE ÁRVORE	<input type="radio"/>
AMEAÇA DE QUEDA DE CAIXA D'ÁGUA	<input type="radio"/>
AMEAÇA DE ROLAMENTO DE PEDRA	<input type="radio"/>
AMEAÇA DE DESPRENDIMENTO DE REBOCO	<input type="radio"/>
IMÓVEL COM RACHADURA	<input type="radio"/>
DESABAMENTO	
DESABAMENTO DE IMÓVEL	<input type="radio"/>
DESABAMENTO DE MARQUISE	<input type="radio"/>
DESABAMENTO DE MURO	<input type="radio"/>
DESABAMENTO DE REBOCO	<input type="radio"/>
DESLIZAMENTO	
DESLIZAMENTO DE BARREIRA	<input type="radio"/>
DESLIZAMENTO DE ENCOSTA	<input type="radio"/>
INCÊNDIO	
INCÊNDIO FLORESTAL EM ÁREAS NÃO PROTEGIDAS	<input type="radio"/>
INCÊNDIO FLORESTAL EM PARQUES, APA OU APP	<input type="radio"/>
INCÊNDIO URBANO EM AGLOMERADOS RESIDENCIAIS	<input type="radio"/>
INCÊNDIO URBANO EM PLANTAS INDUSTRIAIS E DEPÓSITOS	<input type="radio"/>
OUTROS	
ACIDENTE COM PP	<input type="radio"/>
ALAGAMENTO	<input type="radio"/>
CAPTURA DE ABELHA	<input type="radio"/>
CONSTRUÇÃO IRREGULAR	<input type="radio"/>
ENXURRADA	<input type="radio"/>
INFILTRAÇÃO	<input type="radio"/>
INUNDAÇÃO	<input type="radio"/>
OCUPAÇÃO IRREGULAR	<input type="radio"/>
OUTRA OCORRÊNCIA	<input type="radio"/>
PODA / CORTE DE ÁRVORE	<input type="radio"/>
PREVENÇÃO PARA EVENTOS	<input type="radio"/>
QUEDA DE ÁRVORE	<input type="radio"/>
QUEDA DE CAIXA D'ÁGUA	<input type="radio"/>
ROLAMENTO DE PEDRA	<input type="radio"/>
VAZAMENTO DE GÁS	<input type="radio"/>
VISTORIA TÉCNICA	<input type="radio"/>
VISTORIA TÉCNICA EM ÁRVORE	<input type="radio"/>
OUTRO TIPO DE OCORRÊNCIA: <input type="text"/>	

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

Figura 39 – Plataforma do PRODEC - Minha conta

BEM VINDO! MARCELLO DE PAULA CÂMARA

Página Inicial	<div style="text-align: center;">MINHA CONTA</div> <p>Nome: <input type="text" value="MARCELLO DE PAULA CÂMARA"/></p> <p>Matrícula: <input type="text" value="213060089"/></p> <p>CPF: <input type="text"/></p> <p>E-mail: <input type="text" value="marcellocamara@hotmail.com"/></p> <p>Função: <input type="text" value="Desenvolvedor"/></p> <p>Órgão: <input type="text" value="DGDEC"/></p> <p>Permissão: <input type="text" value="Master"/></p> <p>Vistoriante: <input type="text" value="Sim"/></p> <p style="text-align: center;"><input type="button" value="Editar"/></p>
Alterar Senha	
Cadastrar	
Configurar	
Estatísticas	
Exibir Usuários	
Incluir	
Iniciar B.O.	
Mapa de	
Minha Conta	
Nível de Alerta	
Organograma	
Pesquisar	
Relatórios	
Sair	

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

Figura 40 – Plataforma do PRODEC - Alterar senha

BEM VINDO! MARCELLO DE PAULA CÂMARA

Página Inicial	<div style="text-align: center;">ALTERAR SENHA</div> <p>Senha Antiga: <input type="text"/></p> <p>Nova Senha: <input type="text"/></p> <p>Repita Nova Senha: <input type="text"/></p> <p style="text-align: center;"><input type="button" value="Enviar"/></p>
Alterar Senha	
Cadastrar	
Configurar	
Estatísticas	
Exibir Usuários	
Incluir	
Iniciar B.O.	
Mapa de	
Minha Conta	
Nível de Alerta	
Organograma	
Pesquisar	
Relatórios	
Sair	

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

Figura 41 – Plataforma do PRODEC - Documentos

BEM VINDO! MARCELLO DE PAULA CÂMARA

DOCUMENTOS RECEBIDOS E NÃO RESOLVIDOS - PRAZO EXPIRADO								
Total de Documentos Encontrados = 1								
Entrada / Emissão	Documento	Origem / Destino	Assunto	Síntese	Número	Protocolo	Prazo	
20/08/2018	Solicitação de Vistoria	DGDEC	SOLICITAÇÃO DE VISTORIA	Endereço: AVENIDA COELHO DA ROCHA, 1426, 11111, 12, ROCHA SOBRINHO - Referência: N/H - Solicitante: PREFEITO	00001/2018	00001/2018	04/09/2018	

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

Figura 42 – Plataforma do PRODEC - Identificação do Boletim de Ocorrência

BEM VINDO! MARCELLO DE PAULA CÂMARA

IDENTIFICAÇÃO DO BOLETIM DE OCORRÊNCIA						
Data da Solicitação	Horário	Log	B.O.	Protocolo	Protocolo Principal	
20/08/2018	14:36:00		00001/2018	00001/2018	NSA	
Endereço:						
AVENIDA COELHO DA ROCHA, 1426, 11111, 12, Bairro: ROCHA SOBRINHO						
Ocorrência:						
DESLIZAMENTO DE ENCOSTA / Cobrade: 1.1.3.2.1						
Solicitação		Dados da Vistoria		Danos Humanos		Detalhes da Vistoria
Providências		Despachos		Acionamentos		Relatório Fotográfico
Uploads de Arquivos		B.O. Inicial		B.O. Completo		Cópia Autêntica
Afetados Diretos		Mapa da Vistoria		Visualizar B.O.		Movimentações
Risco Geológico			Risco Hidrológico			

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

Figura 43 – Plataforma do PRODEC - Dados da Vistoria

BEM VINDO! MARCELLO DE PAULA CÂMARA

Página Inicial		DADOS DA VISTORIA									
Alterar Senha	IDENTIFICAÇÃO Responsável Ausente? <input type="radio"/> Sim <input checked="" type="radio"/> Não										
Cadastrar Usuário	RESPONSÁVEL:					CPF:					
Configurar Página	EMAIL:					TELEFONE:					
Estatísticas	PROPRIETÁRIO <input type="radio"/>		INQUILINO <input type="radio"/>		FUNCIONÁRIO <input type="radio"/>		PARENTE <input type="radio"/>		OUTROS <input type="radio"/>		
Exibir Usuários	RAZÃO SOCIAL:					CARGO:					
Incluir Ocorrência	CNPJ:					TEL:					
Iniciar B.O.	CLASSIFICAÇÃO DA EDIFICAÇÃO										
Mapa de Vistorias	RESIDENCIAL <input type="radio"/>		COMERCIAL <input type="radio"/>		INDUSTRIAL <input type="radio"/>		MISTO <input type="radio"/>		REUNIÃO DE PÚBLICO <input type="radio"/>		ESCOLAR <input type="radio"/>
Minha Conta	Nº DE PAVIMENTOS:		UNIFAMILIAR <input type="radio"/>		MULTIFAMILIAR <input type="radio"/>		USO ESPECIAL <input type="radio"/>				
Nível de Alerta	ALVENARIA <input type="radio"/>		CONCRETO <input type="radio"/>		MADEIRA <input type="radio"/>		METÁLICA <input type="radio"/>				OUTROS <input type="radio"/>
Organograma	DANOS NA EDIFICAÇÃO										
Pesquisar	NENHUM DANO <input type="radio"/>		DANOS RECUPERÁVEIS <input type="radio"/>		DESTRUIÇÃO PARCIAL <input type="radio"/>		DESTRUIÇÃO TOTAL <input type="radio"/>				
Relatórios	ÁREA/PROPRIEDADE										
Sair	PARTICULAR <input type="radio"/>		PÚBLICA <input type="radio"/>		RESERVA FLORESTAL <input type="radio"/>		OUTRAS <input type="radio"/>				
	OCUPAÇÃO										
	DESORDENADA <input type="radio"/>		POLO INDUSTRIAL <input type="radio"/>		URBANIZADA <input type="radio"/>		RURAL <input type="radio"/>				
	<input type="button" value="Enviar"/> <input type="button" value="Voltar"/>										

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

Figura 44 – Plataforma do PRODEC - Danos Humanos

BEM VINDO! MARCELLO DE PAULA CÂMARA

Página Inicial	DANOS HUMANOS	VITIMAS			AFETADOS			
		PARCIAL	FATAL	TOTAL	DESABRIGADOS	DESALOJADOS	OUTROS	TOTAL
Alterar Senha	LACTENTES	0	0	0	0	0	0	0
Cadastrar Usuário	CRIANÇAS/ADOLESCENTES	0	0	0	0	0	0	0
Configurar Página	ADULTOS	0	0	0	0	0	0	0
Estatísticas	IDOSOS	0	0	0	0	0	0	0
Exibir Usuários	GESTANTES	0	0	0	0	0	0	0
Incluir Ocorrência	PORT. DE NECESSIDADES ESPECIAIS	0	0	0	0	0	0	0
Iniciar B.O.	TOTAL	0	0	0	0	0	0	0
Mapa de Vistorias	<input type="button" value="Enviar"/> <input type="button" value="Voltar"/>							
Minha Conta								
Nível de Alerta								
Organograma								
Pesquisar								
Relatórios								
Sair								

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

Figura 45 – Plataforma do PRODEC - Relatório Fotográfico

BEM VINDO! MARCELLO DE PAULA CÂMARA

Página Inicial	CADASTRO DE IMAGEM		
Alterar Senha	<input type="radio"/> SIM	<input checked="" type="radio"/> NÃO	<input type="button" value="Browse..."/> No files selected.
Cadastrar	<input type="button" value="Enviar"/> <input type="button" value="Voltar"/>		
Configurar Página			
Estatísticas			
Exibir Usuários			
Incluir Ocorrência			
Iniciar B.O.			
Mapa de Vistorias			
Minha Conta			
Nível de Alerta			
Organograma			
Pesquisar			
Relatórios			
Sair			

Fonte: <<http://www.prodec.defesacivil.rj.gov.br/>>

ANEXO B – Documento de Boletim de Ocorrência

CABEÇALHO DO ÓRGÃO

BOLETIM DE OCORRÊNCIA Nº 00908/2017

DADOS DA SOLICITAÇÃO						
PROTOCOLO GERAL Nº 00559/2017			DATA: 27/10/2017		HORA: 14:02:00	
CONDIÇÃO METEOROLÓGICA:						
SOLICITANTE: DAVID VENÂNCIO				TELEFONE: (99) 9999-99999		
ENDEREÇO: ESTRADA DO TINGUÁ				Nº 7416		
BAIRRO: TINGUÁ				COMPLEMENTO: N/H		
PONTO DE REFERÊNCIA: N/H				URG: URG IX		
PROPRIETÁRIO	INQUILINO	VIZINHO	PARENTE	OUTROS	(X)	
EMERGENCIAL	PREVENTIVA			(X)		

OCORRÊNCIA			
AMEAÇA			
AMEAÇA DE ALAGAMENTO	<input type="checkbox"/>	AMEAÇA DE DESABAMENTO DE IMÓVEL	<input type="checkbox"/>
AMEAÇA DE DESABAMENTO DE MARQUISE	<input type="checkbox"/>	AMEAÇA DE DESABAMENTO DE MURO	<input type="checkbox"/>
AMEAÇA DE DESLIZAMENTO DE BARREIRA	<input type="checkbox"/>	AMEAÇA DE ENXURRADA	<input type="checkbox"/>
AMEAÇA DE INUNDAÇÃO	<input type="checkbox"/>	AMEAÇA DE QUEDA DE ÁRVORE	<input type="checkbox"/>
AMEAÇA DE QUEDA DE CAIXA D'ÁGUA	<input type="checkbox"/>	AMEAÇA DE ROLAMENTO DE PEDRA	<input type="checkbox"/>
AMEAÇA DE DESPRENDIMENTO DE REBOCO	<input type="checkbox"/>	IMÓVEL COM RACHADURA	<input type="checkbox"/>
DESABAMENTO			
DESABAMENTO DE IMÓVEL	<input type="checkbox"/>	DESABAMENTO DE MARQUISE	<input type="checkbox"/>
DESABAMENTO DE MURO	<input type="checkbox"/>	DESABAMENTO DE REBOCO	<input type="checkbox"/>
DESLIZAMENTO			
DESLIZAMENTO DE BARREIRA	<input type="checkbox"/>	DESLIZAMENTO DE ENCOSTA	<input type="checkbox"/>
INCÊNDIO			
INCÊNDIO FLORESTAL EM ÁREAS NÃO PROTEGIDAS	<input type="checkbox"/>	INCÊNDIO FLORESTAL EM PARQUES, APA OU APP	<input type="checkbox"/>
INCÊNDIO URBANO EM AGLOMERADOS RESIDENCIAIS	<input type="checkbox"/>	INCÊNDIO URBANO EM PLANTAS INDUSTRIAIS E DEPÓSITOS	<input type="checkbox"/>
OUTROS			
ACIDENTE COM PP	<input type="checkbox"/>	ALAGAMENTO	<input type="checkbox"/>
CAPTURE DE ABELHA	<input type="checkbox"/>	CONSTRUÇÃO IRREGULAR	<input type="checkbox"/>
ENXURRADA	<input type="checkbox"/>	INFILTRAÇÃO	<input type="checkbox"/>
INUNDAÇÃO	<input type="checkbox"/>	OCUPAÇÃO IRREGULAR	<input type="checkbox"/>
OUTRA OCORRÊNCIA	<input type="checkbox"/>	PODA / CORTE DE ÁRVORE	<input type="checkbox"/>
PREVENÇÃO PARA EVENTOS	<input type="checkbox"/>	QUEDA DE ÁRVORE	<input type="checkbox"/>
QUEDA DE CAIXA D'ÁGUA	<input type="checkbox"/>	ROLAMENTO DE PEDRA	<input type="checkbox"/>
VISTORIA TÉCNICA	<input checked="" type="checkbox"/>	VISTORIA TÉCNICA EM ÁRVORE	<input type="checkbox"/>
OUTRO TIPO DE OCORRÊNCIA:			

RODAPÉ DO ÓRGÃO

ANEXO C – Documento de Dados da Vistoria e Danos Humanos

CABEÇALHO DO ÓRGÃO

DADOS DA VISTORIA	
IDENTIFICAÇÃO	
RESPONSÁVEL:	CPF:
EMAIL:	TEL:
PROPRIETÁRIO () INQUILINO () FUNCIONÁRIO () PARENTE () OUTROS ()	
RAZÃO SOCIAL:	CARGO:
CNPJ:	TEL:

CLASSIFICAÇÃO DA EDIFICAÇÃO					
RESIDENCIAL ()	COMERCIAL ()	INDUSTRIAL ()	MISTO ()	REUN. PÚBLICO ()	()
Nº DE PAV:	UNIFAMILIAR ()	MULTIFAMILIAR ()	USO ESPECIAL ()	()	()
ALVENARIA ()	CONCRETO ()	MADEIRA ()	METÁLICA ()	OUTROS ()	()

DANOS NA EDIFICAÇÃO					
NENHUM DANO ()	DANO RECUPERÁVEL ()	DESTRUIÇÃO PARCIAL ()	DESTRUIÇÃO TOTAL ()	()	()

ÁREA / PROPRIEDADE					
PARTICULAR ()	RESERVA FLORESTAL ()	PÚBLICA ()	OUTRAS ()	()	()

OCUPAÇÃO					
URBANIZADA ()	POLO INDUSTRIAL ()	DESORDENADA ()	RURAL ()	()	()

DANOS HUMANOS	VÍTIMAS			AFETADOS			
	PARCIAL	FATAL	TOTAL	DESAB.	DESAL.	OUTROS.	TOTAL
LACTENTES							
CRIANÇAS/ADOLESCENTES							
ADULTOS							
IDOSOS							
GESTANTES							
PORT. DE NECESSIDADES ESPECIAIS							
TOTAL							

RODAPÉ DO ÓRGÃO