

# **UNIVERSIDADE VIRTUAL DO ESTADO DE SÃO PAULO**

Luiz Carlos da Silva  
Marcos Vinicius Navarro  
Mario Benassi Junior  
Sidnei Lopes Ribeiro  
Wellington Wesley da Silva Leite

**Aplicativo de Colaboração Coletiva Socioambiental para *Smartphone* com  
Sistema Operacional *Android***

Araras - SP  
2021

# **UNIVERSIDADE VIRTUAL DO ESTADO DE SÃO PAULO**

## **Aplicativo de Colaboração Coletiva Socioambiental para *Smartphone* com Sistema Operacional *Android***

Relatório Técnico-Científico apresentado na disciplina de Projeto Integrador para o curso de Engenharia de Computação da Universidade Virtual do Estado de São Paulo (UNIVESP).

Vídeo do projeto integrador  
<https://youtu.be/yS4jZP5J9oo>

Araras - SP  
2021

BENASSI Junior, Mario; LEITE, Wellington Wesley da Silva; NAVARRO, Marcos Vinicius; RIBEIRO, Sidnei Lopes; SILVA, Luiz Carlos da. **Aplicativo de Colaboração Coletiva Socioambiental para Smartphone com Sistema Operacional Android.** 63 f. Relatório Técnico-Científico. Curso de Engenharia de Computação – Universidade Virtual do Estado de São Paulo. Tutor: Ms. Paulo César de Lira. Polo Araras, 2021.

## RESUMO

Devido às consequências da crise econômica causada pela pandemia de covid-19 muitas pessoas ficaram desempregadas, perderam suas rendas e até mesmo tiveram que buscar moradias alternativas. Em decorrência disso, há pessoas que buscam as periferias urbanas para se abrigarem em áreas de proteção ambiental, as quais oferecem riscos a seus ocupantes em épocas do ano com fortes chuvas, pois geralmente estas áreas são encostas de morros ou mata ciliar de corpos d'água que sofrerão com as enchentes, desabrigando tais populações. Com isso em mente, o objetivo da pesquisa foi proporcionar a voluntários digitais localizar e fotografar ocupações por população sem moradia de áreas de risco em Piracicaba para auxiliar o Poder Pública a agilizar ações em favor dessas pessoas e do meio ambiente, antes que eventos extremos como enchentes as prejudiquem ainda mais. Como metodologia utilizou-se pesquisa qualitativa, prospecção de ideias via Brainstorm e Design Thinking e revisão de literatura para definir as ações a serem tomadas, que foram o uso do *crowdsourcing* e a prototipagem de um Aplicativo Android para agilizar a comunicação com o Poder Público. Os resultados obtidos foram a construção parcial do aplicativo de crowdsourcing com suas relevâncias social e acadêmica e, como considerações finais, a união do grupo foi essencial para alcançar esta etapa e espera-se que o aplicativo aqui esboçado seja base para outros na comunidade da Univesp.

**PALAVRAS-CHAVE:** Controle Ativo; *Crowdsourcing*; Desigualdade Socioambiental; Direito do Meio Ambiente

## **LISTA DE ILUSTRAÇÕES**

|   |    |
|---|----|
| <b>FIGURA 1</b> – Definição do Problema .....   | 9  |
| <b>FIGURA 2</b> – Bacia do Corumbataí e sua Área de Proteção Ambiental (APA) .....                        | 14 |
| <b>FIGURA 3</b> – Pirâmide de Elementos de Valor .....  | 18 |
| <b>FIGURA 4</b> – Área entre os bairros Bosques dos Lenheiros e Jardim Gilda .....                        | 22 |
| <b>FIGURA 5</b> – Área do projeto, área verde entre os bairros Jardim Gilda e Bosques dos Lenheiros ..... | 23 |
| <b>FIGURA 6</b> – Áreas verdes em foto antiga do <i>Google Maps</i> .....                                 | 23 |
| <b>FIGURA 7</b> – Ocupação da Área 1 .....  | 24 |
| <b>FIGURA 8</b> – Ocupação da Área 2 .....  | 24 |
| <b>FIGURA 9</b> – CRAS de atendimento na região .....   | 25 |
| <b>FIGURA 10</b> – Localização do CRAS Mário Dedini .....   | 25 |
| <b>FIGURA 11</b> – MÓDULO DE Localização pronto e operacional .....                                       | 27 |
| <b>FIGURA 12</b> – MÓDULO DE Imagens pronto e operacional .....   | 29 |
| <b>FIGURA 13</b> – MÓDULO DE E-mail pronto e operacional .....  | 30 |
| <b>FIGURA 14</b> – Tela inicial do aplicativo completo .....  | 31 |

## SUMÁRIO

|  |           |
|--|-----------|
| <b>1. INTRODUÇÃO</b>   | <b>6</b>  |
| <b>2. DESENVOLVIMENTO</b>  | <b>9</b>  |
| 2.1 OBJETIVOS.....   | 10        |
| 2.2. JUSTIFICATIVA E DELIMITAÇÃO DO PROBLEMA.....  | 10        |
| 2.4. METODOLOGIA.....  | 21        |
| <b>FUNDAÇÃO GETÚLIO VARGAS (FGV). BRASIL TEM DOIS DISPOSITIVOS DIGITAIS<br/>POR HABITANTE, REVELA PESQUISA DA FGV. PUBLICADO EM: 21 MAI. 2021.<br/>DISPONÍVEL EM: <a href="https://portal.fgv.br/noticias/brasil-tem-dois-dispositivos-digitais-habiente-revela-pesquisa-fgv/">HTTPS://PORTAL.FGV.BR/NOTICIAS/BRASIL-TEM-DOIS-DISPOSITIVOS-DIGITAIS-HABITANTE-REVELA-PESQUISA-FGV/</a>. ACESSO EM: 11 OUT. 2021.</b> | <b>33</b> |
| <b>APÊNDICES</b>   | <b>36</b> |
| APÊNDICE A – CÓDIGOS DO MÓDULO DE LOCALIZAÇÃO.....   | 36        |
| APÊNDICE B – CÓDIGOS DO MÓDULO DE IMAGEM.....  | 42        |

## 1. INTRODUÇÃO

A água, como recurso hídrico, é importantíssima, porém é escassa e até já foi o motivo desencadeador no ano de 1967, no Oriente Médio, da Guerra dos Seis Dias entre Israel e seus vizinhos árabes, que queriam desviar as nascentes do rio Jordão para privar o país judeu de recurso hídrico<sup>1</sup>. Voltando ao presente, em meados de 2021 ocorreu mais uma situação de escassez hídrica no Estado de São Paulo. Por exemplo, em Ribeirão Preto um estudo detectou que o aquífero Guarani já baixou 120 metros em 71 anos<sup>2</sup> e em Rio Claro foi estabelecida uma multa de R\$ 180,00 a quem desperdiçasse água do abastecimento público fornecida pelo Departamento Autônomo de Água e Esgoto (DAAE)<sup>3</sup>. Dessa forma, percebe-se que o gerenciamento do recurso hídrico é um grande desafio no Estado de São Paulo, pois as reservas subterrâneas estão em depleção e as superficiais apresentam-se poluídas e/ou contaminadas por vários tipos de uso na maioria dos rios do Estado, além da grande diminuição da vazão e do nível em épocas de estiagem.

Como moradores da região, sabe-se que o caudaloso e importante rio Piracicaba está tão poluído e contaminado que há décadas não compensa tanto econômica quanto sanitariamente tratar suas águas para abastecer a cidade. Como o município de Piracicaba não tem água subterrânea suficiente para sua população e o rio Piracicaba está comprometido para abastecimento público, é mais barato e seguro à população tratar o esgoto das cidades da bacia do rio Corumbataí, cuja maior cidade é Rio Claro, do que investir na despoluição do rio Piracicaba. Guardadas as devidas proporções, o que ocorre com Piracicaba é uma situação semelhante à de Israel na época da Guerra dos Seis Dias, pois o município depende das águas da bacia de um único rio que está a montante, onde já há uma população utilizando tal recurso hídrico.

E, como em todo o Brasil, o Estado de São Paulo apresenta problemas socioeconômicos e socioambientais decorrentes das intervenções equivocadas de sua população, apesar da existência de uma legislação ambiental avançada, porém desrespeitada por vários motivos, tais como a falta de fiscalização e ação adequadas por parte do Poder

1 WOLF, Aaron T. *Hydropolitics Along the Jordan River: Scarce water and its impact on the Arab-Israeli conflict*. UNUP, Tokyo, 1995, p. 50-54.

2 Nível do Aquífero Guarani em Ribeirão Preto, SP, cai 120 metros nos últimos 71 anos, diz estudo. Disponível em: <https://g1.globo.com/sp/ribeirao-preto-franca/noticia/2021/09/19/nivel-do-aquifero-guarani-em-ribeirao-preto-sp-cai-120-metros-nos-ultimos-71-anos-diz-estudo.ghtml>. Acesso em: 9 out. 2021.

3 Com a estiagem, Rio Claro vai multar morador que desperdiçar água em R\$ 180; Disponível em: <https://g1.globo.com/sp/sao-carlos-regiao/noticia/2021/09/17/com-a-estiagem-rio-claro-vai-multar-morador-que-desperdicar-agua-em-r-180-veja-as-regras.ghtml>. Acesso em: 9 out. 2021.

Público. Por exemplo, em praticamente todas as cidades de áreas densamente povoadas do país, o Poder Público “fecha os olhos” para ocupação de áreas de risco como colinas declivosas e margens de rios por pessoas desprovidas de condições financeiras para comprar ou alugar habitações em lugares seguros.

Mas, por que as “áreas de risco” como colinas e margens de rios e lagos devem ser preservadas com vegetação florestal? Por uma série de motivos, que todos aprendem na escola, e os fundamentais são: a vegetação protege contra a erosão dos solos; diminui a força do escoamento superficial (enxurrada) durante as chuvas, ajuda a acumular água no subsolo, regulariza a vazão durante o ano, ou seja, a água será liberada no momento oportuno para os rios, diminuindo sua vazão durante a chuva e aumentando no período de estiagem; regula as temperaturas do microclima local, pois diminui a amplitude térmica diária nas suas proximidades, com temperaturas mínimas maiores e temperaturas máximas menores, além de muitos outros benefícios que não estão entre os objetivos desta pesquisa.

Na região de moradia dos integrantes do grupo, um integrante visitou uma ocupação de área de risco para entregar cestas básicas a uma família que habitava um barraco de madeira recém erguido no bairro Jardim Gilda, na periferia do município de Piracicaba. No local constatou-se uma grande concentração de construções precárias, apresentando uma alta densidade demográfica, onde o distanciamento social era impraticável, devido à proximidade das construções e à alta circulação de pessoas sem máscaras.

À frente de diversos barracos percebeu-se que uma atividade importante de geração de renda era a coleta de material reciclável e a maior queixa dos moradores era a falta de renda para pagar aluguel ou financiar uma moradia. Após esta constatação, um dos integrantes do grupo iniciou os contatos com a instituição social da região e com os Centros de Referência de Assistência Social (CRAS) da Prefeitura Municipal de Piracicaba para levantar mais dados sobre os fatos.

Relembrando, o tema da Univesp para esta pesquisa intitula-se “*ferramentas computacionais para apoio à sociedade em relação a doenças contagiosas, pandemias, endemias, epidemias e catástrofes*”. Portanto, todo o exposto até aqui buscou adequar-se ao tema da pesquisa, principalmente quanto aos efeitos colaterais da Pandemia Covid-19 a estes cidadãos sem renda, atingidos pelo desemprego intensificado pela pandemia covid-19, que ocupam moradias precárias, muito próximas umas das outras, instaladas em área de risco de ocorrência de catástrofes causadas por evento climáticos severos como enchentes, vendavais e chuvas de granizo, que podem prejudicar-lhes ainda mais.

Como motivações para a escolha do tema, de início a ideia foi a criação de uma ferramenta de registro de mapas para um aplicativo em *smartphone* para atender ao tema de utilização em pandemia. No entanto, após vivenciar a experiência de visitar uma área habitada

por pessoas necessitadas de auxílio para alimentação, com escassez de renda, que ocuparam uma área de risco próximo a um córrego, que retiraram parte da mata ciliar para construir suas habitações precárias e que estão vivendo em uma área verde que deveria ser preservada, percebeu-se a importância de desenvolver um aplicativo para gerar informações sobre estes impacto ambiental e socioambiental para subsidiar ações do Poder Público em favor destas pessoas.

Para isso, a solução encontrada seria a utilização de uma tendência mundial, a Colaboração Coletiva - em inglês, *crowdsourcing* - equipando voluntários digitais com um aplicativo capaz de enviar informações para centrais que as receberiam e as operariam para chegar a soluções lastreadas tanto em aspectos administrativos quanto logísticos, enviando informações para os usuários. Estes podem ser o Corpo de Bombeiros da Polícia Militar, a Defesa Civil, os fiscais da Prefeitura Municipal, o policiamento integrado, as Secretarias Municipais de Habitação, de Serviço Social, de Meio Ambiente e Urbanismo e demais órgãos da esfera pública habilitados a processar tais informações para simular cenários com as consequências dos eventos extremos climáticos aos habitantes dessas áreas de risco e tomar as providências para remover essas famílias para áreas mais seguras, pois, com a chegada das chuvas, enfrentarão alto risco porque estão no caminho do escoamento das águas de drenagem superficial.

Concluindo esta introdução, como futuros Engenheiros de Computação, delimitou-se o **foco da pesquisa** nas consequências de catástrofes ambientais e seu **problema** é que ocupações irregulares de áreas de risco por populações carentes podem levar a catástrofes humanitárias se as autoridades não forem informadas para intervirem a tempo em favor destes cidadãos (Figura 1).

Figura 1. Definição do Problema



Fonte: Elaborado pelo grupo (2021)

O **objeto** de estudo deste trabalho é a aquisição de informações sobre ocupações irregulares de áreas de proteção ambiental (que geralmente apresentam alto risco aos moradores) por meio da contribuição colaborativa (crowdsourcing). Para isso, o grupo propôs desenvolver um aplicativo para smartphones dotados de sistema operacional Android (o mais difundido atualmente no País), capaz de adquirir e enviar as informações sobre tais ocupações ao Poder Público e a entidades da sociedade civil competentes para reverter a situação.

## 2. DESENVOLVIMENTO

Esta seção da pesquisa trata dos objetivos, da justificativa, da delimitação do problema, da fundamentação teórica (subdividida em área de estudo; legislação de meio ambiente; crowdsourcing & sociedade; e tecnologia escolhida para a construção do aplicativo) e metodologia da pesquisa.

### 2.1 Objetivos

**Utilizar uma ferramenta para evitar catástrofes dentro do conceito de crowdsourcing, aplicando tecnologia de comunicação colaborativa através do desenvolvimento de um aplicativo em Android, para utilização em Smartphones.**

Os objetivos específicos desta pesquisa são:

- Identificar, via comunicação colaborativa, a ocupação irregular de áreas verdes por populações prejudicadas economicamente pela pandemia de Covid-19;
- Desenvolver uma ferramenta em Android para utilização em smartphone;
- Validar com a comunidade a utilização da ferramenta para o objetivo proposto.

## 2.2. Justificativa e delimitação do problema

O **problema desta pesquisa** é criar uma ferramenta colaborativa para identificar áreas sujeitas a catástrofes. Dentro dessa linha, identificamos populações prejudicadas economicamente pela pandemia de Covid-19, que foram levadas a ocupar áreas de risco. Identificamos que uma ferramenta colaborativa voluntária tem sido utilizada no mundo para auxiliar populações afetadas por catástrofes via *crowdsourcing*. Com isso decidiu-se desenvolver uma ferramenta em sistema Android para *smartphones* e permitir a comunicação de voluntários identificando essas áreas, para as instituições responsáveis.

Mas, antes de caracterizar a **relevância social** deste aplicativo, vamos situar o recurso *crowdsourcing*, que nada mais é que a utilização da multidão, que hoje é mais referida nos meios digitais como *network*. Esse recurso é como um alimentador de informações a partir de uma “terceirização” (*outsourcing*) de obtenção de informação. O *crowdsourcing* passou a ser *aplicado principalmente em eventos que geram riscos a população tais como: enchentes, terremotos, epidemias, incêndios naturais (florestas), vulcões, furacões, deslizamentos de terra, ou seja, desastres naturais*. Estes desastres naturais colocam em risco a vida, a alimentação, a saúde física e mental das multidões em geral, tendo como uma das sequelas a perda da moradia, na maior parte das vezes, ao se verem atingidos por esses eventos. Ao utilizar o *crowdsourcing*, pretende-se mitigar os prejuízos para a população com auxílio dessa mesma população, fornecendo informações aos meios de governança, que tenham por missão agilizar o socorro aos atingidos.

A grande disseminação do *crowdsourcing* só foi possível com a utilização da tecnologia, mais especificamente a de comunicação de dados através das redes de telecomunicação digital de celulares, a mesma que suporta o funcionamento dos celulares e telefones e a rede de computadores mundiais interligados a *World Wide Web (WWW)*. E, por consequência, também a utilização das redes de satélites artificiais em órbita do planeta, tanto por imagens como por localização geográfica via Sistema de Navegação Global por Satélite (GNSS), cujo mais conhecido é o Sistema de Posicionamento Global (Global Position System - GPS), que foi o primeiro GNSS a ser implantado no mundo, pela Marinha dos EUA.

Na verdade, três fatores são o tripé desta comunicação: a imagem (vídeo e foto), a voz (áudio) e a localização geográfica (GPS) que são suportados nos “smartphones”. Uma vez certificada essa possibilidade, o passo seguinte foi a ligação logística de milhares de pessoas posicionadas ao redor dos eventos catastróficos. Com isso, foi possível obter um quadro claro da situação em tempo real e disponibilizar ou acionar recursos para minimizar os riscos e também planejar melhor a logística de socorro e estabelecer melhores estratégias de intervenção. As situações em geral são epidemias (Dengue, Sars, Ebola, Covid) e outras utilizações variadas.

A palavra-chave ligada ao *crowdsourcing* é o compartilhamento. E a pergunta que se faz é: *O que se compartilha?* Sendo a resposta: *informação*. Para situar a relevância do recurso em eventos naturais estão elencadas as seguintes opções:

1. Segurança Pública - Brasil (MOTA e LIMA, 2018);
2. Herança cultural - Croácia (ZLODI e IVANJKO, 2013);
3. Gerenciamento de emergências - Canadá (HARRISON e JOHNSON, 2019);
4. Motivação de trabalhadores para eventos de cunho social (suporte a pessoas com deficiências) – Tóquio, Japão (KOBAYASHI et. Al, 2015);
5. Identificação de falta de serviços - Suécia e EUA (BLAIR, KEY e WILSON, 2018);
6. Comportamento humano - Arábia Saudita (ALHALABI et al. 2021);
7. Avaliação de áreas afetadas por furacões - EUA (YUAN e LIU, 2018);
8. Verificação de informações em desastres naturais - Abu Dhabi, Qatar e Reino Unido (POPOOLA et al, 2013);
9. Voluntários digitais em desastres – Colorado, EUA (STARBIRD, 2011);
10. Aplicativo Android para redução de riscos geológicos - China (HE, CHAOYANG et al, 2018);
11. Transportes – coleta de dados (MISRA, ADITI et al.);
12. Levantamento de áreas afetadas por terremotos – USA (BARRINGTON, LUKE et al., 2011);
13. Crowdsourcing com Aplicativo para gerenciamento de risco de áreas de enchente – Filipinas (SAMONTE et al., 2017).

Portanto o desenvolvimento deste aplicativo para situação pretendida é da maior **relevância social**, constituindo ferramenta de compartilhamento, já extensivamente utilizado em várias partes do mundo.

A **relevância cultural** compreende a mudança de paradigma de que os cidadãos deixam todas as ações por conta das instituições públicas, mas passam a colaborar com as ações governamentais, onde cada cidadão passa a ter em suas mãos uma ferramenta de fiscalização e denúncia do estabelecimento de moradias ou outras construções em áreas de risco. Dessa forma, também se aprimora o exercício da cidadania e ainda se difunde a consciência de preservação das Áreas de Preservação Permanente (APPs).

A **relevância acadêmica** deste trabalho é explicada pelo envolvimento de diversas áreas da engenharia de computação que na primeira instância serão as disciplinas envolvidas: banco de dados, programação, comunicação de dados, ciência do ambiente, computação gráfica, engenharia de informação, multimídia e hipermídia, programação orientada a objetos, redes de computadores, segurança da informação, Interfaces Humano-Computador, sistemas operacionais, tecnologia da comunicação, linguagem e compiladores, sistemas de informação e gestão de projetos, dentre outras como Língua Portuguesa e Inglês, por exemplo.

Em segunda instância temos as funções que serão abordadas num projeto que teria aplicação baseada em projeto de sistema operacional Android com uma arquitetura de funções como: localização GPS, coleta de fotos, coleta de vídeos, registro de informações e ferramentas como mapas, previsão do tempo e configuração do sistema e seu gerenciamento. Será explorada ainda a comunicação de dados entre um *smartphone* e um servidor dados de uma rede bem como as aplicações de *webservice*.

Como contribuições da pesquisa para o local onde o projeto será desenvolvido este grupo acredita que a ação conseguirá criar uma ferramenta para agilizar o processo de aquisição de informação e tomada de decisão por parte do Poder Público em favor dessas populações carentes e desprotegidas em momentos de eventos extremos.

## 2. 3. Fundamentação teórica

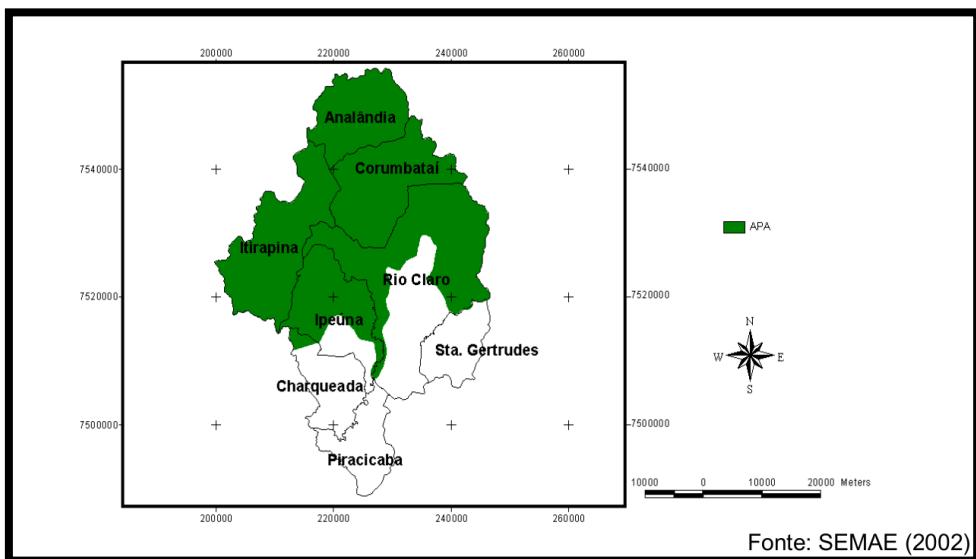
Para melhor organização deste trecho da pesquisa, o grupo dividiu este tópico em: I. Área de estudo, II. Legislação de meio ambiente, III. Crowdsourcing e a sociedade e, por último, IV. Tecnologia escolhida, que serão abordados nesta ordem a partir de agora.

### 2.3.1 Área de Estudo

Em referência à área de estudo trataremos a seguir de sua localização e características ambientais. A bacia do rio Corumbataí *localiza-se* na Depressão Periférica Paulista - DPP

( $22^{\circ} 05' S$  e  $22^{\circ} 30' S$ ;  $47^{\circ} 30'$  e  $47^{\circ} 50' W$ ) (Figura 2), sua área aproxima-se de  $1.710 \text{ km}^2$  ( $171.000 \text{ ha}$ ) e a altitude varia de 470 m no encontro com o rio Piracicaba (Santa Terezinha) a 1.058 m na Serra do Cuscuzeiro (perto de Analândia). O *clima* é tropical, com duas estações bem definidas: uma é quente e chuvosa (verão) e a outra é mais amena e seca (inverno), *Cwa* na classificação de Köeppen e possui temperaturas superiores a  $22^{\circ} \text{C}$  no mês mais quente. A estiagem estende-se de março a setembro, com apenas 20% das chuvas anuais. As massas de ar tropicais e equatoriais predominam em mais da metade do ano, trazem mais de 80% das chuvas anuais, que alcançam cerca de 1.100 mm e o período chuvoso estende-se de outubro a fevereiro, com 60 a 70 dias chuvosos (TROPPMAIR 1975 APUD RIBEIRO 2006, p. 38-9).

Figura 2. Bacia do Corumbataí e sua Área de Proteção Ambiental (APA)



Fonte: Ribeiro (2006, p. 45).

A *Geologia* de bacia do rio Corumbataí compreende-se a nordeste da Bacia Sedimentar do Paraná, na DPP, no centro-leste do Estado de São Paulo, onde afloram rochas paleozoicas (Grupo Itararé, Formação Tatuí e Grupo Passa Dois – formações Irati e Corumbataí), mesozoicas (Formação Piramboia, Grupo São Bento – formações Botucatu e Serra Geral, rochas magmáticas intrusivas – diques e soleiras e Formação Itaqueri) e cenozoicas (Formação Rio Claro). Da nascente em Analândia até a foz no rio Piracicaba, O rio Corumbataí corre sobre diversas litologias, identificadas com as unidades estratigráficas

citadas. Devido à estruturação causada pelo Domo de Pitanga, as unidades estratigráficas do alto curso (formações Botucatu, Piramboia e Corumbataí) e do baixo curso (Formação Corumbataí) são mais jovens que as do médio curso (Grupo Itararé e Formação Tatuí). (PERINOTTO e LINO 2004 apud RIBEIRO 2006, p. 40).

Sobre a *Geomorfologia*, a área ocupa a DPP, localiza-se entre os rebordos pré-cambrianos do Planalto Cristalino e escarpas das *Cuestas* dos derrames basálticos do Planalto Ocidental Paulista. O Rio Corumbataí é “recente-subsequente”, seu traçado obedece à tectônica de falhamento pós-cretácea, concordantes com a direção das camadas permo-triássicas. O Rio Corumbataí surgiu tardivamente na região, pois é o único da DPP a percorrer cerca de 100 km no sentido norte-sul. Na paisagem regional predominam extensas áreas pouco onduladas, apenas interrompidas no contato das escarpas arenítico-basálticas e cortadas pela rede hidrográfica com padronagem dendrítica. As vertentes desprotegidas pelos desmatamentos, se processam de forma acelerada, e contribuem para aprofundar os vales fluviais. *Na recente evolução geomorfológica da bacia, a intensidade de chuvas de verão, que coincide com o fim da atividade agrícola e campos sem cobertura vegetal provoca desgaste do solo* (UNESP, 2004 APUD RIBEIRO, 2006, p. 42-3). Quanto à vegetação, a Mata Atlântica recobria 8.360 ha (4,9%) e os cerrados 2.090 ha (1,2%) da região da bacia do rio Corumbataí (ZAINO e PERINOTTO 1996 APUD RIBEIRO, 2006, p. 42-3).

A hidrografia do Estado de São Paulo foi dividida em 22 Unidades de Gerenciamento dos Recursos Hídricos (UGRHI) e as bacias dos rios Piracicaba, Capivari e Jundiaí compõem a UGRHI 5. A bacia de cada um desses rios foi considerada um compartimento da UGRHI e a bacia do rio Corumbataí ocupa o compartimento de mesmo nome (figuras 5 e 6). Os recursos hídricos da bacia do rio Corumbataí compreendem quatro rios formadores principais: Corumbataí (formado pelos córregos do Veadinho, Retiro, Nova América, São Francisco, Olaria, Santa Terezinha, etc.), o Passa Cinco, o Cabeça e o Ribeirão Claro. O rio Corumbataí tem extensão de cerca de 120 km, nascentes na Serra de Santana a 800 m, vazão anual média de 22 m<sup>3</sup>/s em 1996. Sua foz é no rio Piracicaba, no município de mesmo nome, a 470 m de altitude (SÃO PAULO, ESTADO 2004 APUD RIBEIRO 2006, p. 47-8).

### 2.3.2 Legislação de Meio Ambiente

Quanto à legislação de meio ambiente temos como alicerce de todo o corpo legislativo o *Artigo 225 da Constituição Federal*, do qual citamos seu caput:

Art. 225. Todos têm direito ao meio ambiente ecologicamente equilibrado, bem de uso comum do povo e essencial à sadia qualidade de vida, impondo-se ao Poder Público e à coletividade o dever de

defendê-lo e preservá-lo para as presentes e futuras gerações (BRASIL, 1988).

Como o caput revela, todos os brasileiros têm o dever de defender e preservar o meio ambiente e o aplicativo proposto nesta pesquisa harmoniza-se com este objetivo da Carta Magna em seus objetivos e princípios de funcionamento. Além disso, acrescenta-se que existe uma legislação de meio ambiente bastante extensa, mas serão destacados aqui apenas mais duas leis que são: a Política Nacional de Meio Ambiente (Lei 6938/1981) e a Lei de Proteção da Vegetação Nativa (Lei 12651/2012).

Da Lei 6938/1981 destacamos:

Art. 2º - A Política Nacional do Meio Ambiente tem por objetivo a preservação, melhoria e recuperação da qualidade ambiental propícia à vida, visando assegurar, no País, condições ao desenvolvimento sócio-econômico, aos interesses da segurança nacional e à proteção da dignidade da vida humana [...]

O trecho acima está no mesmo espírito do artigo 225 da Constituição Federal e refere-se à qualidade do meio ambiente como base para o desenvolvimento socioeconômico, ou seja, é o desenvolvimento socioambiental, cujo termo ainda não havia sido criado. Mas continuando com a citação de trechos da Lei 6938/1981, deparamo-nos com:

Art. 4º - A Política Nacional do Meio Ambiente visará:

[...]

V - à difusão de tecnologias de manejo do meio ambiente, à divulgação de dados e informações ambientais e à formação de uma consciência pública sobre a necessidade de preservação da qualidade ambiental e do equilíbrio ecológico;

VI - à preservação e restauração dos recursos ambientais com vistas à sua utilização racional e disponibilidade permanente, concorrendo para a manutenção do equilíbrio ecológico propício à vida;

VII - à imposição, ao poluidor e ao predador, da obrigação de recuperar e/ou indenizar os danos causados e, ao usuário, da contribuição pela utilização de recursos ambientais com fins econômicos (BRASIL, 1981).

Esta pesquisa adequa-se completamente ao inciso V, tanto na divulgação de dados e informações ambientais, que é um dos objetivos do aplicativo aqui proposto, quanto na consciência pública (atualmente chamada de Educação Ambiental) sobre a preservação (ou restauração citada no inciso VI) do equilíbrio do meio ambiente. E os membros deste grupo esperam que o aplicativo seja útil para fazer cumprir o inciso VII com a ajuda da comunidade de voluntários de *crowdsourcing* socioambiental.

Concluindo esta seção sobre a legislação de meio ambiente citamos também a Lei de Proteção da Vegetação Nativa (Lei 12.651/2012) que define Área de Proteção Permanente no seu artigo 3º, §2º, II e a delimita no artigo 4º, tanto nas zonas rurais quanto nas urbanas e trata da proteção das Áreas de Preservação Permanente em seus artigos 7º, 8º e 9º (BRASIL, 2012).

### 2.3.3 Crowdsourcing e a Sociedade

Estamos passando por tempos em que é necessário resolver problemas complexos e com a maior assertividade possível, destinando recursos de tempo e financeiros de forma eficaz e eficiente. Para que isso seja possível, é importante coletar muitos dados e escutar várias vozes antes de iniciar algum projeto. O desenvolvimento de algo inovador precisa estar alinhado às necessidades dos usuários. Esses devem ser os objetivos das instituições (públicas ou privadas), todavia a etapa de pesquisa é bastante dispendiosa. Uma ferramenta ou estratégia que auxilia na investigação, na tomada de decisão e no monitoramento de *feedbacks* é o uso de *crowdsourcing* (CAMPOS, 2018).

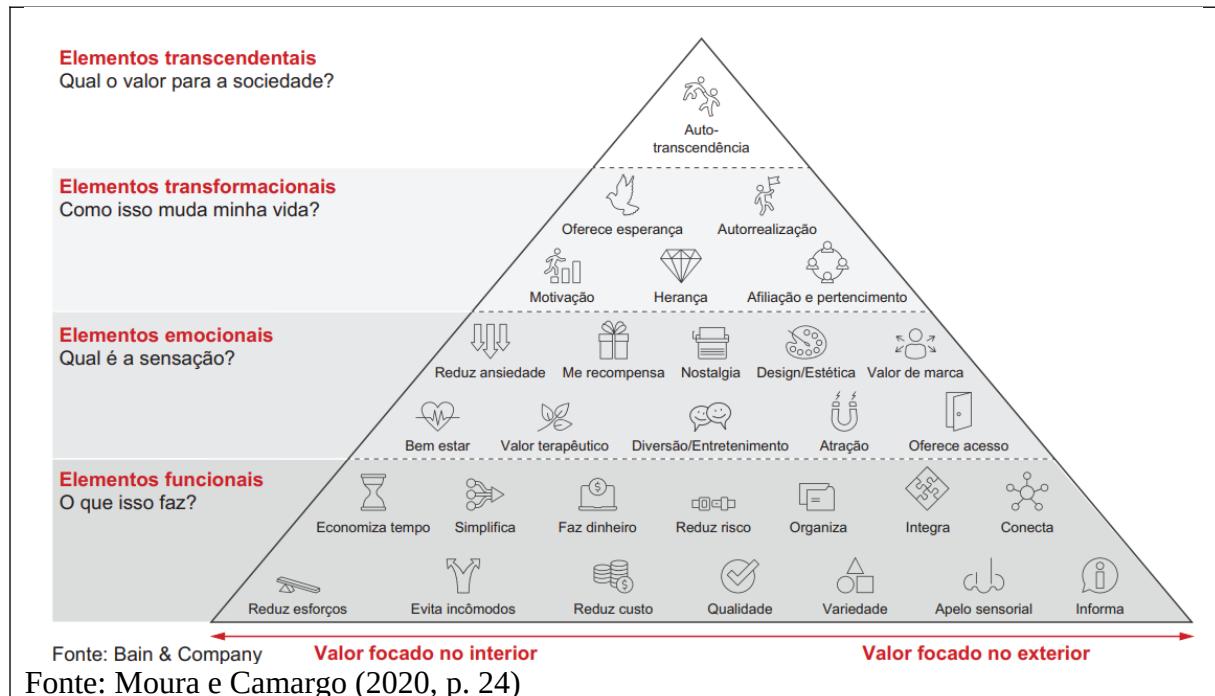
Oriundo da junção das palavras em inglês *crowd* (multidão) e *outsourcing* (terceirização), o *crowdsourcing* se baseia na terceirização de tarefas para grupos de pessoas que contribuem coletivamente no atingimento de um objetivo. Em um projeto, esses grupos podem contribuir alimentando dados, monitorando as soluções promovidas e fornecendo *feedbacks* (CAMPOS, 2018). Nas palavras de seu criador:

Si un grupo de individuos autónomos trata de resolver un problema pero carece de procedimiento para poner en común sus juicios, entonces la mejor solución que puede aspirar a encontrar es la que produzca la persona más inteligente del grupo, yeso no garantiza encontrarla. Pero cuando el mismo grupo tiene una manera de agregar esas opiniones diferentes, la solución colectiva del grupo será posiblemente más inteligente que la del individuo más capaz (SUROWIECKI, 2005 p. 108).

Haja visto que a transformação digital já vem facilitando diversas tarefas do cotidiano das pessoas, com o uso de *crowdsourcing* não é diferente. Hoje, é praticamente impossível a adoção de *crowdsourcing* sem a utilização da internet, uma vez que toda a estrutura de colaboração é feita em plataformas ou aplicativos que alimentam bancos de dados, podendo até dispor do uso de inteligência artificial para análise dos dados e auxiliar na tomada de decisão. Certamente, com o uso da computação, toda essa tarefa atinge muito mais pessoas e potencializa os benefícios dessa ferramenta (CAMPOS, 2018).

Um ponto delicado no uso de *crowdsourcing* é no convencimento das pessoas para colaborarem de maneira voluntária. Uma estratégia é deixar claro para as pessoas os objetivos e benefícios de retorno com soluções para os problemas que estas enfrentam (CAMPOS, 2018). E outra maneira de convencimento é pela formação educacional formal e não-formal na qual a pessoa se conscientiza da necessidade de fazer ao próximo e à natureza. Por exemplo, a imagem da figura 3 nos diz muito:

Figura 3. Pirâmide de Elementos de Valor



Portanto, quando as pessoas têm um smartphone à disposição, não todas, mas muitas delas se deparam com os problemas ambientais e se sentem no “*déver de defender e preservar o meio ambiente*” (art. 225/ CF 1988), elas fazem o uso mais nobre e transcendental para o aparelho, representado no topo dos elementos de valor da **figura 3**, pois contribuem com a sociedade para manter e recuperar o meio ambiente para elas mesmas e para as futuras gerações. Portanto, pensando nessas pessoas e inspirados nestes nobres sentimentos, a partir de agora descreveremos a fundamentação para a tecnologia escolhida, que foi smartphones com sistema operacional Android.

### 2.3.4. Tecnologia escolhida: Android

Como há 242 milhões de smartphones em uso no Brasil, ou seja, mais de um por habitante (FGV 2021), o grupo escolheu desenvolver um aplicativo para smartphone por sua “onipresença” nas mãos dos brasileiros e com o sistema operacional Android, que é o mais difundido em nosso país (MOURA e CAMARGO, 2020).

Como todos já sabem, um dispositivo computacional, seja ele qual for, é composto por sua parte física, denominada hardware, e sua parte lógica, chamada de software. Não é diferente dos dispositivos Mobile como celulares, smartphones ou tablets. Não adiantaria tanta capacidade e design sem um sistema operacional para fazer o dispositivo ganhar vida!

Com a evolução dos equipamentos, a indústria passou a precisar de um sistema genérico, que funcionasse em hardwares diferentes e que estivesse em constante manutenção e evolução, assim como os dispositivos físicos. Em setembro de 2003, para satisfazer as necessidades da indústria, nascia o sistema operacional Android, uma derivação do sistema operacional Linux. A empresa responsável pela criação e desenvolvimento do sistema, diferente do que você imagina, era a empresa Android Inc. (DEITEL et Al., 2015).

Em 17 de agosto de 2005, a Google, poderosa empresa de tecnologia e atual detentora do desenvolvimento do projeto Android, comprou a Android Inc, juntamente com o projeto do sistema operacional, contratando toda a equipe de desenvolvimento do sistema da empresa recém adquirida. A google pretendia entrar no mercado e dispositivos moveis e, em 2007, começou a oferecer um sistema com atualização constante para as principais empresas de hardware do mercado e, logo, começou a fechar acordos comerciais, embarcando seu sistema nesses dispositivos. (DEITEL et Al., 2015).

Outras empresas do setor de telefonia foram convidadas a conhecer e a fazer parte da aceitação ao novo sistema, como também empresas de desenvolvimento de sistemas. Assim, iniciava-se a ampla disseminação do Android 3em todo o mundo. O sistema operacional Android tem seu código aberto pela Google. Muitas empresas fabricantes de aparelhos já trabalham alterando esse código, customizando-o e otimizando-o; porém, com uma versão do sistema que possui código aberto e parte sendo propriedade da empresa do aparelho, em virtude das customizações e otimizações. (DEITEL et Al., 2015).

Com sua grande utilização, o sistema operacional Android abriu as portas de um novo mercado: o mercado de desenvolvimento Mobile. O Android possui uma loja de aplicativos, chamada Play Store, na qual são comercializadas aos usuários ferramentas para o lazer, trabalho, estudo etc. Com a popularização da loja virtual da empresa Google, muitos

desenvolvedores tiveram a oportunidade de publicar seus trabalhos e ferramentas para todo o mundo, saindo do anonimato e se tornando milionários (DEITEL et Al., 2015).

Uma outra grande forma de trabalho para o desenvolvedor é criar projetos para fins particulares, ou seja, desenvolver aplicativos para empresas como bancos, empresas de vendas etc. para atender cada vez mais a necessidade do mercado de desenvolvimento, foi criada a ferramenta *Android Studio*, que auxilia e facilita o desenvolvimento de aplicativos, sejam eles para comercialização na loja da empresa Google ou para atender empresas específicas (DEITEL et Al., 2015).

Assim, já estaremos aptos a instalar e executar nosso primeiro projeto desenvolvido para o sistema operacional Android! A seguir, comentamos o Android Studio. Com o surgimento e aceitação do sistema operacional Android, o mercado passou a necessitar de ferramentas para atender o desenvolvimento de aplicativos, gerados a partir da linguagem Java. Por algum tempo, os desenvolvedores utilizavam a ferramenta Eclipse, mas, para conseguir operar essa ferramenta, eram necessários muitos ajustes e configurações. Muitos desenvolvedores abandonaram os estudos nessa área, pelo simples fato de não conseguirem configurar o ambiente e produzir seu primeiro projeto, tornando numa experiência frustrante seu primeiro contato com o universo da programação Mobile (DEITEL et Al., 2015).

Em maio de 2013, a empresa Google fez o lançamento da sua ferramenta de desenvolvimento para a plataforma Android: o Android Studio. Uma ferramenta de fácil instalação e configuração, que utiliza a linguagem de marcação Extensible Markup Language (XML) para a criação das telas do aplicativo, as chamadas Activities. O Android Studio é a ferramenta oficial de desenvolvimento exclusivo para a plataforma Android, ou seja, esta ferramenta não desenvolve para outros sistemas operacionais (DEITEL et Al., 2015).

Atualmente, no mercado, há outras ferramentas de desenvolvimento, como por exemplo a IDE IntelliJ, base da IDE Android Studio, desenvolvida pela JetBrains, a mesma empresa que criou a linguagem de programação Kotlin. Algumas ferramentas de desenvolvimento permitem programar um único projeto Android e exportá-lo para executar nas principais plataformas de dispositivos móveis, como iOS (Apple) e essas ferramentas são chamadas de multiplataforma (DEITEL et Al., 2015).

Para o desenvolvimento operacional do aplicativo, era utilizada apenas a linguagem de programação Java, porém, como veremos abaixo, atualmente é possível desenvolver aplicativos Android nas linguagens Java e Kotlin. A empresa JetBrains iniciou o desenvolvimento do projeto Kotlin em 2010 e o apresentou em 2011, porque, para Dmitry Jemerov, a maioria das linguagens disponíveis não possuía as características que a JetBrains

procurava. A única era a linguagem Scala, mas o tempo de compilação lento da Scala era uma grave deficiência. Já o Kotlin se mostrou tão capaz quanto o Java na rapidez. Finalmente, em 2012 a JetBrains liberou seu projeto da nova linguagem Kotlin sob a licença de código aberto da Apache (BRESLAV, 2016).

Kotlin é uma Linguagem de programação multiplataforma, compila para a máquina virtual Java (JVM) e também pode ser traduzida para a linguagem JavaScript e compilada para o código nativo Low Level Virtual Machine (LLVM), sendo apresentada oficialmente pela empresa Google como a linguagem oficial para o desenvolvimento do sistema Android. O maior objetivo da linguagem Kotlin é ser uma linguagem de programação concisa e segura para o desenvolvimento web, mobile e desktop, com um amplo conjunto de ferramentas e IDEs fáceis de utilizar. O primeiro lançamento totalmente estável do Kotlin ocorreu em 15 de fevereiro de 2016, a JetBrains se comprometeu a manter a compatibilidade com as versões anteriores a partir da versão Kotlin 1.0 (BRESLAV, 2016).

Mais tarde, em 2017 o Google anunciou que o Kotlin é oficialmente suportado para desenvolvimento móvel no Android. Desde o lançamento do Android Studio 3.0 em outubro de 2017, o Kotlin é incluído como uma alternativa ao compilador Java padrão. O compilador Kotlin do Android permite que o usuário escolha entre o código de bytes compatível com Java 6 ou Java 8 (SILVA, 2019).

#### 2.4. Metodologia

A metodologia empregada foi a pesquisa qualitativa através da realização de prospecção de ideias via Brainstorms e Design Thinking para a ideação do projeto, com auxílio de elementos da literatura para definir a situação dos dois tópicos principais do projeto sendo o uso do *crowdsourcing* e a prototipagem de um Aplicativo Android. A corroboração da área de estudo foi realizada com visita in loco. Os dados foram analisados espacialmente com mapas de satélite e fotos. A viabilização do desenvolvimento do protótipo se deu com uso de programação na Integrated Development Environment (IDE) Android Studio, fornecida gratuitamente pela Google e utilizando a linguagem de programação Kotlin, oficial já há algum tempo, substituindo (em muitos casos, mas nem todos) a linguagem Java no desenvolvimento de aplicativos Android. A seguir detalhamos a metodologia com o Contexto do projeto e outros tópicos após ele.

#### 2.4.1 Contexto do projeto

A situação geradora da ideia principal do projeto ocorreu numa área do município de Piracicaba, O Jardim Bosque dos Lenheiros na divisa com Jardim Gilda, ambos bairros caracterizados por população carente. O endereço específico do local foi na altura do número 1000 da Rua das Oliveiras, uma região onde várias famílias passaram a ocupar áreas de terrenos destinado a áreas verdes, que sofreu um aumento desordenado de moradias precárias (barracos) ocupando a margem destas áreas que na verdade são a confluência da drenagem das áreas mais altas que pode ser visualizada nas Figuras 4, 5, 6, 7, 8, 9 e 10.

Figura 4. Área entre os bairros Bosques dos Lenheiros e Jardim Gilda



Fonte: Grupo (2021)

Figura 5. Área do projeto, área verde entre os bairros Jardim Gilda e Bosques dos Lenheiros



Fonte: Grupo (2021)

Figura 6. Áreas verdes em foto antiga do Google Maps



A imagem acima não mais corresponde à realidade, principalmente a área 1, ocupada por numerosos barracos. A área 2 está ocupada por barracos sob as árvores. Fonte: Grupo (2021).

Figura 7. Ocupação da Área 1



Fonte: Grupo (2021).

Figura 8. Ocupação da área 2



Fonte: Grupo (2021).

Figura 9. CRAS de atendimento na região



Fonte: Grupo (2021).

Figura 10. Localização do CRAS Mário Dedini.



Fonte: Grupo (2021).

#### 2.4.2 Descrição da situação do projeto

Na situação encontrada e junto com as ideias originadas da análise de *Design Thinking*, escolhemos tratar a região objeto como uma área de risco a ser monitorada, dada a precariedade das moradias diante de eventos extremos naturais (chuvas intensas) e também porque a ocupação da área é subproduto dos efeitos da Pandemia do COVID – 19, pois estas pessoas perderam sua renda e impossibilitadas de arcar com as despesas de moradia foram se abrigar ali. Foi implementada a ideia de produzir aplicativo que se insere no conceito de *crowdsourcing*, de modo a permitir que voluntários digitais identifiquem a área e transmitam a informação para as instituições envolvidas.

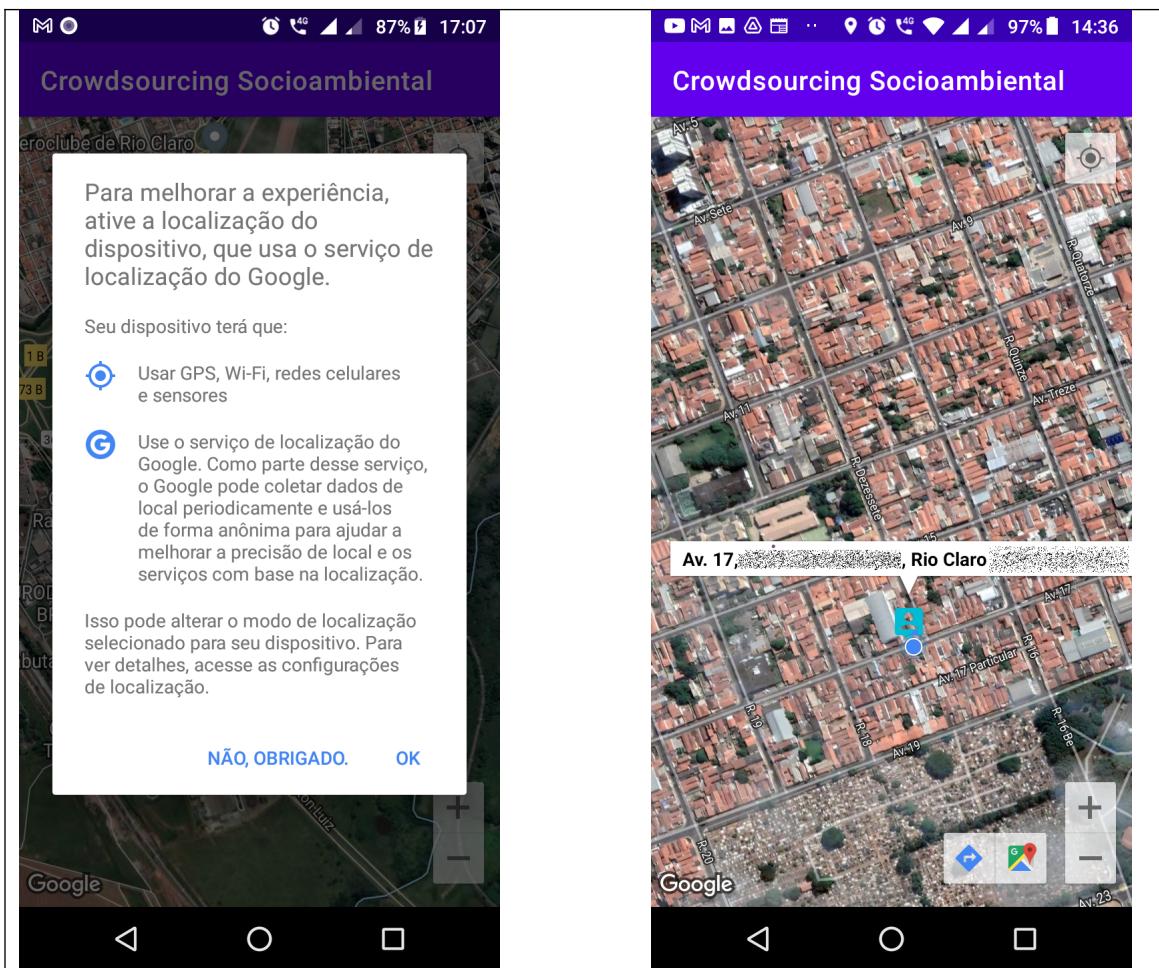
#### 2.4.3 Informações sobre os envolvidos

A comunidade em si não possui representatividade local e foi realizada uma tentativa junto ao CRAS do bairro vizinho, Mário Dedini, conforme Figuras 9 e 10. Ainda não foi possível o contato no referido CRAS, pois não encontramos a pessoa encarregada de tratar da área em questão. Porém, será feito novo contato para viabilizar mais informações e interação com a comunidade.

#### 2.4.4 Desenvolvimento de Protótipo

O desenvolvimento do protótipo acontecerá nas seguintes etapas: a primeira é a da criação do módulo de localização, já implementado (figura 11); na segunda será a criação do módulo de imagens, ainda em desenvolvimento (figura 12); a terceira fase será a de criação do módulo de avisos de eventos extremos e a quarta e última fase será a da integração dos módulos existentes e tentativa de publicação do aplicativo na loja de aplicativos do Google, conhecida como *Google Play Store*.

Figura 11. Módulo de Localização pronto e operacional



Esquerda: Aplicativo implementou com sucesso a permissão para ligar GPS e WiFi do smartphone para localização precisa. Direita: Resultado correto da localização em ambiente de teste controlado. Fonte: Grupo (2021).

Observação: códigos no Apêndice A

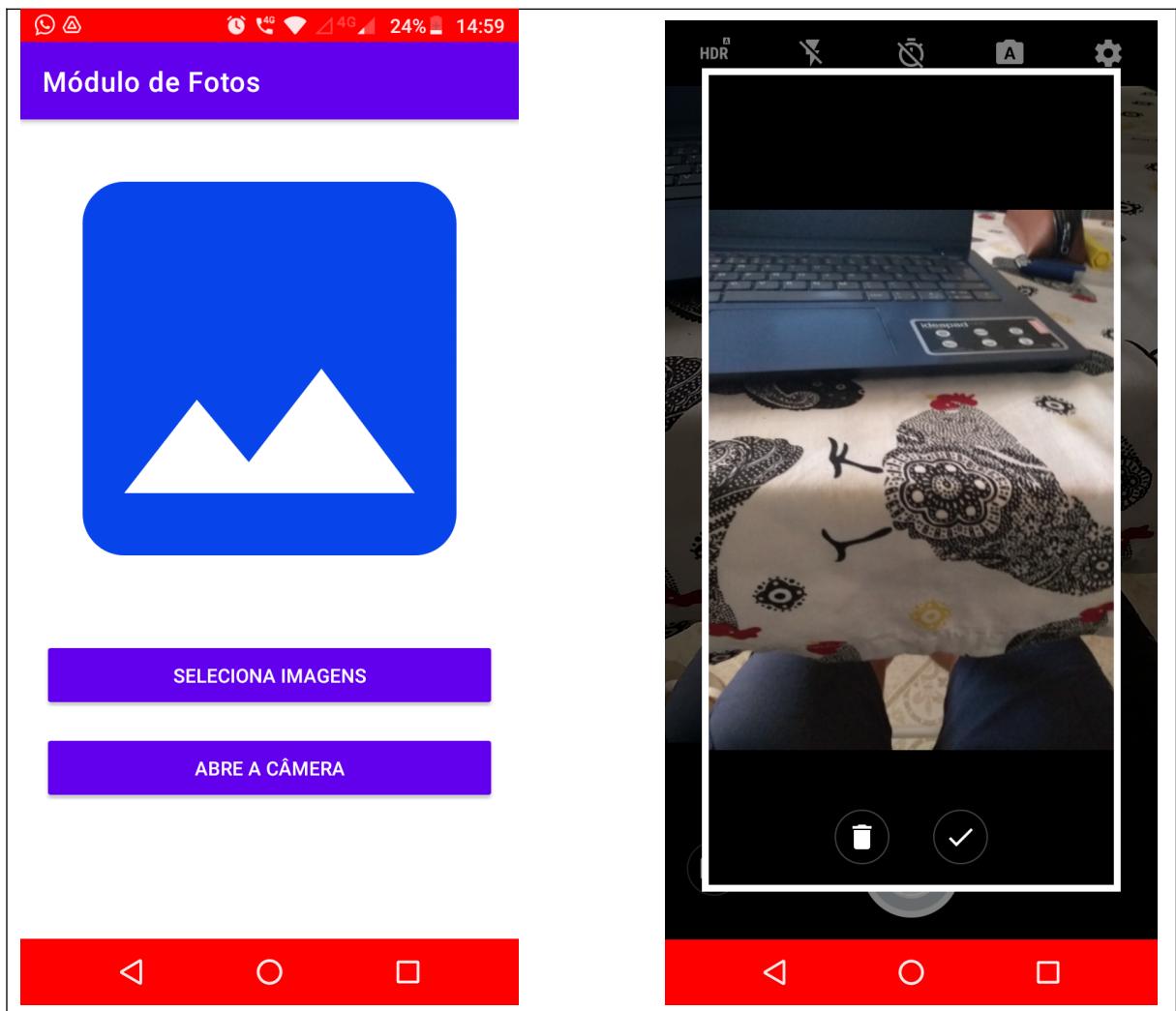
Para ser mais inclusivos, lembrando dos ensinos da disciplina *Interface Computador (IHC)*, em todos os módulos do aplicativo escolheu-se desde a API 16 (Android 4.1 “Jelly Bean”), para funcionar em 99,8% dos dispositivos móveis (smartphones e tablets) dotados de sistema operacional Android, segundo as informações da própria IDE Android Studio. Antes de rodar a aplicação foi necessário adquirir uma conta de desenvolvedor Google, pedir a criação de uma chave para o projeto, copiá-la e inseri-la no arquivo google\_maps\_api.xml (ver apêndice A).

Mas isso não foi tudo, após criar a chave da API, foi necessário concluir o registro do aplicativo no site console.cloud.google.com como tipo de usuário externo (modo teste) foi necessário completar o registro do aplicativo no site do google como Projeto escolar/ tarefa e o acesso vale por 90 dias gratuitamente, sendo possível concluir o PI 8. Também foi informado e-mail, telefone para contato, dados do cartão de crédito e responder a um pequeno questionário sobre por que escolheu a *Google Cloud*. O *Google Temporary Hold* verificou os

dados do desenvolvedor, as permissões da chave foram concedidas e, após isso, ocorreu a codificação do módulo de localização de um ponto no mapa (figura 11 e Apêndice A).

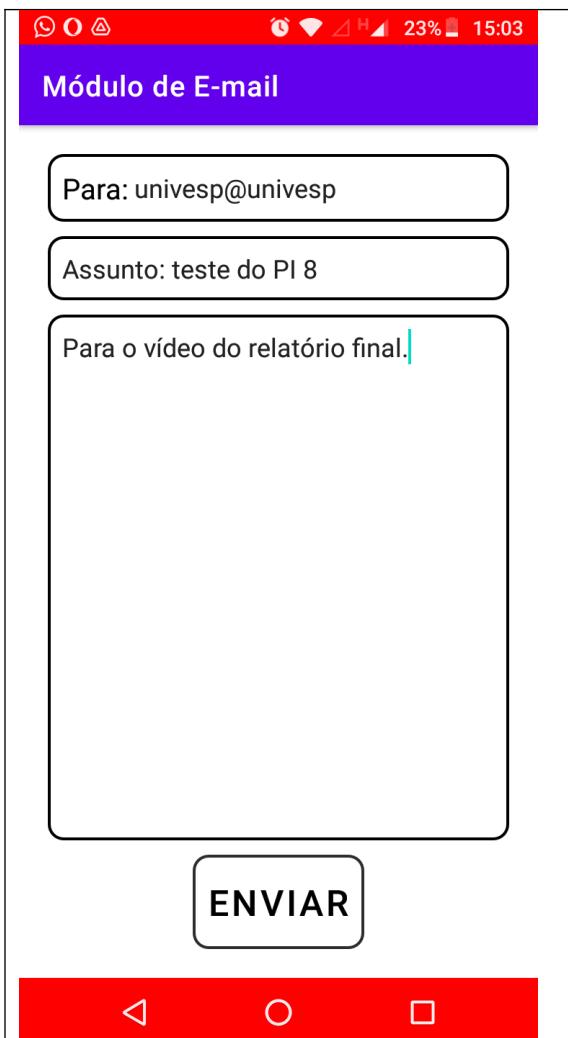
Conseguido isto, o próximo passo foi iniciar a construção do aplicativo para utilizar a câmera do dispositivo (figura 12) para capturar, salvar e recuperar fotografias. Na versão final deste aplicativo, que ficará pronta somente após a entrega deste relatório, o grupo pretende adicionar a função de captura, gravação e recuperação de vídeos.

Figura 12. Módulo de imagens pronto e operacional



Módulo de Fotos pronto e funcional - Fonte: Grupo (2021).  
Observação: ver códigos no Apêndice C.

Figura 13. Módulo de E-mail pronto e operacional



Fonte: Grupo (2021). Observação: ver códigos no Apêndice C.

Com os três módulos principais prontos, foi criada a tela inicial (figura 14) para dar início à integração dos mesmos em um único grande aplicativo. Porém, por falta de tempo hábil para estudo e pesquisa e de conhecimentos especializados não foi possível concluir o projeto no prazo estipulado.

Figura 14. Tela inicial do aplicativo completo.



Fonte: Grupo (2021). Observação: códigos XML no Apêndice D

## **RESULTADOS**

Considerando que o objetivo principal do trabalho foi a utilização de uma ferramenta para evitar catástrofes dentro do conceito de *crowdsourcing*, construindo uma ferramenta baseada em Android *para Smartphones*, o objetivo foi atingido pois foi possível utilizar a comunicação colaborativa de *crowdsourcing* com a construção de ferramenta para Smartphones Android. No entanto, a validação junto à comunidade não foi atingida pois o contato com os responsáveis no CRAS Mário Dedini, em Piracicaba, foi prejudicado pelas dificuldades de contato devidas à situação da Pandemia Covid-19.

Quanto à fundamentação teórica, conseguiu-se estabelecer a **relevância social** através da utilização do recurso de *crowdsourcing*, o qual permite à comunidade utilizá-lo pela via digital como uma “terceirização” da informação de situação de catástrofes tais como: *enchentes, terremotos, epidemias, incêndios naturais (florestas), vulcões, furacões, deslizamentos de terra, ou seja, desastres naturais*. O desenvolvimento do protótipo inicial para um aplicativo com módulos de localização, imagens e avisos foi atingido em sua plenitude. Também foi atingida a **relevância acadêmica** com o *background* fornecido pelo aprendizado das disciplinas da UNIVESP, principalmente linguagens de programação, banco de dados, comunicação de dados, ciência do ambiente, computação gráfica, engenharia de informação, multimídia e hipermídia, programação orientada a objetos, redes de computadores, segurança da informação, Interfaces Humano-Computador, sistemas operacionais, tecnologia da comunicação e linguagem e compiladores.

## **CONSIDERAÇÕES FINAIS**

Em primeiro lugar, nosso grupo obteve sucesso em trabalhar unido para planejar, criar e prototipar um aplicativo atendendo ao desafio do tema sugerido por esta conceituada Instituição Pública de Ensino Superior, aplicando todas as premissas determinadas pela disciplina Projeto Integrado VIII. Foram seguidas todas as etapas pré-estabelecidas, aprofundando-se nos temas colaterais e trabalhando em equipe com responsabilidade e agilidade para propor uma solução a esta problemática sugerida.

Acrescente-se que ainda é preciso aprender mais as linguagens de programação Java e Kotlin para concluir este aplicativo em breve, integrando os três módulos em um único aplicativo e disponibilizá-lo na loja de aplicativos Android, servindo assim à comunidade alvo deste projeto. Finalmente, o grupo deseja que este aplicativo iniciado neste último Projeto Integrador também poderá ser base para ampliações ou adaptações por colegas da Engenharia de Computação para finalidades semelhantes ou diferentes das aqui propostas.

## REFERÊNCIAS

ALHALABI, Wadee; LYTRAS, Miltiadis; ALJOHANI, Nada. Crowdsourcing research for social insights into smart cities applications and services. **Sustainability**, v. 13, n. 14, p. 7531, 2021.

BARNETT, Scott; VASA, Rajesh; GRUNDY, John. Bootstrapping mobile app development. In: **2015 IEEE/ACM 37th IEEE International Conference on Software Engineering**. IEEE, 2015. p. 657-660.

BARRINGTON, Luke et al. Crowdsourcing earthquake damage assessment using remote sensing imagery. **Annals of geophysics**, v. 54, n. 6, 2011.

BLAIR, Amanda; KEY, Thomas Martin; WILSON, Matthew. Crowdsourcing to manage service gaps in service networks. **Journal of Business & Industrial Marketing**, 2019.

BRASIL, PRESIDÊNCIA DA REPÚBLICA (1981). **Lei N° 6.938, de 31 de Agosto de 1981, institui a POLÍTICA NACIONAL DE MEIO AMBIENTE**. Brasília - DF. Disponível em: [http://www.planalto.gov.br/Ccivil\\_03/Leis/L6938.htm#art1](http://www.planalto.gov.br/Ccivil_03/Leis/L6938.htm#art1). Acesso em: 10 out. 2021.

BRASIL, **CONSTITUIÇÃO FEDERAL (1988)**. Capítulo VI – Do Meio Ambiente, art. 225. Congresso Nacional: Brasília - DF. Promulgada em 5 de Outubro de 1988. Disponível em: [http://www.planalto.gov.br/ccivil\\_03/Constituicao/Constituicao.htm](http://www.planalto.gov.br/ccivil_03/Constituicao/Constituicao.htm). Acesso em: 10 out. 2021.

BRASIL, PRESIDÊNCIA DA REPÚBLICA (2012). **Lei N° 12.651, de 25 de Maio de 2012, dispõe sobre proteção da vegetação nativa**. Brasília - DF. Disponível em: [http://www.planalto.gov.br/ccivil\\_03/\\_Ato2011-2014/2012/Lei/L12651.htm](http://www.planalto.gov.br/ccivil_03/_Ato2011-2014/2012/Lei/L12651.htm). Acesso em: 10 out. 2021.

BRESLAV, A. Kotlin 1.0 Release: Pragmatic Language for the JVM and Android. **The Kotlin Blog**. Publicado em: 15 fev. 2016. Disponível em: <https://blog.jetbrains.com/kotlin/2016/02/kotlin-1-0-released-pragmatic-language-for-jvm-and-android/>. Acesso em: 10 out. 2021.

CAMPOS, K. O que é crowdsourcing: plataformas e exemplos de colaboração nas empresas. **Poder da Escuta Corporativa**. Publicado em: 29 mai. 2018. Disponível em: <<https://www.poderdaescuta.com/mas-afinal-o-que-e-crowdsourcing/>>. Acesso em: 9 out. 2021.

DEITEL, H.; DEITEL, P.; DEITEL, A. **Android. Como programar** [Porto Alegre, Bookman, 2015]: Grupo A, 2015. 9788582603482. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788582603482/>. Acesso em: 10 out. 2021.

FUNDAÇÃO GETÚLIO VARGAS (FGV). **Brasil tem dois dispositivos digitais por habitante, revela pesquisa da FGV**. Publicado em: 21 mai. 2021. Disponível em: <https://portal.fgv.br/noticias/brasil-tem-dois-dispositivos-digitais-habitante-revela-pesquisa-fgv/>. Acesso em: 11 out. 2021.

HARRISON, Sara; JOHNSON, Peter. Challenges in the adoption of crisis crowdsourcing and social media in Canadian emergency management. **Government Information Quarterly**, v. 36, n. 3, p. 501-509, 2019.

HE, Chaoyang et al. Designing an android-based application for geohazard reduction using citizen-based crowdsourcing data. **Mobile Information Systems**, v. 2018, 2018.

JOORABCHI, Mona Erfani; MESBAH, Ali; KRUCHTEN, Philippe. Real challenges in mobile app development. In: **2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement**. IEEE, 2013. p. 15-24.

KANADE, Prashant et al. Saviour: Disaster Management and Monitoring Utility. **Available at SSRN 3276164**, 2018.

KOBAYASHI, Masatomo et al. Motivating multi-generational crowd workers in social-purpose work. In: **Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing**. 2015. p. 1813-1824.

MISRA, Aditi et al. Crowdsourcing and its application to transportation data collection and management. **Transportation Research Record**, v. 2414, n. 1, p. 1-8, 2014.

MOTA, João Moisés Brito; LIMA, Afonso Carneiro. Efetividade do Crowdsourcing como Apoio à Segurança Pública. **Revista de Administração Contemporânea**, v. 22, p. 683-703, 2018.

MOURA, L.; CAMARGO, G. **Impacto econômico e social do Android no Brasil**. São Paulo: Bain & Company, 2020. Disponível em:  
<https://www.bain.com/contentassets/a9200a057a0241b8963c05a9b09e33fe/impactos-do-android-no-brasil.pdf>. Acesso em: 11 out. 2021.

POPOOLA, Abdulfatai et al. Information verification during natural disasters. In: **Proceedings of the 22nd international conference on World Wide Web**. 2013. p. 1029-1032.

RIBEIRO, S. L. **Análise da sustentabilidade na bacia do rio Corumbataí (SP)**. Rio Claro, 2006. 143 f. Tese (doutorado em Geologia Regional). Instituto de Geociências e Ciências Exatas. Universidade Estadual Paulista, Rio Claro, 2006. Disponível em:  
<https://repositorio.unesp.br/handle/11449/102984>. Acesso em: 9 out. 2021.

SAMONTE, Mary Jane C. et al. Crowdsourced mobile app for flood risk management. In: **Proceedings of the 3rd International Conference on Communication and Information Processing**. 2017. p. 65-71.

SILVA, J. Kotlin: História e motivos para o crescimento e popularidade da linguagem. **Curitiba TI**. Publicado em: 4 abr. 2019. Disponível em:  
<https://www.curitibati.com.br/Blog/Post/Kotlin-crescimento-popularidade-da-linguagem/>. Acesso em: 10 out. 2021.

SONG, Zhijun; ZHANG, Hui; DOLAN, Chris. Promoting disaster resilience: Operation mechanisms and self-organizing processes of crowdsourcing. **Sustainability**, v. 12, n. 5, p. 1862, 2020.

STARBIRD, Kate. Digital volunteerism during disaster: Crowdsourcing information processing. In: **Conference on human factors in computing systems**. 2011. p. 7-12.

SUROWIECKI, J. **Cien Mejor que uno**: La sabiduría de la multitud o por qué la mayoría siempre es más inteligente que la minoría. Barcelona (España), Ediciones Urano, 2005.

YUAN, Faxi; LIU, Rui. Feasibility study of using crowdsourcing to identify critical affected areas for rapid damage assessment: Hurricane Matthew case study. **International journal of disaster risk reduction**, v. 28, p. 758-767, 2018.

## APÊNDICES

### Apêndice A – Códigos do Módulo de Localização

(códigos principais: MapsActivity.kt, AndroidManifest.xml, google\_maps\_api.xml e strings.xml)

#### 1 - MapsActivity.kt

```
package com.example.crowdsourcingambiental
/*
 * Módulo de localização
 * linguagem: kotlin
 * data: 12 de outubro de 2021
 * Adaptado de curso de Daniel Richter/Digital Innovation One
 */

import android.content.IntentSender
import android.content.pm.PackageManager
import android.graphics.BitmapFactory
import android.location.Address
import android.location.Geocoder
import android.location.Location
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import com.example.crowdsourcingambiental.databinding.ActivityMapsBinding
import com.google.android.gms.common.api.ResolvableApiException
import com.google.android.gms.location.*
import com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.OnMapReadyCallback
import com.google.android.gms.maps.SupportMapFragment
import com.google.android.gms.maps.model.BitmapDescriptorFactory
import com.google.android.gms.maps.model.LatLng
import com.google.android.gms.maps.model.Marker
import com.google.android.gms.maps.model.MarkerOptions
import java.util.*

class MapsActivity : AppCompatActivity(),
    OnMapReadyCallback,
    GoogleMap.OnMarkerClickListener {

    private lateinit var map: GoogleMap
    private lateinit var fusedLocationProviderClient:
    FusedLocationProviderClient
    private lateinit var lastLocation: Location
    private lateinit var binding: ActivityMapsBinding

    /*para atualizar localização do usuário*/
    private lateinit var locationCallback: LocationCallback
    private lateinit var locationRequest: LocationRequest
    private var locationUpdateState = false

    companion object {
```

```

private const val LOCATION_PERMISSION_REQUEST_CODE = 1
private const val REQUEST_CHECK_SETTINGS = 2
}

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = ActivityMapsBinding.inflate(layoutInflater)
    setContentView(binding.root)

    // Obtain the SupportMapFragment and get notified when the map is ready
    // to be used.
    val mapFragment = supportFragmentManager
        .findFragmentById(R.id.map) as SupportMapFragment
    mapFragment.getMapAsync(this)

    fusedLocationProviderClient =
    LocationServices.getFusedLocationProviderClient(this)

    locationCallback = object : LocationCallback (){
        override fun onLocationResult(p0: LocationResult) {
            super.onLocationResult(p0)

            lastLocation = p0.lastLocation
            placeMarkerOnMap(LatLng(lastLocation.latitude,
lastLocation.longitude))
        }
    }

    createLocationRequest()
}

override fun onMapReady(googleMap: GoogleMap) {
    map = googleMap

    //a linha abaixo (em true) habilita os controles de zoom no mapa
    map.getUiSettings().setZoomControlsEnabled(true)
    // a linha abaixo habilita a "escuta" do clique,
    // tornando-o funcional no Google Mapas para traçar rotas e localizar no
    Mapa!
    map.setOnMarkerClickListener(this)
    // tornando-o funcional no Google Mapas para traçar rotas e localizar no
    Mapa!
    setUpMap()
}

private fun setUpMap() {
    if (ActivityCompat.checkSelfPermission(this,
        android.Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this,

```

```

        arrayOf(android.Manifest.permission.ACCESS_FINE_LOCATION),
LOCATION_PERMISSION_REQUEST_CODE)
    return
}

map.isMyLocationEnabled = true

map.mapType = GoogleMap.MAP_TYPE_HYBRID

fusedLocationProviderClient.lastLocation.addOnSuccessListener (this) {
location ->
    // Got last known location. In some rare situations this can be null.

    if (location != null) {
        lastLocation = location
        val currentLatLng = LatLng(lastLocation.latitude,
lastLocation.longitude)
        placeMarkerOnMap(currentLatLng)

map.animateCamera(CameraUpdateFactory.newLatLngZoom(currentLatLng,
15f))
    }
}
}

private fun placeMarkerOnMap(location: LatLng){
    val markerOptions = MarkerOptions().position(location)

markerOptions.icon(BitmapDescriptorFactory.fromBitmap(BitmapFactory.decode
Resource(resources, R.mipmap.ic_user_location)))

    val titleStr = getAddress(location)
    markerOptions.title(titleStr)
    map.addMarker(markerOptions)
}

//captura o endereço do local
private fun getAddress(latLng: LatLng): String{
    val geocoder:Geocoder
    val addresses: List<Address>
    geocoder = Geocoder(this, Locale.getDefault())

    addresses = geocoder.getFromLocation(latLng.latitude, latLng.longitude,
1)

    val address = addresses[0].getAddressLine(0)
    val city = addresses[0].locality
    val state = addresses[0].adminArea
    val country = addresses[0].countryName
    val postalCode = addresses[0].postalCode
    return address
}
//função dialoga com usuário para ligar o GPS e o wi-fi juntos
private fun startLocationUpdates(){
}

```

```
if (ActivityCompat.checkSelfPermission(this,  
        android.Manifest.permission.ACCESS_FINE_LOCATION) !=  
PackageManager.PERMISSION_GRANTED) {  
    ActivityCompat.requestPermissions(this,  
        arrayOf(android.Manifest.permission.ACCESS_FINE_LOCATION),  
        LOCATION_PERMISSION_REQUEST_CODE)  
    return  
}  
fusedLocationProviderClient.requestLocationUpdates(locationRequest,  
locationCallback, null /* Looper */)  
}  
  
private fun createLocationRequest() {  
    locationRequest = LocationRequest()  
  
    /*  
     * Usar estas linhas quando precisar localizar pontos em série (rodovia,  
     * trilha)  
     *locationRequest.interval = 10000  
     *locationRequest.fastestInterval = 5000  
     locationRequest.priority = LocationRequest.PRIORITY_HIGH_ACCURACY  
    */  
  
    val builder = LocationSettingsRequest.Builder()  
        .addLocationRequest(locationRequest)  
  
    val client = LocationServices.getSettingsClient(this)  
    val task = client.checkLocationSettings(builder.build())  
  
    task.addOnSuccessListener {  
        locationUpdateState = true  
        startLocationUpdates()  
    }  
    task.addOnFailureListener { e ->  
        if (e is ResolvableApiException) {  
            try {  
                e.startResolutionForResult(  
                    this@MapsActivity,  
                    REQUEST_CHECK_SETTINGS)  
            } catch (sendEx: IntentSender.SendIntentException) {  
                //Ignore the error  
            }  
        }  
    }  
}  
  
override fun onPause() {  
    super.onPause()  
    fusedLocationProviderClient.removeLocationUpdates(locationCallback)  
}  
  
public override fun onResume() {  
    super.onResume()  
    if (!locationUpdateState) {
```

```

        startLocationUpdates()
    }

    override fun onMarkerClick(p0: Marker?) = false
}

```

## 2 - AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.crowdsourcingambiental">

    <!--
        The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
        Google Maps Android API v2, but you must specify either coarse or fine
        location permissions for the "MyLocation" functionality.
    -->
    <uses-permission
        android:name="android.permission.ACCESS_FINE_LOCATION" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.CrowdsourcingAmbiental">

        <!--
            The API key for Google Maps-based APIs is defined as a string resource.
            (See the file "res/values/google_maps_api.xml").
            Note that the API key is linked to the encryption key used to sign the
            APK.
            You need a different API key for each encryption key, including the
            release key that is used to
            sign the APK for publishing.
            You can define the keys for the debug and release targets in src/debug/
            and src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".MapsActivity"
            android:exported="true"
            android:label="@string/title_activity_maps">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>

```

```
</intent-filter>
</activity>
</application>
```

```
</manifest>
```

### 3 - google\_maps\_api.xml

```
<resources>
```

```
<!--
```

***TODO: Before you run your application, you need a Google Maps API key.***

***To get one, follow this link, follow the directions and press "Create" at the end:***

***[https://console.developers.google.com/flows/enableapi?  
apiid=maps\\_android\\_backend&keyType=CLIENT\\_SIDE\\_ANDROID&r=<...>  
com.example.crowdsourcingambiental](https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=<...>com.example.crowdsourcingambiental)***

***You can also add your credentials to an existing key, using these values:***

***Package name:***

***com.example.crowdsourcingambiental***

***SHA-1 certificate fingerprint:***

```
<....>
```

***Alternatively, follow the directions here:***

***[https://developers.google.com/maps/documentation/android/start#get-  
key](https://developers.google.com/maps/documentation/android/start#get-key)***

***Once you have your key (it starts with "Alza"), replace the "google\_maps\_key" string in this file.***

```
-->
```

```
<string name="google_maps_key"
templateMergeStrategy="preserve" translatable="false">"insert your api
key here!"</string>
</resources>
```

### 4 - strings.xml

```
<resources>
```

```
<string name="app_name">crowdsourcing ambiental</string>
```

```
<string name="title_activity_maps">Módulo de Localização</string>
```

```
</resources>
```

## Apêndice B – Códigos do Módulo de Imagem

(Códigos principais: MainActivity.kt, AndroidManifest.xml, activity\_main.xml, strings.xml)

### Códigos Kotlin:

#### 1 - MainActivity.kt

```
package com.example.photosmodule

import android.Manifest
import android.app.Activity
import android.content.ContentValues
import android.content.Intent
import android.content.pm.PackageManager
import android.net.Uri
import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.provider.MediaStore
import android.widget.Toast
import androidx.core.content.contentValuesOf
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
    var image_uri: Uri? = null

    companion object {
        private val PERMISSION_CODE_IMAGE_PICK = 1000
        private val IMAGE_PICK_CODE = 1001
        private val PERMISSION_CODE_CAMERA_CAPTURE = 2000
        private val OPEN_CAMERA_CODE = 2001
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        pick_button.setOnClickListener {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                if (checkSelfPermission(Manifest.permission.READ_EXTERNAL_STORAGE)
                    == PackageManager.PERMISSION_DENIED) {
                    val permission =
                        arrayOf(Manifest.permission.READ_EXTERNAL_STORAGE)
                    requestPermissions(permission, PERMISSION_CODE_IMAGE_PICK)
                } else {
                    pickImageFromGalery()
                }
            } else {
                pickImageFromGalery()
            }
        }
    }
}
```

```
open_camera_button.setOnClickListener {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (checkSelfPermission(Manifest.permission.CAMERA)
            == PackageManager.PERMISSION_DENIED ||

        checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE)
            == PackageManager.PERMISSION_DENIED) {
            val permissions = arrayOf(
                Manifest.permission.CAMERA,
                Manifest.permission.WRITE_EXTERNAL_STORAGE)
            requestPermissions(permissions,
                PERMISSION_CODE_CAMERA_CAPTURE)
        }
        else {
            openCamera()
        }
    }
    else {
        openCamera()
    }
}

override fun onRequestPermissionsResult(requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray) {
    when (requestCode) {
        PERMISSION_CODE_IMAGE_PICK -> {
            if (grantResults.size > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                pickImageFromGalery()
            } else {
                Toast.makeText(this, "Permissão Negada!",
                    Toast.LENGTH_SHORT).show()
            }
        }

        PERMISSION_CODE_CAMERA_CAPTURE -> {
            if (grantResults.size > 1 &&
                grantResults[0] == PackageManager.PERMISSION_GRANTED &&
                grantResults[1] == PackageManager.PERMISSION_GRANTED) {
                openCamera()
            } else {
                Toast.makeText(this, "Permissão Negada!",
                    Toast.LENGTH_SHORT).show()
            }
        }
    }
}

private fun pickImageFromGalery() {
    val intent = Intent(Intent.ACTION_PICK)
    intent.type = "image/*"
    startActivityForResult(intent, IMAGE_PICK_CODE)
```

```

    }

private fun openCamera() {
    val values = ContentValues()
    values.put(MediaStore.Images.Media.TITLE, "Nova foto.")
    values.put(MediaStore.Images.Media.DESCRIPTION, "Imagen capturada
pela câmera.")
    image_uri =
contentResolver.insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
values)

    val cameraIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, image_uri)
startActivityForResult(cameraIntent, OPEN_CAMERA_CODE)

}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (resultCode == Activity.RESULT_OK && requestCode ==
IMAGE_PICK_CODE) {
        image_view.setImageURI(data?.data)
    }
    if (resultCode == Activity.RESULT_OK && requestCode ==
IMAGE_PICK_CODE){
        image_view.setImageURI(image_uri)
    }
}

public override fun onResume() {
    super.onResume()

}
}

```

## Códigos XML:

### 2 - AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.photosmodule">

    <uses-permission
        android:name="android.permission.READ_EXTERNAL_STORAGE"/>

```

```

<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    tools:ignore="ScopedStorage" />

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.PhotosModule">
    <activity
        android:name=".MainActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>

</manifest>

```

### **3 - activity\_main.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_gravity="center_horizontal"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/image_view"
        android:scaleType="centerCrop"
        android:contentDescription="@string/selected_image"
        android:src="@drawable/image"
        android:layout_width="370dp"
        android:layout_height="370dp"
    />

    <Button
        android:id="@+id/pick_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/image_selected"
        style="@style/Widget.AppCompat.Button.Colored"
    />

```

```

        android:padding="10dp"
        android:layout_margin="16dp"
    />

<Button
    android:id="@+id/open_camera_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/open_camera_button"
    style="@style/Widget.AppCompat.Button.Colored"
    android:padding="10dp"
    android:layout_marginStart="16dp"
    android:layout_marginEnd="16dp"
    />

</androidx.appcompat.widget.LinearLayoutCompat>

```

#### 4. strings.xml

```

<resources>
    <string name="app_name">photosModule</string>
    <string name="selected_image">Imagen a ser selecionada pelo
usuário</string>
    <string name="image_selected">Seleciona imagens</string>
    <string name="open_camera_button">"Abre a Câmera"</string>
</resources>

```

## Apêndice C - Códigos do Módulo de e-mail

### Códigos Kotlin:

```

/*
Este aplicativo baseia-se em:
"How to Send an Email via Intent - Android Studio Tutorial"
Duração: 10m38s
Canal: Coding in Flow
Publicado em: 25 dez. 2017
Disponível em: <https://youtu.be/tZ2YEw6SoBU/>
Acesso em: 16 nov. 2021

```

"How to Send Email in Android Studio | SendEmail | Android Coding"

Duração: 08m03s

Canal: Android Coding

Publicado em: 18 abr. 2019

Disponível em: <<https://youtu.be/aV7-vcwEeRM/>>

Acesso em: 16 nov. 2021

\*/

```
package com.example.mail;
```

```
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.content.Intent;
```

```
import android.os.Bundle;
```

```
import android.widget.Button;
```

```
import android.widget.EditText;
```

```
public class MainActivity extends AppCompatActivity {
```

```
    private EditText mEditTextTo;
```

```
    private EditText mEditTextSubject;
```

```
    private EditText mEditTextMessage;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
```

```
        mEditTextTo = findViewById(R.id.edit_text_to);
```

```
        mEditTextSubject = findViewById(R.id.edit_text_subject);
```

```
        mEditTextMessage = findViewById(R.id.edit_text_message);
```

```
        Button buttonSend = findViewById(R.id.button_send);
```

```
        buttonSend.setOnClickListener(v -> sendMail());
```

```
}
```

```
    private void sendMail() {
```

```

//permite enviar email para várias contas simultaneamente//

String recipientList = mEditTextTo.getText().toString();
String[] recipients = recipientList.split(", ");

String subject = mEditTextSubject.getText().toString();
String message = mEditTextMessage.getText() .toString();

Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, recipients);
intent.putExtra(Intent.EXTRA_SUBJECT, subject);
intent.putExtra(Intent.EXTRA_TEXT, message);

intent.setType("message/rfc822");
startActivity(Intent.createChooser(intent, "Escolha seu provedor de e-mail..."));
}

}

```

### Códigos XML:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout

    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_gravity="center"
    android:padding="20dp"
    tools:context="MainActivity">

<LinearLayout

    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp"
    android:background="@drawable/bg_round">

```

```
<TextView  
    android:layout_width="50dp"  
    android:layout_height="wrap_content"  
    android:text="@string/para"  
    android:textColor="@color/black"  
    android:textSize="20sp"  
    android:layout_gravity="center"  
/>
```

```
<EditText  
    android:id="@+id/edit_text_to"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="@android:color/transparent"  
    android:inputType="textEmailAddress"  
/>  
</LinearLayout>
```

```
<EditText  
    android:id="@+id/edit_text_subject"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/assunto"  
    android:padding="10dp"  
    android:layout_marginTop="10dp"  
    android:background="@drawable/bg_round"  
    android:inputType="textEmailSubject"  
/>
```

```
<EditText  
    android:id="@+id/edit_text_message"  
    android:layout_width="match_parent"  
    android:layout_height="0dp"  
    android:layout_marginTop="10dp"  
    android:lines="10"  
    android:layout_weight="1"
```

```
    android:background="@drawable/bg_round"
    android:gravity="start|top"
    android:hint="@string/escreva_seu_texto"
    android:padding="10dp"
    android:inputType="textWebEditText"/>
```

```
<Button
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginTop="10dp"
    android:background="@drawable/bg_round"
    android:padding="10dp"
    android:text="@string/enviar"
    android:textColor="@color/black"
    android:textSize="25sp"
    app:backgroundTint="#2E2D2D" />
```

```
</LinearLayout>
```

```
<resources>
    <string name="app_name">Módulo de E-mail</string>
    <string name="para">Para: </string>
    <string name="assunto">Assunto: </string>
    <string name="escreva_seu_texto">Escreva seu texto aqui </string>
    <string name="enviar">Enviar</string>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="@android:color/transparent"/>
    <corners android:radius="10dp"/>
```

```
<stroke android:width="2dp" android:color="@android:color/background_dark"/>

</shape>
```

## Apêndice D – Códigos da Tela Inicial

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button_app_information"
        android:layout_width="350dp"
        android:layout_height="100dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="64dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:text="@string/sobre_este_aplicativo"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.511"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button_email" />

    <Button
        android:id="@+id/button_localization"
        android:layout_width="350dp"
        android:layout_height="100dp"
```

```
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="48dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:text="@string/modulo_de_localizacao"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"/> 
```

```
<Button
    android:id="@+id/button_photos"
    android:layout_width="350dp"
    android:layout_height="100dp"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="48dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginBottom="48dp"
    android:text="@string/modulo_de_fotos"
    app:layout_constraintBottom_toTopOf="@+id/button_email"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button_localization" />
```

```
<Button
    android:id="@+id/button_email"
    android:layout_width="350dp"
    android:layout_height="100dp"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="48dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:text="@string/modulo_de_email"
```

```
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button_photos" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

### Códigos Kotlin:

```
package com.example.telainicial
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val buttonLocalization: Button = findViewById(R.id.button_localization)
        val buttonPhotos: Button = findViewById(R.id.button_photos)
        val buttonEmail: Button = findViewById(R.id.button_email)
        val buttonInformation: Button = findViewById(R.id.button_app_information)

        buttonLocalization.setOnClickListener{
            }

        button_photos.setOnClickListener{
            }

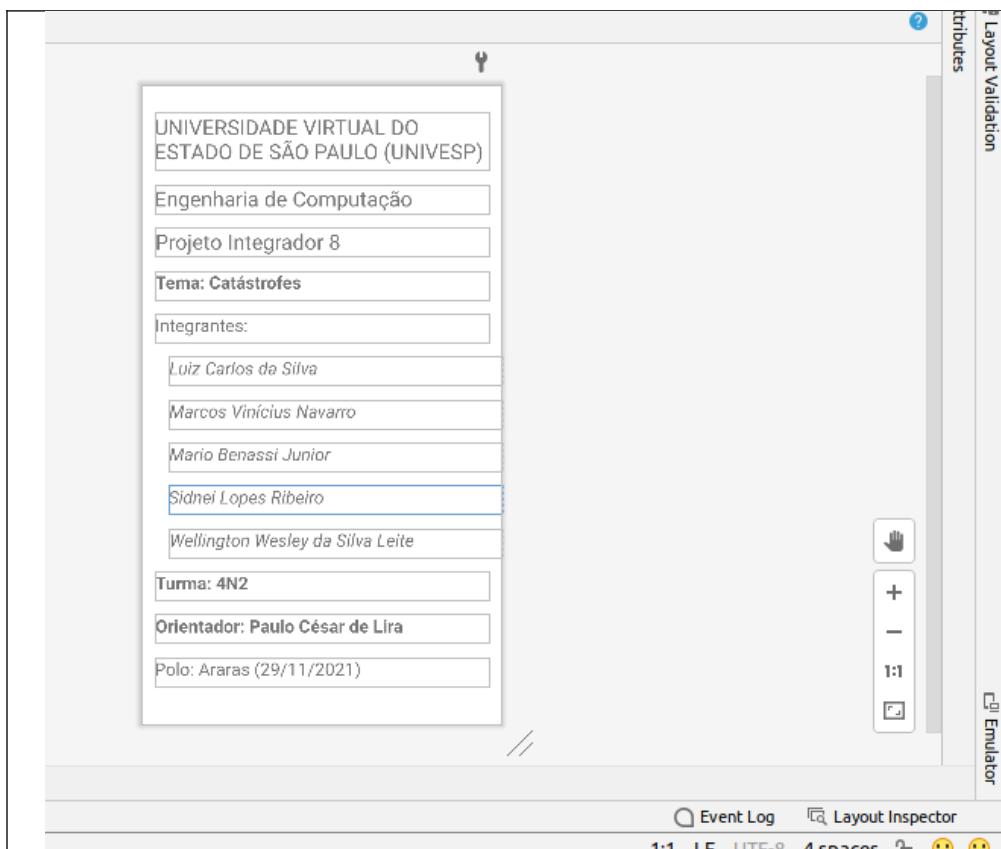
        button_email.setOnClickListener{
            }

        button_app_information.setOnClickListener{
            }
    }
}
```

}

## Apêndice E – Tela Provisória: Ficha Técnica e Integrantes

### 1. Tela



### 2. Códigos

```
<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".ActivityReadme">  
  
<TextView
```

```
    android:id="@+id/textView14"
    android:layout_width="380dp"
    android:layout_height="33dp"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"

    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:text="Polo: Araras (29/11/2021)"
    android:textSize="20sp"
    android:textStyle="normal"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView13" />
```

<TextView

```
    android:id="@+id/textView11"
    android:layout_width="380dp"
    android:layout_height="33dp"
    android:layout_marginStart="32dp"
    android:layout_marginLeft="32dp"

    android:layout_marginTop="16dp"
    android:text="Wellington Wesley da Silva Leite"
    android:textSize="20sp"
    android:textStyle="italic"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView10" />
```

<TextView

```
    android:id="@+id/textView9"
    android:layout_width="380dp"
```

```
    android:layout_height="33dp"
    android:layout_marginStart="32dp"
    android:layout_marginLeft="32dp"

    android:layout_marginTop="16dp"
    android:text="Mario Benassi Junior"
    android:textSize="20sp"
    android:textStyle="italic"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView8" />
```

<TextView

```
    android:id="@+id/textView10"
    android:layout_width="380dp"
    android:layout_height="33dp"
    android:layout_marginStart="32dp"
    android:layout_marginLeft="32dp"

    android:layout_marginTop="16dp"
    android:text="Sidnei Lopes Ribeiro"
    android:textSize="20sp"
    android:textStyle="italic"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView9" />
```

<TextView

```
    android:id="@+id/textView6"
    android:layout_width="380dp"
    android:layout_height="33dp"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
```

```
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:text="Integrantes:"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView5" />
```

```
<TextView
    android:id="@+id/textView12"
    android:layout_width="380dp"
    android:layout_height="33dp"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"

    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:text="Turma: 4N2"
    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView11" />
```

```
<TextView
    android:id="@+id/textView13"
    android:layout_width="380dp"
    android:layout_height="33dp"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"

    android:layout_marginTop="16dp"
```

```
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:text="Orientador: Paulo César de Lira"
    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView12" />
```

```
<TextView
    android:id="@+id/textView7"
    android:layout_width="380dp"
    android:layout_height="33dp"
    android:layout_marginStart="32dp"
    android:layout_marginLeft="32dp"

    android:layout_marginTop="16dp"
    android:text="Luiz Carlos da Silva"
    android:textSize="20sp"
    android:textStyle="italic"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView6" />
```

```
<TextView
    android:id="@+id/textView8"
    android:layout_width="380dp"
    android:layout_height="33dp"
    android:layout_marginStart="32dp"
    android:layout_marginLeft="32dp"

    android:layout_marginTop="16dp"
    android:text="Marcos Vinícius Navarro"
    android:textSize="20sp"
```

```
    android:textStyle="italic"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView7" />
```

```
<TextView
    android:id="@+id/textView5"
    android:layout_width="380dp"
    android:layout_height="33dp"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="16dp"

    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:text="Tema: Catástrofes"
    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView3" />
```

```
<TextView
    android:id="@+id/textView4"
    android:layout_width="380dp"
    android:layout_height="33dp"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="16dp"

    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:text="Engenharia de Computação"
    android:textSize="24sp"
```

```
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />
```

```
<TextView
```

```
    android:id="@+id/textView3"
    android:layout_width="380dp"
    android:layout_height="33dp"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"

    android:layout_marginRight="16dp"
    android:text="Projeto Integrador 8"
    android:textSize="24sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView4" />
```

```
<TextView
```

```
    android:id="@+id/textView"
    android:layout_width="380dp"
    android:layout_height="66dp"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:text="UNIVERSIDADE VIRTUAL DO ESTADO DE SÃO PAULO
(UNIVESP)"
```

```
    android:textSize="24sp"
    app:layout_constraintEnd_toEndOf="parent"
```

```
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```