BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

APLICATIVO ANDROID PARA ENCONTRAR O PONTO DE ENTREGA VOLUNTÁRIA MAIS PRÓXIMO EM RIO VERDE - GOIÁS

PAULO HENRICK

Rio Verde, GO

2022

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO - CAMPUS RIO VERDE
BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

APLICATIVO ANDROID PARA ENCONTRAR O PONTO DE ENTREGA VOLUNTÁRIA MAIS PRÓXIMO EM RIO VERDE - GOIÁS

PAULO HENRICK

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia Goiano - Campus Rio Verde, como requisito parcial para a obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. André da Cunha Ribeiro

Rio Verde, GO

Abril, 2022

Sistema desenvolvido pelo ICMC/USP
Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas - Instituto Federal Goiano

Cabral, Paulo Henrick Freitas
CC117a Aplicativo Android para encontrar o ponto de entrega voluntária mais próximo em Rio Verde - Goiás / Paulo Henrick Freitas Cabral; orientador André da Cunha Ribeiro. -- Rio Verde, 2022.
27 p.

TCC (Graduação em Bacharelado em Ciência da Computação) -- Instituto Federal Goiano, Campus Rio Verde, 2022.

1. Aplicativo android. 2. Google maps. 3. Coleta de lixo. I. Ribeiro, André da Cunha, orient. II. Título.

Responsável: Johnathan Pereira Alves Diniz - Bibliotecário-Documentalista CRB-1 n°2376

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO

PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- ☐ Tese (doutorado)
- ☐ Dissertação (mestrado)
- ☐ Monografia (especialização)
- ☑ TCC (graduação)
- ☐ Artigo científico
- ☐ Capítulo de livro
- ☐ Livro
- ☐ Trabalho apresentado em evento
- ☐ Produto técnico e educacional - Tipo:

Nome completo do autor: Paulo Henrick Freitas Cabral

Matrícula: 2018102201910078

Título do trabalho: APLICATIVO ANDROID PARA ENCONTRAR O PONTO DE ENTREGA VOLUNTÁRIA MAIS PRÓXIMO EM RIO VERDE - GOIÁS

RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: ☑ Não ☐ Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 26 / 04 / 2022

O documento está sujeito a registro de patente? ☑ Sim ☐ Não

O documento pode vir a ser publicado como livro? ☐ Sim ☑ Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

• Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

• Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

• Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Rio Verde (Local) — 26 / 04 / 2022 (Data)

Paulo Henrick Freitas Cabral

Assinatura do autor e/ou detentor dos direitos autorais

Ciente e de acordo: André da Cunha Rabelo

Assinatura do(a) orientador(a)

PAULO HENRICK

APLICATIVO ANDROID PARA ENCONTRAR O PONTO DE ENTREGA VOLUNTÁRIA MAIS PRÓXIMO EM RIO VERDE - GOIÁS

Trabalho de curso DEFENDIDO E APROVADO em 19 de ABRIL de 2022, pela Banca Examinadora constituída pelos membros:

Bruno de Oliveira Costa Couto
Dr. Bruno de Oliveira C. Couto
Instituto Federal Goiano

Marlus Dias Silva
Me. Marlus Dias Silva
Instituto Federal Goiano

André da Cunha Ribeiro
Prof. Dr. André da Cunha Ribeiro
Orientador

Rio Verde, GO

2022

Dedico esse trabalho a minha família, que me deu todo o suporte e apoio para concluir essa longa jornada.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus. Também aos meus pais, que dedicaram suas vidas para me cuidar e moldar quem sou. Por sempre me apoiar e me incentivar nas minhas escolhas.

Aos professores, pelos seus ensinamentos. Em especial, ao professor André Da Cunha Ribeiro, por ter sido meu orientador com dedicação e amizade. E ao professor Bruno de Oliveira Costa Couto, por ter apoiado e sempre se disponibilizado a ajudar.

À Cooperativa de Trabalho de Catadores de Material Reciclável em Geral do Sudoeste Goiano, pela disponibilização dos locais onde possuem Pontos de entrega voluntária na cidade de Rio Verde em Goiás, que foi de grande utilidade para a elaboração deste trabalho.

LISTA DE FIGURAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Com o crescente acúmulo de resíduos sólidos nas cidades brasileiras e a diminuição da capacidade dos aterros sanitários, a opção pela reciclagem representa uma das soluções para esse problema (BIANCHINI, 2001). Um outro ponto é a conscientização da população para alocar os resíduos nos locais corretos, pois ocorre que em locais de difícil acesso, as pessoas desfazem deste lixo de forma mais rápida. E quando o lixo é descartado em locais errados, acaba dificultando a coleta seletiva e trazendo um gargalo para as cooperativa. Pois, antes de iniciarem os processos com este lixo, precisam perder tempo e dinheiro para fazerem uma pré-seleção deste lixo.

Outro fator é o desperdício. Segundo a UNEP (2021) 931 milhões de toneladas de alimentos, ou 17% do total disponível para os consumidores em 2019, foram para a cesta do lixo de domicílios, varejistas, restaurantes e de outros serviços alimentares. De acordo com a revista TERRA (2021), o peso do desperdício global de comida equivale a aproximadamente 23 milhões de caminhões de 40 toneladas totalmente carregados que, se enfileirados, poderiam dar sete voltas na Terra.

Figura 1 – Quantidade de lixo que não passam por transformação

A Compromisso Empresarial Reciclagem (2021), informa que 31,9% dos resíduos urbanos descartados são passíveis de reciclagem, mas apenas 3% passam pela transformação, como mostra a figura acima. De acordo com o Jornal O Tempo (2017) o desperdício de material representa uma perda anual de aproximadamente 120 bilhões de reais para a economia nacional. Logo, se torna urgente que esse quadro seja otimizado através do desenvolvimento de uma aplicação mobile, para que assim seja mais fácil de se conscientizar a população sobre a importância da coleta seletiva e consequentemente mais resíduos descartados sejam reciclados.

Para alcançar o objetivo de reduzir acúmulo de resíduos e o desperdício, realizamos a criação de um aplicativo para celulares com sistema operacional “Android” com o nome de PERV (Nome do aplicativo) a fim de auxíliar as pessoas da comunidade local a localizar o Ponto de Entrega Voluntária (PEV) mais próximo da sua localização atual.

2 REVISÃO DE LITERATURA

2.1 Coleta Seletiva

A coleta seletiva é de extrema importância para o desenvolvimento sustentável e tornou-se uma ação importante na vida moderna devido ao aumento do consumo e consequentemente do lixo produzido. A coleta evita a disseminação de doenças e contribui para que os resíduos se encaminhem para os seus devidos lugares. Separar os resíduos entre plástico, metal, papel e orgânicos também contribui para acabar com poluições tóxicas que contaminam solos e águas de rios, trazendo malefícios imensuráveis ao longo do tempo. Os benefícios da limpeza urbana para a sociedade, em geral, já estão bem estabelecidos, no entanto, questões relativas ao gerenciamento dos resíduos sólidos no Brasil não oferecem uma melhora qualitativa do sistema como um todo (ANJOS, 2001). Há uma forte relação entre a geração de resíduos sólidos e a saúde, seja de forma direta ou indireta, além das agressões ambientais (SANTOS; SILVA, 2009). A população brasileira atual é de 200 milhões de pessoas, de acordo com o Instituto Brasileiro de Geografia e Estatística (IBGE, 2021), também é esperado que cresça a 15% nos próximos 25 anos. Este crescimento trouxe e agravou diversos problemas em âmbitos sociais, econômicos e ambientais. Um destes problemas é a alta produção de resíduos sólidos urbanos que são descartados. Segundo dados do Panorama dos Resíduos Sólidos no Brasil (2020), a geração saiu de 66,7 milhões de toneladas em 2010 para 79,1 milhões em 2019, uma diferença de 12,4 milhões de toneladas. O mesmo estudo diz ainda que cada brasileiro produz, em média, 379,2 kg de lixo por ano, o que corresponde a mais de 1 kg por dia. No Brasil são produzidas cerca de 47.450 mil toneladas de lixo por ano (BLEY JR, 2001). Entre os muitos problemas associados com esse enorme volume de resíduos sólidos, está a sua destinação final. Aproximadamente 76% do lixo produzido no país são dispostos em lixões a céu aberto, o que se constitui na pior forma de deposição. Apenas 1% do lixo passa por algum tipo de tratamento (coleta seletiva, reciclagem e/ou incineração) e o restante é disposto em aterros controlados ou sanitários. Além do impacto ambiental negativo, os lixões colocam em risco a saúde do meio e das populações humanas.

2.2 Tipos de reciclagem

Segundo D'ALMEIDA e VILHENA (2000), várias são as maneiras de classificação do lixo: pela natureza física (seco ou molhado); pela sua composição química (orgânico ou inorgânico); pelos riscos potenciais ao meio ambiente (perigoso, não inerte ou inerte); e pela sua origem (domiciliar, comercial, público, serviços de saúde e hospitalar, portos, aeroportos e terminais rodoviários e ferroviários, industrial, agrícola ou entulho). Adotar a reciclagem significa assumir um novo compromisso diante do ambiente, conservando-o o

máximo possível. Como proposta de educação ambiental, a reciclagem ensina a população não desperdiçar, mas a ver o lixo como algo que pode ser útil (SCARLATO, 1992). Abaixo está especificado os principais tipos de materiais para se reciclar, e em que tipo de categoria ele se encaixa:

- Papel: O que pode ser reciclado: papel de escritório, livros, caixas em geral (sem gordura), jornais (que não tenham sido usados por animais), revistas, cadernos, papel cartão, envelopes, papel de fax, papelão, fotocópias, cartolinas e embalagens longa vida; o que não pode ser reciclado: papéis gordurosos e sujos (por exemplo: caixas de pizza), papel carbono, vegetal, fitas ou etiquetas adesivas, papéis plastificados, fotográficos e parafinados, papel higiênico, lenços de papel, guardanapos.
- Plástico: O que pode: quase tudo o que é feito de plástico pode ser reciclado. CDs e disquetes, sacolas, potes, copos, embalagens, produtos PET (como garrafas de refrigerante), canos, brinquedos, tubos e isopor; o que não pode: fraldas, embalagens metalizadas (de biscoito salgado, por exemplo), esponjas, cabos de panela e plásticos termofixos, com acrílico ou celofane.
- Metal: O que pode: latas de alumínio e de produtos alimentícios (óleo, leite em pó, conservas), tampas de garrafa, pregos, papel-alumínio, embalagens metálicas de congelados, folha-de-flandres; o que não pode: inseticida, aerosol, latas que contenham verniz, além de clipes, tachas, grampos, esponjas de aço, pregos e canos.
- Vidro: O que pode: frascos, garrafas de bebida, potes de produtos alimentícios e copos; o que não pode: lâmpadas, espelhos, cristais, vidros de janelas, vidros de automóveis, vidros temperados, ampolas de medicamento, porcelanas, tubos de TV e de computadores, cerâmicas, refratários e louças de porcelana.
- Tecido: Tecidos podem ser reciclados, mas o processo é muito escasso no Brasil, já que os retalhos costumam ser jogados já degradados e sujos e em pouco volume concentrado. O mais comum é o reuso por artesanato, mas o tecido reciclado pode ser usado como matéria-prima para indústrias de colchões, tecido não tecido (TNT), papel-moeda, produtos medicinais, automobilística, metalúrgica e mobiliária.

Com isso em mente, podemos abordar como reciclar o lixo corretamente. A forma mais simples de fazer a reciclagem de lixo doméstico consiste em separar seus resíduos em duas categorias: recicláveis e não recicláveis. Para isto, basta depositar os lixos em duas sacolas diferentes. É uma ótima ideia que não requer muito esforço.

- LIXO RECICLÁVEL: O lixo reciclável engloba papéis (papelão, jornal, etc.), plásticos, metais e vidros. Cartelas de comprimidos e bandejas de isopor também devem ser descartadas com o lixo seco, pois podem se transformar em matéria-prima para blocos da construção civil.
- LIXO ELETRÔNICO: O lixo eletrônico (pilhas, baterias, celulares e demais eletrônicos) e as lâmpadas devem ser encaminhados para pontos específicos de coleta. No caso de pilhas e baterias de celular, estas podem ser devolvidas diretamente aos

fabricantes, basta entrar em contato com as empresas. Unidades de grandes redes de supermercados também possuem estrutura para receber o descarte desses produtos.

- LIXO ÚMIDO OU LIXO ORGÂNICO: O lixo úmido ou lixo orgânico, é composto de materiais orgânicos (cascas de frutas e legumes, folhas e restos de comida) e não recicláveis (papel higiênico utilizado, materiais de higiene pessoal, plásticos e papéis engordurados, bitucas de cigarro, chiclete, etc.). Ele representa aproximadamente 50% de todo o lixo produzido diariamente por cada brasileiro.

Sabendo melhor como funciona a coleta seletiva e os tipos de reciclagem, podemos avançar e descrever sobre como a computação pode nos auxíliar a gerar uma economia de tempo das pessoas ao realizarem a coleta seletiva e fazer com que mais resíduos descartados sejam reciclados.

2.3 Auxílio da Computação na coleta seletiva

Wing (2006) considera o Pensamento Computacional (PC) uma habilidade fundamental para todos, que permite a resolução de problemas de forma analítica e diz respeito a forma como as pessoas veem e estruturam o pensamento. Dessa forma, o pensamento computacional, bem como a leitura, a escrita e a aritmética, deve fazer parte da capacidade analítica de todas as pessoas, desde a infância. Smith, Sutcliffe e Sandvik (2014) definiram os processos de pensamento utilizados na resolução de um problema, cuja representação permite a execução por um agente de processamento de informações. Este trabalho tem como objetivo criar um aplicativo para dispositivos móveis com sistema operacional "Android", onde o usuário possa encontrar o ponto de entrega voluntária mais próximo de sua localização atual, assim, agilizando seu tempo.

2.4 Dispositivos Móveis

O uso de tecnologia nos dias atuais já é um realidade para boa parte da população. De acordo com RAMÍREZ-CORREA (2015) na ultima década, houve um aumento significativo em busca do acesso a informação através da internet móvel, isso acontece, devido aos avanços tecnológicos. Com esse avanço da tecnologia, os usuários vem buscando por soluções que facilitam ou simplificam sua rotina de alguma forma.

Os dispotivios móveis vieram alterar a facilidade de acesso á informação e á comunicação, fazendo com que a qualquer momento, possamos contactar algo ou sermos contactados. Segundo o Portal de Planos (2021), dispositivos móveis é qualquer aparelho portátil que, por meio de recursos de computação, execute funções e processos, como: reproduzir mídia, navegar na internet, acessar *e-mail*, localização GPS, calculadora e calendário. Os dispositivos móveis podem ser divididos em: *notebooks*, *tablets*, *smartphones*, *smartwatches*, etc.

Os dispositivos móveis, assim como os computadores, precisam de um sistema operacional para processar e coordenar as tarefas executadas. São os sistemas operacionais que permitem que desenvolvedores e programadores criem os diversos aplicativos móveis, ou *apps*, a que temos acesso e usamos nos nossos dispositivos. Mas, além dos sistemas operacionais, muitas vezes também precisamos de acesso a dados móveis para usufruir de diversas funções dos aplicativos e dos dispositivos móveis. O presente trabalho irá desenvolver um aplicativo nativo para dispositivos móveis que possuem o sistema operacional "Android". O motivo de não se desenvolver também para dispositivos móveis com sistema operacional da "Apple", é o seu custo. De acordo com o site oficial da "Apple", Escolhendo uma associação, 2022 , para publicar um aplicativo em sua loja, é preciso fazer parte do programa de desenvolvedores. Este programa, possui um custo de 99 dólares anuais. Já o "Android", o custo é de apenas 25 doláres, pagos uma única vez, que foram pagos por quem desenvolveu o aplicativo.

2.5 Desenvolvimento Mobile

A popularização dos *smartphones* tem contribuído para dispositivos móveis serem mais interativos. Aplicação móvel é aquela que permite a movimentação do usuário durante o seu uso. Segundo SANTOS (2016) o uso de dispositivos móveis tem crescido com consideráveis proporções e vem tornando-se tendência para as mais diversas atividades. Se antes o acesso à informação era mais complexo, com o uso de dispositivos móveis, temos acesso facilitado e, por sua vez, contribuiu para o crescimento do número de usuários em diversos serviços (SANTOS, 2016). O desenvolvimento de aplicativos possuem um nível maior de dificuldade de desenvolvimento em relação a outros tipos de aplicações, como aplicativos *web*. Ao iniciar o desenvolvimento de aplicativos, é necessária a análise de diversos aspectos do projeto para definir pontos técnicos, como as melhores plataformas e metodologias de desenvolvimento (KALEEL; HARISHANKAR, 2013). Com base em tais pontos, nota-se que é importante o contínuo aprimoramento do modo que aplicativos são desenvolvidos, levando em conta não apenas fatores técnicos, mas também fatores ambientais, como o suporte dado para tal atividade.

2.6 Desenvolvimento Android Nativo

O desenvolvimento de aplicativos nativos é a criação de programas executados em dispositivos que rodam o sistema operacional "Android". Para o desenvolvimento é necessário utilizar o kit de desenvolvimento de *software* (SDK) empacotado com Android Studio. O Android Studio é o ambiente de desenvolvimento integrado (IDE), que é um programa de computador. Esse programa reúne todos os requisitos ideais e necessários para que seja possível criar apps para dispositivos móveis "Android".

"Android" é o sistema operacional do Google para dispositivos móveis baseado no Linux. Além disso, a loja virtual Google Play tem aplicativos e jogos tanto gratuitos quanto pagos para os *smartphones* e *tablets* com "'Android".

A plataforma "Android" foi desenvolvida baseada em Linux e sua arquitetura é dividia da seguinte forma:

- Linux é a base de tudo. Todos os drivers de hardware e redes (*Bluetooth*, câmeras, USB e GPS's), sistemas de arquivos e processamento estão incluídos;
- Bibliotecas e Serviços Nativos. Esses são os recursos que já vem com o "Android" para serem utilizados pelo desenvolvedor;
- *Frameworks*. São serviços e bibliotecas (geralmente escritos em Java) que servem para facilitar o desenvolvimento de aplicativos e jogos;
- Aplicativos. São os aplicativos e jogos desenvolvidos em "Java"ou "Kotlin".

O desenvolvimento de aplicativos nativos, nos traz alguns benefícios em relação ao desenvolvimento híbrido (é composto por uma linguagem de programação em conjunto com algum framework, possibilitando a utilização do mesmo código, tanto no sistema IOS como para Android), como:

- Melhor performance: Os aplicativos móveis nativos interagem diretamente com as APIs nativas (*Bluetooth*, câmeras, USB e GPS's) sem depender de bibliotecas externas. Como há menos dependências, os aplicativos móveis nativos são mais rápidos e responsivos do que os aplicativos híbridos. Isso é especialmente importante para aplicativos centrados no desempenho, como jogos e aplicativos com muitos gráficos.
- Aparência consistente: Como os aplicativos móveis nativos são desenvolvidos usando SDKs nativos (kits de desenvolvimento de software), suas UIs parecem consistentes com sua plataforma. Isso garante uma melhor experiência do usuário, pois não há discrepâncias entre o sistema operacional e o design do aplicativo.

Em 2017 durante o Google I/O, evento anual produzido pela Google voltada para o desenvolvimento de aplicações para os seus sistemas operacionais, a empresa adicionou o "Kotlin" à limitada lista de linguagens de programação suportada para o desenvolvimento "Android". Até então, apenas "Java" e "C++" faziam parte dessa lista. Dentre as razões para a escolha do "Kotlin", a Google mencionou o fato da linguagem ser concisa, expressiva e também o fato de que vários desenvolvedores "Android" consideram que o "Kotlin" torna o desenvolvimento mais ágil e divertido. Avram (2022) nós traz o fato do "Kotlin" ser uma linguagem que se integra totalmente ao "Java" e roda na JVM (Java Virtual Machine). Além disso, também é possível invocar código em "C++"/"Android" já que ela suporta JNI (Java Native Interface) por meio de modificadores de acesso externo no código-fonte. A partir do código-fonte do "Kotlin" é possível gerar bytecode para a JVM ou código-fonte "Javascript". Este trabalho irá desenvolver um aplicativo para a plataforma "Android" utlizando a linguagem de programação "Kotlin".

2.7 Representações da informação espacial

Os sistemas de coordenadas com que trabalhamos são uma representação quantitativa da informação espacial. Isto significa que utilizam valores numéricos para indicar de forma unívoca um local na superfície da Terra (FELICE, 2012). Os Sistemas de Informação Geográfica (SIG) utilizam dados quantitativos para armazenar geometrias que representam a informação geográfica. Por serem dados numéricos, estas geometrias podem ser analisadas através de algoritmos estatísticos e de geometria computacional, permitindo a execução de operações complexas como por exemplo interpolação de superfícies e cálculos de rotas (Felice 2012). A localização de qualquer ponto na superficie terrestre pode ser definida quando se dispõe de um sistema de coordenadas como referência. No Sistema de Coordenadas Geográficas WGS-84, a terra é dividida em círculos paralelos ao equador chamados Paralelos e em elipses que passam pelos polos terrestres (perpendiculares aos paralelos) chamados Meridianos. Cada ponto da terra tem um único conjunto de coordenadas geodésicas definidas por latitude e longitude. Com isto, através de uma base de dados composta por dados vetoriais (eixo de ruas, divisas de países), e uma base raster (Os dados raster são compostos por linhas (horizontais) e colunas (verticais) de pixels (também conhecidas como células). Cada pixel representa uma região geográfica, e o valor do pixel representa uma característica dessa região.) composta por um mosaico de imagens de satélites de diversas resoluções e por fotografias aéreas, todas adequadamente georreferenciadas, o Google Maps consegue representar mapas de forma precisa de uma determinada área através de coordenadas geográficas e assim fornecer uma Interface de Programação de Aplicação (API) que permite que o usuário faça calculos de distância entre duas posições diferentes, cálculo de perímetros, entre outros.

2.8 Linguagem de programação Kotlin

Kotlin é uma linguagem de programação criada pela JetBrains e amplamente usada por desenvolvedores Android.

É multiparadigma (suporta vários paradigmas de programação), sendo totalmente orientada a objetos, mas com algumas características de linguagem funcional (O desenvolvimento é feito com base em resultados de funções e a programação é feita com expressões, como se as funções fossem os objetos), com tipagem estática e executada pela Máquina Virtual do Java (Java Virtual Machine). Além disso o kotlin é uma linguagem enxuta e intuitiva, utilizando cerca de 40% menos códigos para representar a mesma coisa que o Java (AGUIAR, 2021). Outro ponto a ser destacado é que por ser estaticamente tipada, essa linguagem oferece mais segurança e performance ao programador Kotlin.

A linguagem de programação kotlin será essencial para este trabalho, através dela o aplicativo proposto por este trabalho será desenvolvido, e também ela que irá interagir com o Kit de desenvolvimento de software (SDK) do Google Maps.

2.9 Interface de Programação de Aplicação

Segundo Melo (2021), Interface de Programação de Aplicação (API) é um termo para designar uma interface de comunicação que um sistema oferece para que outros acessem suas funções, dados e recursos sem que o software ou plataforma externa precise saber como eles foram implementados.

Trata-se de um conjunto de rotinas e padrões muito utilizados na web para facilitar a integração entre diferentes sites e aplicativos. O Google Maps, por exemplo, fornece uma API para que outros produtos utilizem os mapas em seus serviços. Esse comunicação/transferência de dados acontece através de um objeto *Javascript Object Notation* (JSON), que é um formato de arquivo de texto leve, compacto, no qual os dados são guardados no formato de par nome/valor, o qual também pode representar outras estruturas de dados, como arrays e objetos.

O Google Maps, traz um conjunto de APIs para se trabalhar. Eles trazem a *API do Google Places* (Para obter dados de localização para mais de 200 milhões de lugares e adicione detalhes de lugares, pesquisa e preenchimento automático aos seus aplicativos), *Geocoding API* (Converter endereços ou IDs de locais em coordenadas de latitude/longitude e vice-versa), *Geolocation API* (Para retornar um raio de localização e precisão com base nas informações sobre torres de celular e nós WiFi que o cliente móvel pode detectar), *Time Zone API* (A API de fuso horário fornece uma interface simples para solicitar o fuso horário para locais na superfície da Terra, bem como o deslocamento de horário do UTC para cada um desses locais. Você solicita as informações de fuso horário para um par e data específicos de latitude/longitude), *Street View Static API* (Incorporar um panorama ou miniatura do Street View não interativo em uma página da Web), *Roads API* (Identifique as estradas pelas quais um veículo está trafegando e obtenha metadados sobre essas estradas). Apesar dessas diversas funcionalidades, este trabalho fez uso de duas API do Google Maps, a *Matrix API* e *Directions API*. Sobre a *Directions API*, é um serviço para retornar rotas formatadas entre locais. Esta API pode receber rotas para vários meios de transporte, como transporte público, condução, caminhada ou ciclismo. A *Matrix API* é um serviço que fornece distância e tempo de viagem para uma matriz de origens e destinos. A *Matrix API* retorna informações com base na rota recomendada entre os pontos inicial e final, conforme calculado pela API do Google Maps. Quando chamada, é necessário passar alguns parametros, como o ponto de origem (origins), destinos (destinations), modo (mode) que a viagem será feita, além de sempre passar a chave de API única. Um exemplo de requisição para a *Matrix API* é:
https://maps.googleapis.com/maps/api/directions/json?origin=10.3181466,123.9029382
destination=10.311795,123.915864key=APIKEY

Neste exemplo é passado a origem do usuário, e os locais que eu desejo obter a distância, ambos em formato de latitude e longitude, além do modo que a viagem será feita e a chave privada. O retorno é um objetvo JSON, como mostra a Figura 2:

```
{
  "destination_addresses": [
    "R. Nilda De Araujo Q 45, 20 - Santo Agostinho, Rio Verde - GO, 75904-730, Brasil",
    "R. Vinte, N67 - Popular, Rio Verde - GO, 75903-510, Brasil"
  ],
  "origin_addresses": [
    "R. Tiradentes, 434 - Santo Agostinho, Rio Verde - GO, 75904-660, Brasil"
  ],
  "rows": [
    {
      "elements": [
        {
          "distance": {
            "text": "0,2 km",
            "value": 226
          },
          "duration": {
            "text": "1 min",
            "value": 73
          },
          "status": "OK"
        },
        {
          "distance": {
            "text": "1,5 km",
            "value": 1507
          },
          "duration": {
            "text": "4 minutos",
            "value": 255
          },
          "status": "OK"
        }
      ]
    }
  ],
  "status": "OK"
}
```


Figura 2 – Exemplo de retorno da *Matrix API*

Podemos ver que nesse retorno, dentro do objeto *distance*, temos um campo chamado value, que é a distância em metros entre a origem e o destino. Está api possui uma limitação, que será explicada nos materiais e métodos deste trabalho. Para contornar este problema, foi utilizado o haversine.

2.10 Haversine

Para calcular a distância entre 2 pontos, não é adequado usar um cálculo de distância retilínea pois é necessário considerar a circunferência da Terra. Considerando a Terra como um esfera de raio R (correspondente ao raio médio ao longo da linha do equador), então a distância entre 2 pontos $P1$ e $P2$ na superfície da esfera com suas respectivas latitudes e longitudes geográficas a e b e com um diferença entre suas longitudes c pode ser calculada utilizando a formula de Haversine (SINOT, 1984). Dada uma esfera unitária, um "triângulo" sobre a superfície da esfera é definida pelos grandes círculos de ligação de três pontos de u , v , e w sobre a esfera. Se os comprimentos desses três lados são a (de u a v), b (de u a w) e c (de v a w), e o ângulo do canto oposto a c é C , então

a lei dos haversinos afirma :

$$\text{hav}(c) = \text{hav}(a - b) + \sin(a)\sin(b)\,\text{hav}(C). \tag{1}$$

2.11 Kit de desenvolvimento de Software

Kit de desenvolvimento de Software (SDK) consiste em um grupo de recursos de desenvolvimento e códigos com gravação prévia que são utilizados pelos programadores para construir aplicativos destinados a uma plataforma operacional.

Essa tecnologia é, em geral, constituída por um Ambiente Integrado de Desenvolvimento. Esse sistema é equipado de editor que escreve o código, editor visual com a finalidade de estruturar as telas do aplicativo, ferramentas de debug com a função de monitorar e solucionar inconsistências na codificação e um compilador usado na criação do app. Os SDKs contribuem para diminuir o tempo e o esforço que seriam requeridos dos profissionais caso eles fossem escrever os próprios códigos.

Um tipo de SDK é o Android Studio, que consiste em um software usado para desenvolver aplicativos destinados a dispositivos móveis, como tablets e celulares, que, principalmente, tenham sistema operacional Android. Outro exemplo de uma SDK é o Google Maps para "Android", onde é possível adicionar mapas e lidar com os gestos nesse mapa em um app "Android", também é possível fornecer outras informações sobre locais e permitir a interação do usuário adicionando marcadores, polígonos e sobreposições ao seu mapa.

Uma característica importante dessa tecnologia é que ela pode usar APIs na integração de aplicativos, condição que pode ajudar a explicar por que há tanta confusão entre esses dois recursos.

2.12 Kit de desenvolvimento de Software do Google Maps

Para ERLE e GIBSON (2006) Google Maps é um serviço do Google que oferece uma poderosa tecnologia de mapas e informações de locais, incluindo a localização, informações de contatos e direções de condução. Antes de que tivesse uma API pública, alguns desenvolvedores descobriram uma maneira de incorporar os mapas ao seus próprios sites, Isso levou a Google a conclusão que havia a necessidade de uma API pública, e no início de 2005 nas principais localidades dos EUA e posteriormente se expandiu e passou a servir de referência para a busca de endereços e pontos de interesse nos demais centros urbanos de outras nações e continentes - cobrindo várias cidades brasileiras (GOOGLE, 2021a). Para PURVIS, SAMBELLS e TURNER (2006), o grande sucesso e aceitação dos usuários, pouco depois do lançamento oficial, é que os desenvolvedores podem inserir mapas em suas páginas web, contando com a possibilidade de personalização e customização dos mapas como bem entenderem. Para ERLE e GIBSON (2006), a funcionalidade principal

do Google Maps é a exibição de um mapa no website, partindo de uma coordenada que é exibida centralizada na tela. Só isso já basta para usuários que buscam ajuda para localização de ruas e regiões aos redores do endereço fornecido. Como meio de facilitar o entendimento por parte do usuário a visualização do mapa pode ser feita tanto do modo cartográfico - onde aparecem as ilustrações das ruas e quadras - como do modo satélite que exibe uma imagem aérea da área selecionada. O Google Maps foi lançado em 2005 pela Google, atualmente é utilizada por grandes aplicativos no mercado como o Uber. O uso de mapas nas aplicações Android sempre forneceu um aspecto mais profissional ao produto final. Isso se deve em parte à qualidade da própria aplicação Google Maps e de sua imensa adoção entre usuários de todas as plataformas. Com o Kit de desenvolvimento de Software (SDK) do Maps para Android, é possivel adicionar mapas a um app Android, exibições de mapa e respostas de gestos no mapa do Google Maps. Também é possível fornecer outras informações sobre locais e permitir a interação do usuário adicionando marcadores, polígonos e sobreposições ao mapa (GOOGLE, 2021b).

2.13 Banco de dados não relacional

Conhecido inicialmente pelo seu autor Carlo Strozzi em 1998 como "NoSQL", este termo se refere a tipos não relacionais de bancos de dados, e esses bancos de dados armazenam dados em um formato diferente das tabelas relacionais. Outra característica do modelo "NoSQL", que são reconhecidos pela facilidade de desenvolvimento, desempenho escalável, alta disponibilidade e resiliência (BRITO, 2010).

Um banco de dados não relacional não utiliza o esquema de tabela de linhas e colunas encontrado na maioria dos sistemas de banco de dados tradicionais. Em vez disso, os bancos de dados não relacionais usam um modelo de armazenamento otimizado para os requisitos específicos do tipo de dados que está sendo armazenado. Por exemplo, os dados podem ser armazenados como pares chave/valor simples, como documentos JSON ou como um gráfico que consiste em bordas e vértices.

O que esses armazenamentos de dados têm em comum é que eles não usam um modelo relacional. Além disso, eles tendem a ser mais específicos no tipo de dados ao qual dão suporte e no modo como os dados podem ser consultados. Por exemplo, os armazenamento de dados de série temporal são otimizados para consultas de sequências de dados baseadas em tempo, enquanto os armazenamentos de dados de gráficos são otimizados para exploração de relações ponderadas entre entidades. Nenhum dos dois formatos será bem generalizado para a tarefa de gerenciamento de dados transacionais.

2.14 Firebase Firestore

Uma das funções que oferecem suporte a aplicativos móveis de alto desempenho é um banco de dados robusto. Um excelente banco de dados facilita o armazenamento de

informações de maneira bem organizada. Ele também cria uma avenida para recuperar e gerenciar os dados sem esforço, fornecendo funções que ajudam a automatizar várias tarefas de gerenciamento de banco de dados. Uma das plataformas de bancos de dados que oferece tudo isto é o Google Firestore. Ele fornece recursos para gerenciamento avançado de dados e funções em tempo real que aprimoram o desenvolvimento de aplicativos. O Cloud Firestore, também conhecido como Google Firestore, é parte integrante da plataforma Google Firebase. Ele assume a forma de um servidor de banco de dados "NoSQL" baseado em nuvem que faz um excelente trabalho de armazenamento e sincronização de dados. De fato, aplicativos da web e móveis podem interagir diretamente com o Firestore com o uso de SDKs nativos. O Firestore é um banco de dados de alto desempenho que suporta escala automática. Além disso, é muito fácil de usar e muito confiável. Um dos recursos exclusivos é a sincronização de dados em vários aplicativos clientes usando ouvintes em tempo real. O Firestore armazena dados literariamente como documentos que são classificados logicamente em coleções. O documento Firestore oferece suporte para vários tipos de arquivos, números, sequências de caracteres e objetos aninhados. É seguro, confiável e também se integra perfeitamente ao Firebase e ao Google Cloud Platform.

2.15 10 heuristicas de Nielsen

As heurísticas de Nielsen são regras gerais desenvolvidas por Jakob Nielsen em colaboração com Rolf Molich, para realizar "avaliação heurística". Depois de anos, Nielsen evolui o conceito para regras gerais de avaliação do design de interação. As 10 heurísticas são:

- Visibilidade do Status do Sistema: Esta heurística define que é função do sistema informar o seu estado atual em tempo razoável e de forma clara ao usuário, de forma que fique evidente a tarefa executada, ou a função ativada, ou a localização atual; de alguma forma o usuário deve ser capaz de identificar onde ele está e o que ele está fazendo de forma rápida e precisa (NIELSEN, 1994)
- Compatibilidade entre o sistema e o mundo real: Esta heurística define que o sistema deve se adequar a uma linguagem de fácil compreensão para o usuário (NIELSEN, 1994). Compatibilidade entre o sistema e o mundo real, é o conceito que devemos utilizar nas referências do mundo real nos ambientes digitais.
- Controle e liberdade para o usuário: Segundo NIELSEN (1994) o usuário deve ser capaz de realizar a atividade que desejar dentro das possibilidades do sistema, sendo que suas ações não podem fugir as regras de negócio da aplicação, e opções focadas em evitar erros devem ser inclusas entre estas.
- Consistência e Padronização: Padronização de linguagem e representações audiovisuais são importantes a fim demanter o usuário ciente do que está fazendo e facilitar a realização de ações repetitivas; esta padronização evita que o usuário se sinta perdido em diferentes setores do sistema puramente por grande quantidade de diferenças

entres os mesmos (NIELSEN, 1994).

- Prevenção de erros: A fim de evitar que o usuário cometa erros e posteriormente ter de corrigi-los é preferível orientar e informar a respeito de possíveis ou ocorrentes erros antes mesmo de consolidar as ações (NIELSEN, 1994).
- Reconhecimento em vez de memorização: Não deve ser preciso memorizar informações ou procedimentos, ou seja, o usuário deve ser capaz de ter acesso a dados importantes e recomendações de utilização, bem como ajuda e dicas a qualquer momento de maneira fácil e eficiente em qualquer estado do sistema (NIELSEN, 1994).
- Eficiência e flexibilidade de uso: Ao realizar algumas ações com frequência é importante dar ao usuário opções para automatizar ou agilizar estas ocorrências (NIELSEN, 1994).
- Estética e design minimalista: Quanto mais informação disponível ao mesmo tempo mais confusão e dificuldade o usuário terá ao usar o sistema, o design deve conter apenas informações relevantes e a poluição visual deve ser mínima a fim de reduzir distrações minimizando assim erros e má interpretação (NIELSEN, 1994).
- Ajude os usuários a reconhecerem, diagnosticarem e recuperarem-se de erros: Erros devem ser tratados de forma clara e eficiente, as mensagens devem ser diretas e esclarecer o problema além sugerindo uma solução (NIELSEN, 1994).
- Ajuda e documentação: A documentação por si só indica que o sistema não é tão simples de usar, seu entendimento pode ser complicado e algumas atividades podem gerar dúvida por parte dos usuários, mas ela é importante a fim de guia-los em casos de dificuldade, portanto deve ser presente e de preferência simplificada, uma maneira eficiente de tratar a documentação é incorpora-la ao sistema em si (NIELSEN, 1994).

Essas 10 heuristicas podem ser aplicadas a qualquer produto. Mas este trabalho aplica a mesma no desenvolvimento do aplicativo mobile “Android”, para assim trazer uma melhor experiência de uso e interface para o usuário final.

3 MATERIAIS E MÉTODOS

3.1 PEVS e Cooperativa de Lixo

Para saber os PEVS da cidade de Rio Verde em Goiás, a Cooperativa de Trabalho de Catadores de Material Reciclável em Geral do Sudoeste Goiano que é uma sociedade simples, de responsabilidade limitada e sem fins lucrativos, onde foi constituída por deliberação da Assembleia Geral dos Fundadores, realizada no dia 29 de Agosto de 2008, nos termos da Lei 12.690/2012, Lei 5.764/1971 e demais legislações vigentes, forneceu a localização dos PEVS, para serem cadastrados em uma base de dados. A Cooperativa também informou todos os materias que são recebidos. Os itens de coleta são: Alumínio Lata, Alumínio Outros, Cartonado Tetra Pak, Cobre, Eletrônicos, Madeira, Papel Branco, Papelão, Papelão Cimento, PEAD Balde/Bacia, PEAD Branco/Transparente, PEAD Colorido, PEAD Flexível Misto, PEAD Rígido Colorido, PET, PET Branco, PET Colorido, PET Colorido, PET Misto, PET Óleo, Plástico Filme Colorido, Plástico Filme Cristal, PS Colorido, PVC Colorido, Ráfia, Sucata Ferro, Sucata Inox, Sucata Mista, Vidro Caco Colorido, Vidro Caco Transparente.

3.2 Base de dados

Este trabalho utilizou o Cloud Firestore, também conhecido como Google Firestore, que é parte integrante da plataforma Google Firebase, por facilitar o desenvolvimento do aplicativo “Android”, e ser uma plataforma efetiva, rápida e simples. Ele assume a forma de um servidor de banco de dados “NoSQL” baseado em nuvem que faz um excelente trabalho de armazenamento e sincronização de dados. O Firestore é um banco de dados de alto desempenho que suporta escala automática. Esses excelentes recursos explicam o motivo pelo qual este trabalho utilizou Firestore. Foi obtido através do Google Maps, as latitudes e longitudes de cada Ponto de entrega voluntária (PEV) da cidade de Rio Verde em Goiás. Esses dados foram salvos no Firebase Firestore, e utilizados pelo aplicativo “Android” PERV.

3.3 Aplicativo Mobile

Após cadastradas as latitudes e longitudes na base de dados, elas foram utilizadas pelo aplicativo PERV. Foi construído um aplicativo com foco na plataforma “Android”. Para a construção do aplicativo móvel, este trabalho utilizou-se da linguagem de programação “kotlin” e a IDE Android Studio. Para utilizar o mapa dentro do aplicativo, foi utilizado o Kit de desenvolvimento de Software (SDK) do Google Maps, e dentro deste mapa é mostrado todos os Ponto de entrega voluntária (PEV) cadastrados no banco de dados.

O usuário também pode vizualizar os tipos de lixo que podem ser descartados neste local, além de conseguir navegar até este PEV. Para realizar a navegação até o PEV de menor distância, o usuário pode escolher se a rota será feita de carro, bicicleta ou a pé. Logo após escolher é desenhado no mapa a rota que será feita. Está rota é atualizada em tempo real de acordo com a localização do usuário pelo próprio app. Durante o desenvolvimento, foi utilizado a ferramenta de controle de versão de projetos, Git, em um servidor de hospedagem, o GitHub. O código fonte do aplicativo está disponível no endereço: https://github.com/paulohenrickdev/PERV onde todos podem ter acesso por se tratar de um software livre.

3.4 Caminho mínimo

Na teoria de grafos, o problema do caminho mínimo consiste na minimização do custo de travessia de um grafo entre dois nós (ou vértices); custo este dado pela soma dos pesos de cada aresta percorrida. Formalmente, dado um grafo valorado (ou seja, um conjunto V de vértices, um conjunto A de arestas e uma função de peso $f : A \to \mathbb{R}$)
e, dado qualquer elemento v de V, encontrar um caminho P de v para cada v' de V tal que

$$\sum_{p \in P} f(p) \tag{2}$$

é mínimo entre todos os caminhos conectando n a n'.

O Google Maps utiliza-se grafos para realizar o melhor caminho de um destino até aonde queira chegar. Um grafo, como o representado na Figura 3, é uma estrutura formada por nós, representados por pontos, e vértices, linhas que conectam os nós. Onde, os vértices representam as estradas, enquanto os nós são as interseções, ou seja, todos os pontos onde é possível escolher qual estrada seguir (CROVARI, 2019).

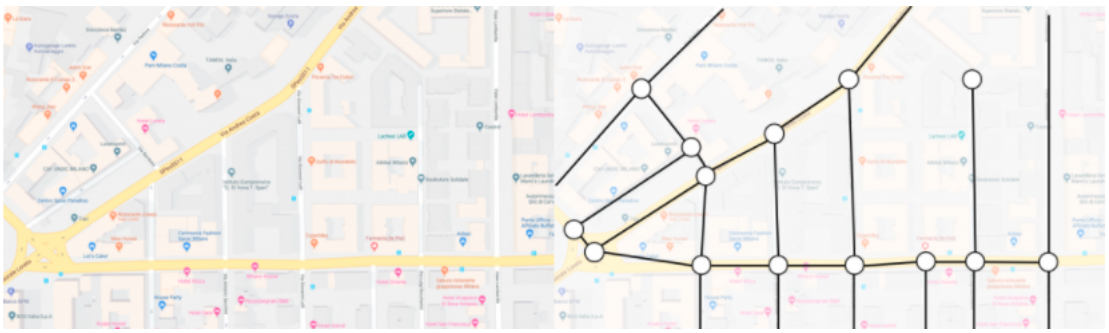


Figura 3 – Fonte (CROVARI, 2019)

3.4.1 Algoritmo de Dijkstra

O algoritmo de Dijkstra soluciona o problema do caminho mais curto num grafo dirigido ou não dirigido com arestas de peso não negativo, em tempo computacional:

$$O(m + n \log n) \tag{3}$$

onde V é o número de vértices e E é o número de arestas.

O Algoritmo 1 apresenta um pseudocódigo do algoritmo de Dijkstra:

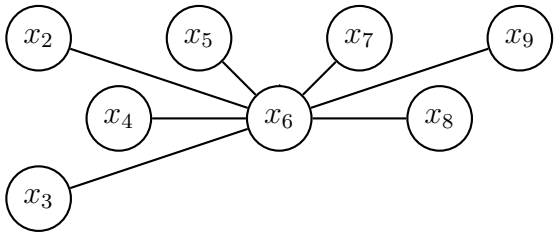
Algoritmo 1: DIJKSTRA(G , w , s)

```
1  início
2  |  INITIALIZE-SINGLE-SOURCE(G,s)
3  |  S = ∅
4  |  Q = V(G)
5  |  enquanto Q ≠ ∅ faça
6  |  |  u = EXTRACT-MIN(Q)
7  |  |  S = S ∪ {u}
8  |  |  para cada vértice v ∈ Adj[u] faça
9  |  |  |  RELAX(u, v, w)
10 |  |  fim
11 |  fim
12 fim
```

O problema de caminho mínimo é um dos problemas genéricos intensamente estudados e utilizados em diversas áreas como Engenharia de Transportes, Pesquisa Operacional, Ciência da Computação e Inteligência Artificial. Isso decorre do seu potencial de aplicação a inúmeros problemas práticos que ocorrem em transportes, logística, redes de computadores e de telecomunicações, entre outros. Neste trabalho foi utilizado a API do Google Maps, que para calcular a menor distância utiliza este algoritmo. Isso ajudará a resolver o problema de quando é preciso retornar ao usuário a menor distância para o Ponto de entrega voluntária (PEV) mais próximo de sua localização atual.

3.5 Haversine

O haversine foi utilizado para contornar o problema de limite da *Distance Matrix API*. A fórmula de Haversine determina a distância do grande círculo entre dois pontos em uma esfera, dadas suas longitudes e latitudes. Importante na navegação, é um caso especial de uma fórmula mais geral da trigonometria esférica, a lei dos haversines , que relaciona os lados e os ângulos dos triângulos esféricos.



No grafo acima, é demonstrado como o haversine foi aplicado. Considerando que o nó $x6$ é a latitude e longitude do usuário, e os outros nós são as latitude e longitude dos PEV's cadastrados na base de dados, é feito o calculo de $x6$ para cada nó. Assim é obtido as menores distâncias em linha reta. Quando obtido essas distâncias, são passadas para a *Distance Matrix API* do Google Maps.

Neste trabalho, este cálculo foi desenvolvido utilizando a linguagem de programação kotlin. Abaixo está o código utilizado:

```kotlin
fun calculationByDistance(
    initialLat: Double,
    initialLong: Double,
    finalLat: Double,
    finalLong: Double
): Double {
    val PI_RAD: Double = Math.PI / 180.0
    val phi1: Double = initialLat * PI_RAD
    val phi2: Double = finalLat * PI_RAD
    val lam1: Double = initialLong * PI_RAD
    val lam2: Double = finalLong * PI_RAD

    val distance: Double =
        6371.01 * acos( x: sin(phi1) * sin(phi2) + cos(phi1) * cos(phi2) * cos( x: lam2 - lam1))

    return distance
}
```

Figura 4 – Código na linguagem de programação kotlin para calcular o Haversine

A Figura 4 apresenta a função que realiza este cálculo utilizando a linguagem de programação kotlin, onde ela recebe 4 parâmetros: Latitude e Longitude inicial, onde estes são respectivos a localização do usuário, e também Latitude e Longitude final, que representa o PEV. Após este processamento é retornado a distância entre esses pontos.

3.6 Google Maps SDK

Este trabalho utilizou a SDK e as APIs *Directions API* e *Distance Matrix API* do Google Maps, para adicionar um mapa ao aplicativo "Android", e assim adicionar marcadores, desenhar e calcular rotas. Ao utilizar a *Distance Matrix API* foi encontrada uma restrição, onde por solicitação, é possível passar um máximo de 25 destinos. Atualmente na base de dados, possui cerca de 60 pontos cadastrados, sendo assim não é possível enviar

todos os pontos cadastrados para a API. Considerando isto, foi utilizado o haversine, para encontrar as menores 25 distâncias, enviando estas distâncias para a API. Com o retorno da API, é dado um retorno dentro do elemento *distance*, que é a distâncias entre a origem do usuário e o PEV. Para realizar essa chamada a API, foi utilizado o "Retrofit" (Biblioteca fornecendo um padrão simples de implementação para transmissão de dados entre aplicação e servidor, que faz uso do JSON). A Figura 5 mostra um exemplo de código que realiza uma chamada para a *Distance Matrix API* :

```
interface ApiMapsDistanceMatrix {
    @GET( value: "distancematrix/json")
    suspend fun getDistanceMatrix(@QueryMap parameters: Map<String, String>): DistanceMatrix
}
val api = Retrofit.Builder()
    .addConverterFactory(GsonConverterFactory.create())
    .baseUrl( baseUrl: "https://maps.googleapis.com/maps/api/")
    .build()
    .create(ApiMapsDistanceMatrix::class.java)

fun getDistanceMatrix(actualLocation: LatLng?, locationsDb: Map<Double, LatLng>?) {
    if(actualLocation == null) return

    var locationsString = ""

    for (i in 0..24) {
        locationsString += "${
            locationsDb?.toList()?.get(i)?.second?.latitude
        },${locationsDb?.toList()?.get(i)?.second?.longitude}|"
    }
    viewModelScope.launch {  this: CoroutineScope
        val newValue = try {
            val mapQuery: MutableMap<String, String> = HashMap()
            mapQuery["origins"] =
                "${actualLocation.latitude}, ${actualLocation.longitude}"
            mapQuery["destinations"] = locationsString
            mapQuery["mode"] = ModeSingleton.modeSelected.mode
            mapQuery["key"] = BuildConfig.MAPS_API_KEY
            api.getDistanceMatrix(mapQuery)
        } catch (e: Exception) {
            Log.i( tag: "MATRIX",  msg: "Error $e")
            throw IllegalArgumentException("Erro ao chamar a API de distancias")
        }

        _locationDistanceMatrix.value = newValue
        Log.i( tag: "MATRIX", newValue.toString())
    }
```

Figura 5 – Código que realiza a chamada API para a *Distance Matrix API*

Com o retorno da API, é obtido qual PEV possui a menor distância, e salvo essa latitude e longitue para ser usada na *Directions API*. A *Directions API*, é um serviço para retornar rotas formatadas entre locais. A API retorna as rotas mais eficientes ao calcular as rotas. O tempo de viagem é o principal fator otimizado, mas a API também

pode levar em consideração outros fatores, como distância, número de curvas e muito mais, ao decidir qual rota é a mais eficiente. É passado para essa API a latitude e longitude atual do usuário e o PEV com a menor distância.


```
"steps": [
  {
    "distance": {
      "text": "0,1 km",
      "value": 106
    },
    "duration": {
      "text": "1 min",
      "value": 19
    },
    "end_location": {
      "lat": 10.3190675,
      "lng": 123.9032266
    },
    "html_instructions": "Siga na direção <b>nordeste</b> na <b>Archbishop Reyes Ave</b> em direção à <b>Apitong St</b>",
    "polyline": {
      "points": "wg~}@swfsVo@u@}@["
    },
    "start_location": {
      "lat": 10.3182038,
      "lng": 123.9028155
    },
    "travel_mode": "DRIVING"
  },
```


Figura 6 – Exemplo de retorno *Directions API*

A Figura 6 mostra um exemplo de retorno desta API, onde dentro do campo *steps*, onde possui um campo *polyline*, e dentro possui um objeto chamado *points*. A própria SDK trata o campo *points*, para decodificar ele. Assim, eu posso obter a rota e desenhar no mapa, auxiliando o usuário a navegar até o Ponto de entrega voluntária (PEV) mais próximo.

4 RESULTADOS E DISCUSSÃO

Foi desenvolvido o aplicativo PERV, baseando nas heuristicas de Nielsen. Suas heuristicas, ajudam a ter um aplicativo com um desing minimalista e boa usabilidade.

Assim que o aplicativo é iniciado, o usuário verá uma tela igual a esta:

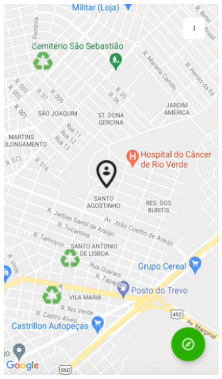


Figura 7 – Página inicial do aplicativo

A Figura 7, mostra onde o usuário está posicionado no mapa através de um ícone preto. Os ícones de reciclagem são onde possuem os pontos de entrega voluntária (PEV) em Rio Verde - Goiás.

Além disso, como mostra a figura abaixo, o usuário poderá escolher como deseja navegar até até o PEV, de acordo com os meios de locomoção disponíveis.

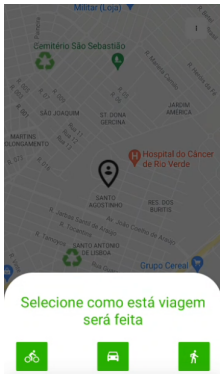


Figura 8 – Escolhendo rota

Outro ponto é que o usuário além de ver sua rota até o PEV mais próximo, sua localização é atualizada em tempo real, como mostra a Figura 9:

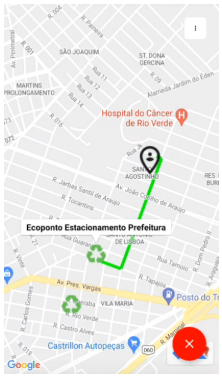


Figura 9 – Rota até o PEV mais próximo

Como dito anteriormente, este trabalho fez o uso das heuristicas de Nielsen. Abaixo, mostra algumas telas que fizeram utilização das heuristicas:

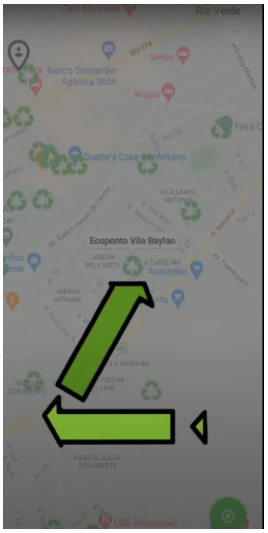


Figura 10 – Exemplo do uso da heurística de visibilidade do status do sistema

No PERV, foi aplicada a heurística de visibilidade do status do sistema. O status está sempre visível ao usuário. Por exemplo, quando o cálculo da rota é iniciado, é mostrado um status de *loading* enquanto o usuário aguarda.

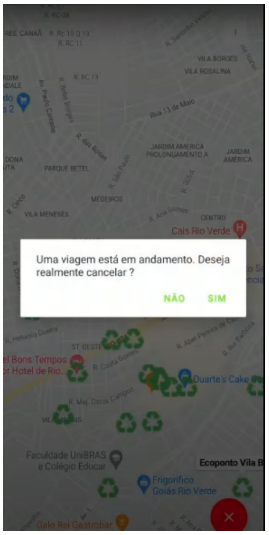


Figura 11 – Exemplo do uso da heurística de compatibilidade entre o sistema e o mundo real

O PERV fez uso da heurística de compatibilidade entre o sistema e o mundo real. O aplicativo utiliza uma linguagem de fácil compreensão para o usuário. Na figura acima, é facil compreender que está prestes a cancelar uma viagem que está em andamento.

Figura 12 – Exemplo do uso da heurística de consistência e padronização

No aplicativo, foi aplicada a heurística de consistência e padronização. Os botões, textos e cores do PERV mantém sempre os mesmos padrãos de tamanho e fonte.

Figura 13 – Exemplo do uso da heurística de reconhecimento em vez de memorização

O usuário não precisa memorizar todos os tipo de lixo que podem ser descartados, basta ele clicar em um botão que saberá isso. Sendo assim, não precisará memorizar. Está heurística é a de reconhecimento em vez de memorização.

Apesar do uso com sucesso das heurística de Nielsen, este trabalho passou por um problemas. Um deles foi ao utilizar a *Matrix API*, onde possuia uma limitação, que por solicitação era possível enviar no máximo 25 origens ou destinos. Este problema foi contornado com a utilização do haversine. Através do cálculo de haversine, foi obtida as menores 25 distâncias entre dois pontos (localização atual do usuário e o PEV), e assim enviadas para a API. Outro obstáculo enfrentado, foi no cadastramentos das latitudes e longitudes dos PEV, pois a inserção foi feita de forma manual, o que acabou consumindo muito tempo.

Apesar dos desafios encontrados, os resultados foram bastante satisfatórios, dado o sucesso no desenvolvimento do aplicativo “Android”, onde o usuário consegue escolher como deseja navegar até o PEV mais próximo, além de sua localização ser atualizada em tempo real. O usuário consegue acessar tudo que é possível ser descartado em um PEV e acessar conteúdos que auxiliam na conscientização. Outro fator de sucesso foi o contorno dos obtáculos citados.

5 CONCLUSÃO

Neste trabalho, buscamos desenvolver um aplicativo para disposítivos móveis com sistema operacional “Android” para que as pessoas da comunidade local consigam localizar e navegar até o PEV mais próximo de sua localização atual.

A base de dados levantada com as latitudes e longitudes dos PEV existentes em Rio Verde - Goiás, foi salvo no “Firebase Firestore” e utilizada com sucesso pelo aplicativo. Com esses dados salvos, foi desenvolvido o PERV utilizando a linguagem de programação “Kotlin” e a IDE “Android Studio”, onde dentro do aplicativo, utilizando a SDK do Google Maps para “Android”, foi possível mostrar no mapa todos os PEV para o usuário. O aplicativo também fez uso das heurísticas de Nielsen com sucesso para uma melhor experiência do usuário. Além disso, o usuário consegue encontrar o PEV mais próximo, selecionando como vai realizar a navegação (A pé, bicicleta ou carro), e sua localização é atualizada em tempo real. Assim, é gerado uma maior economia de tempo do usuário, além de diminuir o lixo comum.

Outro fator importante foi o de instruir o usuário. Dentro do aplicativo possui vídeos que auxíliam sobre a importância e os benefícios da coleta seletiva de lixo, e seus impactos para o meio ambiente. Outro ponto é que o PERV mostra todos os materiais que podem ser descartados em um PEV. Sendo assim, este trabalho atua na conscientização da população de Rio Verde sobre a importância e os benefícios da coleta seletiva de lixo, e seus impactos para o meio ambiente.

5.1 Trabalhos Futuros

O aplicativo PERV pode ser desenvolvido para alcançar dispositivos com sistema operacional *iOS*. Atingindo assim, uma quantidade maior de usuários.

Outro fator relevante, seria a criação de um sistema para cadastrar um PEV. Assim, a localização GPS do PEV, dado por suas respectivas latitudes e longitudes poderiam ser alteradas, deletadas e cadastradas de forma mais fácil e simples, e não diretamente no banco de dados.

Referências

AGUIAR, P. H. C. *Introdução a Kotlin: um guia para começar*. 2021. Disponível em: <https://blog.geekhunter.com.br/introducao-a-kotlin/>. Citado na página 7.

ANJOS, F. Aspectos de saúde coletiva e ocupacional associados à gestão dos resíduos sólidos municipais. *Cad. Saúde Pública*, p. 689–686, 2001. Citado na página 2.

AVRAM, A. *Kotlin agora é uma linguagem oficial no Android*. 2022. Disponível em: <https://www.infoq.com/br/news/2017/06/android-kotlin/#:~:text=Outra%20raz%C3%A3o%20importante%20%C3%A9%20o,acesso%20externo%20no%20c%C3%B3digo%2Dfonte.>. Acesso em: 01 Abril 2022. Citado na página 6.

BIANCHINI, T. Coleta seletiva é a saída. *Ecologia e Desenvolvimento*, 2001. Citado na página 1.

BLEY JR. Lixo no brasil e no mundo. *Seminário Nacional de Resíduos Sólidos e Limpeza Urbana*, 2001. Citado na página 2.

BRITO, R. W. *Banco de Dados NoSQL x SGBDs Relacionais: Análise Comparativa.* 2010. Disponível em: <http://www.infobrasil.inf.br/userfiles/27-05-S4-1-68840-Bancos%20de%20Dados%20NoSQL.pdf>. Acesso em: 28 outubro 2021. Citado na página 11.

Compromisso Empresarial Reciclagem. *Um panorama reciclagem no brasil*. 2021. Disponível em: <http://cempre.org.br/download.php?arq=b18xYWJvNW42MmsxcmEwMTY2ajFobHMxZmEwMTllY2EucGRm>. Acesso em: 04 agosto 2021. Citado na página 1.

CROVARI, P. *Google Maps e teoria dos grafos*. 2019. Disponível em: <https://magazine.impactscool.com/en/speciali/google-maps-e-la-teoria-dei-grafi/>. Citado 2 vezes nas páginas e 15.

D'ALMEIDA e VILHENA. Lixo municipal: manual de gerenciamento integrado. *IPT / CEMPRE*, 2000. Citado na página 2.

ERLE, S.; GIBSON, R. Google maps hacks. Editora O'Reilly, 2006. Citado na página 10.

FELICE, G. D. Reasoning with mixed qualitative-quantitative representtations of spatial knowledge. *Universität Bremen*, 2012. Citado na página 7.

GOOGLE. *Google Maps Api*. 2021. Disponível em: <https://developers.google.com/maps/documentation>. Acesso em: 28 outubro 2021. Citado na página 10.

GOOGLE. *Marcadores*. 2021. Disponível em: <https://developers.google.com/maps/documentation/android-sdk/marker?hl=pt-br>. Citado na página 11.

IBGE. *Projeção da população do Brasil e das Unidades da Federação*. 2021. Disponível em: <https://www.ibge.gov.br/apps/populacao/projecao/index.html?utm_source=portal&utm_medium=popclock&utm_campaign=novo_popclock>. Acesso em: 23 outubro 2021. Citado na página 2.

Jornal O Tempo. *O Brasil perde R$ 120 bilhões por ano ao não reciclar lixo*. 2017. Disponível em: <https://www.otempo.com.br/economia/brasil-perde-r-120-bilhoes-por-ano-ao-nao-reciclar-lixo-1.1423628>. Acesso em: 08 agosto 2021. Citado na página 1.

KALEEL, S. B.; HARISHANKAR, S. Applying agile methodology in mobile software engineering: Android application development and its challenges. *Ryerson University Digital Commons*, p. 213–220, 2013. Citado na página 5.

MELO, D. *O que é uma API? [Guia para iniciantes]*. 2021. Disponível em: <https://tecnoblog.net/responde/o-que-e-uma-api-guia-para-iniciantes/#:~:text=API%20%C3%A9%20um%20acr%C3%B4nimo%20para,vantagens%20neste%20guia%20para%20iniciantes.> Citado na página 8.

NIELSEN, J. Usability inspection methods. New York: John Wiley Sons, 1994. Citado 2 vezes nas páginas 12 e 13.

Panorama dos Resíduos Sólidos no Brasil. *Panorama dos Resíduos Sólidos no Brasil*. 2020. Disponível em: <https://abrelpe.org.br/panorama/>. Acesso em: 11 outubro 2021. Citado na página 2.

Portal de Planos. *Dispositivos móveis: entenda o que entra nessa categoria*. 2021. Disponível em: <https://portaldeplanos.com.br/artigos/dispositivos-moveis/>. Acesso em: 12 março 2021. Citado na página 4.

PURVIS, M.; SAMBELLS, J.; TURNER, C. Beginning google maps applications with php and ajax: From novice to professional. 2006. Citado na página 10.

RAMÍREZ-CORREA, P. Aceptación de internet móvil en estudiantes universitarios brasileños: Un estudio empírico usando modelado de ecuaciones estructurales. v. 36, n. 13, 2015. Citado na página 4.

SANTOS, A. e. a. Combining challenge-based learning and scrum framework for mobile application development. In: . [S.l.: s.n.], 2016. p. 189–194. Citado na página 5.

SANTOS, G. O.; SILVA, L. F. F. Estreitando nós entre o lixo e a saúde – estudo de caso de garis e catadores da cidade de fortaleza, ceará. *Revista Eletrônica do Problema, Fortaleza*, p. 83–102, 2009. Citado na página 2.

SCARLATO. Do nicho ao lixo: ambiente, sociedade e educação. *Atual*, 1992. Citado na página 3.

SINOT, R. "virtues of the haversine,"sky and telescope. v. 68, p. 159, 1984. Citado na página 9.

Smith, Sutcliffe e Sandvik. bringing programming to uk primary schools through scratch. paper presented at the proceedings of the 45th acm technical symposium on computer science education. 2014. Citado na página 4.

TERRA. *Desperdício global de alimento alcança 121 kg per capita; maior parte é em casa*. 2021. Disponível em: <https://www.terra.com.br/economia/desperdicio-global-de-alimento-alcanca-121-kg-per-capita-maior-parte-e-em-casa,b720846d21d5f59fcd24afde99015b299vcptv9n.html>. Acesso em: 21 fevereiro 2022. Citado na página 1.

UNEP. Unep food waste index report 2021. 2021. Citado na página 1.

WING, J. M. Computational thinking. communications of the acm. p. 33–35, 2006. Citado na página 4.