UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

RELATÓRIO FINAL

Navegação em ambientes internos utilizando dispositivos móveis e realidade aumentada

Projeto Orientado em Computação I

Aluno: Giuliano Penido de Paula Júnior
Orientador: Douglas Guimarães Macharet

Belo Horizonte
2023

1. INTRODUÇÃO

A orientação espacial é um desafio que acompanha a humanidade desde os seus primórdios e, ao longo da história, a invenção e o aprimoramento de técnicas de localização e navegação foram divisores de água para o avanço tecnológico e econômico. Os desenvolvimentos da cartografia e de instrumentos como a bússola e o astrolábio, por exemplo, foram fatores centrais para o início da expansão marítima e comercial europeia do século XV, que marcou o início do processo de globalização. A partir da década de 1960, com a idealização do GPS pelo departamento de defesa norte-americano, tornou-se possível determinar posicionamentos ao longo de toda a superfície terrestre com precisão.

Somadas às outras tecnologias como a internet e os dispositivos móveis, o GPS consegue suprir a maior parte das demandas relacionadas à navegação de usuários. Porém, quando o problema é a navegação em ambientes pequenos e fechados, a precisão do GPS não é o suficiente para desenvolver uma solução de qualidade. Esse tipo de serviço tem muitos tipos de aplicações, como por exemplo a orientação dentro de um shopping center ou aeroporto e a localização de produtos em um estoque ou galpão logístico.

Existem muitas técnicas e tecnologias utilizadas em soluções de navegação interna e *wayfinding*, porém, a grande maioria exige uma infraestrutura de comunicação para localizar os agentes no ambiente, como por exemplo *beacons* de *Bluetooth* ou *wifi*. Apesar de conseguirem solucionar o problema, a necessidade de investir na compra e na instalação de equipamentos torna essas soluções inviáveis em alguns casos. Nesse contexto, a alternativa de utilizar visão computacional e/ou *pedestrian dead-reckoning* torna-se muito interessante, levando em conta que é uma proposta que envolve apenas *software*, explorando os artefatos de *hardware* de dispositivos cotidianos dos usuários, como *smartphones* e *smartwatches.*

Com base nas motivações por trás da navegação em ambientes internos e o potencial da visão computacional na solução desse problema, a proposta desse projeto é o desenvolvimento de uma aplicação mobile capaz de mapear ambientes e realizar a navegação de usuários utilizando realidade aumentada.

2. REFERENCIAL TEÓRICO

Sistemas de navegação interna para humanos são compostos por três diferentes módulos[1]: módulo de posicionamento, módulo de navegação e módulo de interação humano-computador. O sistema de posicionamento estima a posição do usuário dentro do ambiente, o de navegação faz o roteamento para o destino final com base na posição atual e o

módulo de interação mostra ao usuário de forma gráfica as instruções de deslocamento e permite com que ele interaja com o sistema.

2.1. TECNOLOGIAS E TÉCNICAS DE POSICIONAMENTO INTERNO

Dentre essas três partes de um sistema de navegação interna, o grande desafio de desenvolvimento encontra-se na parte de posicionamento, tendo em vista a alta precisão exigida nas estimativas de posição dos usuários e dos destinos. As soluções encontradas no mercado propõem diferentes formas de determinar posições.

2.1.1. Tecnologias de comunicação

Os métodos mais utilizados até então para definir o posicionamento de usuários em ambientes internos envolvem a utilização de alguma rede de equipamentos que se comunicam entre si e conseguem estimar a posição dos usuários com base na intensidade dos sinais recebidos ou no tempo de resposta[2]. Os princípios por trás dessa alternativa são parecidos com os utilizados pelo GPS, porém, no caso da navegação interna, são necessários equipamentos capazes de prover maiores precisões, como por exemplo *beacons* de *Bluetooth* ou de *wifi* e *VLCs (Visible Lighting Communication)*.

Ao implementar uma rede de comunicação desse tipo, a escolha do equipamento geralmente leva em conta a disponibilidade de energia, o orçamento e a precisão mínima necessária no projeto. As soluções de *BLE(Bluetooth Low Energy)*, por exemplo, são mais acessíveis financeiramente e consomem menos energia do que as que utilizam *VLC*, porém são menos precisas[3].

2.1.2. Visão Computacional

As soluções que utilizam visão computacional trabalham em cima de imagens coletadas por câmeras espalhadas pelo ambiente ou até mesmo a do próprio celular do usuário. Essas imagens podem ser utilizadas como entrada para algoritmos de deep learning, por exemplo, para estimar a distância da câmera para objetos reconhecidos no ambiente e, assim, determinar posições[4].

2.1.3. Pedestrian dead-reckoning (PDR)

O PDR é um método de posicionamento que busca estimar a posição do usuário com base em dados fornecidos por sensores *IMU (Inertial Measurement Unit)* como acelerômetros, giroscópios e magnetômetros. Os sistemas que utilizam PDR conseguem inferir como o usuário se movimentou ao longo do tempo combinando as saídas de todos esses sensores [5].

2.1.4. Localização e mapeamento simultâneos (SLAM)

O SLAM é uma técnica computacional de mapear um ambiente desconhecido enquanto simultaneamente rastreia-se a movimentação de um agente. Os algoritmos de SLAM podem ser baseados em sistemas chamados de *fusion systems*[6], que são a união de duas ou mais técnicas de posicionamento dentre as citadas até então. No caso deste projeto, utilizou-se uma ferramenta de realidade aumentada para celulares desenvolvida pelo Google chamada ARCore[7], que baseia-se na junção dos dados das imagens da câmera com os dados coletados pelos sensores IMU. O ARCore detecta objetos e planos nas imagens e os utiliza para calcular mudanças de localização. A informação visual é combinada com as medições inerciais para estimar a posição e orientação da câmera em relação ao ambiente ao longo do tempo[8].

2.2. PROPOSTA DE SOLUÇÃO

A proposta do projeto é o desenvolvimento de um aplicativo para dispositivos Android que oferece aos usuários dois modos de utilização dentre os quais será possível alternar na tela inicial da aplicação.

Figura 1 - Menu inicial do app

2.2.1. Modo mapeamento

O primeiro modo é o de mapeamento, em que o usuário define um ponto de início do mapa que será construído. Esse primeiro ponto será definido a partir da identificação de um QR Code que contém o identificador do local, que poderá representar o número de uma sala ou o nome de um produto, dependendo da utilização do app.

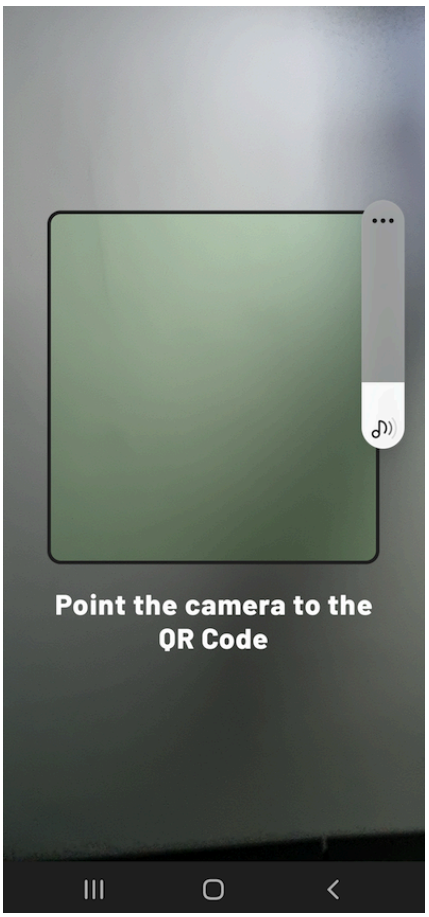

Figura 2 - Tela de scan de QR Code

Partindo desse ponto inicial, o usuário poderá percorrer o restante do ambiente e fazer marcações de pontos e conexões por todo o local a fim de criar um grafo representativo do espaço.

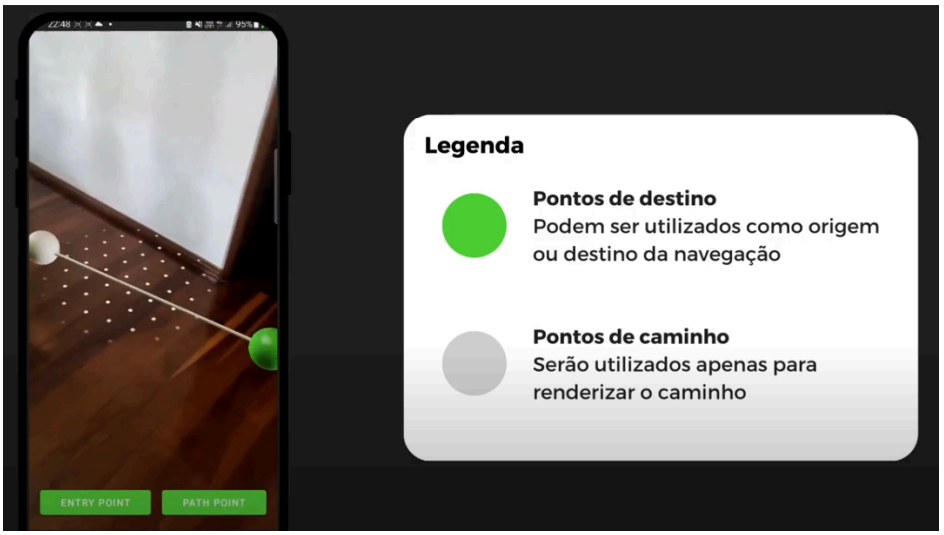

Figura 3 - Exemplo de grafo em construção

O modo mapeamento permite ao usuário inserir dois tipos distintos de marcações no ambiente. Os *Path Points* são marcações intermediárias entre dois QR Codes espalhados pelo local e tem como única finalidade popular o grafo a ser construído. Esses pontos são úteis para criar curvas, cruzamentos e desviar de possíveis obstáculos existentes nos caminhos. Já os Entry Points, identificados pela coloração verde, são pontos que devem ser colocados junto aos QR Codes e sempre possuem um identificador associado.

2.2.2. Modo navegação

Dado um ambiente já mapeado, é possível utilizar o segundo modo do app: navegação. Ao iniciar o modo navegação, o usuário deverá ler o QR Code correspondente ao local em que ele se encontra e depois selecionar um destino final dentre os pontos do grafo para visualizar o percurso encontrado.

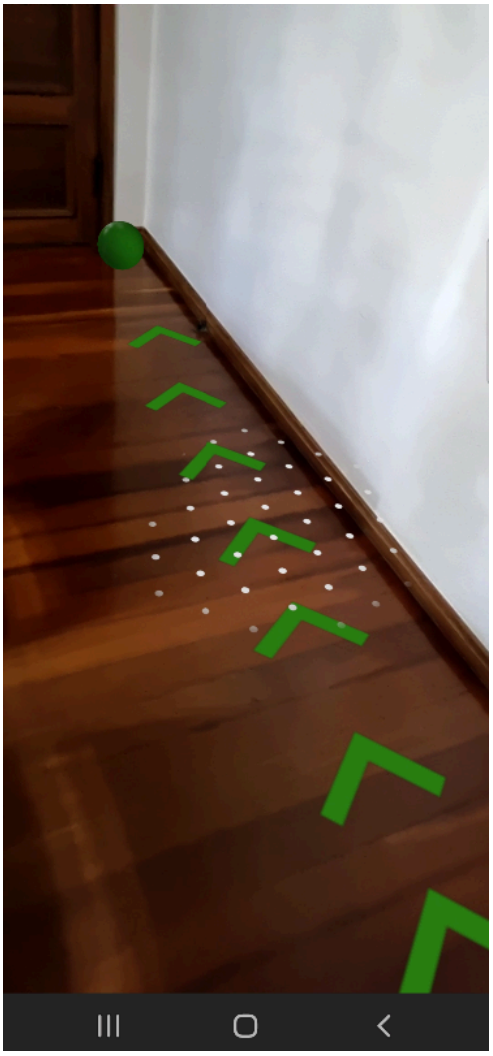

Figura 4 - Exemplo de caminho rota traçada pelo app

3. DESENVOLVIMENTO

3.1. AMBIENTE DE DESENVOLVIMENTO

Foi desenvolvida uma aplicação mobile nativa para o sistema operacional Android utilizando o SDK Android 33 e a linguagem Kotlin na versão 1.9.0. A compilação do projeto foi automatizada utilizando um automatizador muito comum no ecossistema de desenvolvimento Java/Kotlin chamado Gradle, na versão 7.4.

O requisito mínimo para a instalação do app em um dispositivo Android é a versão mínima do sistema operacional 7, que já abrange cerca de 94%[9] dos dispositivos em uso no mundo no ano de desenvolvimento do projeto. Outro requisito para uso do app é a concessão de permissões de acesso à câmera e localização do dispositivo, exigências da biblioteca ARCore, que precisa de acesso à localização para complementar os dados utilizados pelo algoritmo de SLAM.

3.2. MÓDULOS DO SISTEMA

Com base na proposta apresentada, subdividiu-se o projeto nos mesmos três módulos que compõem um sistema de navegação interna e nas seções a seguir serão apresentados detalhes técnicos da implementação do app.

3.2.1. Módulo de posicionamento

O módulo de posicionamento do projeto baseia-se na biblioteca ARCore do Google para desenvolvimento mobile, que utiliza técnicas de SLAM para definir o posicionamento do usuário no ambiente.

Ao iniciar a aplicação, independente do modo em que o usuário selecionou, o local do ambiente é desconhecido e de imediato a biblioteca inicia o mapeamento do ambiente e o *tracking* da posição. Paralelamente, o app inicia a busca por um QR Code coletando e processando frames da câmera do dispositivo a cada meio segundo para identificar a presença do código. O processamento dessas imagens é feito utilizando uma biblioteca também do Google chamada ML Kit[10].

3.2.1.1. ML Kit e QR Codes

O ML Kit é um SDK mobile open-source desenvolvido pelo Google que implementa uma série de algoritmos de Machine Learning voltados para visão computacional. Dentre as possíveis funcionalidades, utilizou-se o *barcode scanning*[11] para buscar QR Codes dentro das imagens coletadas e extrair os dados neles contidos.

Existem diferentes tipos de QR Code[12] que diferem-se entre si pelo tipo de dado que ele codifica. Os códigos utilizados no projeto são do tipo *URL* e representam um endereço que deve possuir o seguinte formato:

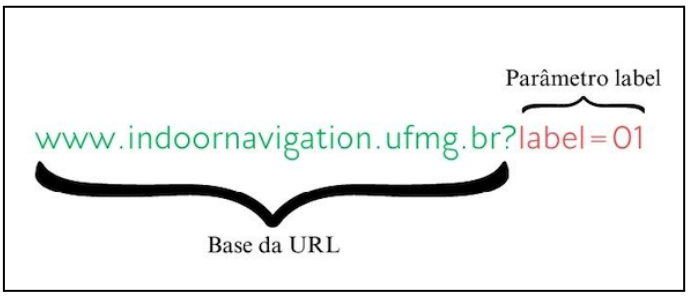

Figura 5 - Estrutura da URL do QR Code

A *URL* armazena o identificador do local que o QR Code representa no parâmetro *label*. Esse identificador é do tipo *String* e será posteriormente utilizado para buscar o local que ele marca.

Assim que um dos processamentos do ML Kit retorna que encontrou um QR Code válido em um frame analisado, o app inicia o processo de posicionar uma marcação de ponto na tela do usuário. Para conseguir renderizar e fixar um ponto nas imagens visualizadas pela câmera, é necessário extrair informações de posicionamento do local onde o QR Code se encontra através de um procedimento chamado hit-test[13].

3.2.1.2. Hit Test

O hit-test é um recurso disponibilizado pelo ARCore capaz de determinar coordenadas dentro do ambiente e dados de rotação do primeiro objeto geométrico identificado na cena. No caso de uso do projeto, o hit-test funciona como se um raio fosse disparado pela câmera na direção das coordenadas centrais do QR Code identificado pelo ML Kit e percorresse o ambiente até colidir com um objeto ou plano. Assim, a partir dos dados de posicionamento obtidos pela colisão do hit-test, é possível ancorar um modelo 3D ou imagem

no ambiente de forma que ele se mantenha fixo na cena mesmo com movimentação e rotação do celular. Esse processo de ancoragem é feito com o sistema de âncoras do ARCore.

3.2.2. Módulo de navegação

Uma vez que o sistema de posicionamento do app já é capaz de identificar a posição do celular do usuário e posicionar pontos de marcação no ambiente, torna-se possível construir o sistema de navegação.

3.2.2.1. Grafo de navegação

No modo mapeamento do app, o usuário poderá inserir uma série de pontos no ambiente de forma a construir um grafo que o represente. Cada novo ponto inserido será ligado ao último ponto que está selecionado. Para selecionar um novo ponto, basta tocá-lo e uma seta indicativa será posicionada para mostrar que a seleção foi bem-sucedida. Essa seleção baseia-se no ARCore que consegue identificar cliques em âncoras posicionadas no ambiente, ou seja, é possível interceptar cliques nos nós do grafo e selecioná-los.

Figura 6 - Exemplo de nó selecionado

O motivo por trás da escolha de mapear o local por meio de pontos e ligações ao longo do espaço tem como objetivo reduzir o problema de *wayfinding* da navegação interna ao problema de encontrar caminhos entre dois pontos de um grafo. Entretanto, para que isso seja possível ainda é necessário uma maneira de salvar esses pontos de forma que eles não sejam perdidos entre diferentes sessões do app.

3.2.2.2. Persistência dos dados entre sessões

O funcionamento do sistema exige que as informações de localização dos pontos marcados durante o posicionamento e as ligações entre eles sejam persistidas para que o grafo do ambiente possa ser reutilizado em sessões subsequentes do app.

No caso da versão desenvolvida para a primeira disciplina desse projeto, todas as informações são armazenadas localmente no celular do usuário por meio de um banco de dados relacional SQLite. Em outras palavras, o mapeamento precisa ser realizado no mesmo dispositivo que será utilizado para navegar no ambiente. No entanto, é importante ressaltar que a arquitetura utilizada no desenvolvimento permite a substituição desse método de persistência por outro que armazena os dados remotamente de maneira muito simples. A arquitetura será abordada com mais detalhes posteriormente.

É importante ressaltar que essa primeira versão do app também só permite a execução do modo mapeamento uma vez e para um único ambiente. Em outras palavras, o mapeamento do ambiente completo deve ser feito em uma mesma sessão e, caso o modo seja iniciado novamente depois, o banco de dados será reiniciado para receber um novo ambiente, resultando na perda dos pontos anteriores. Essa decisão foi tomada com o objetivo de simplificar a parte de persistência de dados na primeira versão do projeto, mas nada impede que avanços sejam desenvolvidos de forma a comportar múltiplos armazenamentos em uma versão posterior.

Utilizou-se também um *Object Relational Mapper* (ORM) muito comum no ecossistema de desenvolvimento Android chamado Room[14]. O Room é capaz de abstrair a construção de queries e o mapeamento de resultado de consultas para classes Kotlin de forma a aumentar a produtividade durante o desenvolvimento.

A modelagem do banco utilizado segue o seguinte diagrama:

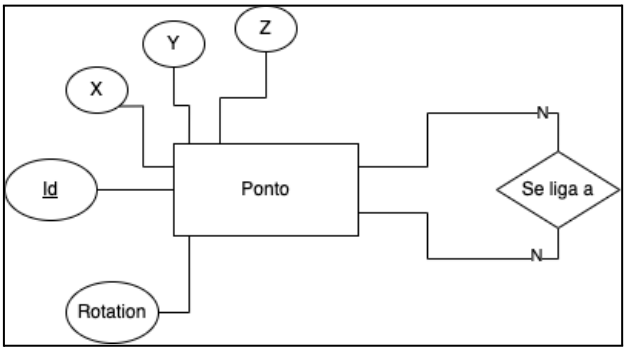

Figura 7 - Modelagem do banco de dados da aplicação

3.2.2.3. Carregamento do grafo entre sessões

A cada vez que o app é reiniciado, todas as informações obtidas do ambiente pela tecnologia de SLAM do ARCore são perdidas e a coleta é reiniciada. Assim, não há nenhuma garantia que as coordenadas dos pontos de interesse entre sessões subsequentes sejam as mesmas. Sendo assim, é necessário uma forma de atualizar as antigas informações de posição dos pontos armazenados durante o mapeamento do ambiente para adequá-los ao novo contexto.

Esse processo é realizado usando como base a posição do primeiro QR Code encontrado durante a navegação e que será utilizado como ponto inicial do percurso. Ao encontrar esse QR Code, obtém-se as coordenadas nas quais ele se encontra e elas são utilizadas para calcular o deslocamento que existe entre as coordenadas do mapeamento com as novas obtidas. Essa diferença é utilizada para fazer o deslocamento de todos os outros pontos da base, tornando-os assim compatíveis com a sessão atual.

3.2.2.4. Busca de caminhos

Quando o usuário inicia o app no modo navegação e escaneia o QR Code mais próximo, é apresentado um modal que o permite selecionar um possível destino dentre todos os *Entry Points* representados pelo identificador do QR Code. A partir da seleção, cabe ao app encontrar um possível caminho entre esses dois pontos do grafo do ambiente.

Para essa tarefa, considerou-se que o grafo não é direcionado e que há um peso em cada aresta correspondente à distância euclidiana entre os pontos que a originam. Com base nessas premissas, optou-se pela utilização do algoritmo A* para solucionar o problema de encontrar o caminho com uma heurística que estima a distância dos pontos até o ponto de destino com a distância euclidiana.

O motivo para a utilização desse algoritmo, além da simplicidade de implementação, deve-se à sua performance, que pode ser um diferencial caso o projeto seja utilizado em ambientes muito grandes e com muitos nós. É importante ressaltar que, dado que a heurística escolhida é admissível, ou seja, nunca superestima a distância entre os pontos, também é possível afirmar que o algoritmo sempre resulta no caminho mais curto possível.

3.2.3. Módulo de interação humano-computador

Toda a parte visual da aplicação foi desenvolvida por meio do sistema de hierarquia de views do Android que utiliza layouts em XML para criar as interfaces gráficas. Além disso, utilizou-se um componente do SDK Android chamado Activities para associar os layouts XML com o código Kotlin que continha a lógica de controle de UI do app.

A seguir, serão apresentados os componentes e telas com os quais os usuários podem interagir e uma explicação por trás do seu funcionamento

3.2.3.1. Menu inicial

A tela inicial do aplicativo é um menu estático que apresenta uma breve explicação de cada um dos modos de funcionamento do app para o usuário. Nesse momento, o usuário poderá selecionar em qual dos dois modos ele deseja inicializar o app ao clicar em um dos dois *cards* apresentados.

3.2.3.2. Barra de ações do modo mapeamento

Figura 8 - Barra de ações do app

Enquanto o usuário navega pelo ambiente no modo mapeamento, é apresentada uma barra de opções que permite que ele interaja com o app para marcar os pontos no ambiente. É importante ressaltar que essa barra só é disponibilizada após a identificação do primeiro QR Code correspondente ao primeiro nó do grafo criado.

O usuário poderá criar *Path Points* ao pressionar o botão identificado pelo texto “Path” e criar Entry Points ao pressionar o outro botão. Entretanto, para habilitar o botão de Entry Points, o campo de texto logo acima não poderá estar vazio, caso contrário, seria criado um nó correspondente a um QR Code sem um identificador válido.

3.2.3.3. Modal de seleção de destino de navegação

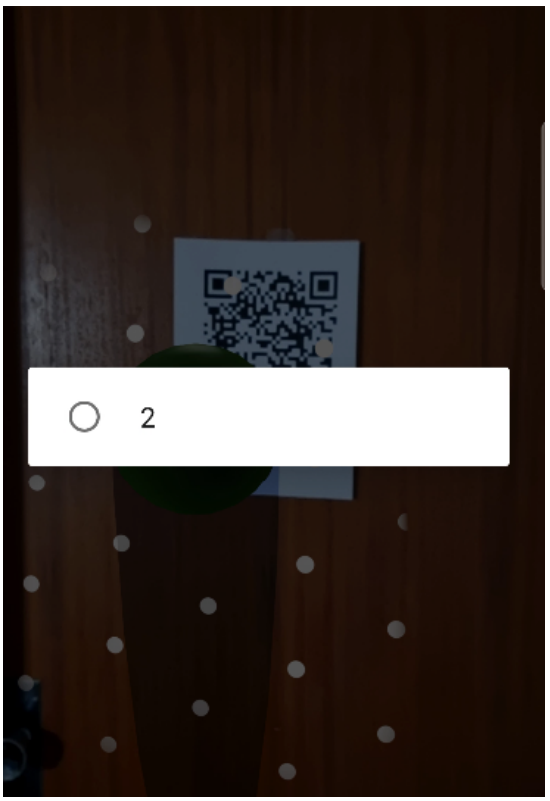

Figura 9 - Modal de seleção de destino

Após identificar o QR Code de início do percurso no modo navegação, um modal com os possíveis destinos do usuário será apresentado. Nesse momento, o usuário deverá buscar e pressionar o destino desejado para disparar os algoritmos responsáveis por encontrar e mostrar o caminho encontrado.

3.3. ESTRUTURA DO PROJETO

3.3.1. Organização de arquivos

Os arquivos criados no projeto foram organizados segundo a seguinte estrutura, de forma a separar arquivos relacionados à parte visual do app, à lógica dos algoritmos e à persistência de dados.

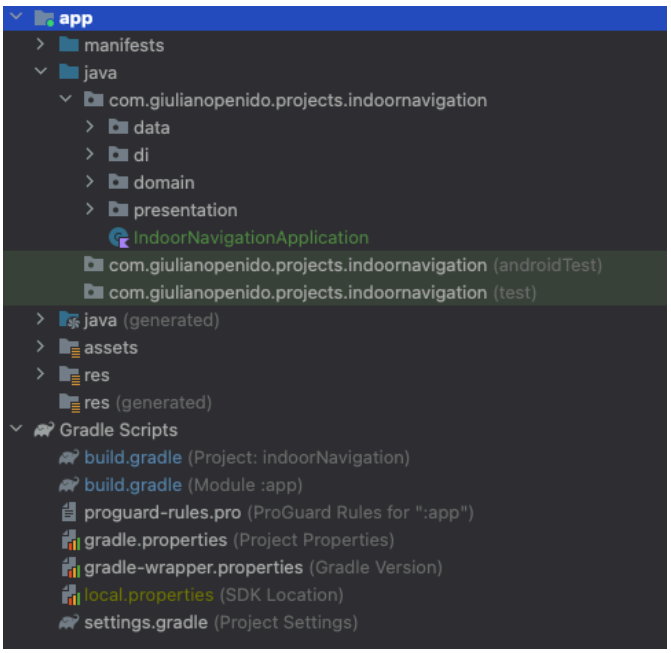

Figura 10 - Estrutura de diretórios do projeto

3.3.2. Arquitetura utilizada

O aplicativo foi desenvolvido utilizando uma arquitetura muito comum no desenvolvimento de aplicações mobile chamada *Model-View-ViewModel* (MVVM)[15]. Essa arquitetura tem o intuito de quebrar o controle da aplicação em três diferentes entidades. A *View* é a entidade que tem como única e exclusiva responsabilidade lidar com o controle e construção das interfaces. A *Model* é responsável pelas regras de negócio das aplicações e persistência dos dados. Já a *ViewModel* é a camada intermediária entre as outras duas, responsável por chamar e tratar os resultados da camada *Model* e transmitir para a *View* as alterações necessárias na UI.

4. RESULTADOS OBTIDOS

No final do processo de desenvolvimento, o resultado foi uma aplicação que ficou de acordo com as expectativas iniciais do projeto. A solução proposta consegue solucionar o problema de navegação interna e *wayfinding* levantado inicialmente, tendo em vista que o usuário consegue navegar de maneira eficiente entre 2 pontos de um ambiente pequeno e fechado com uma precisão satisfatória.

Um dos principais desafios do projeto era desenvolver uma estratégia para garantir que o grafo construído durante o mapeamento pudesse ser reconstruído em uma sessão

posterior de uso do app. No final, a solução proposta de translocar os pontos a partir do cálculo de deslocamento de um ponto inicial se mostrou satisfatória para ambientes pequenos, com uma margem de erro tolerável a depender da sua aplicação.

A utilização da ARCore para fazer mapeamento do ambiente e renderização de formas geométricas ancoradas no espaço com o objetivo de abstrair um pouco a complexidade dos algoritmo de visão computacional foi uma escolha também satisfatória. A agilidade de desenvolvimento que a biblioteca provê permitiu que a solução avançasse com mais confiabilidade.

A opção de persistir os dados de mapeamento localmente no celular utilizando um banco de dados relacional, embora não seja a melhor alternativa pensando em um produto, reduziu a complexidade da solução e permitiu que um foco maior fosse dado aos demais algoritmos por trás do wayfinding e renderização dos componentes visuais.

5. CONCLUSÃO

Conforme levantado inicialmente, a navegação interna em ambientes fechados é um grande desafio com inúmeras aplicações cujas soluções mais utilizadas até então sofrem pela necessidade de alto investimento em equipamentos ou pela falta de precisão. Partindo desse problema, o intuito do projeto era desenvolver um sistema para dispositivos celulares capaz de fazer mapeamento e wayfinding nesse tipo de ambiente utilizando realidade aumentada. Ao final do projeto, é possível afirmar que chegou-se em um protótipo que cumpre com os requisitos iniciais apresentados.

Entretanto, conforme foi sinalizado na proposta do trabalho, esperava-se que essa solução tivesse um problema de escalabilidade pelo seguinte motivo: tendo em vista que o modo de mapeamento depende da criação de rotas por meio de muitos pontos com pequena distância entre si, torna-se inviável a sua utilização em ambientes muito grandes. Na fase de testes do projeto, notou-se que não só o mapeamento é um problema para ambientes grandes, mas quanto maiores as rotas percorridas, maiores são os erros acumulados de translocamento dos pontos entre diferentes sessões, o que pode tornar o caminho incorreto dentro do ambiente. Além disso, adotou-se uma estratégia de implementação que carrega todo o grafo do ambiente em memória ao iniciar a aplicação, o que também se tornaria um problema de escalabilidade, dado a limitação de memória dos celulares.

Esses pontos podem ser trabalhados em uma segunda etapa do projeto, de forma a adotar uma estratégia mais eficiente de mapeamento capaz de reduzir o tempo e esforço do processo. Estratégias para carregar pequenas partes do grafo de cada vez na memória à

medida que o usuário percorre o caminho e descartar as partes anteriores também seriam uma evolução natural do projeto. Outra possível sugestão de melhoria é adicionar a funcionalidade de reabrir um grafo mapeado em uma sessão anterior no modo de mapeamento e continuar a sua construção, dado que no atual estágio do projeto, o banco de dados é reiniciado a cada vez que o modo mapeamento é aberto.

6. REFERÊNCIAS

[1] Kunhoth, J., Karkar, A., Al-Maadeed, S. et al. Indoor positioning and wayfinding systems: a survey, 2020. Hum. Cent. Comput. Inf. Sci. 10, 18.

[2] Indoor Navigation Technology: A Comparison. Waymapnav, 2022. Disponível em: https://waymapnav.com/indoor-navigation-technology-a-comparison/

[3] El-Sheimy, N., Li, Y. Indoor navigation: state of the art and future trends, 2021. Satell Navig 2, 7.

[4] Hui Ng, Xin; Ning Lim, Woan. Design of a Mobile Augmented Reality-based Indoor Navigation System, 2019. International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Istanbul, Turkey, 2020, pp. 1-6

[5] Wu, Y., Zhu, HB., Du, QX. et al. A Survey of the Research Status of Pedestrian Dead Reckoning Systems Based on Inertial Sensors, 2019. Int. J. Autom. Comput. 16, 65–83.

[6] Perez-Navarro, A.; Montoliu, R.; Torres-Sospedra, J. Advances in Indoor Positioning and Indoor Navigation. Sensors 2022, 22, 7375.

[7] Overview of ARCore and supported development environments. ARCore, 2022. Disponível em: https://developers.google.com/ar/develop

[8] Fundamental concepts. ARCore, 2022. Disponível em: https://developers.google.com/ar/develop/fundamentals

[9] Belinski, Eugene. Android API Levels. 2023. Disponível em: https://apilevels.com/

[10] ML Kit guides. ML Kit, 2022. Disponível em: https://developers.google.com/ml-kit/guides

[11] Barcode Scanning. ML Kit, 2022. Disponível em: https://developers.google.com/ml-kit/vision/barcode-scanning

[12] QR Code Security: What are QR codes and are they safe to use? AO Kaspersky Lab. 2023. Disponível em: https://www.kaspersky.com/resource-center/definitions/what-is-a-qr-code-how-to-scan

[13] Hit-tests place virtual objects in the real world. ARCore, 2022. Disponível em: https://developers.google.com/ar/develop/hit-test
[14] Room. Developers Android, 2023. Disponível em: https://developer.android.com/jetpack/androidx/releases/room?hl=pt-br
[15] Anderson, C. The Model-View-ViewModel (MVVM) Design Pattern, 2012. In: Pro Business Applications with Silverlight 5. Apress, Berkeley, CA.