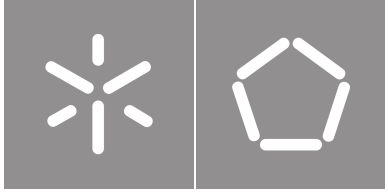
Universidade do Minho
Escola de Engenharia

Ricardo Loureiro da Silva

Aplicação móvel para inventário

〈Março〉 de 〈2023〉



Universidade do Minho
Escola de Engenharia

Ricardo Loureiro da Silva

Aplicação móvel para inventário

Dissertação de Mestrado
Mestrado Integrado em Engenharia Informática

Trabalho efetuado sob a orientação de:
José Francisco Creissac

⟨Março⟩ de ⟨2023⟩

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

_____, _____

(Local) (Data)

_____

(Ricardo Loureiro da Silva)

Resumo

Aplicação móvel para inventário

Esta dissertação foi desenvolvida sobre no contexto de um projeto proposto pela empresa Altice Labs, para o desenvolvimento de um aplicação móvel para o catálogo e gestão de equipamentos e infraestruturas dispersos no terreno, que integrará o sistema de gestão de inventário, provisionamento de clientes e projeto de rede, **Netwin**. Estudou-se também a possibilidade de complementar a solução através da utilização de tecnologia de Realidade Aumentada.

O projeto surge devido à necessidade de uma solução adaptada à utilização no terreno. O sistema Netwin possui uma interface Web, mas esta não se destina à utilização no exterior por dispositivos móveis, não só por não ser capaz de funcionar offline, mas também pela dificuldade que a complexidade e extensão desta impõem a um operacional que apenas pretenda catalogar/consultar infraestrutura e equipamento de rede.

O projeto foi dividido em duas partes: A primeira do projeto consiste na conceção de uma aplicação que permite cumprir um conjunto de use cases definidos, que passam pelas operações de catálogo/consulta de infraestrutura e equipamento de rede. A segunda parte corresponde a um estudo, especificação e prototipagem de algumas soluções de Realidade Aumentada utilizando diferentes ferramentas e abordagens de modo a investigar a viabilidade destas no projeto e de que modo podem acompanhar a aplicação.

A solução permite aos operacionais no terreno consultar/catalogar e atualizar o estado dos equipamentos, o que permite que o sistema tenha uma visão mais completa e atualizada sobre o estado da rede.

Palavras-chave: Aplicação Móvel, Aplicação Móvel Multi-Plataforma, Sistemas de Suporte a Operações, Levantamento de dados, Inventário, Realidade Aumentada

Abstract

Mobile App for Inventory

This dissertation was developed in the context of a project proposed by the company Altice Labs, for the development of a mobile application for the catalog and management of equipment and infrastructure dispersed in the field. This application which will integrate the inventory management system, provisioning of customers and network project, **Netwin**. The possibility of complementing the solution through the use of Augmented Reality technology was also studied.

The project arises due to the need for a solution adapted to use in the field. The Netwin system has a web interface, but this is not intended for outdoor use by mobile devices. Not only because it is not able to work offline, but also due to the difficulty that its complexity and extension impose on an operator who only wants to catalog/consult infrastructure and network equipment.

The project is was divided into two parts: The first part of the project starts with the design of an application that allows the fulfillment of a set of defined use cases, which include catalog/consult operations of infrastructure and network equipment. The second part corresponds to a study, specification and prototyping of Augmented Reality solutions, using different tools and approaches, in order to investigate their viability in the project, and how they can accompany the application.

The solution will allow field operators to consult/catalog and update the status of the equipment, which will allow the system to have a more complete and up-to-date view of the network status.

Keywords: Mobile Application, Multi-Platform Mobile Application, Operations Support Systems, Data Collection, Inventory, Augmented Reality

Índice

Índice de Figuras

Índice de Tabelas

Siglas

API Aplication Programming Interface 9, 16, 17, 18, 20, 21, 22, 24, 32, 41
AR Augmented Reality 3, 4, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 32, 33, 35, 40, 59

CAD Computer-Aided Design 49

E2E End to End 1

GPS Global Positioning System 13, 21, 35, 42, 57

LiDAR Light detection and Ranging 35, 36, 51, 53, 54, 57

OS Operating System 18, 40, 41

SDK Software Development Kit 7, 9, 16, 17, 18, 19, 20, 21, 40, 46
SLAM Simultaneos Location and Mapping 21
SSO Sistema de Suporte às Operações 1, 2

UI User Interface 8, 9
UX User Experience 10

VPS Visual Positioning System 22, 41, 55, 58, 59
VR Virtual Reality 11, 18

1 Introdução

1.1 Contexto

Esta dissertação encontra-se enquadrada num projeto proposto pela equipa Netwin da empresa Altice Labs, com o propósito de desenvolver uma aplicação mobile com possível recurso a tecnologias de realidade aumentada, integrada no sistema de desenvolvimento de rede e inventário com o mesmo nome, **Netwin**.

O sistema Netwin foi projetado para fins de gestão de inventário (equipamentos e infraestruturas no terreno), projeto de rede e provisionamento de clientes. Suporta os processos de negócios envolvidos no planeamento, fornecimento e desenvolvimento de serviços de rede e telecomunicações. O sistema oferece uma representação dos recursos físicos em localizações georreferenciadas e/ou das estruturas organizacionais associadas ao negócio. A solução inclui todos os níveis de inventário de informação, desde o inventário dos recursos físicos associados às infraestruturas (Fábrica Externa e Fábrica Interna), passando pelos vários níveis físicos e lógicos de rede (suportando uma gama de domínios técnicos como fibra ótica, cobre e rede coaxial etc.) para gestão de serviço, fornecendo assim cobertura ponta a ponta (Rede e Serviços).

Esta solução simplifica muito todos os aspetos de gestão de inventário e projeto de rede. Serve como um stock mestre de referência para toda a empresa, com sua visão End to End (E2E) de *make-up* de serviços. Isso significa que os processos podem ser agilizados, aumentando assim a sua eficiência e, ao mesmo tempo, aumentando o potencial de ganhos. Também permite o ajuste fino dos processos aos requisitos de negócios e oferece suporte para uma ampla gama de tecnologias.

1.1.1 Sistemas de Suporte às Operações

Um Sistema de Suporte às Operações (SSO) é um conjunto de software, ou um sistema de Tecnologia da Informação, usado por fornecedores de serviços de telecomunicação para monitorizar, controlar, analisar e gerir um sistema de rede de telecomunicações ou Internet. O software SSO é especificamente dedicado a fornecedores de serviços de telecomunicações e usado principalmente para apoiar processos de rede

para manter o inventário da rede, configurar componentes de rede, fornecer serviços e gerir falhas.

Os quatro principais elementos SSO são os seguintes:

- Processos: Sequência de eventos
- Dados: A informação que é gerida
- Aplicações: Os componentes que implementam processos para gerir dados
- Tecnologia: Como são implementadas as aplicações

1.1.2 Gestão de inventário

A gestão de inventário refere-se ao processo de solicitar, armazenar, usar ou vender stock de uma empresa. Isso inclui a gestão de matérias-primas, componentes e produtos acabados, bem como o armazenamento, processamento e manutenção de tais produtos. No contexto desta dissertação, o stock corresponde ao equipamento de rede que se encontram nas centrais ou distribuídos no terreno. Em vendas, manufatura, serviços e outros setores com uso intensivo de stock, os equipamentos e produtos acabados de uma empresa são o centro dos seus negócios. A falta de stock quando e onde é necessário pode ser extremamente prejudicial.

Aplicações móveis para gestão de inventário

São sistemas que acompanham e gerem o stock atual, incluindo as operações de entrada, movimento e saída de produtos/equipamentos. Se o software funcionar corretamente, ajuda as empresas a garantir que têm a quantidade certa de stock, no momento certo e no local correto. Um sistema complexo permite controlar o stock, monitorizar o seu movimento e reabastecê-lo quando necessário.

Um dos problemas encontrados mais frequentemente na gestão de inventário é a dificuldade de manter a informação sobre o stock atualizada, as alterações deste são frequentes, e manter uma folha de cálculo ou base de dados atualizados é um processo demorado e custoso, e possivelmente até propício a erros. Desta forma, as aplicações móveis são uma peça essencial para um sistema de gestão de inventário bem sucedido, já que podem acompanhar os trabalhadores no terreno ou armazém, permitindo-lhes assim atualizar as alterações de stock em tempo real.

1.2 Motivação

O funcionamento e interface da aplicação web Netwin, porém, não são adequados à utilização por um operacional no terreno. Não só pela impossibilidade de funcionar offline, mas também pela dificuldade que a complexidade e extensão desta impõe a um operacional que apenas pretenda catalogar/consultar infraestrutura e equipamento de rede. Desta forma, surge a necessidade de uma aplicação adaptada a estas necessidades.

1.3 Objetivos

O objetivo desta dissertação consiste na conceção e desenvolvimento de uma aplicação móvel adaptada às operações em causa, denominada de aplicação **Surveying**, que permita a identificação de elementos físicos da rede de acesso (Fibra, Cobre, Coaxial) que se encontram dispersos no terreno ou nas centrais, de modo a possibilitar a sua utilização no terreno e assim facilitar o funcionamento da plataforma. Com base na identificação de cadastro a aplicação permitirá consultar, complementar e acrescentar nova informação ao cadastro, facilitando a sua ligação com o sistema de Projeto e Cadastro da rede da Altice Labs – **NETWIN**.

Aquando do início desta dissertação já se encontrava disponível um protótipo da aplicação. Tinha sido definido que a aplicação teria de suportar os sistemas operativos móveis **Android** e **iOS** e, para tal, foi utilizada a framework para aplicações hibridas **Ionic** acompanhada da framework javascript **Angular**.

Pretende-se também explorar a possibilidade de utilização de tecnologias de Realidade Aumentada, na medida em que estas possibilitam simplificar ainda mais a interação com a aplicação e facilitam a localização e identificação dos equipamentos no terreno pelo utilizador.

1.4 Estrutura do Documento

Este documento segue a seguinte estrutura:

- **Capítulo 1 - Introdução:** É um capítulo introdutório, dedicado à contextualização, motivação e objetivos do tema, apresentando assim alguns conceitos básicos para a compreensão da dissertação.

- **Capítulo 2 - Estado da Arte:** Debate o estado da arte do desenvolvimento de aplicações móveis multiplataforma e o estado do desenvolvimento de aplicações de realidade aumentada. Este capítulo explora o conceito de aplicação móvel multiplataforma e aplicação híbrida, o mérito destas abordagens relativamente a aplicações nativas e apresenta alguns dos frameworks mais populares para tal, fazendo uma análise comparativa destes. Em seguida, faz um estudo sobre o estado da arte da realidade aumentada em aplicações móveis e web, procurando exemplos de trabalhos prévios e provas de conceito sobre aquilo que será possível atingir no desenvolvimento desta dissertação. Expõe também algumas ferramentas de desenvolvimento Augmented Reality (AR) disponíveis em contexto mobile que mostram ser relevantes para o problema em causa.

- **Capítulo 3 - Desenvolvimento da aplicação mobile Netwin:** Descreve o trabalho realizado durante a primeira fase do projeto, que corresponde ao desenvolvimento da aplicação móvel. É exposto o método de trabalho adotado, as ferramentas utilizadas para a gestão e desenvolvimento do projeto, os *Use Cases* e especificações definidas para a aplicação e alguns dos problemas que tiveram de ser resolvidos durante a sua conceção.

- **Capítulo 4 - Desenvolvimento da aplicação de Realidade Aumentada:** Este capítulo marca o início da segunda parte do projeto, que corresponde ao estudo, e prototipagem de alguns tipos de soluções de realidade aumentada AR. Particularmente, este capítulo faz um estudo dos requisitos impostos ao funcionamento destes protótipos e como estes se conjugam com a tecnologia presente num smartphone que utilize esses os mesmos.
- **Capítulo 5 - Soluções e ferramentas de Realidade Aumentada:** Este capítulo foca-se na seleção das soluções mais relevantes de entre as expostas no Estado do Arte e no desenvolvimento das aplicações de protótipo para cada uma destas, de modo a melhor entender até que ponto estas poderão ser úteis para o problema em análise.
- **Capítulo 6 - Conclusão:** Conclui este documento e discute o trabalho que foi realizado e aquilo que foi possível alcançar durante cada uma das fazes do projeto, bem como o seguimento natural de cada uma.

2

Estado da Arte

Este capítulo começa por estudar a necessidade de aplicações móveis no contexto empresarial de modo a verificar o valor de uma solução móvel para o problema em concreto. Estuda também o estado do mercado móvel de modo a verificar a necessidade de suportar **Android** e **iOS**.

De seguida, expõe várias soluções de desenvolvimento de aplicações que permitem suportar vários sistemas operativos, como é o caso das aplicações multiplataforma.

Posteriormente expõe o estado e a utilidade de aplicações de realidade aumentada, a sua distinção de realidade virtual e casos de estudo da utilização desta em contexto mobile.

2.1 Estado do mercado móvel

2.1.1 Sistemas Operativos Móveis

Apesar de nos últimos anos termos observado uma consolidação do mercado de sistemas operativos móveis[1], este ainda está dividido entre **Android** e **iOS**, como podemos observar na Figura 1, com frações de mercado de 73% e 26%, respetivamente[2]. É essencial que as aplicações suportem ambas as plataformas de modo a não perder uma percentagem significativa do mercado.

2.1.2 Presença Móvel

A presença no mercado móvel, é hoje em dia indispensável para muitas empresas. Estudos mostram que os consumidores passam cada vez mais tempo a utilizar aplicações móveis diariamente [24] e que os downloads das mesmos crescem a um ritmo acelerado[3], como podemos ver na Figura 2. Contrariamente, a presença da web mobile corresponde apenas a uma fração do tempo que os consumidores passam a utilizar o telemóvel [36]. Estes fatores mostram que cada vez mais as empresas precisam de se preocupar em ter uma presença nas **App Stores**, ou simplesmente possuírem soluções móveis para uso interno,

[1] Quota de mercado dos sistemas operacionais móveis a nível mundial de 01/2012 a 06/2021, visitado em 2021
[2] Quota de mercado mundial dos sistemas operativos móveis, visitado em 2021
[3] Downloads de aplicações (iOS, Android) trimestrais, visitado em 2021

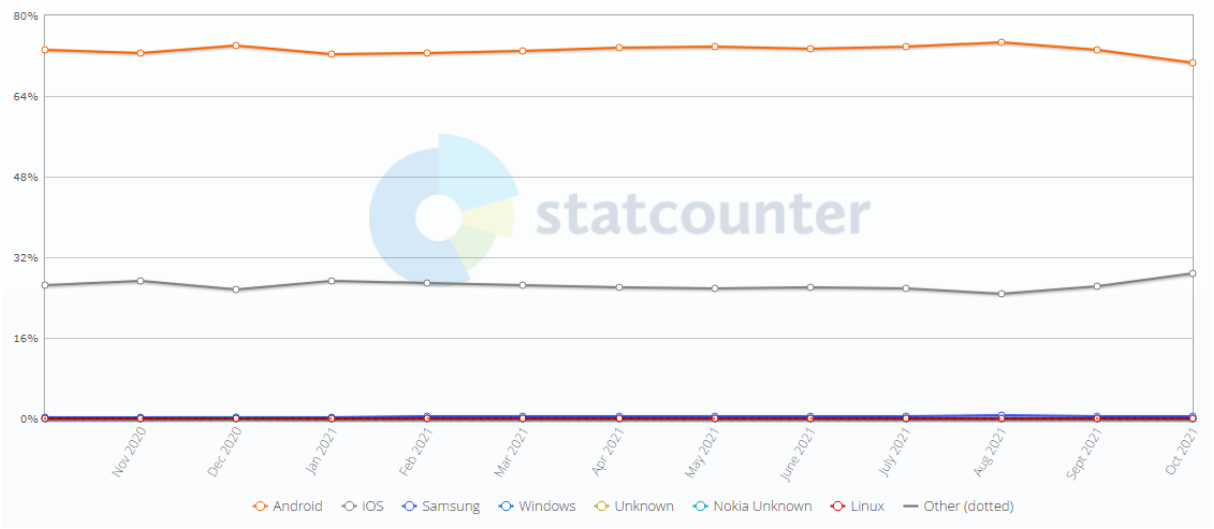


Figura 1: Market Share dos sistemas operativos móveis, Mundialmente 2020/2021

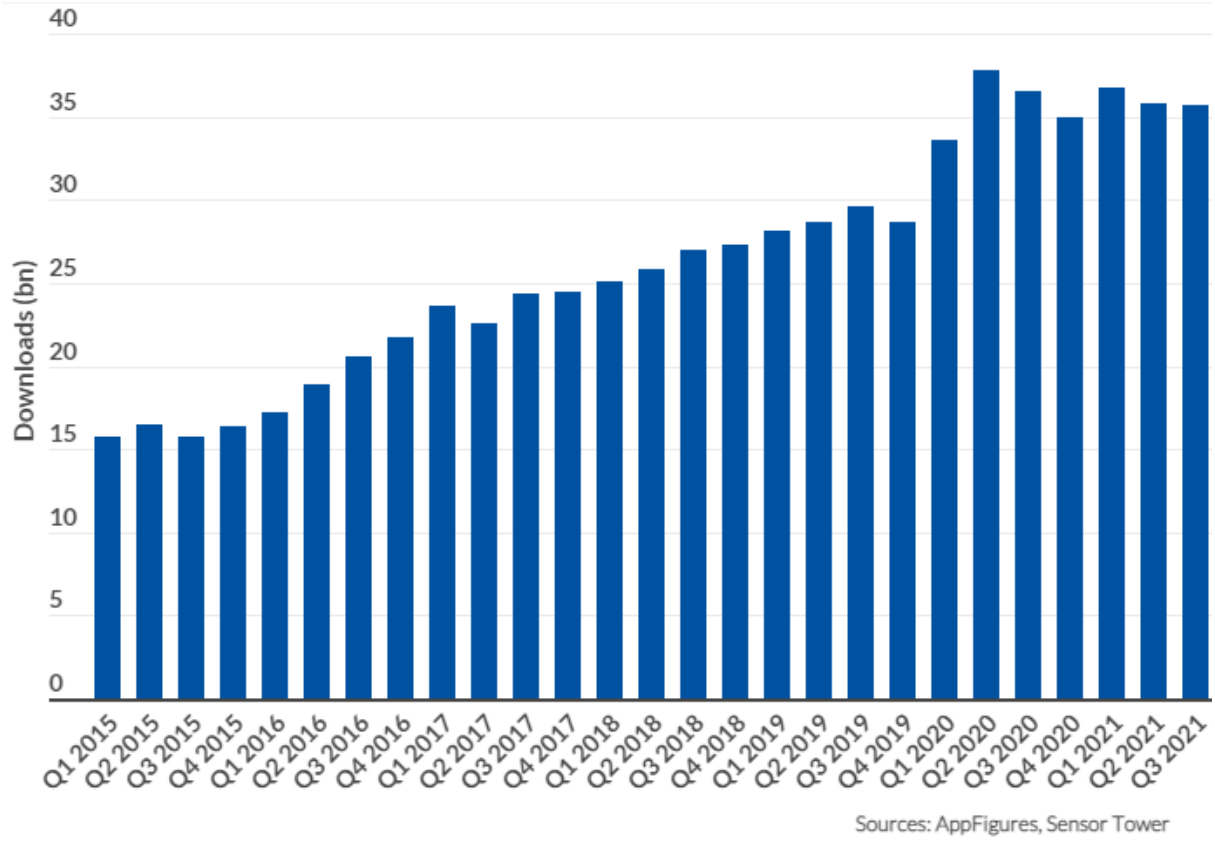


Figura 2: Crescimento dos downloads de aplicações móveis 2015/2021

como é o caso da aplicação em causa, ao invés de apenas manterem um website capaz de ser utilizado em dispositivos móveis.

2.2 Desenvolvimento de Aplicações Híbridas e Multiplataforma

2.2.1 Aplicações híbridas

Uma aplicação híbrida é um aplicação que combina elementos de aplicações nativas e aplicações web. Aplicações híbridas são essencialmente aplicações web colocadas numa *shell* de aplicação nativa. Depois de descarregadas e instaladas, a *shell* é capaz de se conectar a quaisquer recursos que a plataforma móvel fornece por meio de um *browser* que está embutido na aplicação. O *browser* e os seus *plug-ins* são executados no *back-end* e são invisíveis para o utilizador final.

As aplicações híbridas são populares porque permitem que os desenvolvedores escrevam código para uma aplicação móvel e website uma vez e acomodem várias plataformas. Como as aplicações híbridas adicionam uma camada extra entre o *source-code* e a plataforma de destino (no caso mobile), podem ter um desempenho um pouco mais lento do que as versões nativas ou web da mesma aplicação [44]. Dentro deste paradigma de desenvolvimento de aplicações, o *Framework* mais popular é **Ionic**.

2.2.2 Framework Ionic

Ionic é um Software Development Kit (SDK) *Open Source* completo para desenvolvimento de aplicações móveis híbridas criado por Max Lynch, Ben Sperry e Adam Bradley da **Drifty Co.**. A versão original foi lançada em 2013 e construída sobre **AngularJS** e **Apache Cordova** [22]. No entanto, a versão mais recente foi reconstruida como um conjunto de *Web Components*, permitindo ao desenvolvedor escolher qualquer *framework* de interface[4], como **Angular**, **React** ou **Vue.js**. **Ionic** fornece ferramentas e serviços para o desenvolvimento de **Aplicações móveis híbridas**, **Desktop** e **Progressive Web Apps** com base em tecnologias e práticas de desenvolvimento web modernas, usando tecnologias da web como **CSS**, **HTML**. Em particular, aplicações móveis podem ser construídas com essas tecnologias da Web e, em seguida, distribuídas através de aplicações nativas para serem instaladas em dispositivos utilizando **Apache Cordova** ou **Capacitor** [12].

Frameworks Javascript suportados pelo Ionic

O desenvolvimento através do *framework* **Ionic** acarreta a necessidade de fazer mais uma escolha de tecnologia, já que suporta os três *frameworks* de javascript mais populares, sendo estes:

[4] Apresentando o Ionic 4: Ionic para Todos, visitado em 2021

- **React** (também conhecido como **React.js** ou **ReactJS**) é uma biblioteca Javascript front-end gratuita e *open source* para construir interfaces baseadas em componentes de User Interface (UI) [32]. É mantido pela **Meta** e por uma comunidade de desenvolvedores individuais e empresariais. Pode ser usado como base no desenvolvimento de aplicações de página única ou móveis. No entanto, **React** está preocupado apenas com a gestão de estado e renderização desse estado para o **DOM**, portanto, o desenvolvimento de aplicações **React** geralmente requer o uso de bibliotecas adicionais para *routing*, bem como certas funcionalidades do lado do cliente. **React** foi criado por Jordan Walke, um engenheiro de *software* do **Facebook**, que lançou um protótipo inicial do **React** chamado **FaxJS**[5]. Foi utilizado pela primeira vez no *feed* de notícias do **Facebook** em 2011 e, posteriormente, no **Instagram** em 2012. O *source code* foi *open sourced* na **JSConf US** em Maio de 2013.

- **Angular** é um *framework* de aplicações web gratuito e de *open source*, baseado em **TypeScript**, liderado pela equipa **Angular** da **Google** e por uma comunidade de indivíduos e empresas [1]. Contrariamente a **React** apresenta-se como uma solução completa para o desenvolvimento de aplicações, incluindo ferramentas para *routing*, *forms*, *testing* entre outras. **Angular** foi anunciado na conferência **ng-Europe** 22–23 Outubro de 2014[6]. Em 30 de abril de 2015, os desenvolvedores do **Angular** anunciaram que este mudou de *Alpha* para *Developer Preview*. **Angular** mudou para *Beta* em dezembro de 2015, e o primeiro candidato a lançamento foi publicado em maio de 2016. A versão 1.0 foi lançada em 14 de setembro de 2016 [2].

- **Vue.js** ou **Vue** é um *framework* JavaScript de *front-end open source* para a construção de interfaces e aplicações de página única. O **Vue** apresenta uma arquitetura incrementalmente adaptável que se concentra na renderização declarativa e na composição de componentes [42]. A biblioteca central está focada apenas na camada de visualização. Recursos avançados necessários para aplicações complexas, como *routing*, *state management* e *build tools*, são oferecidos por meio de bibliotecas e pacotes de suporte mantidos oficialmente. **Vue** foi criado por Evan You após trabalhar para a **Google** usando **AngularJS** em vários projetos. A primeira contribuição para o projeto foi datada de julho de 2013, e a versão 1.0 foi lançada em fevereiro de 2014[7].

2.2.3 Aplicações multiplataforma

As aplicações móveis multiplataforma são aplicações móveis desenvolvidas através de um *framework* para funcionar em várias plataformas, mais notavelmente plataformas móveis, como **iOS** e **Android**. A nível mobile distinguem-se das aplicações híbridas por eliminarem a necessidade de uma camada extra entre a aplicação e o sistema nativo, o que permite um desempenho melhor e uma interação com o utilizador superior.

[5] A história do React.js numa timeline, visitado em 2021
[6] Uma espreitadela ao radicalmente novo **Angular** 2.0, visitado em 2021
[7] Lançamentos Vue.js, visitado em 2021

Com o desenvolvimento de aplicações móveis multiplataforma, os desenvolvedores podem construir aplicações que podem ser executadas em plataformas diferentes com um único sistema de código. Isto permite lançar o produto com mais rapidez e qualidade, compatível com diversos sistemas operativos, o que permite à aplicação atingir um público mais amplo [45].

Exemplos de *frameworks* multiplataforma populares nos dias de hoje são **React Native** e **Flutter**.

React Native

React Native é um *framework* de software de *User Interface open source* criado pela **Meta**. Pode ser usado para desenvolver aplicações para **Android**, **Android TV**, **iOS**, **macOS**, **tvOS**, **Web**, **Windows** e **UWP**, permitindo que os desenvolvedores usem a biblioteca **React** junto com recursos da plataforma nativa [34].

Os princípios de funcionamento do **React Native** são virtualmente idênticos ao **React**, exceto que o **React Native** não manipula o **DOM** por meio do **Virtual DOM**. É executado num processo em segundo plano (que interpreta o JavaScript escrito pelos desenvolvedores) diretamente no dispositivo final e comunica com a plataforma nativa através de dados serializados[8].

Os componentes do **React** envolvem o código nativo existente e interagem com as Aplication Programming Interfaces (APIs) nativas por meio do paradigma de UI declarativo do **React** e do **Javascript** .

Embora o *styling* do **React Native** tenha uma sintaxe semelhante ao **CSS**, na verdade não usa **HTML** ou **CSS**. Em vez disso, as mensagens da thread **Javascript** são usadas para manipular *Views* nativas. **React Native** também permite que os desenvolvedores escrevam código nativo em linguagens como **Java** ou **Kotlin** para Android, **Objective-C** ou **Swift** para **iOS** e **C++/WinRT** ou **C#** para **Windows 10**, o que o torna ainda mais flexível [33].

Flutter

Flutter é um kit de desenvolvimento de software de UI *open source* criado pela **Google**. É usado para desenvolver aplicações multiplataforma para **Android**, **iOS**, **Linux**, **MacOS**, **Windows**, **Google Fuchsia** e **Web**, a partir de uma única base de código[9] . As aplicações **Flutter** são escritas na linguagem **Dart** e fazem uso de muitos dos recursos mais avançados da linguagem.

No **Windows**, **MacOS** e **Linux**, **Flutter** é executado na máquina virtual **Dart**, que possui um mecanismo de execução *just-in-time*. Para melhor desempenho, as versões de produção de aplicações **Flutter** direcionadas a **Android** e **iOS** são compiladas antecipadamente (**AOT**)[10]. O *engine* do **Flutter**, escrito principalmente em **C++**, fornece suporte de renderização de baixo nível usando a biblioteca de gráficos **Skia** do **Google**. Além disso, faz interface com SDKs específicos da plataforma, como os fornecidos

[8] Bridging em React Native, visitado em 2021
[9] Google inicia um impulso para o desenvolvimento de aplicações multi-plataforma com o **Flutter SDK**, visitado em 2021
[10] Padrões de Compilação em Flutter, visitado em 2021

pelo **Android** e **iOS**. O **Flutter Engine** é um runtime portátil para hospedar aplicações **Flutter**. Este implementa as principais bibliotecas do **Flutter**, incluindo animação e gráficos, ficheiros e *I/O* de rede, suporte de acessibilidade, arquitetura de *plug-in*, o *runtime* **Dart** e uma cadeia de ferramentas de compilação. A maioria dos desenvolvedores interage com o **Flutter** por meio do **Flutter Framework**, que fornece uma estrutura reativa e um conjunto de plataforma, *layout* e *widgets* (componentes) de base [17].

2.2.4 Vantagens e desvantagens

A utilização de tecnologias hibridas ou multiplataforma porém, não é sem *tradeoffs* associados. Duas das críticas principais a este tipo de tecnologias são de que comprometem a User Experience (UX) e impõem uma sobrecarga de desempenho significativa.

Estudos mostram que, apesar de tal se registar, na realidade, as diferenças em UX [16] e a sobrecarga de desempenho [10], poderão não ser suficientemente significativos para justificar os custos de desenvolver duas aplicações nativas separadamente. Tal pode ser observado na prática em aplicações de gigantes tecnológicos como **Facebook** e **Instagram** [11] que foram desenvolvidas com a *framework* **React Native**

2.3 Desenvolvimento de Aplicações de Realidade Aumentada

2.3.1 Realidade Aumentada

A realidade aumentada corresponde a uma experiência interativa de um ambiente do mundo real onde os objetos reais são melhorados por informações percetivas geradas por computador, através de múltiplas modalidades sensoriais, incluindo visual, auditiva, tátil, e olfativa. AR pode ser definida como um sistema que incorpora três recursos básicos: uma combinação de mundos real e virtual, interação em tempo real e registo 3D preciso de objetos virtuais e reais. As informações sensoriais sobrepostas podem ser construtivas (ou seja, aditivas ao ambiente natural) ou destrutivas (ou seja, mascaramento do ambiente natural). Esta experiência está perfeitamente integrada com o mundo físico, de forma que é percebida como um aspeto envolvente do ambiente real [35].

O principal valor da realidade aumentada é a forma pela qual os componentes do mundo digital se misturam com a perceção do mundo real, não como uma simples exibição de dados, mas por meio da integração de sensações, que são percebidas como partes naturais de um ambiente. Os primeiros sistemas de AR funcionais que forneceram experiências imersivas de realidade aumentada para utilizadores foram inventados no início de 1990, começando com o sistema Virtual Fixtures desenvolvido no Laboratório Armstrong da Força Aérea dos EUA em 1992. As experiências comerciais de realidade aumentada

[11] Quem está a usar React Native, visitado em 2021

foram introduzidas pela primeira vez em negócios de entretenimento e jogos [23]. Posteriormente, as aplicações de realidade aumentada espalharam-se por setores comerciais, como educação, comunicações, medicina e entretenimento [35].

2.3.2 Realidade Aumentada vs Realidade Virtual

Na realidade virtual, a perceção da realidade dos utilizadores é totalmente baseada em informações virtuais. Na realidade aumentada, o utilizador recebe informações adicionais geradas por computador dentro dos dados recolhidos da vida real que aumentam sua perceção da realidade [35]. Por exemplo, em arquitetura, Virtual Reality (VR) pode ser usada para criar uma simulação do interior de um novo edifício; e AR pode ser usado para mostrar as estruturas e sistemas de um edifício sobrepostos com a estrutura real. Algumas aplicações de AR, como o **Augment**[12], permitem que os utilizadores coloquem objetos digitais em ambientes reais, permitindo que as empresas usem dispositivos de realidade aumentada como uma forma de visualizar seus produtos no mundo real. Da mesma forma, também pode ser usado para demonstrar a aparência dos produtos num ambiente imersivo para os clientes, conforme demonstrado por empresas como **Mountain Equipment Co-op** ou **Lowe's**, que usam realidade aumentada para permitir que os clientes visualizem como os produtos ficariam em sua casa através de modelos 3D[13].

2.3.3 Espectro de Realidade Mista

De modo a melhor compreender a diferença entre realidade aumentada e realidade virtual podemos utilizar a escala **Reality-Virtuality (RV) continuum**, proposta por Paul Milgram [7] que varia desde um ambiente completamente real, a realidade, a um completamente virtual, uma virtualidade. Esta escala é contínua e engloba, portanto, todas as variações e composições possíveis de objetos reais e virtuais. A área entre os dois extremos, onde tanto elementos reais como virtuais são apresentados ao utilizador, é chamada de realidade mista. Diz-se que esta, por sua vez, é composta ambas, realidade aumentada, onde o virtual aumenta o real, e virtualidade aumentada, onde o real aumenta o virtual.

A localização de qualquer ambiente imersivo informatizado ao longo desta escala coincide com a sua localização ao longo de uma escala paralela, o espectro da modelação do ambiente. Como ilustrado na Figura 4, no extremo direito da escala encontram-se ambientes virtuais (que precisam de ser completamente modelados para serem apresentados). Contudo, se aumentarmos a granularidade da escala podemos identificar a Realidade Aumentada, como um ambiente onde mais de metade do cenário é ambiente real sem qualquer modelação; e a Virtualidade Aumentada , como um ambiente onde mais de metade do cenário é ambiente virtual.

É de salientar que esta escala trata do conhecimento do mundo no sentido em que este é apresentado ao utilizador, isto é, da crescente necessidade de conhecer melhor as formas e propriedades dos objetos

[12] Augment está a trazer a revolução de AR aos negócios, visitado em 2021
[13] O retalho está a ser reimaginado com a realidade aumentada, visitado em 2021

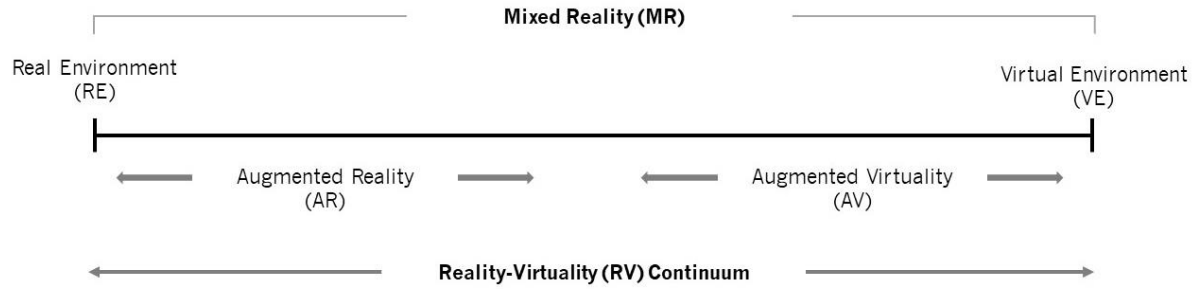


Figura 3: Reality-Virtuality Continuum, AR, and Augmented Virtuality

reais de modo a poderem ser virtualizadas. No capítulo 5, quando é referido que uma ferramenta precisa que os objetos no mundo estejam modelados ou que tira partido destes modelos, tal não afeta a posição da aplicação produzida nesta escala, já que esta modelação não afeta o conteúdo virtual que é apresentado ao utilizador, mas apenas a capacidade da ferramenta colocar o conteúdo aumentado com mais precisão por reconhecer os objetos do mundo real através dos modelos fornecidos. Esta escala trata apenas do conteúdo aumentado apresentado ao utilizador, e, desta forma, todas as ferramentas utilizadas se encontram no mesmo ponto da escala, na zona de realidade aumentada.

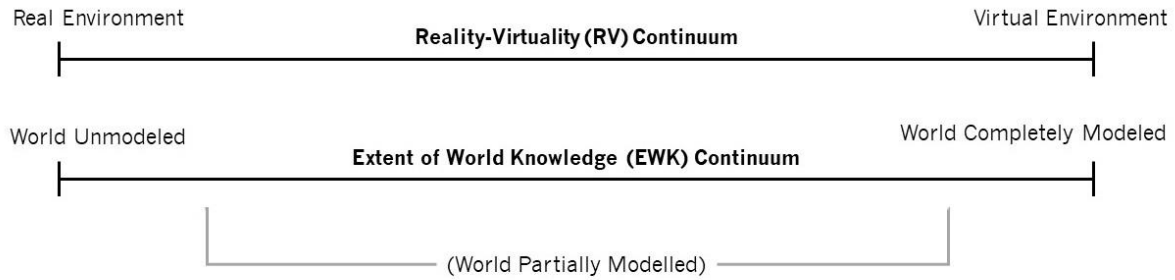


Figura 4: Reality-Virtuality Continuum

2.3.4 Tipos de Realidade Aumentada

Existem várias aplicações para realidade aumentada em uso hoje em dia. Do marketing, retalho, manufatura aos jogos, há muitos negócios na fase de exploração da utilização desta tecnologia emergente. A diversidade de *Use-Cases* para esta tecnologia levou ao desenvolvimento e categorização de vários tipos de realidade aumentada. Sendo que tomamos como mais relevantes para o problema em questão os seguintes:

Marker-based AR

A AR baseada em marcadores tira partido de marcadores colocados em pontos de interesse para desencadear uma experiência aumentada. Os marcadores, muitas vezes feitos com padrões distintos como códigos *QR* como poder ser observado na Figura 5 ou outros desenhos únicos, atuam como âncoras para a tecnologia. Quando um marcador no mundo físico é reconhecido por uma aplicação de realidade aumentada, o conteúdo digital é colocado em cima dele. O marcador de realidade aumentada é normalmente utilizado para fins de *marketing* e *retail*. Podemos tomar como exemplo cartões de visita que falam e brochuras que se movem [31].

Figura 5: Exemplo de utilização de AR baseada em marcadores

Markerless AR

A AR sem marcadores é mais versátil do que a AR baseado em marcadores, uma vez que permite ao utilizador decidir onde colocar o objeto virtual como pode ser observado na figura 6. A realidade aumentada sem marcadores depende do hardware do dispositivo, incluindo a câmara, Global Positioning System (GPS), bússola digital e acelerómetro, para recolher a informação necessária para que o software de AR seja capaz de posicionar o objeto virtual com precisão [31].

Location-based AR

A AR baseado na localização liga o conteúdo digital e a experiência que cria a um lugar específico. Os objetos são mapeados para que quando a localização de um utilizador corresponde ao local predeterminado, estes sejam exibidos no ecrã, tal como podemos ver na Figura 7. O jogo que trouxe a realidade aumentada às massas, **Pokemon Go** é um exemplo de AR baseado em localização. A experiência traz *Pokemons* virtuais ao nosso mundo através das funcionalidades do *smartphone* [31].

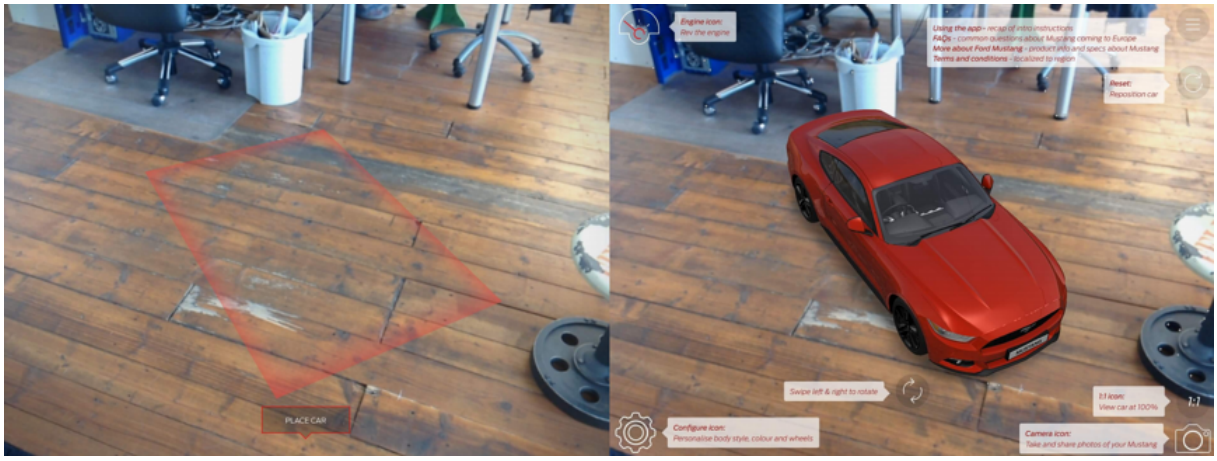


Figura 6: Exemplo de utilização de AR sem marcadores

Figura 7: Exemplo de utilização de AR baseada em localização

Outlining AR

O *Outlining AR* reconhece limites e linhas para ajudar em situações em que o olho humano não pode. O esboço da realidade aumentada utiliza o reconhecimento de objetos para compreender o ambiente imediato de um utilizador. Alguns exemplos de uso deste tipo de AR seriam a assistência à condução em condições de pouca luz ou delimitar a estrutura de um edifício a partir do exterior [31].

Sumário

Neste projeto estaremos mais focados em *Location-based AR* sem marcadores, já que o objetivo é que a aplicação seja capaz de colocar objetos virtuais da forma mais precisa possível nos locais dos equipamentos reais. A quantidade destes, aliada ao facto de estarem expostos aos elementos exclui qualquer possibilidade de desenvolver uma solução baseada em marcadores. Porém, não está excluída a opção do desenvolvimento de alguma funcionalidade de **Outlining AR** (que neste caso teria também de ser aliada

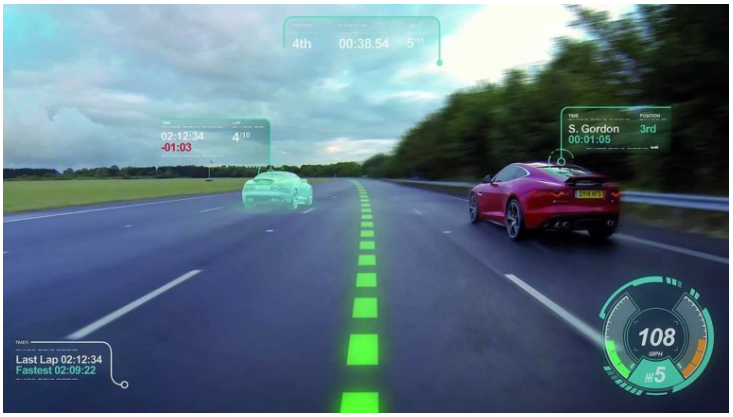


Figura 8: Exemplo da utilização de *Outlining AR*

à funcionalidade de localização), que poderia proporcionar uma experiência mais suave se for possível detetar os equipamentos através de *Reconhecimento de imagens*.

2.4 Desenvolvimento de Aplicações de Realidade Aumentada em contexto mobile

Esta secção apresenta brevemente um exemplo da utilização de AR em aplicações móveis, mas foca-se na exposição das ferramentas e soluções mais relevantes atualmente para o desenvolvimento destas aplicações.

As primeiras utilizações comerciais de realidade aumentada foram usadas largamente no entretenimento e na indústria dos videojogos, mesmo que agora outras indústrias estejam também interessadas nas possibilidades da realidade aumentada em áreas como a educação, comunicação, medicina e gestão de inventários.

O caso mais notável da utilização de AR em contexto mobile é sem duvida a aplicação **Pokémon GO**, apresentada na Figura 9, que se tornou viral em 2016. A aplicação para além de utilizar os sensores de localização e orientação do *smartphone*, faz uso das ferramentas de reconhecimento de imagens e superfícies incluídas nos *smartphones* modernos para criar um ambiente imersivo que enriquece a experiência dos utilizadores na interação com esta.

2.4.1 Frameworks para o desenvolvimento de AR em contexto móvel

Existem atualmente diversas tecnologias disponíveis para implementar a solução de Realidade Aumentada. Mesmo que esta seja uma possibilidade, não é obrigatório que a solução esteja integrada diretamente na aplicação **Surveying**, o que deixa uma larga variedade de opções à escolha.

Esta secção começa por apresentar os *frameworks* relevantes sobre os quais pode ser desenvolvida a aplicação de AR e, posteriormente, expõe ferramentas que funcionam sobre estes *frameworks* e oferecem

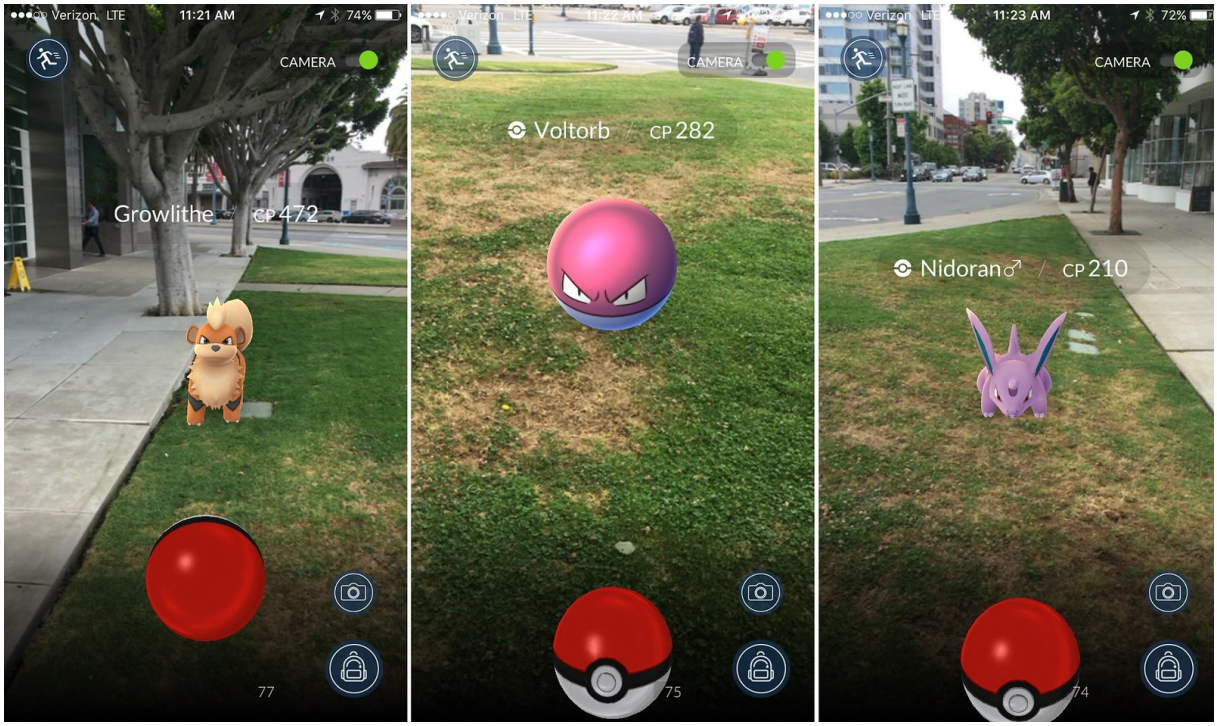


Figura 9: Utilização da realidade aumentada na aplicação Pokemon GO

as funcionalidades de AR necessárias para a aplicação.

SDKs Nativos Móveis

De modo a melhor compreender e situar as ferramentas e plataformas apresentadas posteriormente nesta secção, faz sentido primeiro entendermos que todas estas, exceto aquelas que se baseiam somente nos sensores de localização do *smartphone*, estão construídas sobre os SDKs nativos para desenvolvimento de AR de cada um dos sistemas operativos em questão, **ARCore** para **Android** (Também disponível para **iOS**) e **ARKit** para **iOS**. Estes SDKs estão atualmente presentes em virtualmente todos os *smartphones*, salvo os mais antigos (anteriores a 2018) e, desta forma é expectável que todas as ferramentas sejam capazes de funcionar nestes dispositivos, mesmo que com resultados de qualidade variável, pois nem todos estes *smartphones* possuem o hardware ou suportam as versões destes SDKs mais recentes.

Desta forma os SDKs em questão são os seguintes:

- **ARCore:** É o SDK do Google para a construção de experiências de realidade aumentada. Fornecendo diferentes APIs, **ARCore** permite ao *smartphone* detetar e compreender o seu ambiente e interagir com a informação capturada pela câmara. O **ARCore** utiliza três funcionalidades chave para integrar conteúdo virtual com o mundo real, tal como visto através da câmara do *smartphone*:
  - Deteção de posições e movimentos
  - Deteção de cena e plano horizontal

  - Estimativa da iluminação

  Algumas destas APIs estão também disponíveis em packages para **iOS**, o que permite reutilizar esta ferramenta para desenvolver aplicações de Realidade Aumentada em **iOS**. O **ARCore** foi lançado em Março de 2017 [29] [5].

- **ARKit:** È o SDK da **Apple** que permite aos desenvolvedores criar jogos e ferramentas de realidade aumentada para **iOS**. Possui funcionalidades como:

  - Deteção de posições e movimentos
  - Deteção de cena e plano horizontal
  - Estimativa da iluminação

  O **ARKit** foi anunciado aos programadores na **WWDC** de 2017[14], e as demonstrações mostraram aos programadores como poderiam criar experiências AR para as suas aplicações [6] [11].

Podemos verificar que **ARCore** e **ARKit** apresentam funcionalidades básicas semelhantes. Isto indica que as ferramentas apresentadas seguidamente serão capazes de funcionar de modo semelhante quanto construídas sobre qualquer um destes SDKs (existindo também a possibilidade de utilizar **ARCore** em **iOS**, ainda que com performance inferior expectada), mesmo se com resultados de qualidade variável, como acima mencionado.

O facto de **ARCore** estar disponível para **iOS** torna-se relevante na medida em que **ARCore** disponibiliza APIs que aparentam ser relevantes para o problema em questão, nomeadamente **Cloud Anchors** e **Geospatial API**, que serão apresentadas neste capítulo.

Abordagens e Frameworks relevantes para desenvolvimento

As abordagens relevantes para o desenvolvimento desta app são as mesmas da aplicação **Surveying**, isto é, pode ser desenvolvida como aplicação nativa, híbrida ou multiplataforma. Da mesma força os *frameworks* relevantes disponíveis para tal coincidirão, isto é, temos a opção de fazer o desenvolvimento através de **Android** e **iOS** nativos para desenvolver uma aplicação nativa, **Ionic** para desenvolver uma aplicação híbrida e **Flutter** ou **React Native** para desenvolver uma aplicação multiplataforma, a única diferença surge nesta última abordagem com a adição de **Unity** à lista de *frameworks* relevantes. Este secção expõe então as capacidades destes *frameworks* no contexto de desenvolvimento de aplicações **AR** móveis.

- **Android e iOS nativos:** A opção de desenvolver a aplicação sobre as plataformas nativas é atrativa na medida em que, devido à sua integração nativa nestas e à sua utilização em massa, é expectável que sejam capazes de disponibilizar as ferramentas mais avançadas e proporcionem o melhor desempenho.

[14] Conferência **WWDC** 2017, visitado em 04/2023

- **Web - Ionic:** A atratividade de desenvolver a aplicação para Web surge na medida em que esta plataforma, em principio, oferecerá maior facilidade no desenvolvimento e distribuição da aplicação, já que esta seria capaz de ser distribuída como uma aplicação para **Android** e **iOS**, desenvolvida sobre **Ionic** tal como a aplicação **Surveying**, ou simplesmente distribuída como uma página **Web**, que, apesar de não ser o seu objetivo final, traria facilidades a nivel de desenvolvimento e testagem.

  Se pretendermos aceder a funcionalidades de AR suportadas pelos SDKs nativos dos *smartphones* a partir da plataforma **Web** tal será possível através da API **WebXR**, atualmente em desenvolvimento:

  - **WebXR** é um grupo de normas que são utilizadas para permitir o desenvolvimento de aplicações de AR ou VR em contexto Web. O **WebXR Device API** implementa o núcleo do conjunto de características **WebXR**, gerindo a seleção de dispositivos de saída, renderizando a cena 3D no dispositivo escolhido à taxa de fotogramas apropriada, e gerindo os vetores de movimento criados utilizando controladores de entrada.

    WebXR pode ser utilizado através de um browser compatível com esta tecnologia, num *smartphone* que possua o SDK nativo do Operating System (OS) (**ARCore** ou **ARKit**).

  Devido à natureza ainda experimental desta API, não é expectável que tenha acesso às funcionalidades mais recentes dos SDKs nativo, mas poderá ser útil para use-cases mais simples.

Os *frameworks* multiplataforma mencionados têm disponíveis soluções que permitem ter acesso a funcionalidades dos SDKs nativo, de modo a adicionar funcionalidades de realidade aumentada às aplicações. A estes junta-se o *framework* **Unity**, que apesar de ser geralmente visto como destinado a jogos, possui uma extensa quantidade de ferramentas de desenvolvimento 3D que permitem desenvolver aplicações com realidade aumentada.

- **React Native** Se pretendermos usar React Native, temos disponível a ferramenta **ViroReact**. **ViroReact** é uma plataforma para os criadores construírem rapidamente experiências de AR e VR. Os programadores escrevem em React Native, e Viro faz a ligação aos SDKs nativos das plataformas móveis de VR (incluindo **Google Daydream**, **Samsung Gear VR**, e **Google Cardboard** para **iOS** e **Android**) e AR (**iOS ARKit** e **Android ARCore**) [41]. As funcionalidades mais recentes de **ARCore** mencionadas acima, **Cloud Anchors** e **Geospatial API**, estão atualmente em fase de testes e portanto ainda não disponíveis para utilização nesta plataforma.

- **Flutter** Em flutter existem algumas ferramentas que fazem a ligação às ferramentas de realidade aumentada nativas dos dispositivos e que permitem interagir com estas através da aplicação flutter. A ferramenta mais notável a explorar será **ar_flutter_plugin** que disponibiliza uma API que permite fazer uso das capacidades de AR de **Android** e **iOS** simultaneamente [25]. Esta ferramenta suporta a funcionalidade **Cloud Anchors** de **ARCore** mas não **Geospatial API**.

- **Unity** Em Unity existem várias ferramentas para criar experiências de realidade aumentada, o que indica que possivelmente este será o *framework* que proporcionará mais funcionalidades no desenvolvimento da aplicação. Nesta plataforma está disponível a ferramenta **AR Foundation**, que tal como as ferramentas das plataformas anteriores, faz a ligação aos SDKs nativos, porém, esta é melhor suportada e especialmente para **ARCore** existem guias para Unity utilizando **AR Foundation** no website oficial do SDK. Estes guias incluem **Cloud Anchors** e **Geospatial API**, suportados por **AR Foundation**, o que destaca **Unity** dos *frameworks* anteriores.

  Existem também outros SDKs de desenvolvimento AR disponíveis para **Unity**, os mais relevantes são:

  - Vuforia
  - Wikitude

  Estes SDKs serão apresentados na próxima secção.

  Uma outra funcionalidade que **Unity** possui, que torna este *framework* desejável, é a possibilidade de incluirmos aplicações **Unity** dentro de outras aplicações nativas através do paradigma **Unity as a library** [40], que oferece controlos que lhe permitem gerir quando e como carregar/ativar/descarregar o **Unity Runtime** dentro da aplicação nativa. Esta funcionalidade está disponível para projetos **iOS Xcode** e **Android Gradle**.

2.4.2 Soluções para desenvolvimento AR

Existem várias soluções de realidade aumentada disponíveis que se adaptam ao problema em causa. Estas soluções utilizam abordagens bastante diversas, mas podem ser categorizadas para o caso em questão como utilizando uma ou ambas das abordagens seguintes:

- Deteção de Objetos
- Localização através de sensores

As soluções que utilizam a abordagem de deteção de objeto são capazes de colocar o conteúdo virtual após detetarem um equipamento físico em frente à câmara do *smartphone*.

As que utilizam a abordagem de localização através de sensores colocam o conteúdo virtual após obterem os dados de localização e orientação do *smartphone*, de modo a saberem onde e como está orientado o *smartphone*.

Para este capítulo foram selecionadas as soluções mais populares, com melhor suporte, e, à partida, mais acessíveis.

AR.JS (Localização)

AR.JS é uma biblioteca *lightweight* para realidade Aumentada na Web, que vem com recursos como reconhecimento de imagens e marcadores e AR baseado em localização [4]. À primeira vista será simples de utilizar e integrar na aplicação **Surveying**, se tal for pretendido, já que também utiliza tecnologia Web.

Cloud/Spatial Anchors (Deteção de Objetos + Localização)

Cloud Anchors correspondem a abordagem mais avançada que, apesar de também utilizar os dados de localização do *smartphone*, faz principalmente uso de algoritmos de visão por computador para colocar o conteúdo aumentado.

Para melhor compreendermos este conceito faz sentido primeiro explorarmos o conceito de Ancora nos SDKs de realidade aumentada.

As âncoras correspondem a pontos fixos e estáveis no ambiente **3D** captado pelo *smartphone*. A necessidade das âncoras surge porque a compreensão ambiental dos SDKs de AR varia ao longo de uma experiência AR. Isto é, a compreensão que o SDK tem sobre o ambiente que rodeia o *smartphone* e a sua posição neste varia à medida que o *smartphone* recolhe mais informação visual, ou se houver movimentos por parte do ambiente ou do *smartphone*. Isto faz com que objetos virtuais colocados "soltos"no mundo pareçam instáveis ou se afastem do local onde foram colocados e dos outros objetos virtuais. Isto pode ter impacto na experiência proporcionada pela aplicação. As âncoras são pontos fixos em locais de fácil localização e compreensão para o SDK. Outros conteúdos poderão ser depois colocados em posições relativas à posição da âncora. Desta forma, o conteúdo virtual ancorado mantém uma posição e orientação no espaço mais estável relativamente ao ambiente e ao outro conteúdo virtual ajudando a manter a ilusão de objetos virtuais colocados no mundo real.

Uma **Cloud Anchor** é um tipo especial de âncora que pode ser usada para persistir experiências de AR no mundo real. Com **Cloud Anchors**, é possível criar conteúdos virtuais interativos e ancorá-los a locais reais, criando experiências que podem ser partilhadas ao longo do tempo por múltiplos utilizadores através de dispositivos diferentes.

O caso de uso em questão porém, não é o ideal para o uso desta solução, já que esta abordagem beneficia de múltiplos registos de âncoras no mesmo local para que melhor possa entender o ambiente, o que não acontece no caso dos equipamentos mais dispersos. Mesmo assim, os resultados poderão ser satisfatórios.

Existem duas soluções de Cloud Anchors disponíveis a considerar:

- Azure Spatial Anchors
- Google Cloud Anchors (Disponível como API para ARCore)

Wikitude (Deteção de Objetos ou Localização)

O **Wikitude** é um SDK de realidade aumentada para dispositivos móveis. Lançado pela primeira vez em Outubro de 2008, o SDK inclui reconhecimento e seguimento de imagem, renderização de modelos **3D**, sobreposição de vídeo e AR baseado em localização. Em 2017, a **Wikitude** lançou a sua tecnologia Simultaneos Location and Mapping (SLAM) que permite o reconhecimento e o seguimento de objetos.

O SDK está disponível para os sistemas operativos **Android**, **iOS** e **Windows**, sendo também otimizado para vários dispositivos inteligentes de óculos.

Vuforia (Deteção de Objetos ou Localização)

Vuforia é um SDK de realidade aumentada para dispositivos móveis [43]. Utiliza tecnologia de visão por computador para reconhecer e seguir imagens planares e objetos 3D em tempo real. Esta capacidade de registo de imagens permite aos programadores posicionar e orientar objetos virtuais, tais como modelos 3D e outros suportes, relativamente a objetos do mundo real, quando estes são vistos através da câmara de um dispositivo móvel. O objeto virtual segue então a posição e orientação da imagem em tempo real para que a perspetiva do utilizador sobre o objeto corresponda à perspetiva sobre o alvo. Assim, parece que o objeto virtual faz parte da cena do mundo real.

O **Vuforia SDK** suporta uma variedade de tipos de alvo 2D e 3D, incluindo Alvos de Imagem 'sem marcador' e Alvo de Modelo 3D.

Vuforia fornece APIs em **C++**, **Java**, **Objective-C**, e as linguagens **.NET** através de uma extensão para a plataforma Unity. Deste modo, o SDK suporta tanto o desenvolvimento nativo para **iOS**, **Android**, e **UWP**, como o desenvolvimento de aplicações AR em **Unity** que são facilmente portáteis para ambas as plataformas.

Existe também um plugin que permite a utilização de AR baseado em localização através deste SDK.

ARCore Geospatial API (Deteção de Objetos + Localização)

A **ARCore Geospatial API** corresponde a uma ferramenta adicionada recentemente à plataforma **ARCore**. Esta ferramenta utiliza dados dos modelos **Google Earth 3D** e dados de imagem **Street View** do **Google Maps** para permitir a sua aplicação para experiências de realidade aumentada imersiva, à escala global e baseada em localização.

A principal vantagem desta ferramenta, em contraste com uma ferramenta que se baseie somente nos dados de GPS é o ganho em termos de precisão da localização. A utilização exclusiva de GPS e outros dados de sensores para determinar a localização não é normalmente suficiente para alcançar a precisão necessária para aplicações AR deste tipo. A **Geospacial API** utiliza dados de sensores de dispositivos (como o GPS) juntamente com dados de imagens capturadas para determinar a localização precisa. Cria um mapa de imagem que é processado para encontrar as partes reconhecíveis do ambiente e faz a

sua correspondência com o modelo de localização Visual Positioning System (VPS). O resultado é uma localização e pose (posição e orientação) mais precisa para a aplicação.

O VPS da Google fornece este modelo de dados, extraído de milhares de milhões de imagens em todo o mundo. As imagens são convertidas num mapa de pontos 3D de alta definição para localização visual. Redes neuronais profundas identificam e descrevem então as partes da imagem que são suscetíveis de serem reconhecidas durante longos períodos de tempo. Este modelo de localização consiste em dezenas de triliões de pontos e abrange todas as zonas com cobertura **Street View**.

Geospatial API vs Cloud Anchors

As âncoras no **ARCore** também utilizam mapas de imagem para determinar a pose. No entanto, estes mapas de imagem são criados localmente, por exemplo, com o API **Cloud Anchors**. A partilha destes mapas de imagem fora da aplicação que os criou não é prontamente suportada pela API e requer trabalho extra.

A API Geospacial prevê a posição horizontal (latitude e longitude) e vertical (altitude) de uma âncora de acordo com a especificação **WGS84** [46]. Com o **Geospacial API**, é possível colocar uma âncora em quase qualquer parte do mundo a uma dada latitude, longitude e altitude. A geometria 3D do edifício é uma parte ainda em desenvolvimento do modelo de localização VPS. A colocação de âncoras em planos geométricos também será apoiada, uma vez que estão planeadas melhorias a estes e outros aspetos do modelo de localização VPS para futuras versões do **Geospatial API**.

Soluções Alternativas

Existem algumas soluções disponíveis, apesar de ainda em fases iniciais, que possibilitam o desenvolvimento de aplicações AR através da mesma abordagem que **Geospatial API**, através de VPS, como as tecnologias propostas pela **Niantic**[15] e **Apple**[16], porém estas abordagens prevêm apenas a utilização em grandes centros urbanos (atualmente limitadas a cidades como San Francisco, Los Angeles...) e, desta forma, não se adaptam ao problema em questão.

2.5 Conclusão do capítulo

Como foi mencionado no início do capítulo, este estudou essencialmente 3 questões:

- Necessidade de suportar dois sistemas operativos móveis, **iOS** e **Android**
- Validade da escolha do *framework* **Ionic + Angular** para desenvolver a aplicação
- Estado e utilidade de soluções de realidade aumentada

[15] Áreas e cidade suportadas pelo **City AR** da Niantic, visitado em 06/2022
[16] Áreas e cidade suportadas pelo **City AR** da Apple, visitado em 06/2022

Relativamente ao primeiro ponto foi possível verificar que o mercado de sistemas operativos móveis está essencialmente dividido entre **iOS** e **Android**, daí se puder dar como justificada a escolha de suportar ambos os sistemas operativos.

No que diz respeito ao segundo ponto, podemos verificar que **Ionic**, apesar de ser menos popular que *frameworks* multiplataforma como **React Native** e **Flutter**, é mesmo assim relativamente popular, e o facto de ser uma *framework* para aplicações hibridas ao invés de multiplataforma impõe uma curva de aprendizagem menor, devido à possibilidade de reutilizar conhecimento de desenvolvimento de aplicações **Web**, o que é uma característica também desejável. Desta forma dá-se como válida a escolha deste *framework*.

Por fim, podemos verificar que já existem exemplos de sucesso relativamente a aplicações de realidade aumentada, o que demonstra o seu mérito e prova que a tecnologia necessária para o seu desenvolvimento já se encontra disponível, o que justificará o esforço no seu estudo.

3

Estado do desenvolvimento da aplicação Surveying

Este Capítulo descreve o trabalho realizado na primeira fase do projeto, que corresponde ao desenvolvimento da aplicação mobile Surveying e a abordagem usada para o seu desenvolvimento.

3.1 Estado da aplicação: Serviço Web

O sistema **Netwin**, já mencionado anteriormente, foi projetado para fins de gestão de stock, desenvolvimento de rede e provisionamento. Para isso oferece uma aplicação web, apresentada na Figura 10, que permite executar o extenso conjunto de use cases também já mencionados na introdução. Este sistema inclui uma API Rest de dados, Figura 11, que permite executar as funcionalidade da aplicação web, mas que se encontra disponível para ser utilizada também por outras aplicações. A aplicação móvel a ser desenvolvida fará uso desta API para fazer as alterações ao estado da rede necessárias.

3.2 Trabalho prévio

Como já brevemente referido na introdução, anteriormente a esta dissertação, já tinha sido realizado um protótipo da aplicação **Surveying**.

Também já tinha sido definida a *framework* que seria utilizada no desenvolvimento da aplicação, sendo esta **Ionic** com a *framework* javascript **Angular**. Esta escolha, bem como as justificações para tal, foram validadas também pelo estudo do estado da arte desta dissertação. Assim, o desenvolvimento da aplicação mantém a utilização destas tecnologias. A escolha é fortalecida também pelo facto de permitir uma maior colaboração com os outros membros da equipa **Netwin**, que também desenvolve a plataforma Web **Netwin** através de **Angular**.

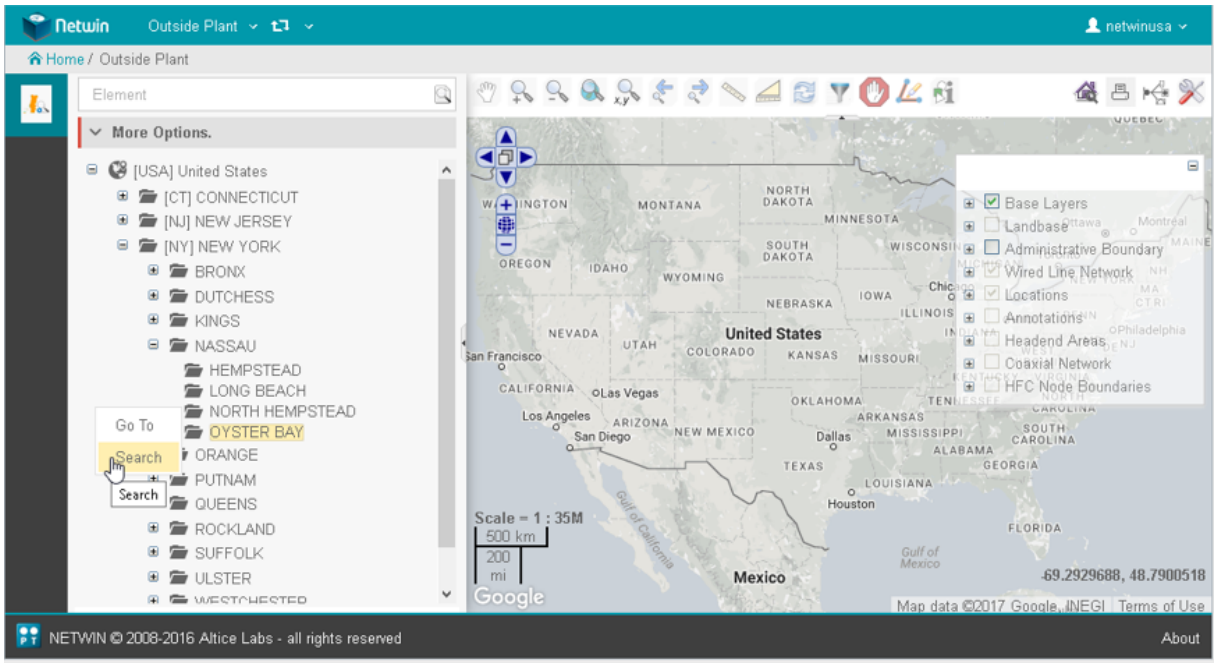


Figura 10: Interface da aplicação web Netwin

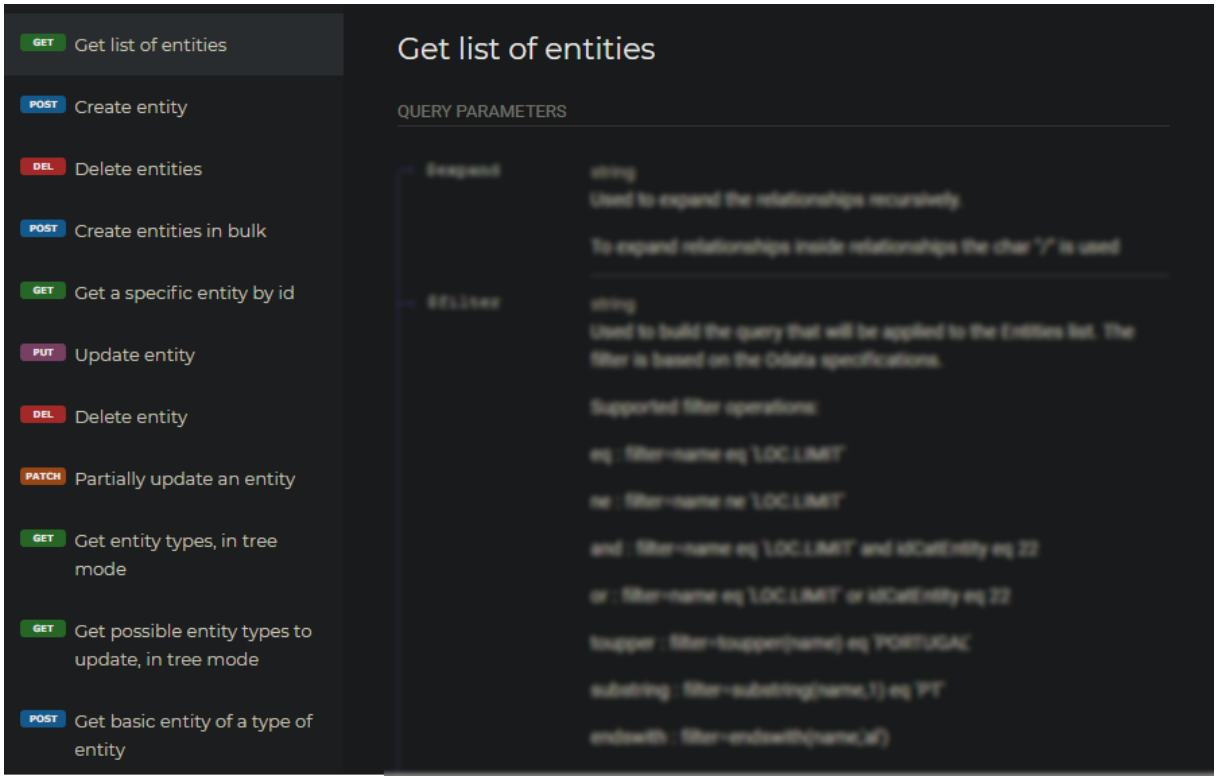


Figura 11: Documentação da API Rest do sistema Netwin

3.3 Abordagem de desenvolvimento adotada no projeto

O projeto na empresa **Altice Labs** encontra-se integrado na equipa **Netwin**, desta forma, o processo de desenvolvimento acompanhará aquele já definido pela equipa. A equipa **Netwin** segue um padrão de desenvolvimento bem estruturado e documentado, utilizando a *framework* **Agile - Scrum** para a gestão do processo de desenvolvimento, bem como uma série de ferramentas para a gestão de *issues*, controlo de versões, *code reviews* e documentação.

3.3.1 Agile - Scrum

Na gestão de projetos, o Scrum é uma estrutura para desenvolver, entregar e sustentar produtos dentro de um ambiente complexo. È projetado para equipas de dez ou menos elementos, que dividem seu trabalho em metas que podem ser concluídas em iterações *time-box*, chamadas de *Sprints*, com duração não superior a um mês, e, mais comummente, duas semanas. A equipa **Scrum** avalia o progresso em reuniões diárias com limite de tempo de 15 minutos ou menos, chamadas de **Scrums** diárias (uma forma de reunião stand-up). No final da *Sprint*, a equipa realiza mais duas reuniões: a revisão da *Sprint*, que demonstra o trabalho feito aos *stakeholders* para obter *feedback*, e a retrospetiva da *Sprint*, que permite que a equipa reflita e melhore.

3.3.2 Gestão de issues - Jira

Os *issues* da *Sprint* são geridos num quadro Kanban, como demonstra a figura 12. Os quadros **Kanban** representam visualmente o trabalho em vários estágios de um processo, usando cartões para representar itens de trabalho e colunas para representar cada estágio do processo. Os cartões são movidos da esquerda para a direita para mostrar o progresso e ajudar a coordenar as equipas que executam o trabalho.

O estado das *issues* têm um *workflow* pré-definido de modo a permitir o seu acompanhamento que pode ser observado na figura 13.

3.3.3 Controlo de versões - SVN

O sistema de controlo de versões utilizado foi o **SVN**. **SVN**, abreviatura de **Apache Subversion** é um sistema de controle de versão e revisão de software distribuído como código aberto sob a Licença Apache. Neste, o código é organizado em *Trunk* (versão mestra do código) e *Branches* (versões segundarias) e permite uma elevada quantidade de ações para a gestão do código como commits, merges, resolução de conflitos... Apesar de atualmente ser muito menos popular que **Git**, a sua escolha é justificada pela integração que é feita com todo o *workflow* das plataformas da **Altice Labs**. A extensão das funcionalidades desta que foram utilizadas durante o desenvolvimento é mínima, devido a apenas existirem dois

Figura 12: Quadro Kanban da *Sprint* onde são geridas as *issues*

membros da equipa a submeterem *commits* ativamente. Os *commits* eram aplicados diretamente no trunk e os conflitos, quando ocorriam, podiam ser rapidamente resolvidos.

3.3.4 Code reviews - Crucible

Quando algum *issue* se dá por completo, o *workflow* dita que seja aberta uma review, como observado na figura 14, para que as alterações possam ser avaliadas por um outro membro da equipa. Para isso é utilizada a plataforma **Crucible**, uma aplicação Web de revisão de código colaborativo da empresa de software australiana **Atlassian** voltada principalmente para empresas, e fornece recursos que permitem a revisão por pares de uma base de código. O **Crucible** é especialmente adaptado para equipas distribuídas e facilita a revisão assíncrona e comentários no código. **Crucible** também se integra a ferramentas populares de controle de origem, como **Git** e **Subversion**. **Crucible** não é de código aberto, mas os clientes podem visualizar e modificar o código para seu próprio uso.

3.3.5 Wiki

Para cada projeto é mantida uma página de wiki, como observado na figura 15, onde não só são descritas as especificações da aplicação, mas também o processo de desenvolvimento e as decisões tomadas neste. Desta forma, a integração de novos membros, a utilização da aplicação por membros externos à equipa ou até mesmo a cooperação entre membros da equipa que trabalhem em secções diferentes é facilitada.

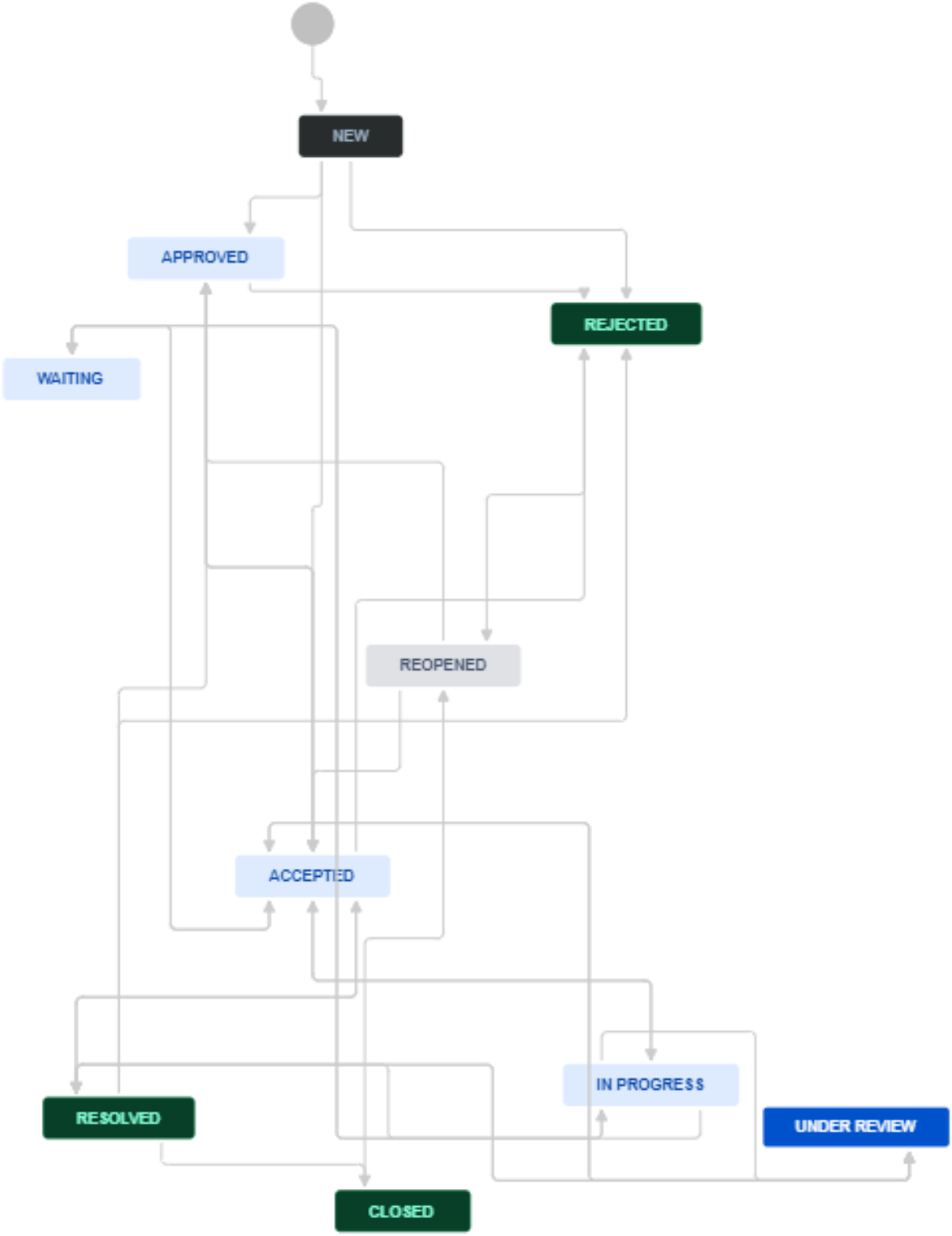


Figura 13: Workflow de um *issue*

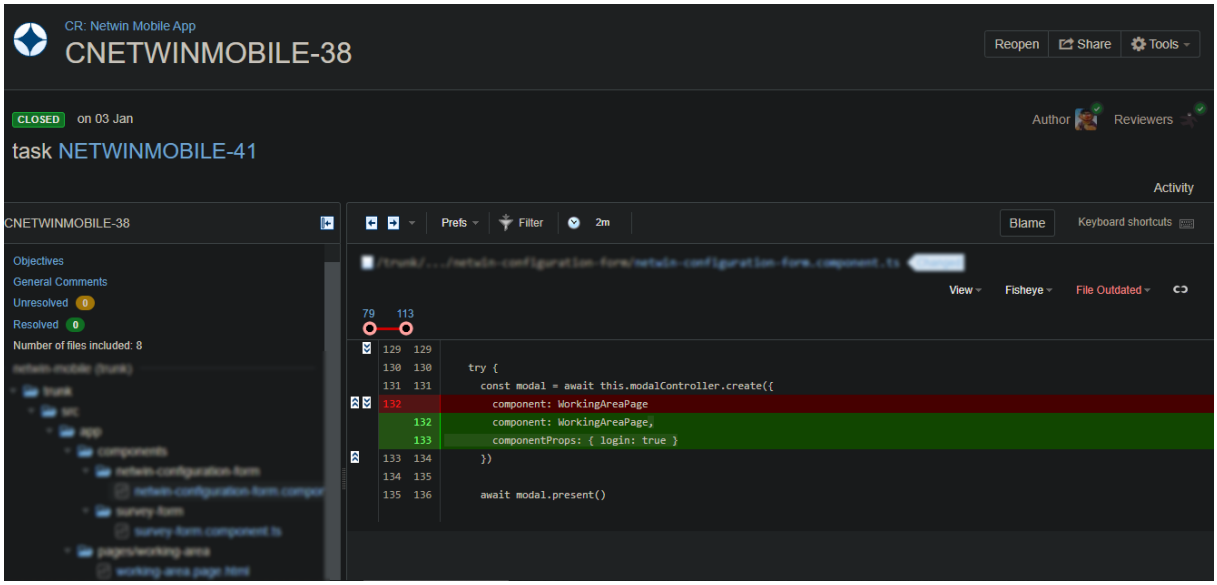


Figura 14: Exemplo de um code review

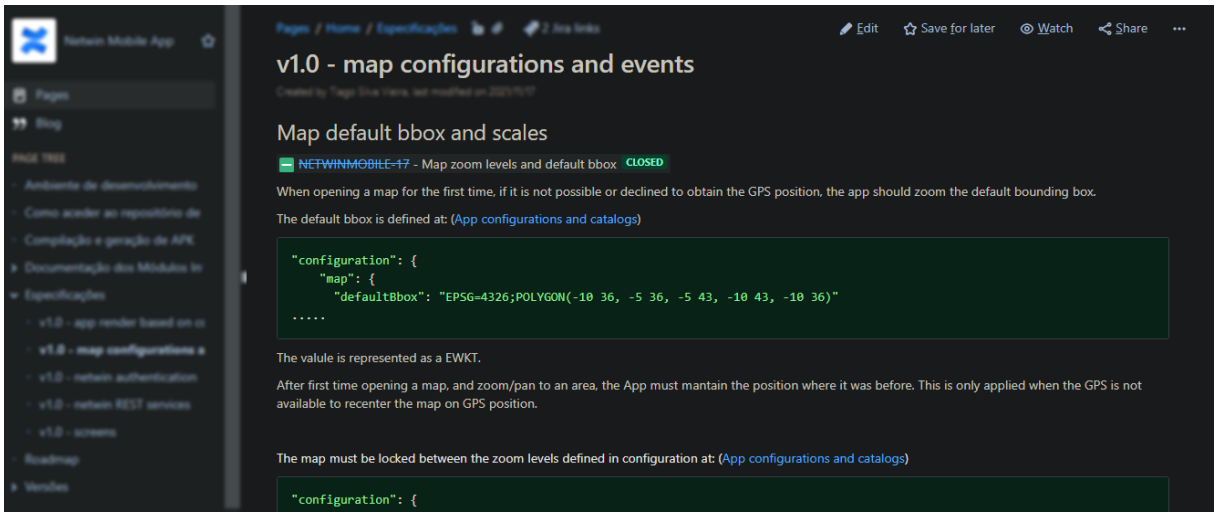


Figura 15: Exemplo de uma página da wiki

3.4 Especificação 1.0

De modo a ter uma evolução previsível e controlada da aplicação, foi definido um conjunto de funcionalidades que deveriam ser concretizadas para que a aplicação fosse minimamente viável, estas concretizam-se nos ecrãs presentas na figura 16.

- **Spash Screen:** Corresponde ao ecrã inicial que é apresentado ao utilizador enquanto a aplicação carrega, deverá apresentar o logótipo da aplicação
- **Login:** É primeiro e único ecrã a que o utilizador tem acesso se não estiver logado na aplicação, o processo de login corresponde à autenticação do utilizador na plataforma através das suas credencias da empresa, e posteriormente à definição de uma Área de trabalho no ecrã seguinte,

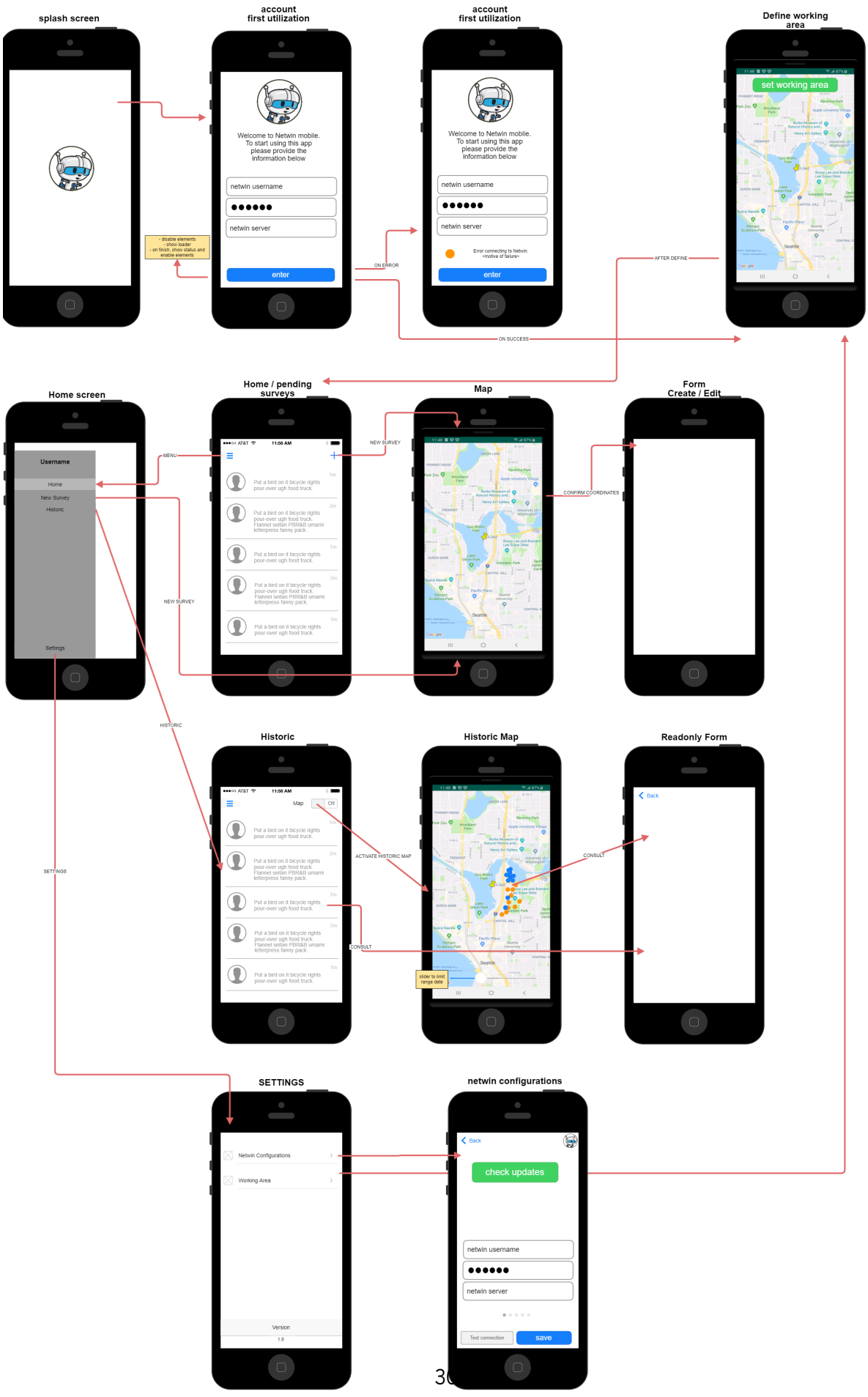


Figura 16: Ecrãs definidos para a especificação 1.0 da aplicação

se algum destes passos falhar o utilizador deve ser reencaminhado para este ecrã com a devida mensagem de erro.

- **Mapa Área de Trabalho:** Este ecrã permite ao utilizador definir a sua área de trabalho, para isto, começa por mostrar ao utilizador uma área selecionada a qual este pode ajustar até se encontrar como pretende, de seguida faz o download da informação da área, rotas e registos anteriores realizados na mesma área e mostra entes também ao utilizador, se este prosseguir a aplicação guarda os novos dados e ajusta-se ao trabalho na nova área, se não, o mapa permite ao utilizador redefinir a área.

- **Home Screen:** Corresponde a um conjunto de tabs sobre os quais assenta toda a funcionalidade da aplicação, sendo estes a **Lista de registos**, **Histórico** e **Configurações**

- **Lista de Registos:** Este ecrã apresenta a lista dos registos que foram efetuados pelo utilizador mas ainda não foram submetidos para a plataforma, aqui, o utilizador pode selecionar um ou mais registos para editar/submeter se assim o pretender, inclui também um menu que permite encaminhar o utilizador para a página de criação de um novo registo

- **Form de criação/edição/visualização de Registos:** Esta página corresponde ao form através do qual o utilizador pode criar/editar/visualizar os seus registos, é dinâmica e os campos que aparecem no form são gerados de acordo com a configuração da aplicação.

- **Mapa de Registo:** A criação de um registo implica a seleção de um ponto geográfico para tal, este mapa serve essa funcionalidade, apresenta ao utilizador a sua área de trabalho e os registos já efetuados e permite-lhe colocar um marco onde pretender que seja a localização do registo.

- **Histórico:** Esta página corresponde ao histórico de ações na aplicação, criação, edição, remoção, submissão e erros de submissão irão gerar registos que aqui serão apresentados.

- **Mapa de Histórico:** Corresponde a um mapa onde o utilizador poderá ver o seu histórico de registos submetidos.

- **Configurações:** Esta página destina-se à gestão e atualização das configurações da aplicação, permite ao utilizador redefinir a sua área de trabalho, fazer logout se pretender atualizar as suas credenciais, e atualizar as configurações de form/roteiros da aplicação.

3.5 Estado do desenvolvimento da aplicação

No momento da entrega deste documento, a fase de desenvolvimento da aplicação encontra-se quase concluída, a aplicação é capaz de cumprir os requisitos delineados na especificação 1.0 e está disponível para ser compilada para **Android** e **iOS** como foi estipulado.

Durante o seu desenvolvimento tiveram de ser resolvidos problemas como a persistência e modelação de dados, utilização da API **Netwin**, utilização de livrarias externas como **leaflet** para definição de mapas, resolução de conflitos de dependências, entre outros.

Com a conclusão da especificação, conseguimos ter uma melhor noção de como o levantamento de equipamentos no terreno será efetuado e os seus dados registados. Desta forma, já temos alguma noção prévia da informação disponível e das limitações que serão impostas ao protótipo da aplicação de AR que será estudada nos próximos Capítulos.

4

Desenvolvimento da aplicação de Realidade Aumentada

Este Capítulo marca o início da segunda parte do projeto, que, como já mencionado, focar-se-á no estudo, e prototipagem de alguns tipos de soluções de AR.

Particularmente, faz um estudo dos requisitos impostos ao funcionamento destes protótipos, e como estes se conjugam com a tecnologia presente num *smartphone* que utilize esses mesmos protótipos.

4.1 Requisitos

Esta aplicação não estará ligada à fase de catálogo dos equipamentos no terreno por operadores, que é suportada pela aplicação **Surveying**, mas será orientada ao *use case* de um operador no terreno que pretenda localizar, ou obter informação, sobre um equipamento já catalogado no sistema. Desta forma, esta aplicação não cobre o use case do catálogo dos equipamentos, porém considera-se a possibilidade de calibrar a posição dos equipamentos ou alterar o estado destes. É desejável que pudesse funcionar offline, tal como a aplicação surveying. Se possível, a solução deveria suportar **Android** e **iOS**, mas considera-se suficiente nesta fase que apenas suporte **Android**.

Sendo esta uma fase mais ligada à investigação, não está definida a abordagem que a aplicação utilizará, nem que sensores ou funcionalidades do *smartphone* poderão ou não ser utilizados, pelo que, à partida, não se exclui nenhuma das ferramentas expostas no Capítulo 2.

4.2 Tecnologia Necessária

Como mencionado acima, para já, não se exclui a possibilidade de utilizar qualquer sensor de localização e orientação, câmara ou funcionalidade do *smartphone*, pelo que, as soluções definidas poderão fazer uso de qualquer um destes. Assim, faz sentido explorar e expor melhor as funções e as ordens de precisão (m,cm) de dados que estes são capazes de registar, de modo a desde já, sermos capazes de identificar

limitações nas ferramentas propostas e , futuramente, sermos capazes de identificar qual o *smartphone* mais eficaz para a utilização de algum tipo de solução proposta.

Desta forma foram identificados os seguintes sensores/funcionalidades como sendo relevantes para o problema em questão:

4.2.1 Magnetómetro (Bússola)

Um magnetómetro é um instrumento para medir a força e, por vezes, a direção dos campos magnéticos. Neste caso, o campo magnético da Terra. Os *smartphones* vêm equipados com um magnetómetro para que sejam capazes de detetar a sua orientação no espaço, e desta forma, determinar a sua orientação em relação ao Norte (ou Sul) Magnético [14] [27].

A exatidão do sensor pode variar muito dependendo da posição geográfica do *smartphone* e do ruído magnético presente, mas podemos esperar um erro de 1 a 4 graus relativamente à deteção da sua orientação em relação ao Norte (ou Sul) Magnético [37].

4.2.2 Acelerómetro

O acelerómetro é uma ferramenta embutida num *smartphone* para medir a sua aceleração. Deteta os diferentes movimentos como agitar, inclinar, oscilar e rodar e, consequentemente, permite ao *software* do *smartphone* determinar a orientação atualizada do mesmo.

Apesar de ser relativamente preciso, a acumulação de erros (o cálculo da posição através da aceleração envolve um duplo integral[1] [28]) impede que possa ser usado isoladamente para a localização do *smartphone* a partir de um ponto de referência. Tal acontece também para a orientação, pelo que é necessária a assistência de um giroscópio para tal.

4.2.3 Giroscópio

Um Giroscópio pode ser entendido como um dispositivo que é utilizado para manter uma direção de referência ou fornecer estabilidade na navegação, estabilizadores, etc. Da mesma forma, um giroscópio ou um sensor Giroscópio está presente nos *smartphones* para detetar velocidade angular de rotação e aceleração. Juntamente com o acelerómetro, é capaz de detetar a orientação do *smartphone* [21]. Os giroscópios dos *smartphones* atuais são relativamente precisos, com erros expectáveis de 0.5º a 3º [39].

4.2.4 Barómetro

Um barómetro é um instrumento científico que é utilizado para medir a pressão do ar num determinado ambiente. Esta informação, quando calibrada com informação atmosférica pode ser usada para determinar a altitude do *smartphone* [9] [8].

[1] Qual é a precisão do mundo real dos acelerómetros telefónicos quando utilizados para posicionamento? 2022

Porém, é de notar que apesar da informação de altitude ser bastante precisa, é pouco exata [19]. Isto é, apesar de ser capaz de detetar com precisão variações na altitude, a informação sobre altitude ou elevação absoluta pode ter erros de 20 a 30 metros [13]. Estas características fazem com que, no caso de uma aplicação de AR que dependa da informação de elevação, a informação de localização seja muito precisa dentro de uma mesma sessão, mas numa sessão futura ou sessão de outro utilizador, seja necessário recalibrar a elevação dos objetos relativamente ao *smartphone*.

4.2.5 Sensores de Localização

Os *smartphones* atuais usam uma variedade de ferramentas para serem capazes de obter uma posição exata da sua localização, entre estas estão GPS, sinal de torres celulares, sinal Wi-Fi, e outras [20]. A utilização destas ferramentas em conjunto é capaz de gerar estimativas de posição muito exatas (cerca de 1-2 metros de erro [15] [38]), mas, como existem vários fatores variáveis que podem afetar a qualidade desta estimativa (Satélites GPS obstruídos, Sinal móvel fraco, Sinal Wi-Fi fraco, ...), é expectável que esta exatidão varie muito consoante o local onde está o *smartphone*.

Para avaliar a exatidão do sinal no local onde serão testados os "protótipos"desenvolvidos, foi criada uma aplicação web para permitir visualizar a posição e o percurso do *smartphone* em tempo real, e, posteriormente, percorrido um percurso linear de modo a poder visualizar as posições registadas. O resultado inicial é apresentado na Figura 17.

Podemos ver que a exatidão da localização é baixa (cerca 4 metros de distância, por vezes chegando aos 20 metros). Uma exatidão da localização com esta grandeza não será ponderável para a utilização em contexto de Realidade Aumentada, porque este erro de localização, isoladamente, já seria suficiente para causar desvios não aceitáveis no objetos virtuais.

Estes resultados porém, devem-se ao facto da funcionalidade de localização web, por omissão, ser muito pouco exata, com o objetivo de ser mais rápida e poupar energia. Existe, no entanto, a possibilidade de requisitar uma localização mais precisa ativando a *flag* "enableHighAccuracy"(disponível em todos os Browsers). Esta flag indica à aplicação que pretendemos obter o melhor resultado de localização possível, mesmo que em troca de tempos de resposta mais lentos ou maior consumo de bateria. O resultado obtido utilizando a localização mais exata é apresentado na Figura 18.

Estes resultados mostram-se muito mais promissores e permitem prever posicionamentos de objetos virtuais muito mais rigorosos.

4.2.6 Mapas 3D e Tecnologia LiDAR

Se pretendermos desenvolver uma aplicação que não se baseie somente nos sensores de localização do *smartphone* então é necessário que este possua a tecnologia necessária para obter um mapa 3D do ambiente que o rodeia. Para isto existem duas abordagens significativas, *Computer Vision* e Light detection and Ranging (LiDAR).

Figura 17: Localização registada durante o percurso

LiDAR significa *Light Detection And Ranging*, mas também pode ser comummente referido como "3D laser scanning"ou alguma variação sobre o mesmo. A tecnologia funciona enviando ondas de luz que refletem em superfícies e medindo o tempo de voo para determinar a forma e distância dos objetos na área com maior precisão do que apenas a câmara fotográfica é capaz de obter. Oferece também possibilidades como a oclusão de conteúdo aumentado se este for colocado atrás de algum objeto físico [26].

Esta tecnologia está presente apenas nos dispositivos móveis **iOS** (*iPhones* e **iPads**) mais recentes [26]. A tecnologia LiDAR já esteve presente em dois *smartphones* **Android** (**Lenovo Phab2 Pro** e **Asus ZenFone AR**), que utilizando o sensor LiDAR juntamente com a plataforma **Google Tango** [18] permitiam atingir o objetivo mencionado acima. Esta tecnologia porém, acabou por ser abandonada [18] e a **Google** acabou por optar por utilizar *Computer Vision* para atingir este propósito. Recentemente, foi disponibilizada, no contexto de **ARCore**, para novos modelos **Android** a tecnologia **Depth API** [3], que tem como objetivo melhorar os mecanismos de deteção do ambiente e oferece também possibilidades como a oclusão de conteúdo aumentado.

Figura 18: Localização registada durante o percurso com a flag "enableHighAccuracy"ativa

4.2.7 Considerações com a bateria

Num contexto de computação móvel, a utilização dos sensores de localização, Câmara + Algoritmos de deteção de imagens ou ambos, terá sempre um grande impacto no consumo de bateria deste. Pelo que tal deve ser levado em consideração. Não é expectável que a bateria de um *smartphone* atual seja capaz de permitir a utilização prolongada da aplicação, porém esta tem como objetivo ser utilizada apenas brevemente permitindo ao operador encontrar rapidamente o equipamento que procura. Assim, não é expectável que o operador tenha problemas com descarregamento da bateria. Em qualquer caso, se tal problema se colocar existe também a possibilidade de este (se já não o fizer), se fazer acompanhar de uma bateria externa portátil, pelo que, este fator, para já, não parece ser limitativo para a aplicação.

4.3 Conclusão

É de esperar que soluções que se baseiem somente nos sensores de localização (**AR.JS**) e orientação do *smartphone* sejam muito pouco precisas, bem menos do que é expectável para a aplicação, se bem que, este tipo de soluções sejam, à partida, mais simples de desenvolver. Por outro lado, é expectável que as soluções baseadas na deteção de objetos ou imagens (**Vuforia**), consigam atingir um nivel de precisão muito mais elevado, apesar de poderem apresentar mais dificuldades na utilização destas, na medida em que a deteção de certos equipamentos com formas não regulares, como, por exemplo, alguns postes, poderá ser bastante difícil. Assim, as soluções que fizeram uso de ambas possibilidades (**Cloud Anchors**) poderão permitir tanto uma melhor experiência de desenvolvimento, como uma maior precisão do conteúdo, mas poderão introduzir requisitos não adequados a esta solução (ligação à Internet, maior uso de bateria), pelo que, se considera que até aqui não faz sentido a exclusão de um estudo mais aprofundado e realização de testes de nenhuma das ferramentas expostas.

5

Soluções e ferramentas de Realidade Aumentada

Este Capítulo foca-se na seleção das soluções mais relevantes entre aquelas expostas no Capítulo 2 e no desenvolvimento das aplicações de protótipo para cada uma destas, de modo a melhor entender até que ponto estas poderão ser úteis para o problema em análise.

5.1 Comparação entre as soluções

As soluções apresentadas na Secção 2.4.2, dividem as suas abordagens em dois métodos:

- Deteção de Objetos
- Localização através de sensores

Porém, algumas delas suportam os dois métodos separadamente e outras utilizam uma mistura dos dois para criar a experiência de realidade aumentada. Desta forma, podemos sub-classificar as soluções nas seguintes categorias:

- Localização através de sensores (**AR.JS**, **Vuforia**, **Wikitude**)
- Cloud Anchors (**Google Cloud Anchors**, **Azure Spatial Anchors**)
- Localização através de VPS (**Geospatial API**)
- Deteção de Objetos (**Vuforia**, **Wikitude**)

Duas categorias anteriores, **Cloud Anchors** e **Localização através de VPS**, fazem uso simultâneo da localização através de sensores e deteção de objetos. Faz sentido, devido às duas diferentes abordagens, explorar estas quatro categorias, teremos portanto de determinar a opção mais relevante para cada uma destas.

Esta secção dedica-se à escolha da solução que será utilizada para cada uma.

5.1.1 Localização através de sensores

Das soluções apresentadas, três suportam o posicionamento de conteúdo aumentado baseado apenas na localização do *smartphone*, sendo estas:

- **AR.JS**
- **Wikitude**
- **Vuforia**

Entre estes, **AR.JS** destaca-se claramente para o problema em questão, por três razões:

- **Acessibilidade:** O facto de ser desenvolvido sob tecnologia web facilita o desenvolvimento e permite que a aplicação seja disponibilizada para web, em adição às plataformas suportadas pelas outras soluções (uma aplicação web pode ser disponibilizada como aplicação mobile).
- **Custo:** Os SDKs **Vuforia** e **Wikitude** necessitam de uma licença para serem utilizados profissionalmente, o que não é justificável se apenas pretendermos utilizar a funcionalidade de localização através de sensores.
- **Simplicidade:** Por se tratar de uma biblioteca Web e não de um SDK completo para o desenvolvimento de aplicações de AR, é expectável que seja mais simples de utilizar.

AR.JS é assim a solução selecionada para esta abordagem.

5.1.2 Cloud Anchors

Existem duas soluções disponíveis para a utilização de Cloud Anchors atualmente:

Soluções de Spatial Anchors			
Provider	Preço	Plataformas Suportadas	OSs Suportados
Azure Spatial Anchors	10.000/Mês Grátis + 0.02$/request	Android Nativo & iOS Nativo & Unity	Android & iOS
Google Cloud Anchors	Grátis? (Max 300Req/Min)	Android Nativo & iOS Nativo & Unity	Android & iOS

Tabela 1: Características das Cloud Anchors disponíveis

Estas soluções são bastante semelhantes em termos de funcionalidade e plataformas suportadas (apesar de ser possível utilizar **Google Cloud Anchors** com **flutter**, esta opção foi excluída por ser pouco relevante).

Azure Spatial Anchors foi a solução selecionada para esta abordagem, pelo facto de ter um modelo de preços melhor definido (não é clara na página da **Google Cloud Anchors** a limitação nem o custo de

utilização desta solução), o que poderá indicar que este é um projeto melhor suportado. O facto de ser disponibilizada uma aplicação de demonstração para **Azure Spatial Anchors** também favorece esta ferramenta, na medida em que encurta o tempo necessário para realizar os testes com esta.

Resta definir a plataforma para utilizar esta solução. Aqui, a escolha torna-se mais simples já que existe documentação oficial em ambas as soluções para **Unity**. Esta plataforma, para além de ter a vantagem de suportar tanto **iOS** e **Android** tem também as outras vantagens mencionadas no Capítulo 2 (Unity as a library, Ambiente de desenvolvimento 3D), o que a torna a escolha mais atrativa.

5.1.3 Localização através de VPS

Relativamente à localização com VPS só temos uma opção disponível, já que se trata de uma abordagem relativamente recente. Assim, a solução utilizada será **Geospatial API**.

Tal como mencionado no Capítulo 2 esta solução está disponível como uma API para **ARCore** e desta forma pode ser utilizada em **Android** e **iOS** nativos. Das ferramentas multiplataforma, só **Unity** suporta **Geospatial API**, porém, esta plataforma é bem suportada e possui documentação oficial. Desta forma, devido às razões mencionadas na secção anterior, daremos preferência a **Unity** para a utilização desta solução.

5.1.4 Deteção de Objetos

Existem duas soluções disponíveis para a utilização de Deteção de Objetos atualmente:

Soluções para Deteção de Objetos			
Provider	Preço	Plataformas Suportadas	OSs Suportados
Vuforia	Personalizado	Android Nativo & iOS Nativo & Unity	Android & iOS
Wikitude	2490€/ano	Android Nativo & iOS Nativo & Unity	Android & iOS

Tabela 2: Características das Soluções para Deteção de Objetos disponíveis

Estas soluções colocam o mesmo dilema que encontramos com **Cloud Anchors**, na medida em que são bastante semelhantes e, o problema em questão, mais do que perceber qual é a ideal para o caso em questão, é entender se faz sentido o uso de uma solução com esta abordagem para tal.

Vuforia é assim a solução selecionada para esta abordagem, por ser ligeiramente mais popular e possuir mais recursos de aprendizagem, sendo, desta forma, mais acessível.

Relativamente à plataforma escolhida, esta será mais uma vez **Unity**, pelas razões já mencionadas na secção das **Cloud Anchors**.

5.2 Testes Realizados

Para avaliar os resultados produzidos pelas ferramentas em questão, foi planeado um teste simples que procura, se bem que de um modo simplista, simular a utilização da aplicação pelos operadores no terreno.

Este teste consiste no catálogo de alguns equipamentos de rua, que neste caso serão postes de Luz e árvores. O objetivo será posteriormente localizar estes "equipamentos", apontando o *smartphone* para a localização onde foram registados. A satisfação com o resultado do teste estará dependente de quão estável e de quão perto o objeto virtual está do objeto real registado.

Todos estes testes, exceto se houver menção em contrário, foram realizados num *smartphone* **Samsung Galaxy A40**[1].

5.2.1 AR.JS

A primeira ferramenta a utilizar, por razões já anteriormente especificadas, foi **AR.JS**.

Para realizar o teste, foi desenvolvida uma aplicação mínima através de um único documento **HTML**. A utilização desta biblioteca num projeto mais extenso poderá obrigar a uma instalação mais complexa, mas neste caso limita-se apenas à inclusão de algumas *Tags* de **Script** na **Head** do documento. Surgiram, mesmo assim, alguns problemas na obtenção destes **Scripts**, relacionados com problemas de versões ou *Rate-Limiting*, porém, tais puderam ser resolvido fazendo o download local dos mesmos.

Depois de solucionado este problema, o desenvolvimento da aplicação foi relativamente simples. Os objetos são posicionados através de uma Latitude, Longitude e Altitude fornecidas, desta forma para fazer o registo de um equipamento basta colocar o *smartphone* no local onde se pretende registar o equipamento e obter as coordenadas através dos dados de GPS e Altitude do *smartphone*. Posteriormente esses equipamentos são colocados na **Scene** para que possam ser observados.

Como podemos observar pela Figura 19, a dependência total nos sensores de localização e orientação do *smartphone* leva a que a posição dos objetos colocados se desvie significativamente da sua posição real, já que estes sensores incorrem em erros.

5.2.2 Azure Spatial Anchors

Em seguimento dos testes com **AR.JS**, estes foram realizados utilizando **Azure Spatial Anchors**. Para realizar os testes com esta ferramenta, porém, não foi necessário o desenvolvimento de uma aplicação de teste, já que é disponibilizada uma na documentação da mesma. A inclusão deste ferramenta de mecanismos de deteção de superfícies e reconhecimentos de imagens também possibilita a sua utilização *Indoors*, como podemos observar na Figura 20.

Podemos observar na Figura 21, que o nível de precisão da localização dos equipamentos é bastante superior aos resultados obtidos anteriormente. Estes resultados já estão a um nível que se possa considerar aceitável para a utilização por um operador no terreno.

[1] Samsung Galaxy A40

Figura 19: Localização de equipamentos com AR.JS

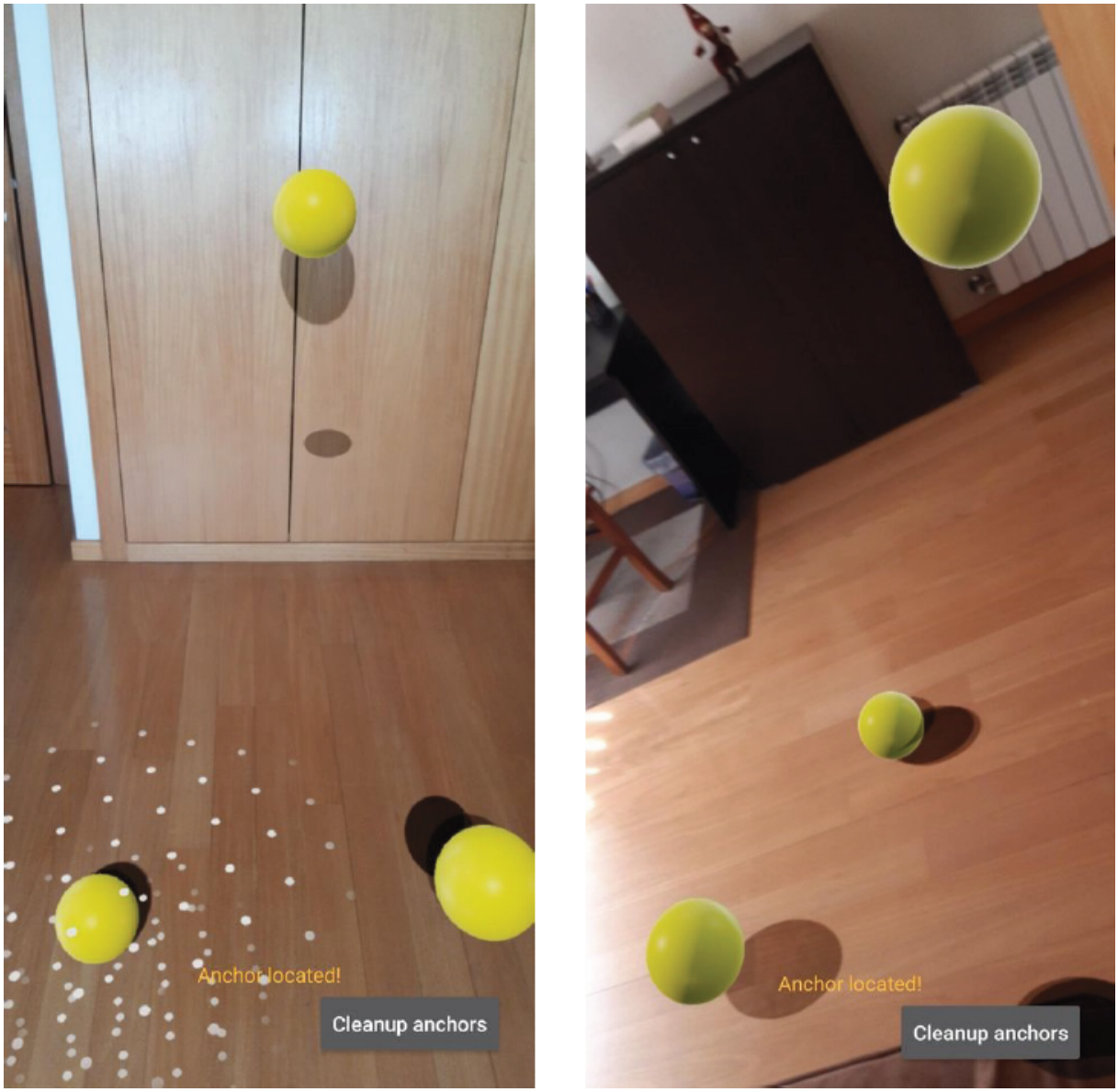


Figura 20: Localização de equipamentos com Azure Spatial Anchors Indoors

Porém, esta solução tem duas desvantagens que entram em conflito com os requisitos especificados acima. Por um lado, necessita de acesso à Internet sempre que é necessário registar, posicionar uma âncora, o que a torna incongruente com a app **Surveying** neste aspeto. Por outro lado, requer que todos os equipamentos sejam registados neste sistema, o que obriga a um registo manual e obtenção de fotografias de todos os equipamentos já registados no sistema **Netwin**, o que não será viável.

Desta forma, a procura de soluções segue na expectativa de que possamos encontrar uma que não possua estas desvantagens.

Figura 21: Localização de equipamentos com Azure Spatial Anchors

5.2.3 Vuforia

A terceira solução a ser testada, será **Vuforia**. Devido às diferentes abordagens e pré requisitos impostos pela abordagem de deteção de imagens/modelos utilizando **Vuforia**, foram realizados diferentes testes com esta solução de acordo com as suas abordagens e serão apresentados separadamente. Tal como decidido na Secção 5.1.4, os exemplos demonstrados nesta secção foram desenvolvidos utilizando a extensão para Unity.

Tipos de alvos suportados

Como referido anteriormente, o SDK suporta vários tipos de alvos para ancorar os objetos virtuais. Para o *use-case* em causa, foi considerado que os mais relevantes seriam **imagens**, **Alvos Múltiplos** e **modelos 3D**. Para cada um destes tipos de alvos foram gerados alguns exemplos de modo a poder perceber o nível de precisão e eficácia da deteção de cada um destes por parte do SDK.

Deteção de imagens

Os **Alvos de Imagem** representam imagens que o **Vuforia Engine** pode detetar e seguir. O *engine* deteta e segue a imagem comparando as características naturais extraídas da imagem da câmara com uma base de dados de recursos alvo conhecida. Uma vez detetado o **Alvo da Imagem**, o **Vuforia Engine** irá rastrear a imagem e aumentar o seu conteúdo[2].

O *engine* é capaz de detetar várias imagens simultaneamente (o número específico pode ser ajustado de acordo com os requerimentos e o desempenho do *smartphone*) e é capaz de detetar imagens com forte inclinação ou em fracas condições de luminosidade, sendo que a eficácia da deteção depende muito da qualidade da imagem. Idealmente uma imagem terá os seguintes atributos:

- Rica em detalhes
- Bom contraste
- Sem padrões repetitivos

É de notar que estes atributos continuam relevantes para as imagens que constituem os **Alvos Múltiplos** e para as superfícies dos **Alvos de Modelo**.

Introdução aos exemplos

Para manter alguma consistência entre os exemplos dos diferentes tipos de alvos, foi utilizado um objeto recorrente para todos estes. O objeto foi o livro apresentado na Figura 22:

[2] Alvos de Imagem Vuforia, visitado em 2022

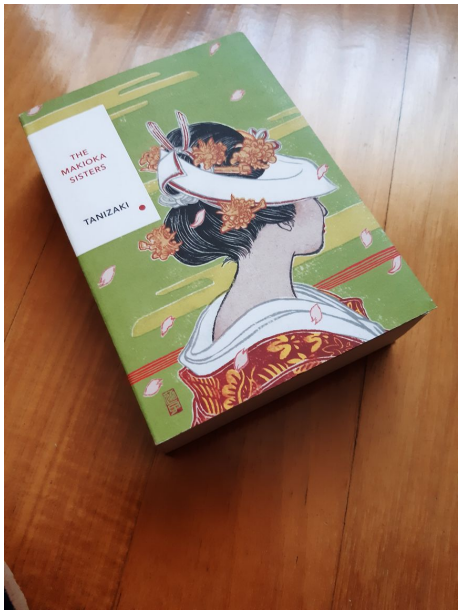


Figura 22: Fotografia do livro

Este objeto é um bom alvo de exemplo já que contém todas as características desejáveis para um alvo mencionadas acima, e tem uma forma geométrica regular, o que permite ser detetado através de um **Alvo de Imagem**, **Alvo Múltiplo** ou **Alvo de Modelo**.

Exemplo

Para o teste de deteção de Alvos de Imagem foi utilizada a imagem presente na Figura 23.

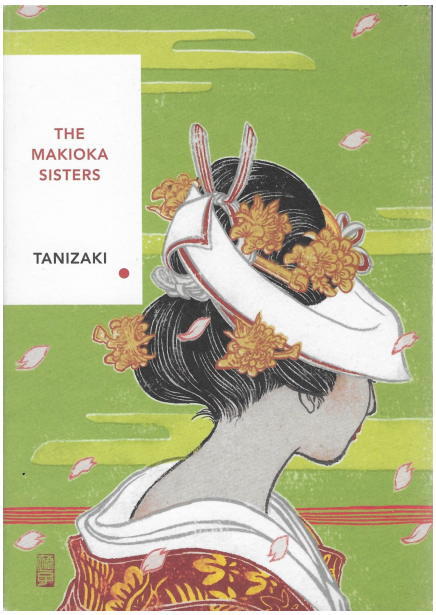


Figura 23: Capa do livro digital

Após a deteção da imagem, o Engine foi capaz de adicionar o conteúdo aumentado pretendido, como pode ser observado na Figura 24.

Figura 24: Conteúdo aumentado colocado após a deteção da capa do livro

Como podemos verificar, o conteúdo é adicionado sem qualquer consideração daquilo de deverá ficar visível ou oculto, isto ocorre porque no caso dos **Alvos de Imagem**, o *engine* apenas usa a imagem como uma âncora para o conteúdo aumentado, esta não é tomada como um objeto físico, e daí não existir qualquer ocultação da parte do conteúdo aumentado que ficaria atrás da imagem.

Uma outra desvantagem da utilização de **Alvos de Imagem** para a deteção de objetos com várias faces distintas é a de que se pretendermos que o objeto seja detetável através de mais do que uma face, teremos de adicionar um **Alvo de Imagem** novo para cada face, e será necessário solucionar o problema que surge quando várias faces forem visíveis e detetáveis simultaneamente, e o *engine* gerar múltiplas vezes o conteúdo aumentado (uma vez por cada face visível).

Desta forma podemos concluir que **Alvos de Imagem** serão úteis para equipamentos em que somente uma face seja visível, para outros equipamentos é expectável que **Alvos Múltiplos** ou **Alvos de Modelo** sejam mais úteis.

Deteção de Alvos Múltiplos

Um **Alvo Múltiplo** é uma coleção de Alvos de Imagem múltiplos combinados numa disposição geométrica definida, tal como caixas. Isto permite rastrear e detetar o alvo de todos os lados sendo que o *engine* trata este conjunto de imagens como uma única entidade, isto é, não adiciona o conteúdo virtual múltiplas vezes no caso de conseguir detetar mais de uma das imagens que constituem o **Alvo Múltiplo**[3].

[3] Alvos Múltiplos Vuforia, visitado em 2022

Uma outra vantagem da utilização de **Alvos Múltiplos** é a possibilidade de o *engine* ocultar partes do conteúdo aumentado que não estariam visíveis devido à forma do alvo, o que torna a utilização de **Alvos Múltiplos** desejável, mesmo para caixas em que apenas uma das faces seja detetável (isto é, tenha as características especificadas anteriormente).

Para a utilização de **Alvos Múltiplos**, porém, é necessário que os alvos tenham formas geométricas restritas, mais especificamente, têm de ser um cuboide ou cilindro, para outras formas geométricas, será necessário usar um **Alvo de Modelo**.

Exemplos

Para o teste de deteção de **Alvos de Múltiplos** foi utilizado o mesmo livro dos testes anteriores.

Como podemos observar nas Figuras 25, 26 e 27, o *engine* já é capaz de gerar a oclusão das partes do conteúdo aumentado que seriam ocultadas pelo próprio alvo. Utilizando os sensores de movimento do *smartphone* e algoritmos de visão, o *engine* também é capaz de continuar a posicionar o conteúdo aumentado mesmo que, pelo movimento do alvo ou do *smartphone*, deixem de existir faces detetáveis visíveis, é exemplo disso a Figura 27 (a face da lombada, apesar de ter alguns contrastes, não é muito facilmente detetável, mesmo assim, o *engine*, após ter detetado a capa ou contracapa, é capaz de detetar o movimento do livro e manter o conteúdo posicionado), não é o caso se ocorrer um movimento demasiado brusco no entanto.

Deteção de Alvos de Modelo

Os **Alvos de Modelos** permitem que as aplicações construídas com o **Vuforia Engine** reconheçam e sigam determinados objetos no mundo real, com base na forma e na superfície do objeto. Este será, em princípio o tipo de alvo mais adequado para a deteção dos equipamentos no terreno, já que se espera que a maior parte destes tenham superfícies complexas e/ou não facilmente detetáveis individualmente através do reconhecimento de imagens[4].

Preparação de modelos

Para gerar um **Alvo de Modelo** para um determinado objeto é necessário ter acesso aos dados de um modelo 3D do mesmo, tal como um modelo Computer-Aided Design (CAD) 3D ou uma digitalização 3D do objeto. Porém, somente um modelo do objeto não é suficiente para a deteção deste por parte do *engine*, este tem primeiro se ser processado para que se adapte ao algoritmo de deteção do *engine*, para isso está disponível a aplicação **Model Target Generator**.

O **Model Target Generator** toma como entrada um modelo 3D representando o objeto que deseja rastrear, verifica a sua adequação e permite configurá-lo para uma deteção ótima com vistas de guia e

[4] Alvos Modelo Vuforia, visitado em 2022

Figura 25: Conteúdo aumentado colocado após a deteção da capa do livro

Figura 26: Conteúdo aumentado colocado após a deteção da contracapa do livro

Figura 27: O conteúdo aumentado visto a partir da lombada do livro

vistas avançadas. O **Model Target Generator** gera uma base de dados **Vuforia** que pode utilizar com a integração em **Unity** do **Vuforia Engine** a fim de detetar o objeto.

Além disso, o **Model Target Generator** permite treinar as bases de dados através de um processo de treino de aprendizagem profunda baseado na *Cloud*, produzindo uma base de dados de **Alvos de Modelos Avançados**. Tal base de dados contém um ou mais **Alvos de Modelo** em que cada um contém uma ou mais **Visões-Guia** com um alcance de reconhecimento até 360 graus. Os **Alvos de Modelo Avançado** suportam o reconhecimento e deteção de um objeto a partir de qualquer posição do intervalo de reconhecimento definido sem exigir que o utilizador alinhe manualmente a vista com o objeto físico.

Obtenção dos modelos

Para a utilização de aplicação anterior com o propósito de gerar o modelo que pode ser usado em Unity é preciso primeiro obter o modelo 3D do alvo pretendido. Para este fim, existem várias aplicações, gratuitas ou não, disponíveis nas **App Stores**, neste caso concreto, não foi considerado justificável efetuar uma análise comparativa entre soluções e foi utilizada a aplicação **Polycam**, que está disponível para **Android** e **iOS**. A aplicação efetua a captura do modelo 3D através de fotogrametria [30] e é capaz de fazer uso de um sensor LiDAR (nos dispositivos **iOS** que o possuírem) para permitir gerar o modelo mais rapidamente e com melhor qualidade. Para gerar os modelos dos exemplos seguintes não foi possível utilizar tais dispositivos, pelo que foram gerados apenas através de imagens capturadas pela da câmara da *smartphone*.

Exemplos

Para realizar os testes da deteção de **Alvos de Modelo** foram capturados dois modelos. O primeiro, é mais simples, e, à partida, mais facilmente detetável (livro). O segundo corresponde a um equipamento real (caixa de distribuição) e, devido à maior complexidade que possui em termos de geometria e ao facto de ter faces mais homogéneas, é expectável que seja mais difícil de detetar.

Livro

A deteção do livro por parte do *engine*, em geral, não é imediata e exige que posicionemos o *smartphone* em alguns ângulos até ser capaz de o detetar, mas quando o consegue, é capaz de posicionar o conteúdo aumentado com precisão, mesmo que o *smartphone* esteja em movimento.

Figura 28: Modelo gerado do livro

Figura 29: Livro acompanhado de conteúdo aumentado

Caixa de Distribuição

Por se expectar que os **Alvos de Modelo** sejam mais adequados à maioria dos equipamentos em questão (já que possuem formas geométricas irregulares), e de modo a avaliar a capacidade de deteção de um equipamento real, foi realizado um segundo teste numa caixa de distribuição que se encontra no terreno.

Figura 30: Fotografia da Caixa de Distribuição

Figura 31: Modelo gerado da Caixa de Distribuição

Neste caso, não foi possível capturar uma imagem com o conteúdo aumentado uma vez que o *engine* não conseguiu detetar a caixa de distribuição através deste modelo. Dado o facto de a caixa possuir um modelo geométrico regular, coloca-se a possibilidade de tentar efetuar a deteção através de um modelo genérico com as mesmas dimensões como o da figura 32. Neste caso concreto, esta abordagem foi capaz de produzir melhores resultados e o *engine* foi capaz de detetar a caixa e colocar o conteúdo aumentado, tal como observado na figura 33.

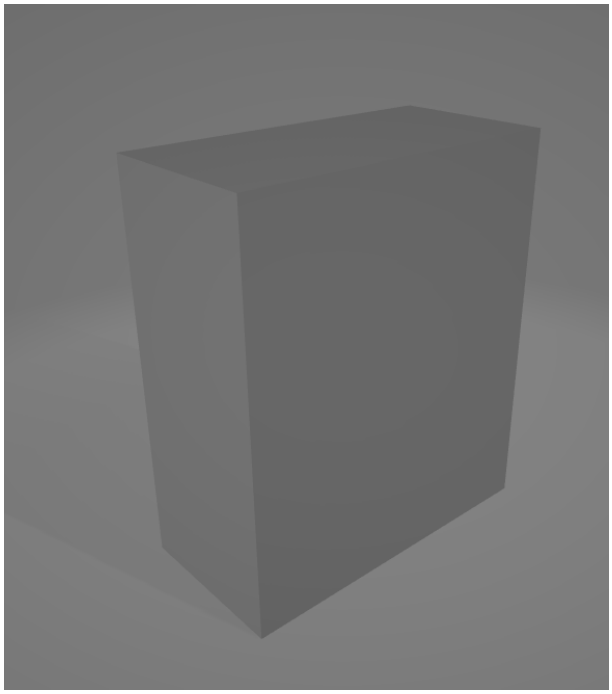


Figura 32: Modelo genérico de uma caixa com as dimensões pretendidas

Figura 33: Caixa acompanhada de conteúdo aumentado

Problemas possíveis

A incapacidade e dificuldade do *engine* em detetar certos modelos pode dever-se a algumas causas não mutuamente exclusivas. A primeira será o facto de o *engine* ter dificuldades em detetar modelos deste tipo, isto é, com faces demasiado homogéneas e sem grande diferença de cor entre estas. O segunda corresponde ao facto da qualidade dos modelos ser demasiado baixa. Estes modelos capturados são obviamente de má qualidade (um *smartphone* **iOS** com tecnologia LiDAR seria capaz de gerar um modelo com mais qualidade) e não foram devidamente cortados (a aplicação **Polycam** só disponibiliza essa funcionalidade para **iOS**), o que pode ter impactado o funcionamento do *engine*. Por fim, estas dificuldades também se podem dever ao facto do *smartphone* utilizado não possuir a tecnologia de deteção

de profundidade que seria recomendada para este tipo de modelos[5]. Melhores resultados poderiam potencialmente ser obtidos num dispositivo **Android** com suporte para **Depth API** ou num dispositivo **iOS** que possua um sensor LiDAR.

De qualquer forma, este tipo de solução baseada na deteção de modelos terá sempre dificuldades na deteção de equipamentos com formas irregulares, e, individualmente distintas como por exemplo postes, e mesmo que fosse capaz de o fazer, ancorar conteúdo aumentado nestes traria ainda mais desafios.

5.2.4 ARCore Geospatial API

Tal como aconteceu no caso das **Azure Spatial Anchors**, também esta solução apresentava uma aplicação de demonstração disponível em **Unity**, desta forma, os testes aqui apresentados foram realizados usando essa mesma aplicação. Os resultados desta aplicação podem ser observados nas Figuras 34 e 35.

Figura 34: Localização com Geospatial API

[5] Dispositivos com suporte para **Depth API**, visitado em 2022

Figura 35: Localização com Geospatial API

Em segunda análise, foi possível descobrir que estes resultados foram obtidos sem a localização com **Geospatial API**. A utilização em **Unity** revelou-se complexa e não foi possível obter um exemplo na aplicação demo nesta *framework* em que o modo de localização com VPS estivesse ativo. Porém, esta experiência permitiu observar que é possível fazer uso destas âncoras espaciais locais sem a localização com **Geospatial API**, apesar da degradação da precisão. Isto indica que uma solução que utilize esta ferramenta será capaz de funcionar offline, apesar de ser preferível que tenha acesso à Internet.

Em alternativa a **Unity** foi utilizada a aplicação de demo em **Android Nativo**, em que o acesso e configuração da primitivas **ARCore** e **Geospatial API** é mais simples. Os resultados desta nova aplicação podem ser observados na Figura 36.

Foi possível também verificar que a aplicação é capaz de funcionar sem acesso à Internet e em locais onde não se encontra disponível **Street View**. Em ambos os casos a precisão da aplicação simplesmente reverte para a verificada na aplicação anterior, sem qualquer outro impedimento ao seu funcionamento.

Como podemos observar, os resultados obtidos não foram ideais mas mostram-se mais precisos que

Figura 36: Localização precisa com Geospatial API

os obtidos na aplicação **AR.JS** que só utilizava os sensores de GPS. Em particular, a aplicação, se for capaz de captar o espaço envolvente com um certo nível de completude, é capaz de atingir precisão vertical na ordem de um metro, significativamente mais preciso que a aplicação **AR.JS**. A precisão é visivelmente inferior à obtida com as **Azure Spatial Anchors**, porém, esta ferramenta apenas necessita das coordenadas de posicionamento do equipamento e não de um registo prévio do equipamento como âncora, requerimento esse que tornou as **Azure Spatial Anchors** numa solução pouco atrativa.

Em especial, esta ferramenta destaca-se das outras pois, apesar de, tal como a solução **Spatial Anchors**, requerer acesso à Internet, cumpre com os restantes requisitos (em especial não necessita de um registo ou de um modelo do equipamento para ancorar o conteúdo aumentado), e é capaz de utilizar todos os sensores disponíveis no *smartphone* para atingir o objetivo proposto, o que a coloca numa situação favorável para apresentar melhores resultados em caso de melhoria destes, isto é, são esperados resultados mais precisos por parte desta aplicação se for utilizada num *smartphone* que possua **Depth API** ou tecnologia LiDAR, já que beneficia de um melhor mapeamento do espaço envolvente. Da mesma forma, é expectável uma melhoria na precisão no caso de ser utilizada por um *smartphone* com um melhor sensor de GPS (posicionamento em geral, wifi, rede, ...), ou se for utilizada numa área com melhor recessão destes sinais.

5.3 Conclusão

O estudo e os testes realizados neste Capítulo permitiram-nos ter maior clareza relativamente às características e funcionamento das ferramentas e técnicas em questão. Desta forma, para cada ferramenta foram determinadas as seguintes vantagens e desvantagens:

- **Localização (AR.JS)**
  - Vantagens
    - Gratuito
    - Funciona offline
  - Desvantagens
    - Baixa Precisão
- **Cloud Anchors (Azure Spatial Anchors)**
  - Vantagens
    - Muito Preciso
  - Desvantagens
    - Impõe custos adicionais

        - Requer acesso à Internet
        - Requer que todos os equipamentos sejam registados neste sistema

- **Deteção de Objetos (Vuforia)**
    - Vantagens
        - Muito Preciso
        - Funciona offline
    - Desvantagens
        - Impõe custo adicionais
        - Requer a criação de modelos 3D para todos os equipamentos
        - A deteção de certos equipamentos pode ser difícil

- **Localização com VPS (Geospatial API)**
    - Vantagens
        - Gratuito
        - Funciona offline, apesar de com menor precisão
        - Preciso, se tiver acesso à Internet
    - Desvantagens
        - Requer acesso à Internet para fornecer maior precisão.
        - Só é capaz de fornecer localização precisa em locais com Google Street View disponível.

Tendo em conta este conjunto de características, a escolha da ferramenta mais indicada para o problema em questão torna-se mais simples do que era aparente anteriormente. **Deteção de Objetos (Vuforia)** e **Cloud Anchors (Azure Spatial Anchors)** mostraram-se incompatíveis com o problema em questão, ambas por imporem requisitos que são impraticáveis tendo em conta a quantidade e a diversidade de equipamentos que já se encontram dispersos no terreno e registados no sistema Netwin.

Desta forma, a escolha cai entre **Localização (AR.JS)** e **Localização com VPS (Geospatial API)**. Porém, como podemos observar nos testes realizados, **Geospatial API** é capaz de funcionar baseada somente em localização quando os seus requerimentos de Internet ou **Google Street View** não são atendidos. A precisão neste caso, aparenta até ser superior à observada nos testes com **AR.JS**. Isto leva-nos a concluir que esta será uma solução mais atrativa que **AR.JS** mesmo nos casos em que os seus requerimentos para uma localização mais precisa não são satisfeitos.

Geospatial API é, desta forma, a melhor alternativa. Funciona razoavelmente sem Internet, e torna-se bastante preciso quanto tem Internet sem impor mais nenhum requisito.

6

Conclusão

Relembrando a motivação desta dissertação, esta está divida em duas partes, sendo que termina com a conclusão de ambas. A primeira parte foi encerrada quando se deram como concretizados os requerimentos expostos na Secção 3.4. Entretanto, foi dado seguimento ao projeto, já fora do espetro desta dissertação, e a aplicação entrou no ciclo de desenvolvimento usual da metodologia **Agile**. O protótipo inicial já foi entregue aos clientes e aos utilizadores finais para obter o seu *feedback*, ideias de design e melhoramento, que será tido em conta para os desenvolvimentos do próximo ciclo com vista a melhorar continuamente a aplicação em curtos ciclos tal como especificado na metodologia.

A segunda parte, que termina com o final da dissertação, foi dada como concluída com o estudo e prototipagem das ferramentas que foram tomadas como relevantes para o problema em questão. Podemos observar que, apesar existirem várias abordagens no que diz respeito a AR por localização, nenhuma se adequa perfeitamente ao problema em questão, existindo sempre o *tradeoff* entre restrições/requerimentos e precisão. Porém, a última solução estudada (**Geospatial API** - localização com o VPS da **Google**) destaca-se das outras como a mais provavelmente viável para o desenvolvimento desta aplicação. Não obstante, este é um setor ainda numa fase inicial de desenvolvimento, e, desta forma, ainda muito volátil no que diz respeito aos resultados que podem ser obtidos com certas abordagens, e mesmo em relação às abordagens disponíveis (a **Geospatial API** da **Google** foi lançada durante o desenvolvimento desta dissertação).

O seguimento natural da segunda parte será um estudo juntamente com os operadores no terreno de modo a perceber se as ordens de precisão registadas são satisfatórias para uma utilização eficaz deste tipo de abordagem. Se sim, a próxima fase deste projeto seria um levantamento mais aprofundado e estruturado dos requisitos para a utilização desta aplicação de modo a poder delinear o desenvolvimento da aplicação em grande escala.

Bibliografia

[1] *Angular (web framework)*. Wikipedia. 2021. url: https://en.wikipedia.org/wiki/Angular_(web_framework) (ver p. 8).

[2] *Angular Changelog*. Github. 2014. url: https://github.com/angular/angular/blob/master/CHANGELOG.md#200-rc0-2016-05-02 (ver p. 8).

[3] *AR Core Depth API*. Google. 2022. url: https://developers.google.com/ar/develop/depth (ver p. 36).

[4] *AR.js - Augmented Reality on the Web*. Arjs.org. 2022. url: https://ar-js-org.github.io/AR.js-Docs/ (ver p. 20).

[5] *ARCore*. Wikipedia. 2022. url: https://en.wikipedia.org/wiki/ARCore (ver p. 17).

[6] *ARKit Apple Docs*. Apple. 2022. url: https://developer.apple.com/documentation/arkit (ver p. 17).

[7] *Augmented reality: A class of displays on the reality-virtuality continuum*. Paul Milgram, Haruo Takemura. 1994. url: https://www.researchgate.net/publication/228537162_Augmented_reality_A_class_of_displays_on_the_reality-virtuality_continuum (ver p. 11).

[8] B. Ball. *Why does my phone have a barometer?* NextNav. 2020. url: https://nextnav.com/why-phones-have-barometers/ (ver p. 34).

[9] *Barometer*. Wikipedia. 2022. url: https://en.wikipedia.org/wiki/Barometer (ver p. 34).

[10] A. Biørn et al. *An empirical investigation of performance overhead in cross-platform mobile development frameworks*. Empirical Software Engineering. 2020. url: https://link.springer.com/content/pdf/10.1007/s10664-020-09827-6.pdf (ver p. 10).

[11] C. Bohon. *Apple's ARKit*. Tech Republic. 2021. url: https://www.techrepublic.com/article/apples-arkit-everything-the-pros-need-to-know (ver p. 17).

[12] *Core Concepts*. Ionic. 2021. url: https://ionicframework.com/docs/core-concepts/fundamentals (ver p. 7).

[13] *Elevation Tracker*. Elevation Tracker. 2022. url: http://www.elevationtracker.com/main/accuracy.html (ver p. 35).

[14] T. E. of Encyclopaedia Britannica. *Magnetometer*. Britannica. 2022. url: https://www.britannica.com/technology/magnetometer (ver p. 34).

[15] T. E. of Encyclopaedia Britannica. *Magnetometer*. Britannica. 2022. url: https://medium.com/@importanttech/we-tested-mobile-gps-gnss-accuracy-and-found-some-surprising-results-b9ec35873e2e (ver p. 35).

[16] X. F. Esteban Angulo. *A Case Study on Cross-Platform Development Frameworks for Mobile Applications and UX*. Conference: the XV International Conference. 2014. url: https://www.researchgate.net/publication/301399328_A_Case_Study_on_Cross-Platform_Development_Frameworks_for_Mobile_Applications_and_UX (ver p. 10).

[17] *Flutter (software)*. Wikipedia. 2021. url: https://en.wikipedia.org/wiki/Flutter_(software) (ver p. 10).

[18] *Google Tango (platform)*. Wikipedia. 2017. url: https://en.wikipedia.org/wiki/Tango_(platform) (ver p. 36).

[19] T. G. Haibo Kai Dong. *HiMeter: Telling You the Height Rather than the Altitude*. 2018. url: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6022159/ (ver p. 35).

[20] J. Hildenbrand. *How does GPS work on my phone?* androidcentral. 2020. url: https://www.androidcentral.com/how-does-gps-work-my-phone (ver p. 35).

[21] *How does a Gyroscope Sensor work in your smartphone?* techahead. 2022. url: https://www.techaheadcorp.com/knowledge-center/how-gyroscope-sensor-work-in-smartphone/ (ver p. 34).

[22] *Ionic (mobile app framework)*. Wikipedia. 2021. url: https://en.wikipedia.org/wiki/Ionic_(mobile_app_framework) (ver p. 7).

[23] A. Javornik. *The Mainstreaming of Augmented Reality: A Brief History*. Harvard Business Review. 2018. url: https://hbr.org/2016/10/the-mainstreaming-of-augmented-reality-a-brief-history (ver p. 11).

[24] D. Kristianto. "Winning the Attention War: Consumers in Nine Major Markets Now Spend More than Four Hours a Day in Apps". Em: (2021). url: https://www.data.ai/en/insights/market-data/q1-2021-market-index (ver p. 5).

[25] lars.carius.io. *ar_flutter_plugin*. pub.dev. 2022. url: https://pub.dev/packages/ar_flutter_plugin (ver p. 18).

[26] M. Lowe. *What is LIDAR and why is it in the iPhone 13 Pro?* Pocket-lint. 2021. url: https://www.pocket-lint.com/phones/news/apple/151476-what-is-lidar-ipad-why-arkit-measure (ver p. 36).

[27] *Magnetometer*. Nasa. 2022. url: https://spacemath.gsfc.nasa.gov/Sensors/GuideMagnetism.docx (ver p. 34).

[28] OpenStax. *Finding Velocity and Displacement from Acceleration*. OpenStax. 2022. url: https://phys.libretexts.org/Bookshelves/University_Physics/Book%5C%3A_University_Physics_(OpenStax)/Book%5C%3A_University_Physics_I_-_Mechanics_Sound_Oscillations_and_Waves_(OpenStax)/03%5C%3A_Motion_Along_a_Straight_Line/3.08%5C%3A_Finding_Velocity_and_Displacement_from_Acceleration (ver p. 34).

[29] *Overview of ARCore and supported development environments*. Google. 2022. url: https://developers.google.com/ar/develop (ver p. 17).

[30] *Photogrammetry*. Wikipedia. 2022. url: https://en.wikipedia.org/wiki/Photogrammetry (ver p. 51).

[31] B. Poetker. *What Is Augmented Reality? (+Most Common Types of AR Used Today)*. G2. 2019. url: https://www.g2.com/articles/augmented-reality (ver pp. 13, 14).

[32] *React (JavaScript library)*. Wikipedia. 2021. url: https://en.wikipedia.org/wiki/React_(JavaScript_library) (ver p. 8).

[33] *React Native*. Wikipedia. 2021. url: https://en.wikipedia.org/wiki/React_Native (ver p. 9).

[34] *React Native Out-of-Tree Platforms*. Meta. 2022. url: https://reactnative.dev/docs/out-of-tree-platforms (ver p. 9).

[35] *Realidade aumentada*. Wikipedia. 2021. url: https://pt.wikipedia.org/wiki/Realidade_aumentada (ver pp. 10, 11).

[36] P. Saccomani. "Consumers now average 4.2 hours per day in apps, up 30% from 2019". Em: (2019). url: https://www.mobiloud.com/blog/mobile-apps-vs-the-mobile-web (ver p. 5).

[37] R. Speiden. *SAR Expert: Testing the Best Cell Phone Compass Apps*. ActionHub. 2020. url: https://www.actionhub.com/how-to/2020/10/08/best-cell-phone-compass-apps-wilderness/ (ver p. 34).

[38] S. Technologies. *We Tested Mobile GPS/GNSS Accuracy and Found Some Surprising Results*. Medium. 2020. url: https://frameboxxindore.com/android/how-accurate-is-location-services-on-android.html (ver p. 35).

[39] P. G.
bibinitperiod U.-D. R. Tim Kuhlmann. *Smartphone sensor accuracy varies from device to device in mobile research: The case of spatial orientation*. 2021. url: https://link.springer.com/article/10.3758/s13428-020-01404-5#Sec2 (ver p. 34).

[40] *Unity as a Library*. Unity. 2022. url: https://unity.com/features/unity-as-a-library (ver p. 19).

[41] *Viro React*. Viro Community. 2022. url: https://github.com/ViroCommunity/viro (ver p. 18).

[42] *Vue.js*. Wikipedia. 2021. url: https://en.wikipedia.org/wiki/Vue.js (ver p. 8).

[43] *Vuforia Augmented Reality SDK*. Wikipedia. 2022. url: https://en.wikipedia.org/wiki/Vuforia_Augmented_Reality_SDK (ver p. 21).

[44] *What is a Hybrid Application*. Tech Target. 2019. url: https://searchsoftwarequality.techtarget.com/definition/hybrid-application-hybrid-app (ver p. 7).

[45] *What is Cross-Platform App Development?* Full Scale. 2020. url: https://fullscale.io/blog/what-is-cross-platform-app-development/ (ver p. 9).

[46] *World Geodetic System*. Wikipedia. 2023. url: https://en.wikipedia.org/wiki/World_Geodetic_System (ver p. 22).