



Universidade do Minho

Escola de Engenharia

André Gonçalo Ribeiro da Silva Lopes

**Guias de Realidade Aumentada para a
divulgação de Património
Arquitetónico**

Dissertação de Mestrado

Mestrado Integrado em Engenharia de
Telecomunicações e Informática

Trabalho efetuado sob a orientação de

Professor Doutor Luís Gonzaga Magalhães

Doutor Paulo Bernardes

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



Atribuição-NãoComercial-SemDerivações
CC BY-NC-ND

<https://creativecommons.org/licenses/by-nc-nd/4.0/>

AGRADECIMENTOS

No decorrer da realização desta dissertação contei com o apoio e auxílio de inúmeras pessoas, sem as quais a conclusão da mesma não teria sido possível. Expresso assim os meus agradecimentos mais sinceros.

Aos meus pais, visto que sem eles não seria possível a conclusão do Curso. Deram-me forças para continuar à medida que ia desistindo. O mais sincero obrigado.

Aos meus irmãos, que sempre me ajudaram a escrever a tese com bom humor e disposição.

Ao meu orientador, o Professor Luís Gonzaga, agradeço a oportunidade concedida para a realização deste tema de dissertação, assim como com a sua ajuda e contribuição com sugestões.

Ao meu coorientador, Paulo Bernardes, que foi o principal pilar durante toda esta caminhada e que sem ele seria impossível ter terminado. Um muito obrigado por todas as horas investidas a auxiliar-me e guiar-me em cada etapa do projeto e ao facto de nunca ter desistido de mim e de me puxar para cima nos momentos mais complicados.

Por último, mas não menos importante, agradeço a todos os meus amigos e principalmente à minha namorada pela constante amizade, carinho e apoio que vêm demonstrado ao longo do tempo. Acredito que esta forte amizade tenha tido um contributo fundamental para a conclusão do meu percurso académico.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho acadêmico 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.

RESUMO

A Realidade Aumentada (RA) caracteriza-se pela inserção de elementos virtuais no mundo real, em tempo real. A popularidade da utilização de dispositivos móveis aliada ao aumento das capacidades gráficas e de processamento dos mesmos abriu caminho para uma nova forma de visualização de ambientes de realidade aumentada em qualquer local. A RA tem vindo a ser implementada em diversas áreas do saber, sendo uma delas a divulgação e conservação do Património Arquitetónico e Arqueológico.

Neste projeto é proposto e implementado um sistema de RA para dispositivos móveis capaz de apoiar visitas através de guias previamente configurados. Durante a visita um utilizador direciona a câmara do dispositivo para os marcadores colocados ao longo do espaço de interesse patrimonial de modo a visualizar e obter informação relevante acerca desse espaço. A estes marcadores podem estar associados modelos 3D de artefactos da época, bem como diversos tipos de conteúdos multimédia, áudio, vídeo ou imagem. De forma a facilitar a configuração de uma visita fez-se uso de um único ficheiro de configuração, responsável por cada uma das visitas criadas.

Foi também desenvolvida uma aplicação Web capaz de gerir, configurar e criar os guias de RA de uma forma simples, dinâmica e intuitiva. A utilização desta aplicação estará a cargo dos administradores de visitas.

De forma a testar a aplicação de RA em ambiente real, utilizou-se como caso de estudo o Convento de São Francisco em Real, Braga. Por intermédio dos testes conduzidos conseguiu observar-se a potencialidade de aplicações de RA na valorização e divulgação do património arqueológico. Através desta os visitantes têm acesso a informação que complementa o existente no local, ou podem visualizar elementos de interesse, que graças às vicissitudes do tempo, já não estão no local de origem.

Palavras-Chave: Dispositivos móveis, Património Arquitetónico e Arqueológico, Realidade Aumentada

ABSTRACT

Augmented Reality (AR) is characterized by the combination of virtual elements in the real world, in real-time. The popularity of the use of mobile devices combined with the increase in their graphic and processing capabilities paved the way for a new way of viewing augmented reality environments anywhere. AR has been implemented in several areas of knowledge, one of which is the dissemination and conservation of the Architectural and Archaeological Heritage.

This project proposes and implements an AR system for mobile devices, capable of supporting visits through previously configured guides. During the visit, a user directs the device's camera to the markers placed on the areas of heritage interest to visualize and obtain relevant information about that heritage element. These markers can be associated with 3D models of artifacts, as well as different types of multimedia, audio, video, or image content. To simplify the configuration of a visit, a single configuration file was used, responsible for each of the created visits.

A web application capable of managing, configuring, and creating AR guides in a simple, dynamic and intuitive way was also developed. The use of this application will be the responsibility of the visit administrators.

To test the application of AR in a real environment, the Convento de São Francisco in Real (Braga, Portugal) was used as a case study. Through the tests carried out, it was possible to observe the potential of AR applications in the valorisation and dissemination of archaeological heritage. Through this application, visitors have access to information that complements the existing information on the site, or that visualizes elements of interest, which, due to the changes over time, are no longer in the place of origin.

Keywords: Mobile devices, Architectonic and Archaeological heritage, Augmented reality

Índice

Capítulo 1. Introdução	1
1.1. Enquadramento e motivação	1
1.1. Objetivo da dissertação	2
1.2. Estrutura da dissertação	2
Capítulo 2. Estado da Arte	4
2.1. Resenha Histórica	4
2.2. Sistemas de RA	11
2.2.1. Arquitetura de um Sistema Típico de RA	12
2.2.1.1. Subsistema de Aquisição de Imagem	13
2.2.1.2. Subsistema Gerador de Objetos Virtuais	13
2.2.1.3. Subsistema de Realidades Mistas	14
2.2.1.4. Subsistema de Caracterização do Mundo Real	14
2.2.1.5. Subsistema de <i>Tracking</i>	14
2.2.1.6. Subsistema de Apresentação	14
2.2.2. <i>Tracking</i>	15
2.2.2.1. Tipos de <i>tracking</i>	15
2.2.2.1.1. <i>Tracking</i> baseado em sensores	16
2.2.2.1.2. <i>Tracking</i> baseado na visão	16
2.2.2.1.3. <i>Tracking</i> híbrido	18
2.2.3. Bibliotecas de desenvolvimento de aplicações de RA	18
2.2.3.1. ARToolKit	19
2.2.3.2. Vuforia	20
2.2.3.3. ARCore	21
2.2.3.4. AR Foundation	22
2.3. Áreas de Aplicação	22
2.3.1. Medicina	23
2.3.2. Jogos	23
2.3.3. Educação	24
2.3.3.1. Biologia e Química	25
2.3.3.2. Matemática	25
2.3.3.3. Programação	26
2.3.3.4. História	26
2.3.4. Arquitetura e Construção	27
2.3.4.1. Morpholio AR Sketchwalk	28

2.3.4.2. DAQRI Smart helmet	28
2.3.4.3. Aplicações de medição para Android e IOS.....	28
2.3.4.4. GAMMA AR.....	28
2.3.5. Herança Cultural	29
2.3.5.1. Trabalhos relacionados	29
Capítulo 3. Especificação da solução	32
3.1. Definição da experiência.....	33
3.2. Requisitos do Sistema	35
3.3. Arquitetura do Sistema	36
3.4. Configuração da Visita	38
Capítulo 4. Implementação	39
4.1. Ferramentas utilizadas.....	39
4.2. Modelo de dados	40
4.3. Aplicação Web	44
4.3.1. Páginas Web	46
4.4. Aplicação de Realidade Aumentada	56
4.4.1. Ficheiro de configuração	56
4.4.2. Aplicação Móvel de Realidade Aumentada.....	58
4.4.2.1. Cena de Rastreio.....	59
4.4.2.1.1. <i>Download</i> dos marcadores.....	61
4.4.2.1.2. Biblioteca de criação de marcadores em <i>runtime</i>	64
4.4.2.1.3. <i>Download</i> dos modelos 3D.....	65
4.4.2.1.4. Carregamento dos modelos 3D.....	68
4.4.2.1.5. Unzip dos modelos 3D.....	70
4.4.2.1.6. <i>Download</i> dos ficheiros multimédia.....	70
4.4.2.1.7. <i>Download</i> de imagens	74
4.4.2.1.8. <i>Download</i> de vídeos	75
4.4.2.1.9. <i>Download</i> de áudio	76
4.4.2.1.10. Leitura de marcadores.....	77
4.4.2.2. Cena de Visualização	82
4.4.2.2.1. Carregamento do modelo 3D	83
.....	84
4.4.2.2.2. Carregamento dos botões multimédia	84
4.4.2.2.3. Botão de imagens	86
4.4.2.2.4. Classe responsável pelo painel de imagens	89

4.4.2.2.5. Botão de vídeos	91
4.4.2.2.6. Classe responsável pelo painel de vídeos	93
4.4.2.2.7. Botão de áudio	97
Capítulo 5. Avaliação do Sistema	101
5.1. <i>Hardware</i> utilizado.....	101
5.2. Preparação do teste.....	101
5.3. Realização do teste	104
Capítulo 6. Conclusões	114
6.1. Sumário e discussão dos resultados	114
6.2. Trabalho futuro	115
Referências Bibliográficas.....	117

Índice de Figuras

Figura 1 - Exemplo de Sensorama	5
Figura 2 – Exemplo do Primeiro Head Mounted Display	5
Figura 3 - Funcionário da Boeing a utilizar o sistema de RA	6
Figura 4 - Sistema KARMA	7
Figura 5 - Reality Virtuality Continuum	7
Figura 6 - Sistema Studierstube.....	8
Figura 7 - Sistema Touring Machine	9
Figura 8 - Exemplo do uso do SDK ARToolKit	10
Figura 9 - Jogo ARQuake	10
Figura 10 - Sistema The Invisible Train	11
Figura 11 - Arquitetura de um sistema típico de RA	13
Figura 12 - Tipos de Tracking	16
Figura 13 - Exemplo tracking com marcadores	17
Figura 14 - Exemplo tracking sem marcadores	18
Figura 15 - Arquitetura da aplicação de Realidade Aumentada (Back-end).....	37
Figura 16 - Arquitetura da aplicação de Realidade Aumentada (Front-end).....	37
Figura 17 - Diagrama ER da Base de Dados	40
Figura 18 - Diagrama de navegação da aplicação Web.....	45
Figura 19 - Página inicial da aplicação Web	46
Figura 20 - Página de registo da aplicação Web.....	46
Figura 21 - Página principal da aplicação Web.....	47
Figura 22 - Página de criação de novo Perfil de Visitante na aplicação Web.....	48
Figura 23 - Página de listagem de perfis de visitante na aplicação Web	49
Figura 24 - Página de importação de ficheiros multimédia na aplicação Web	50
Figura 25 - Página de listagem de ficheiros multimédia na aplicação Web	50
Figura 26 - Página de criação de novos Pontos de Interesse na aplicação Web.....	51
Figura 27 - Página de listagem dos Pontos de Interesse	52
Figura 28 - Página de criação de um novo Guia de RA na aplicação Web	52
Figura 29 - Página de listagem de Guias de RA na aplicação Web.....	53
Figura 30 - Ficheiro de configuração no formato de código QR na aplicação Web	54
Figura 31 - Página de visualização de um guia específico na aplicação Web	54
Figura 32 - Exemplo de imagem vista no modal	55
Figura 33 - Exemplo de Ponto de Interesse sem qualquer ficheiro de vídeo associado	55
Figura 34 - Exemplo ficheiro de configuração	57
Figura 35 - Fluxograma da Cena de Rastreio	59
Figura 36 - Código responsável pelo download do ficheiro de configuração	60
Figura 37 - classes responsáveis pela conversão dos pares chave-valor vindos do ficheiro de configuração	60
Figura 38 - Fluxograma da função DownloadMarkers	62
Figura 39 - Código que carrega os marcadores de uma diretoria o que realiza o download dos mesmos.....	62
Figura 40 - Código responsável pelo Download dos marcadores vindos do servidor ...	63
Figura 41 - Código responsável por adicionar os marcadores à biblioteca de marcadores, criada e gerida em runtime	64
Figura 42 - Fluxograma da função AddImage.....	65

Figura 43 - Fluxograma da função DownloadModels.....	66
Figura 44 - Código que itera cada um dos pontos de interesse e realiza o download dos mesmos.....	67
Figura 45 - Código responsável por realizar o download dos modelos 3D	67
Figura 46 - Fluxograma da função DownloadZipFile	68
Figura 47 - Código responsável pelo carregamento para a tela de cada um dos modelos 3D.....	69
Figura 48 - Código responsável pelo unzip do modelo 3D	70
Figura 49 – Código responsável pelo mapeamento ou Download de cada um dos ficheiros multimédia presente no guia.....	71
Figura 50 - Fluxograma da função DownloadMedia	72
Figura 51 - Código responsável pelo download de cada um dos diferentes tipos de multimédia.....	73
Figura 52 - Fluxograma da função DownloadPoiMedia	74
Figura 53 - Código responsável pelo download dos ficheiros de imagem	75
Figura 54 - Código responsável pelo download dos ficheiros de vídeo	76
Figura 55 - Código responsável pelo download dos ficheiros de áudio	77
Figura 56 - Modos de operação da função OnTrackedImageChanged	78
Figura 57 - Código responsável por mostrar o modelo e o nome do mesmo.....	78
Figura 58 - Código responsável por mostrar ou omitir o modelo, nome e botão de detalhes	79
Figura 59 - Código responsável por levar o utilizador para uma nova cena, após carregar no botão de detalhes.....	80
Figura 60 - Código responsável por verificar se utilizador trocou de cena	81
Figura 61 - Fluxograma da classe ViewDetailsPoi.....	82
Figura 62 - Código inicial da Cena de Visualização	83
Figura 63 - Código responsável pelo carregamento do modelo na Cena de Visualização	84
Figura 64 - Código responsável pelo carregamento dinâmico dos botões de multimédia	84
Figura 65 - Código responsável por indicar os tipos de multimédia que um determinado ponto de interesse possui	85
Figura 66 - Código responsável por instanciar os botões de multimédia na tela de visualização.....	86
Figura 67 - Fluxograma da função OnClickImageButton	87
Figura 68 - Código responsável por mostrar na tela do utilizador as imagens e a respetiva descrição relativas a um ponto de interesse	88
Figura 69 - Função inicial da classe que carrega as imagens e as respetivas descrições	89
Figura 70 - Função de apoio que retorna o número total de imagens relativas a um Ponto de Interesse específico	90
Figura 71 - Código responsável por instanciar cada um dos painéis	90
Figura 72 - Código responsável por instanciar as imagens e a respetiva descrição	91
Figura 73 - Fluxograma da função OnClickVideoButton	92
Figura 74 - Código responsável por mostrar na tela do utilizador os vídeos e a respetiva descrição relativos a um ponto de interesse	93
Figura 75 - Primeira função a ser invocada mal o utilizador carrega no botão de visualização de vídeos	94

Figura 76 - Função que retorna o número total de vídeos presentes numa diretoria específica	95
Figura 77 - Código responsável pelo carregamento dos painéis de visualização dos vídeos	96
Figura 78 - Código responsável por carregar vídeo e respetiva descrição para a tela do dispositivo.....	96
Figura 79 - Fluxograma da função OnClickAudioButton	97
Figura 80 - Código responsável por mostrar na tela do utilizador o áudio relativo ao ponto de interesse em visualização	98
Figura 81 - Código responsável por obter o clip de áudio e invocar a sub-rotina de carregamento do mesmo	99
Figura 82 - Código responsável pelo carregamento do ficheiro de áudio para o clip de áudio.....	99
Figura 83 - Funções de controlo do painel de áudio	100
Figura 84 - Código responsável por trocar a cena de visualização pela cena de rastreio	100
Figura 85 - Exemplo de marcador	102
Figura 86 - Leitura código QR na aplicação de RA.....	102
Figura 87 - Loading screen na aplicação de RA	103
Figura 88 - Exemplo ficheiro de configuração retornado pela leitura do código QR... ..	104
Figura 89 - Modelo do chafariz visualizado na aplicação de RA.....	105
Figura 90 - Chafariz visto de frente na aplicação de RA	106
Figura 91 - Chafariz rodado na aplicação de RA.....	106
Figura 92 - Menu de multimédia aberto na aplicação de RA.....	107
Figura 93 - Exemplo painel de imagens na aplicação de RA	108
Figura 94 - Exemplo de informação acerca de um Ponto de Interesse na aplicação de RA	109
Figura 95 - Modelo da arca visualizado na aplicação de RA.....	109
Figura 96 - Arca vista de frente na aplicação de RA	110
Figura 97 - Menu de multimédia aberto do modelo da arca na aplicação de RA.....	111
Figura 98 - Exemplo de imagem no primeiro painel na aplicação de RA.....	112
Figura 99 - Exemplo de imagem no segundo painel na aplicação de RA	112
Figura 100 - Exemplo de vídeo no primeiro painel na aplicação de RA	113
Figura 101 - Exemplo painel de áudio na aplicação de RA.....	113

Índice de Tabelas

Tabela 1 - Ferramentas de trabalho	39
------------------------------------------	----

Capítulo 1. Introdução

1.1. Enquadramento e motivação

Ao contrário da Realidade Virtual (RV), que nos transporta para um mundo completamente virtual, a Realidade Aumentada (RA) caracteriza-se pela inserção de elementos virtuais no mundo real, em tempo real [1].

Atualmente a RA é cada vez mais um suporte inovador para a divulgação do Património Cultural. Atua como um guia para os visitantes e como um suplemento a conteúdos relativos ao património visitado, contexto histórico, etc, proporcionando assim uma experiência mais imersiva e informativa. Este tipo de experiência potencia a adesão de novos visitantes aos locais de herança cultural, pela sua característica diferenciadora.

A RA aplicada à herança cultural permite um maior aprofundamento do conhecimento das diferentes fases e épocas da história da humanidade ao mesmo tempo que preserva o ambiente existente na altura.

Como já foi referido, esta aplicação terá como caso de estudo o Convento de São Francisco, em Real, uma vez que possui um enorme valor arqueológico e patrimonial. Devido a tal facto, surgiu a necessidade de registar essa mesma riqueza.

Com este projeto de dissertação pretende-se que os visitantes do Convento sejam capazes de observar as descobertas resultantes das escavações previamente realizadas. Os modelos 3D presentes na aplicação de RA serão fornecidos pela Unidade de Arqueologia da Universidade do Minho.

Esta aplicação móvel pretende assim aumentar a interatividade dos visitantes com o local visitado através de modelos 3D fiéis aos existentes antigamente e de ficheiros multimédia. Por outro lado, uma vez que se torna numa forma dinâmica e única de realizar visitas, é de se esperar um aumento no número de visitantes ao complexo monástico.

1.1. Objetivo da dissertação

Esta dissertação tem como objetivo principal o desenvolvimento um sistema de criação e gestão de guias de realidade aumentada para dispositivos móveis, com a finalidade de servir como um elemento de suporte ao visitante, capaz de complementar a realidade observável em contexto de visita a um espaço arquitetónico de interesse cultural.

O sistema a desenvolver será composto por dois componentes. Uma aplicação web que permita definir e configurar guias de realidade aumentada e uma aplicação móvel que permita carregar e visualizar o(s) guias(s) relativos ao local que o utilizador está a visitar.

1.2. Estrutura da dissertação

Esta dissertação encontra-se dividida em seis distintos capítulos.

No Capítulo 1, intitulado “Introdução” faz-se um enquadramento de todo o trabalho efetuado no decorrer desta dissertação, explicitando os objetivos fundamentais deste.

No Capítulo 2, “Estado da Arte”, faz-se um levantamento teórico do que foi descoberto relativamente à tecnologia de RA. São também expostos alguns trabalhos que aplicam esta tecnologia na área de conservação e divulgação de Património Arquitetónico.

No Capítulo 3, denominado “Especificação da solução”, é exposta a experiência de RA, bem como os requisitos do sistema, respetiva arquitetura e configuração das visitas.

No Capítulo 4, “Implementação” é descrito o processo de desenvolvimento tanto da aplicação web como da aplicação móvel. Neste capítulo é feito também um levantamento sobre as ferramentas utilizadas. Encontra-se dividido em três subcapítulos principais que dizem respeito ao ficheiro de configuração, aplicação Web e aplicação móvel.

No Capítulo 5, “Avaliação do Sistema”, são apresentados todos os testes práticos realizados sobre a aplicação de RA no Convento de São Francisco.

No Capítulo 6, “Conclusões” é apresentado um resumo e discussão dos resultados, assim como propostas para trabalhos futuros. Ainda neste capítulo é feito um levantamento dos obstáculos encontrados em termos de programação ao longo do trabalho.

Capítulo 2. Estado da Arte

A Realidade Aumentada (RA) caracteriza-se por ser uma tecnologia na qual o mundo real é aumentado por elementos gerados por computador, ou seja, objetos de caráter virtual são inseridos no mundo real, em tempo real [1]. Esta tecnologia começou a ter um desenvolvimento mais considerável no início da década de 90.

Ao contrário da Realidade Aumentada, a Realidade Virtual (RV) transporta o utilizador para um mundo completamente gerado por computador. Segundo *J. Zheng et al.*, a RV define-se como sendo uma interface avançada que simula um ambiente real, ou seja, o utilizador é levado para um ambiente completamente simulado em computador [2].

Assim sendo, pode dizer-se que a Realidade Aumentada se encontra entre o mundo virtual e o mundo real. Devido a um aumento das capacidades gráficas e de processamento dos dispositivos móveis ao longo do tempo, aliados à vulgarização dos mesmos, potenciou-se o uso da RA em qualquer espaço. Tal tecnologia tem vindo a ser aplicada em diversas áreas distintas, como jogos, medicina, educação, engenharia, indústria, arquitetura, etc [3].

2.1. Resenha Histórica

A primeira aparição do termo Realidade Aumentada surge por volta dos anos 50 do século passado quando *Morton Heilig*, um cinematografo, considerado o pai da realidade virtual, pensou no cinema como uma atividade capaz de atrair o espetador de uma forma mais imersiva através da utilização dos cinco sentidos de uma maneira efetiva. A partir desta premissa, concebeu-se, em 1962, o que seria denominado por *Sensorama* (ver Figura 1) uma máquina do tipo *arcade* (máquina que requer algum tipo

de crédito por partida) construída para estimular os diferentes sentidos [1]. Várias curtas-metragens foram desenvolvidas para este revolucionário aparelho.



Figura 1 - Exemplo de Sensorama

Sensorama de Morton Heilig, 1962, retirada a 10 de fevereiro de 2020 em https://www.repubblica.it/tecnologia/2021/08/28/news/il_sensorama_la_macchina_per_realta_virtuale_che_non_venne_capita-315540396/

Uns anos mais tarde, em 1966, *Ivan Sutherland* concebeu primeiro HMD (*Head Mounted Display*), podendo ser visto por intermédio da Figura 2. Em 1968 *Sutherland* foi a primeira pessoa a criar um sistema de Realidade Aumentada funcional, através da utilização do HMD. Devido ao seu peso este era suspenso no teto, tendo sido apelidado de “Espada de Dâmocles” [4].



Figura 2 – Exemplo do Primeiro Head Mounted Display

Espada de Dâmocles de Ivan Sutherland, 1968, retirada a 10 de fevereiro de 2020 em <https://www.informit.com/articles/article.aspx?p=2516729&seqNum=2>

Em 1975, *Myron Krueger* criou o *Videoplace*, uma sala que permite aos utilizadores a interação com objetos virtuais [5].

Uns anos mais tarde, em 1992, *Tom Caudell* e o seu colega de trabalho, *David Mizell*, ambos trabalhadores da *Boeing*, cunham o termo Realidade Aumentada. Na altura, o sistema da Boeing consistia na utilização de grandes placas de contraplacado com as instruções da cablagem por aeronave construída. *Caudell*, juntamente com o seu colega *Mizell* foram convidados a substituir o sistema então existente por um outro montado na cabeça (HMD) dos trabalhadores responsáveis pela construção dos aviões. As informações da cablagem eram projetadas nas placas de contraplacado em forma de Modelo de linhas (ver Figura 3). Tal inovação permitiu que as placas pudessem ser polivalentes e reutilizáveis, visto que a reconfiguração de cada uma passou de manual para automática, personalizada de uma forma rápida e eficaz com recurso a um sistema computacional [4][6]. Com isto, *Tom* e *Mizell* deram início à utilização da realidade aumentada num ambiente industrial.



Figura 3 - Funcionário da Boeing a utilizar o sistema de RA

fotografia de um trabalhador da Boeing a utilizar RA, n.d, retirada a 10 fevereiro 2020 de <https://medium.com/scape-technologies/augmented-reality-from-research-to-ubiquity-in-30-years-1570a8653d63>

No ano seguinte, em 1993, *Steven Feiner*, *Blair MacIntyre* e *Doree Seligmann* apresentaram em [7] um protótipo de um sistema AR denominado KARMA (*Knowledge-based Augmented Reality for Maintenance Assistance*). Este sistema, com recurso ao HMD, explica ao consumidor final de uma forma simples a manutenção de uma impressora laser sem a necessidade da constante consulta do manual de instruções como se pode observar na Figura 4 [7].



Figura 4 - Sistema KARMA

Sistema KARMA, n.d., retirada a 10 fevereiro 2020 de <https://medium.com/scape-technologies/augmented-reality-from-research-to-ubiquity-in-30-years-1570a8653d63>

Em 1994 *Paul Milgram et al.* define o Contínua da Virtualidade, apresentado na Figura 5 através da descrição de uma taxonomia que determina como é que a realidade aumentada e a realidade virtual se encontram associadas entre si [8].

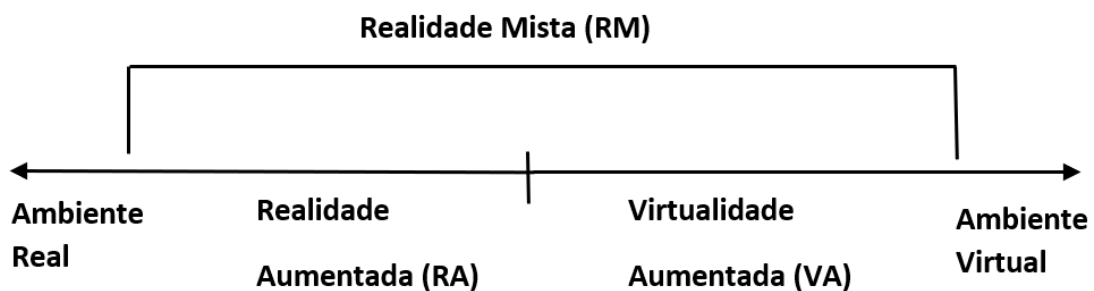


Figura 5 - Reality Virtuality Continuum

A Figura 5 apresenta a diferença entre Ambiente Virtual, onde todos os objetos são virtuais, Virtualidade Aumentada, onde o ambiente virtual é aumentado por objetos reais, Ambiente Real, objetos do mundo real e Realidade Aumentada, onde objetos virtuais são colocados no mundo real [1][9].

A Realidade Aumentada permite-nos visualizar imagens geradas virtualmente por computador no mundo real, em tempo real. Na realidade aumentada, o utilizador interage com o mundo real de uma forma natural. A grande diferença entre esta tecnologia e a Realidade Virtual assenta no facto de que na última (RV) o utilizador é completamente imerso num mundo artificial.

Em 1996 foi criado o primeiro sistema colaborativo de RA, produzido por *Schmalstieg et al.*, conhecido por *Studierstube*. Tal sistema permitiu que diferentes colaboradores manipulassem o mesmo objeto num mesmo espaço partilhado (ver Figura 6). Para que tal fosse possível, recorreu-se ao uso do HMD, anteriormente mencionado [10].

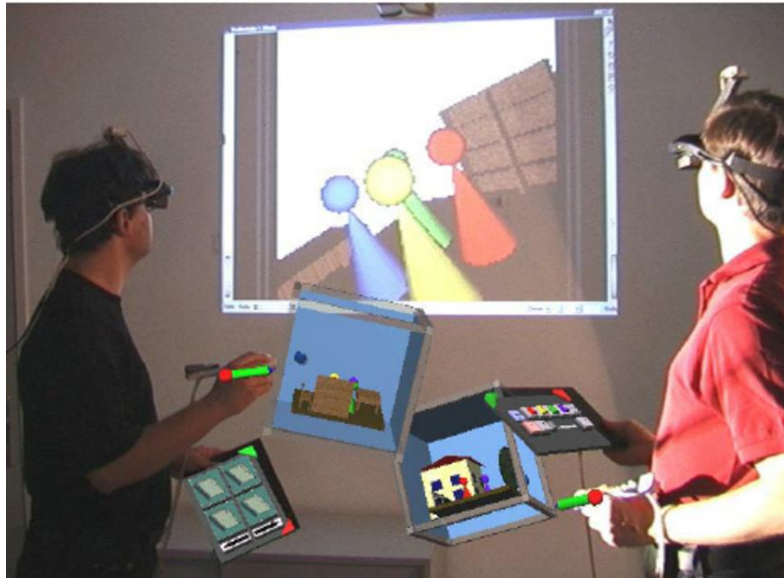


Figura 6 - Sistema Studierstube

Sistema Studierstube, n.d., retirada em 12 de fevereiro de 2020 em https://www.researchgate.net/figure/Studierstube-augmented-reality-system-SFH-01_fig3_30508785

No ano seguinte, *Mann* desenvolveu o primeiro sistema ao ar livre tendo por base o GPS. Este fornecia assistência de navegação para pessoas invisuais com recurso ao áudio [11].

Nesse mesmo ano, *Feiner et al.* criam um protótipo de MARS (Mobile Augmented Reality System), denominado *Touring Machine*, o primeiro sistema 3D de RA no mundo, capaz de detalhar informação acerca de edifícios e pontos de interesse que o utilizador vê em tempo real [12]. Com o auxílio a um monitor portátil o utilizador conseguia obter ainda mais informação sobre o ponto em questão (ver Figura 7). Tal sistema provou que o uso da realidade aumentada no dia a dia das pessoas seria uma aliada poderosa devido a uma potencial melhoria na forma de ver o mundo [4].



Figura 7 - Sistema Touring Machine

Sistema Touring Machine, n.d., retirada em 12 de fevereiro de 2020 em <https://emag.directindustry.com/2020/06/15/a-brief-history-of-augmented-reality/>

Ronald Azuma et al. descreve, no seu *survey paper*, que a realidade aumentada possui três características distintas e que apenas estas chegam para definir propriamente o que é de facto a Realidade Aumentada [13]:

- Realidade aumentada irá combinar objetos reais e virtuais num ambiente real;
- Interativa em tempo real;
- Objetos reais e virtuais registados em 3D;

Em 1999 foi desenvolvido o primeiro *Software Development Kit* (SDK) para realidade aumentada, o ARToolKit, por Hirokazu Kato e Mark Billinghurst no HIT lab. Este ajudava na construção de sistemas de RA de uma forma mais rápida (ver Figura 8) [14].

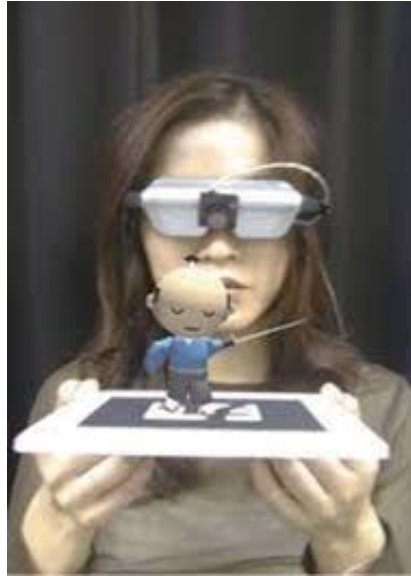


Figura 8 - Exemplo do uso do SDK ARToolkit

Exemplo SDK ARToolkit, n.d., retirada a 13 fevereiro 2020, em <http://www.hitl.washington.edu/artoolkit.html>

Uns anos mais tarde, em 2000, *Bruce Thomas et al.* desenvolvem o primeiro jogo de realidade aumentada ao ar livre, o *ARQuake*. Neste jogo os utilizadores do mundo real eram misturados com personagens fictícias, num ambiente de combate imaginário (ver Figura 9) [15].



Figura 9 - Jogo ARQuake

Jogo ARQuake, n.d., retirada a 12 de fevereiro 2020, em <https://www.timetoast.com/timelines/realidad-aumentada-eb6b9b3b-d0a6-43fc-95c1-f4882cd3ebe8>

Em 2004 aparece a primeira aplicação de realidade aumentada para telemóvel, *The Invisible Train*. Esta é uma aplicação de multiutilizadores na qual cada um deles controla comboios virtuais, aumentados numa pista de madeira real. Estes comboios virtuais apenas são visíveis aos jogadores através do respetivo PDA, uma vez que não existem no mundo real (ver Figura 10) [16].

Sendo assim, os jogadores interagem com o jogo, podendo mudar trilhos, bem como ajustar a velocidade do seu comboio. A sincronização entre todos os participantes é realizada através de rede *wireless*. O objetivo final do jogo é evitar que os comboios virtuais colidam uns com os outros.

O sucesso do jogo ilustra bem as vantagens da já falada *framework*, *Studierstube*.



Figura 10 - Sistema The Invisible Train

Sistema The Invisible Train, n.d., retirada a 13 de fevereiro 2020, em https://www.google.com/search?q=The+Invisible+Train+augmented+reality&rlz=1C1FCXM_pt-PTPT978PT978&sxsrf=AJOqlzWxm5w6BITcCsVvIYLOGibADJiwXw:1676712207940&source=lnms&tbm=isch&sa=X&ved=2ahUKEwj5v63c3579AhUEzaQKHxv9CEYQ_AUoAXoECAEQAw&biw=1536&bih=714&dpr=1.25#imgrc=M5LWlk2KYcPITM

Nos seguintes anos mais e mais tecnologias de realidade aumentada vieram a ser desenvolvidas, especialmente na vertente móvel, uma vez que com o passar dos anos e a exponencial evolução da tecnologia, os telemóveis tiveram uma enorme evolução tanto a nível de *software* como de *hardware*, permitindo aos investigadores novas soluções para problemas comuns do quotidiano.

Por outro lado, a realidade aumentada tem vindo a ser utilizada em diversas áreas de estudo, sendo utilizado em grande escala em organizações devido a uma melhoria de serviço que provém da mesma.

2.2. Sistemas de RA

Os sistemas de RA apresentam-se como sendo uma tecnologia relativamente moderna, em constante desenvolvimento. Assim sendo, estes têm vindo a ser classificados tendo por base o sistema de apresentação utilizado.

De uma maneira geral, podemos definir um sistema de RA como sendo a união de uma câmara, um *software* de RA e um computador. Como sabemos, existe uma imensidão de aplicações e dispositivos, o que torna o alcance de um modelo concreto uma tarefa mais complexa. De forma a facilitar a compreensão de um sistema, podemos e devemos estruturá-lo e classificá-lo em diferentes subsistemas.

2.1.1. Arquitetura de um Sistema Típico de RA

Vallino faz a distinção de três tipos de sistemas conforme o dispositivo de apresentação usado [17]:

- Baseados num monitor
- Baseados num HMD com câmara de vídeo agrupada
- Baseados em HMD de lentes semireflexivas

Mencionando ainda *Vallino*, este faz a representação de um sistema típico de realidade aumentada, no qual distingue cinco subsistemas:

1. Sistema de captação de imagem (SCI)
2. Sistema de seguimento de localização e orientação do SCI (SSSCI)
3. Sistema gráfico para geração de objetos virtuais (SG)
4. Sistema Misturador de Imagem (SMI)
5. Sistema de apresentação (SA)

José Braz e João Madeiras descrevem, no seu trabalho TARCAST, a necessidade de expandir a arquitetura acima referida. Para tal, propuseram o acréscimo de três novos subsistemas aos anteriormente mencionados [18]. São eles:

1. Sistema Real de Manipulação de Objetos (tanto pode manipular objetos reais como virtuais)
2. Sistema de seguimento do Sistema Real de Manipulação de Objetos (SRMO)
3. Sistema de seguimento de objetos reais que se movimentam no ambiente aumentado

A arquitetura típica de um sistema de RA é conseguida através da adição do Sistema Real de Manipulação de Objetos ao sistema proposto por Vallino e do agrupamento do Sistema de seguimento do SRMO com o Sistema de seguimento de objetos reais num único sistema global (ver Figura 11).

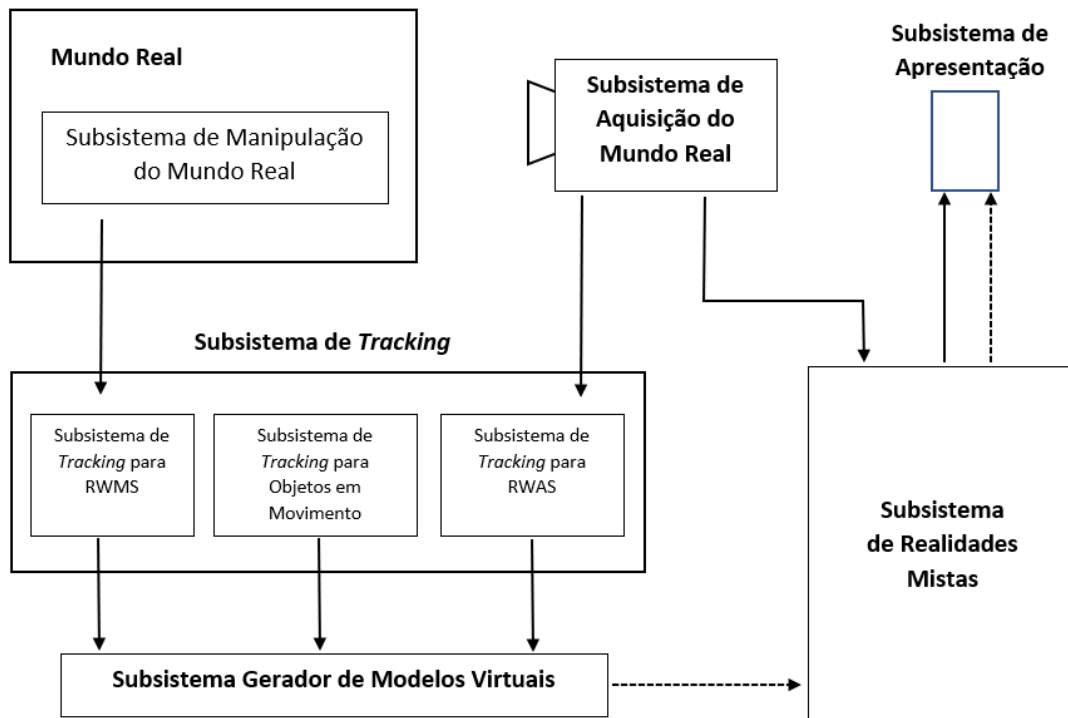


Figura 11 - Arquitetura de um sistema típico de RA

Sabemos então que um sistema de RA pode ser fragmentado em 6 diferentes subsistemas, segundo José Braz e João Madeiras [18].

2.2.1.1. Subsistema de Aquisição de Imagem

Este subsistema possui como função a obtenção de sons e imagens provenientes do mundo real, tipicamente através da utilização de uma câmara.

2.2.1.2. Subsistema Gerador de Objetos Virtuais

Subsistema encarregue da construção da parte gráfica. A sua caracterização é realizada conforme os sentidos que permite estimular. A partir deste é possível classificar o modelo gráfico relativamente ao seu grau de realismo, dinamismo, interatividade com o utilizador e tecnologia utilizada.

2.2.1.3. Subsistema de Realidades Mistas

Neste subsistema é efetuada a junção entre o real e o virtual. Tal combinação possibilita que a tecnologia de mistura utilizada seja identificada, sabendo que pode ser caracterizada segundo duas tecnologias distintas, ótica ou eletrónica.

2.2.1.4. Subsistema de Caracterização do Mundo Real

Este subsistema tem em conta que as características do sistema de manipulação são determinadas pela tecnologia vinculada ao mesmo e ainda que essas mesmas características diferem de acordo com o tipo de tecnologia utilizado. Assim sendo, este subsistema tem por obrigação gerir a interatividade tanto em ambiente real, como em ambiente virtual.

2.2.1.5. Subsistema de *Tracking*

O subsistema de *tracking* apresenta-se como sendo o mais importante, mas ao mesmo tempo o mais complexo de todos. Uma boa leitura da localização do utilizador e dos objetos virtuais que o rodeiam com tamanha precisão é um dos maiores obstáculos que dificulta a construção de sistemas de RA eficazes. Podemos classificar este subsistema de acordo com o tipo de *tracking* (baseado em sensores, baseado na visão ou híbrido), estudados com maior detalhe no próximo subcapítulo, e o grau de complexidade existente.

2.2.1.6. Subsistema de Apresentação

O subsistema de apresentação possui diversas tecnologias de visualização da experiência relativamente aos dispositivos possíveis de serem utilizados [8]. Possui também a tarefa de identificar o ponto de vista do utilizador, sabendo que este se pode encontrar em diferentes níveis de imersividade. Em termos de escala, e de acordo com *Milgram et Al.*, podemos dividir os sistemas de visualização entre os que possibilitam visualizar o mundo numa escala de 1:1 e os que não o possibilitam.

2.2.2. Tracking

No seguinte capítulo irão ser discutidos com maior detalhe os diferentes tipos de *tracking* presentes no subsistema com o mesmo nome, possíveis de serem utilizados num sistema de RA. Através do subcapítulo acima, sabemos que o subsistema de *tracking* está encarregue da leitura da localização do utilizador e dos objetos virtuais que o rodeiam. Sabendo que a correta localização do utilizador num dado instante de tempo é um dos elementos fundamentais para que um sistema de RA funcione corretamente, o subsistema de *tracking* apresenta-se como sendo o de maior grau de complexidade, possuindo ao mesmo tempo o maior número de obstáculos, entre os seis subsistemas estudados no subcapítulo anterior. De forma a contornar a grande dificuldade descrita anteriormente, localização precisa e longo raio de ação, novos tipos de *tracking* tiveram de ser inventados ou reinventados, passando assim a ser mais complexos, necessitar da junção de diversas tecnologias de forma a mitigar as desvantagens de cada uma e tirando proveito das suas vantagens. Desde o final dos anos 80 e inícios dos anos 90, vários autores têm vindo a escrever e a propor novas técnicas de *tracking*, como por exemplo [19] [20].

Em suma, o elevado grau de complexidade no subsistema de *tracking* pode ser um impedimento para a correta caracterização do mesmo, visto que ainda não existe um total consenso entre qual ou quais tecnologias utilizar e quais as melhores metodologias.

2.2.2.1. Tipos de *tracking*

O *tracking* entre objetos gerados em computador e objetos do mundo real é um desafio crucial para o desenvolvimento de uma aplicação de realidade aumentada.

Quando um utilizador muda a sua posição ou ponto de vista, os objetos virtuais devem permanecer alinhados com a posição e orientação dos objetos do mundo real. Sendo assim, o alinhamento entre objetos virtuais e reais depende de um *tracking* preciso.

Zhou *et al.* categorizam, em 2008, três tipos de *tracking*: baseado em sensores, baseado na visão e híbrido (ver Figura 12) [21].

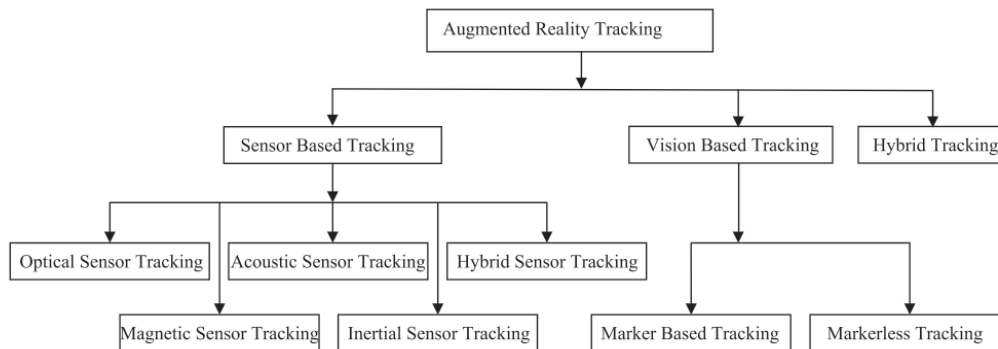


Figura 12 - Tipos de Tracking

2.2.2.1.1. Tracking baseado em sensores

Sensores ativos são utilizados no *tracking* baseado em sensores, posteriormente aplicados no rastreamento da posição do movimento da câmara, ou seja, este tipo de *tracking* relaciona-se com dispositivos que emitam qualquer tipo de sinal para se proceder à sua localização. Os sensores podem ser óticos, acústicos, inerciais, magnéticos ou ultrassônicos.

Cada sensor possui vantagens e desvantagens e a seleção de um ou outro vai depender de diferentes fatores, tais como a calibração, precisão, custo, temperatura, ambiente, pressão, resolução e raio de ação.

2.2.2.1.2. Tracking baseado na visão

Técnicas de *tracking* baseadas na visão utilizam informações de imagem para rastrear a posição e orientação da câmara. Este tipo de *tracking* é o mais investigado na área de realidade aumentada [21].

Existem dois métodos distintos:

- **Com recurso a marcadores**

Neste tipo de *tracking* marcadores artificiais são colocados no meio ambiente para serem posteriormente consumidos pela aplicação de realidade aumentada (ver Figura 13).

A posição e orientação dos objetos virtuais na cena depende assim da posição e orientação do marcador a si associado.



Figura 13 - Exemplo tracking com marcadores

Exemplo tracking com marcadores, n.d., retirada a 16 fevereiro de 2020, em <https://www.areteproject.eu/newsevents/body,493507,en.html>

- **Sem recurso a marcadores**

Caracteriza-se por ser um método mais complexo que o referido anteriormente, sendo o método mais utilizado e pesquisado nos dias correntes.

O *tracking* é realizado através das características visuais interpretadas pela câmara, ou seja, as informações presentes no ambiente real são recolhidas dinamicamente pela câmara do dispositivo (ver Figura 14).

Assim sendo não é necessária a inserção de um objeto artificial no ambiente real de forma a se poder visualizar um certo objeto virtual.



Figura 14 - Exemplo tracking sem marcadores

Exemplo tracking sem marcadores, n.d., retirada a 16 fevereiro 2020, em <https://mobile-ar.reality.news/news/usens-introduces-markerless-tracking-for-mobile-apps-0178948/>

2.2.2.1.3. Tracking híbrido

Os dois tipos de *tracking* acima mencionados possuem as suas limitações. Devido a essas mesmas limitações e uma vez que uma solução de *tracking* robusta não é possível para algumas aplicações de realidade aumentada, numa tentativa de ultrapassar essas limitações, desenvolveram-se métodos híbridos, que combinam as vantagens de cada um dos dois tipos acima mencionados.

Em suma, técnicas de *tracking* híbrido caracterizam-se por serem a combinação entre *tracking* baseado em sensores e *tracking* baseado na visão e são a forma mais promissora de lidar com soluções *indoor* e *outdoor* aplicadas à realidade aumentada.

2.2.3. Bibliotecas de desenvolvimento de aplicações de RA

Antes mesmo de falarmos na evolução dos SDK de RA e no contributo de cada um, é necessário ter em mente o conceito de SDK. Assim sendo, um SDK nada mais é do que uma coleção de diversas ferramentas oferecidas na maior parte das vezes pelo próprio fornecedor de hardware, um determinado sistema operacional (SO) ou uma linguagem de programação em particular.

Através dos SDK, os desenvolvedores de software, são capazes de criar programas ou aplicações para uma determinada plataforma, sistema operacional ou linguagem de programação.

Em suma, quando se faz uso de um SDK, a concretização de uma determinada aplicação torna-se mais simples de ser executada num menor intervalo de tempo, ao mesmo tempo que existe uma melhoria significativa no tratamento de erros devido às ferramentas fornecidas. No caso da RA, o uso de SDKs específicos não foge à regra. Muitos dos componentes dentro da aplicação de RA tornam-se mais acessíveis devido ao uso de SDKs específicos, como por exemplo o reconhecimento, *tracking* e a renderização de conteúdos.

Ao longo do capítulo irão ser referenciados diferentes SDK de RA que contribuíram ao longo dos anos para que aplicações de RA pudessem ser mais interativas e precisas.

Os SDKs de RA podem ser divididos em três grandes categorias: (1) Navegadores de geolocalização, (2) baseados em marcadores e (3) rastreamento de características naturais [9].

Os Navegadores de geolocalização de RA possibilitam que os utilizadores concebam aplicações de RA de geolocalização, através da utilização do GPS e do IMU, totalmente disponíveis na maior parte dos telemóveis atuais. O SDKs baseados em marcadores servem-se de marcadores e imagens características de forma a poder criar a experiência de realidade aumentada. Por último, os SDK de Rastreamento de Características Naturais baseiam-se nas propriedades existentes no mundo natural para a realização do aumento dos objetos, através do *tracking* de imagens.

Nos próximos subcapítulos irão ser definidos os SDK mais comuns usados na realidade aumentada de forma a facilitar o desenvolvimento de aplicações para esta tecnologia.

2.2.3.1. ARToolKit

Como já foi mencionado, o primeiro SDK de realidade aumentada surgiu em 1999, o ARToolKit por Kato e Billinghurst [10]. Este é uma biblioteca de *tracking* baseada em marcadores. Tanto a localização como a posição do objeto virtual são definidas de acordo com a respetiva posição e localização do marcador, ou seja, o objeto está intrinsecamente conectado ao marcador a si associado.

Um ano mais tarde, em 2000, foi lançado como um *software open-source*, passível de ser utilizado por qualquer pessoa, sem qualquer tipo de custo associado. Devido ao facto de ser pioneiro na matéria, tornou-se numa das bibliotecas de RA mais utilizadas a nível mundial, tendo facilitado a construção de aplicações de RA.

Ao longo dos anos, diversas *frameworks* de RA foram desenvolvidas tendo por base o ARToolKit, tais como SLARToolKit, FLARToolKit e FLARManager [22].

2.2.3.2. Vuforia

Atualmente o Vuforia apresenta-se como sendo uma das bibliotecas de RA e RM mais utilizadas à escala mundial, tendo sido desenvolvido pela empresa Qualcomm, muito devido ao facto de ser multiplataforma, dando suporte ao desenvolvimento de aplicações nativas para android, IOS e Windows. Por outro lado, permite também o desenvolvimento de aplicações de RA no Unity, visto que são facilmente portáveis para as plataformas já mencionadas.

O SDK faz uso de uma técnica eficiente e estável de reconhecimento de imagem baseada em técnicas de visão por computador de forma a reconhecer marcadores e modelos 3D, em tempo real. Simultaneamente, fornece diversas funcionalidades que permitem que programadores possam utilizar diferentes técnicas outrora limitadas [9].

Uma aplicação de AR baseada no Vuforia possui, na maior parte das vezes, os seguintes componentes:

- **Câmara** - responsável por capturar imagens e passar o conteúdo para o *tracker*
- **Conversor de imagens** - converte a imagem tirada pela câmara para um formato que possa ser posteriormente renderizado
- **Tracker** - possui a possibilidade de carregar diferentes marcadores em simultâneo
- **Código Aplicacional** - quando um novo alvo é detetado pela câmara do dispositivo é realizada uma *query* sobre ele resultando numa atualização da lógica da aplicação com os novos dados de entrada e na renderização dos gráficos

- **Base de Dados do Dispositivo** – armazena os marcadores no próprio dispositivo, em memória
- **Base de Dados na *cloud*** – armazena o alvo na *cloud*

Este SDK suporta inúmeros tipos de *targets*, tanto 2D como 3D, podendo também ser configurado para conseguir ler múltiplos marcadores ou até mesmo imagens sem recurso a marcadores.

Por último, o Vuforia fornece diferentes API's, nomeadamente Java, C++, Objective C++ e as linguagens .NET. Aplicações de realidade aumentada concebidas por meio do Vuforia são compatíveis com uma vasta gama de dispositivos pessoais móveis [23].

2.2.3.3. ARCore

O ARCore é uma *framework* e SDK de RA desenvolvido pela própria google no início de 2018, com o propósito de criar aplicações de realidade aumentada de uma forma gratuita focadas somente em Android. Atualmente, é possível criar experiências de RA tanto para Android como para IOS devido às várias API's oferecidas pelo ARCore. Também é possível utilizar a plataforma com os populares *game engine* Unity e Unreal.

Este oferece três funcionalidades principais que o distinguem dos restantes SDK, permitindo a construção de projetos de realidade aumentada robustos[24]:

- **Rastreamento de movimento** – na realidade aumentada é essencial que os objetos 3D pareçam realistas quando vistos de todos os ângulos possíveis. Através do ARCore é possível garantir tal funcionalidade por meio do alinhamento da câmara virtual 3D com a própria câmara do dispositivo.
- **Compreensão ambiental** – o ARCore é capaz de detetar planos e pontos característicos para que seja possível a colocação de objetos virtuais em superfícies planas e reais, como por exemplo em mesas, chão ou paredes.
- **Iluminação** – Com recurso à câmara do dispositivo pessoal, o ARCore é capaz de detetar posições de iluminação no mundo real. Logo os objetos virtuais podem ser iluminados da mesma forma que os objetos reais, de forma a adicionar a sensação de realismo à cena.

2.2.3.4. AR Foundation

Para o desenvolvimento da parte prática da dissertação foi pedido que se usasse a nova *framework* criada especificamente para Unity, o AR Foundation que nada mais é do que um conjunto de funções e APIs focadas para dispositivos que suportem uma série de conceitos aplicados na realidade aumentada. Esta permite que se trabalhe com SDKs de realidade aumentada em multiplataformas, nomeadamente android e IOS, dentro do próprio Unity. No entanto possui a particularidade de não implementar qualquer funcionalidade de AR por si só, apesar de apresentar uma interface para os desenvolvedores do Unity utilizarem, caso queiram. Consequentemente, de modo a ser possível a utilização do AR Foundation num dispositivo móvel é necessário utilizar outros pacotes para as plataformas alvo suportadas oficialmente pelo Unity, como por exemplo o ARCore (anteriormente falado), ARKit, Magic Leap e Windows [25].

Assim sendo, para este projeto em concreto foi utilizado o AR Foundation em junção com o ARCore, visto tratar-se de uma aplicação somente focada para o sistema operativo android. Em acréscimo o AR Foundation utiliza o ARCore, o que o torna numa ferramenta extremamente poderosa, uma vez que possui tanto as suas próprias funções como as do ARCore.

2.3. Áreas de Aplicação

Nos últimos anos a tecnologia de realidade aumentada tem vindo a ter um crescimento sem precedentes tornando-se num dos principais motores da economia tecnológica. Muito deste aumento exponencial deve-se ao uso que os grandes líderes do mercado lhe dão, nomeadamente Microsoft, Apple, Facebook, Google e Amazon, vulgarmente conhecidos como *Big Five* [26].

Aplicações de RA, *headsets* e óculos inteligentes prometem acrescentar valor à maior parte da indústria virtual – desde a indústria do retalho à indústria farmacêutica. Cada vez mais a realidade aumentada é utilizada para tentar resolver alguns dos maiores problemas impostos à sociedade [27].

De entre as várias áreas onde a realidade aumentada pode ser utilizada de modo a facilitar a vida quotidiana da população em geral, podemos destacar sete delas: jogos, medicina, educação, arquitetura e cultura.

2.3.1. Medicina

No ramo da medicina, a RA destaca-se no sentido de permitir unir imagens geradas virtualmente com o mundo real, incluindo pessoas reais e objetos reais. Em termos práticos tal tecnologia possibilita que médicos, cirurgiões e/ou enfermeiros consigam visualizar partes do corpo humano, ocultas a olho nu, como por exemplos veias de um braço ou perna, ossos partidos, tumores, registos de saúde e até mesmo raio x. A diferença entre este tipo de abordagem em detrimento da convencional assenta no facto dos profissionais de saúde não necessitarem de desviar a sua atenção dos pacientes para poderem olhar para outro ecrã, visto disporem de toda a informação à frente dos olhos [28].

Por outro lado, esta tecnologia pode e deve auxiliar o cidadão comum em termos de aplicações para telemóvel destinadas a mostrar onde se encontram os desfibrilhadores mais próximos, por exemplo. Uma aplicação prática que faz exatamente o descrito denomina-se por AED4EU (*Automated External Defibrillators*) e permite que utilizadores desta adicionem locais nas mediações de AED's de forma a ser possível que qualquer pessoa saiba onde pode encontrar tais dispositivos [29].

2.3.2. Jogos

Quando se ouve falar em realidade aumentada aplicada à indústria dos jogos eletrónicos, o primeiro nome que vem à cabeça é o famoso "Pokemon Go", mesmo que nunca se tenha jogado ou visto, existente desde 2016. De facto, o jogo tornou-se tão popular que é considerado por muitos o pai das aplicações de RA orientadas para jogos. Existem inúmeras reportagens, debates, artigos, etc. sobre este, um dos maiores fenómenos do mundo *gaming* da cultura atual. Tal jogo permite a jogadores de todo o mundo interagirem com os famosos Pokémons, virtualmente aumentados no mundo real através de um ecrã de telemóvel. Para que seja possível a utilização da aplicação, o dispositivo móvel tem de estar equipado com câmara, giroscópio, relógio e GPS para que seja possível obter um ambiente de realidade aumentada baseado na localização em tempo real do jogador [30].

Ao longo dos últimos anos, a indústria de jogos eletrónicos tem vindo a saturar-se devido à falta de algo inovador, o que leva a uma falta de interesse por parte de

possíveis jogadores a investirem nos jogos. Assim sendo, a AR está a ser atualizada nesta área para tentar envolver os jogadores de todo o mundo, elevando os jogos para outro patamar, através dos seguintes pontos [31]:

- **Renascimento do 3D** – com o crescente aumento de videojogos de realidade aumentada, mais pessoas podem desfrutar das experiências reais e imersivas proporcionadas pelos gráficos 3D, que são o suporte dos jogos. Através dos efeitos 3D, os jogadores são presenteados com uma jogabilidade cativante e envolvente.
- **Aumentar o apelo dos jogos** – no desenvolvimento de jogos em RA também se faz uso da geolocalização e do *tracking* 3D para dar aos jogos um lado mais criativo inovativo. Esta altera a perspetiva dos utilizadores relativamente aos jogos através de conteúdo animado em 3D, disponibilizando jogos com maior potencial do que a tecnologia convencional.
- **Substituição de sistemas de jogos convencionais** – nos dias correntes a maior parte dos jogos encontram-se apenas presentes para smartphones. No entanto, novos *gadgets* estão a ser estudados e lançados para o mercado, como por exemplo os *headsets*. Com o constante e rápido avanço da tecnologia, cada vez mais utilizadores poderão vivenciar a delicada união entre o mundo virtual e o mundo real.

2.3.3. Educação

No ramo da educação a realidade aumentada promete mudar o paradigma de que como as aulas são lecionadas fazendo com que esta recente tecnologia esteja a aumentar gradualmente de popularidade em todas as escolas a nível mundial. Permitirá aos estudantes aprender de uma maneira mais interativa [32]. Em acréscimo, os educadores serão capazes de melhorar o real aproveitamento dos alunos através de um aumento de envolvimento, potenciando aprendizagem de competências e conhecimentos tais como a colaboração e comunicação com outros alunos e a resolução de problemas.

Assim sendo, a RA aplicada à educação oferece aos estudantes a possibilidade de aprofundarem os seus conhecimentos em diversas áreas do saber, nomeadamente [33]:

- Trabalhar com números
- Jogos
- Criação de conteúdos
- Leitura
- Cenários e ambientes do mundo real
- Conceitos espaciais

Ao longo dos últimos anos, diversas aplicações de RA foram criadas com o principal objetivo de auxiliar os professores no melhoramento do real aproveitamento dos estudantes, bem como na ajuda com a resolução de problemas.

De seguida, irá ser feito um levantamento de diversas aplicações e ferramentas de RA atualmente existentes nas diferentes áreas do saber [34].

2.3.3.1. Biologia e Química

A RA aplicada no ramo das ciências permite aos professores ensinar ciência de uma forma mais envolvente, recorrendo a aulas interativas, uma vez que podem combinar objetos de AR, animações e vídeos de modo a ajudar os alunos com as suas investigações científicas.

Uma das muitas aplicações existentes denomina-se Chem101 e possui como principal objetivo o de assistir os alunos na compreensão de compostos complexos, tais como óxidos e ácidos. Recorrendo a marcadores desenhados para o efeito, os estudantes são capazes de modificar estruturas moleculares, podendo até criar novas substâncias [35].

2.3.3.2. Matemática

A matemática caracteriza-se como uma das disciplinas onde os alunos sentem mais dificuldade, criando-se uma certa aversão à mesma. De forma a tentar incentivar os alunos no estudo desta, existem diversas ferramentas de RA que ajudam os docentes a criar temáticas de matemática educativas e envolventes, que consigam aguçar a curiosidade e o pensamento crítico dos alunos.

Um dos muitos exemplos de aplicativos atualmente existentes no auxílio a esta matéria é a Photomath que, através do uso da câmara do dispositivo, possibilita aos

alunos a digitalização de um problema de matemática [36]. Após lido, é mostrada a solução passo a passo, desde o início até ao final com o auxílio de animações. A RA aplicada à matemática também pode ser empregue num dos ramos mais antigos desta, a geometria. Nestas aplicações, os alunos são capazes de entender melhor conceitos matemáticos complexos por meio da visualização de modelos 3D interativos. Por exemplo, o Merge Cube proporciona aos alunos a aprendizagem da geometria de uma forma interativa, permitindo que os alunos observem e rodem um cubo virtual [37].

2.3.3.3. Programação

A realidade aumentada possui inúmeros benefícios no campo da educação, sendo um deles a possibilidade de alunos se envolverem mais ativamente no processo de desenvolvimento das aulas juntamente com os respetivos professores. Por esta razão, os professores podem utilizar esta tecnologia imergente no ramo da educação para seu próprio proveito, através da utilização de plataformas específicas para aulas de codificação.

Um das muitas aplicações atualmente utilizadas é a Tynker que permite que estudantes desenvolvam projetos de RA para videojogos em contexto de sala de aula com o auxílio dos professores [38].

2.3.3.4. História

Nas áreas do saber acima discriminadas podemos ver um padrão predominante no que diz respeito ao porquê de se usar a RA, a interatividade que esta oferece. Também em história, ferramentas de RA são utilizadas para ajudar os alunos a experimentá-la de maneira interativa. Ferramentas como Timelooper [39] e 360Cities [40] permitem que sejam realizadas visitas completamente virtuais a sites a nível mundial de forma a ensinarem perspetivas culturais e históricas. Em museus e locais históricos, professores e estudantes podem utilizar os respetivos dispositivos pessoais devidamente equipados com aplicações de RA que fornecem informações e contextos complementares acerca de peças ou artefactos históricos em exposição.

2.3.4. Arquitetura e Construção

No ramo da arquitetura, a realidade aumentada promete revolucionar a forma como os arquitetos projetam os edifícios. Em oposição à realidade virtual, a RA possibilita o casamento entre desenhos arquitetônicos e a realidade do local de construção, aumentando a precisão e eficiência, ao mesmo tempo que minimiza a possível ocorrência de erros, economizando dinheiro, recursos e tempo [41].

Em termos práticos, permite colocar modelos 3D de um projeto sobrepostos num espaço existente, com o auxílio de dispositivos. Muitos dos locais de construção encontram-se sujos, desarrumados e barulhentos. Aliado a este facto, há sempre a possibilidade da ocorrência de erros e dúvidas durante o processo da construção, sabendo que estes custam muito tempo e dinheiro a corrigir. O uso da RA na arquitetura e construção permite que aplicações forneçam uma vista mais precisa do que será construído, através da inclusão dos diferentes tipos de materiais e instalações, como por exemplos tubos de esgoto, muitas vezes difíceis de visualizar por meio de simples desenhos e esquemas. De forma a tentar combater esta ideia são concebidos modelos e hologramas de modelos 3D para que seja possível compreender e executar um projeto. Mesmo durante a construção, a habilidade de se conseguir ver objetos escondidos por entre paredes ou outras divisórias, simplifica o processo enquanto reduz a chance de ocorrência de erros [42].

Em suma, o uso da RA na arquitetura e construção auxilia as seguintes tarefas muito comuns na construção [43]:

- Controlo de qualidade e inspeções
- Visualização de modelos 3D de design tanto em *outdoor* como *indoor*
- Exposição de objetos ocultos por delimitadores físicos

De seguida irão ser mencionadas várias aplicações atualmente em uso no ramo da arquitetura e construção que vão de encontro ao que foi anteriormente mencionado.

2.3.4.1. Morpholio AR Sketchwalk

Esta aplicação de RA possibilita que designers utilizem a realidade aumentada nos seus esboços de modo a proporcionarem uma sensação de espaço mais próxima da realidade tanto aos seus clientes como a si mesmos. Através de um iPad, posiciona-se o esboço no local da construção, sendo possível andar sobre ele e até mesmo fazer crescer paredes. Tal facto faz com que a experiência de apresentação de um projeto seja muito mais interativa e compreensível para os clientes [44].

2.3.4.2. DAQRI Smart helmet

O DAQRI *Smart helmet* é um capacete inteligente que permite a visualização de projetos e modelos 3D por meio da realidade aumentada. Equipas de trabalhadores podem comparar o trabalho que se encontra a ser desenvolvido com o design original [45].

2.3.4.3. Aplicações de medição para Android e IOS

Atualmente, com o avanço dos telemóveis e da realidade aumentada, começam a existir aplicações de RA que já vêm instaladas de fábrica no próprio dispositivo, particularmente as de medição de espaços. Estas funcionam tal e qual como régua digital, fazendo o cálculo das distâncias em espaços reais recorrendo à câmara do dispositivo móvel. Através da aplicação é possível medir objetos e desenhar os planos de um espaço real. Como exemplos concretos deste tipo de aplicações temos a AirMeasure e MeasureKit [42].

2.3.4.4. GAMMA AR

GAMMA AR apresenta-se como sendo uma aplicação de monitorização de locais de construção que utiliza a tecnologia de RA para sobrepor edifícios BIM (*Building Information Modeling*) 3D recorrendo a *smartphones* ou *tablets*. Permite comparar o trabalho real com a informação presente no projeto original, fazendo com que os modelos 3D possam ser vistos antes e durante a totalidade do processo de construção,

o que cria uma melhor compreensão do projeto, evita erros e reduz os custos de construção [46].

2.3.5. Herança Cultural

Atualmente, a RA é, cada vez mais, um suporte inovador para a divulgação do Património Cultural. Atua como um guia para os visitantes e como um suplemento a conteúdos relativos ao património visitado, contexto histórico, etc., proporcionando assim uma experiência mais imersiva e informativa [3]. Este tipo de experiência potencia a adesão de novos visitantes aos locais de herança cultural. O facto de na realidade aumentada acontecer a subtil união entre o real e o virtual, permite que a visualização de artefactos, locais históricos, monumentos e obras de arte, outrora existentes, possam ser devidamente apreciados e estudados pelos visitantes. Para além disso, com a realidade aumentada aliada ao Património Cultural, consegue-se obter um maior e aprofundado conhecimento sobre a arquitetura das diferentes épocas da história da humanidade, garantindo sempre que o ambiente atual que nos rodeia se mantém intacto.

Com a melhoria da tecnologia e do avanço dos dispositivos pessoais, novas técnicas de RA são criadas, o que permite que a imersão de um visitante num determinado local seja mais real. Esta possibilidade de imersão, com o auxílio de imagens virtuais aumentadas no mundo real, tem levado a um enorme progresso na área da conservação e divulgação de Património Arquitetónico, através do desenvolvimento de aplicações de RA para esse mesmo fim.

Inúmeros *papers* e trabalhos de investigação já demonstraram que a RA se apresenta como uma tecnologia que possui cada vez mais valor para a inovação e experiência do visitante [47].

2.3.5.1. Trabalhos relacionados

Nos seguintes parágrafos irão ser referenciados diversos projetos de realidade aumentada com aplicação direta à divulgação de Património Arquitetónico.

A aplicação de realidade aumentada, KnossosAR, possui como principal propósito a utilização da tecnologia MAR (*Mobile Augmented Reality*) no auxílio a jovens

estudantes no contexto educacional, sozinhos ou em grupo, através de visitas guiadas em locais arqueológicos ao ar livre. O local arqueológico de Knossos é o maior sítio arqueológico da Idade do Bronze, tendo sido promovido a património Mundial da Unesco. Os seus principais objetivos são:

- Dar a conhecer o palácio de Knossos através da sua arquitetura, decoração, artefactos e frescos.
- Adquirir conhecimentos elementares acerca da civilização Minoica, a sua soberania nos mares e a propagação da civilização Minoica através do comércio.
- Dar a conhecer o estilo de vida da civilização *Minoan*. As suas ocupações, casas, culinária, tipos de roupa e a respetiva *toilette*.
- Compreender as causas que levaram à destruição da civilização *Minoan*.

Archeoguide (Augmented Reality based Cultural Heritage On-site GUIDE), foi um projecto financiado pela União Europeia para desenvolvimento de um sistema de realidade aumentada destinado a visitantes de um local arqueológico [48]. Teve como caso de estudo o local arqueológico histórico de Olímpia, situado na Grécia, onde eram realizados outrora os Jogos Olímpicos.

A principal motivação para a realização deste projeto prendeu-se na enorme afluência de turistas a este local histórico. Uma vez que este tipo de local possui apenas poucos vestígios de ruínas, o cuidado para a preservação do mesmo deve ser adicional. Sendo assim, o Archeoguide veio ajudar na medida em que preserva completamente o local histórico e o espaço envolvente, ao mesmo tempo que ajuda os visitantes a terem uma perceção de elevada precisão da arquitetura do local visitado.

O sistema é constituído por uma interface de navegação, modelos 3D dos templos do passado e das respetivas estátuas ornamentais e de avatares a simular os competidores na mítica corrida histórica no antigo estádio olímpico. Na parte das comunicações é utilizada a WLAN, porém a localização é conseguida através do uso do GPS, devido à sua enorme precisão [13]. Relativamente ao sistema propriamente dito, diversas unidades móveis podem ser utilizadas, todas escaláveis, desde HMDs, passando para computadores do tipo Palmtop e computadores de bolso.

Em 2002 surge o projeto LIFEPLUS, por *Papagiannakis et al.*, com o principal propósito de dar a conhecer Pompeia antiga aos visitantes, através da reconstrução da fauna e flora, bem como de grupos sociais que demonstram algum tipo de comportamento da vida na quotidiana da época. Uma outra finalidade ainda do projeto era a de transcender os limites da época em questão em termos da tecnologia de realidade aumentada, onde os visitantes conseguissem obter uma experiência com um elevado grau de imersão interativa [49].

Neste projeto, foi utilizado o HMD para a captura das cenas reais, auricular e *wireless TrackBall* de forma que o utilizador conseguisse ouvir e visualizar o mundo virtual, com recurso à RA. Logo após, era possível renderizar simulações 3D realistas da fauna e da flora de carácter virtual, em tempo real, bem como de cenários baseados no ambiente da época. Todos os humanos concebidos virtualmente possuem uma estrutura corporal e estão acompanhados por falas, expressões faciais e roupa da época. [50]. O sistema de *tracking* tinha por base o sem marcadores, realizado através da localização do utilizador no interior do local arqueológico.

Apesar do projeto ter sido concebido especificamente para centros de património cultural, o paradigma não se limita apenas para temas relacionados com património cultural, abrangendo todos os tipos de entretenimento baseados em localização.

Capítulo 3. Especificação da solução

Tendo em conta o desafio proposto, guias de realidade aumentada para a divulgação de património arquitetónico, a especificação da solução passa por definir dois componentes funcionais:

- Uma aplicação web, que permite a gestão dos conteúdos e a configuração de uma visita virtual ao espaço museológico/de divulgação do património;
- Uma aplicação móvel de visualização dos conteúdos e apoio ao visitante;

Antes mesmo de se começar a trabalhar na arquitetura do sistema, foram definidos os seguintes requisitos para o sistema:

- Existem três perfis standard (perito, adulto, infantil);
- Alguns artefactos já se encontram presentes na base de dados no servidor;
- Os guias para os respetivos perfis já existem na base de dados do servidor;
- Todos os ficheiros multimédia já existem na base de dados do servidor;
- Caso o Visitante pretenda, é criado um guia específico;
- Visitante lê um QRCode a partir da aplicação de forma a poder começar a experiência de RA;
- Ficheiro de configuração em formato JSON;
- Uso de marcadores;

Os três perfis existentes dizem respeito ao nível de conhecimento que cada visitante possui sobre um determinado tema ou área, ou seja, os modelos 3D, a informação e todo o conteúdo multimédia são escolhidos pelos Administradores ou Utilizadores do sistema adaptados ao grau de conhecimento de cada indivíduo em particular. Atente-se ainda que o mesmo ficheiro de configuração pode ser utilizado por vários visitantes em simultâneo, uma vez que pode haver vários utilizadores com o mesmo perfil de visitante, bem como com a mesma visita.

Neste capítulo irá ser apresentada a solução encontrada para o problema proposto. Para tal é inevitável definir de uma forma clara e precisa a experiência de RA com os devidos requisitos do sistema de forma que todo o processo de lógica e programação vá de encontro aos mesmos.

Assim sendo, irá ser realizada uma divisão clara entre os constituintes fundamentais que compõem a solução, nomeadamente os requisitos do sistema, a sua arquitetura e o respetivo modelo de dados.

3.1. Definição da experiência

De uma forma geral, pretende-se que qualquer visitante de um determinado museu ou local arqueológico se guie através de uma aplicação, como acontece com as visitas guiadas tradicionais. A única diferença entre ambas prende-se no facto da primeira ser realizada de uma forma completamente autónoma pelo próprio visitante, através da utilização de um dispositivo pessoal.

De forma a simplificar o processo da escolha da experiência por parte dos visitantes, só será possível realizar uma única visita de cada vez, devendo dirigir-se ao gestor do local visitado para uma nova visita, caso seja de interesse. A possibilidade de um mesmo visitante dispor de várias visitas em simultâneo foi pensada e planeada, contudo não foi posta em prática. Assim sendo, a experiência total rege-se pelas seguintes definições:

1. O gestor do local visitado cria uma ou várias visitas personalizadas.
2. O(s) visitantes(s):
 - 2.1. Descarrega(m) a aplicação de RA no seu dispositivo e consome(m) a visita por si escolhida.
 - 2.2. Iniciam o percurso.
 - 2.3. Ao longo do percurso, nos locais onde existem marcadores, foca(m) a câmara no mesmo de forma a poder visualizar o seu conteúdo.

Na base de uma visita encontram-se os marcadores a ela associados. Inicialmente foi necessário determinar o sistema de coordenadas onde o artefacto iria ser exibido, dado que todos os dados presentes no ficheiro de configuração se encontram referentes ao modelo escolhido. Existem três tipos distintos de AR:

- **Baseado em marcadores** – faz-se uso de marcadores de forma a ser possível a amostra de um elemento virtual no mundo real, apontando a câmara do dispositivo para o respetivo marcador.
- **Sem marcadores** – o utilizador move o objeto virtual de livre vontade, sem este se encontrar preso a um marcador específico, o que permite uma maior flexibilidade na hora de posicionar os objetos.
- **Baseado em localização** – cria-se uma ligação entre uma localização concreta e o conteúdo de realidade aumentada.

Tendo estes três tipos distintos de AR em mente, decidiu utilizar-se o primeiro, baseado em marcadores, visto que a sua implementação apresenta um menor nível de dificuldade e o erro de localização é menor, ou seja, com o uso de marcadores existe a vantagem de haver sempre um menor erro de posicionamento do objeto a visualizar, enquanto com o uso das outras tecnologias pode haver erros derivados às coordenadas de GPS.

Tal como numa visita tradicional, onde somos guiados para locais específicos ao longo do percurso, também na visita virtual o visitante segue o caminho indicado pelos marcadores. Inicialmente foi ponderada a implementação de um mapa do local a visitar com cada um dos pontos de interesse (marcadores) devidamente destacados e posicionados. No entanto a ideia acabou por ser descartada devido à complexidade adicional que iria trazer, não sendo um componente essencial ao problema proposto. Cada marcador encontra-se associado a um objeto. Assim, quando um marcador for identificado pela câmara do dispositivo, este retorna o respetivo objeto e informação acerca do mesmo.

No entanto, um objeto tridimensional por si só não é suficiente para definir uma visita, apesar de possuir um papel fundamental no conhecimento acerca do mesmo, uma vez que é possível a sua visualização 3D. Nem todos os espaços de um sítio arqueológico possuíam, outrora, artefactos. Podem ser apenas pontos de interesse, como por exemplo jardins ou salas, sem objetos físicos. Surge então a necessidade de acrescentar informação adicional relativa a um marcador, para que seja possível a um visitante descobrir mais a respeito de um determinado artefacto ou ponto de interesse. Para tal, a cada marcador está também associado um conjunto de ficheiros multimédia,

de acordo com o guia escolhido e as necessidades do visitante, tais como imagens, vídeos, áudio ou textos complementares.

De forma a ser possível a um visitante a visualização de conteúdos multimédia relativos ao respetivo ponto de interesse é crucial que a associação entre a interface e o marcador seja bem realizada por parte do gestor de conteúdos aquando da criação do guia de RA. De forma a melhorar a UX do utilizador, uma nova cena com todos os conteúdos multimédia e o objeto 3D (caso exista) é criada, dando assim oportunidade ao visitante de visualizar esses mesmo conteúdos na passagem entre dois marcadores, sem a constante necessidade da deteção de um marcador, o que faria com que tal conteúdo apenas pudesse ser apreciado num único instante. Esta abordagem possibilita assim ao visitante usufruir da reconstituição de um determinado artefacto em tempo real no local onde outrora se encontrava e do referente conjunto multimédia de forma independente.

3.2. Requisitos do Sistema

Definida a experiência de realidade aumentada, os requisitos que o sistema tem de conter tornam-se mais evidentes. De seguida irá ser exposta a listagem relativa a esses mesmo requisitos.

- Possibilidade de importar marcadores.
- Possibilidade de listar marcadores.
- Possibilidade de eliminar marcadores.
- Possibilidade de importar ficheiros multimédia.
- Possibilidade de listar ficheiros multimédia.
- Possibilidade de eliminar ficheiros multimédia.
- Possibilidade de importar objetos tridimensionais.
- Possibilidade de listar objetos tridimensionais.
- Possibilidade de eliminar objetos tridimensionais.
- Criar e configurar visitas de realidade aumentada.
- Cada utilizador apenas possui uma única visita de cada vez.
- Garantir a total privacidade do utilizador.

3.3. Arquitetura do Sistema

Foi criada assim uma arquitetura para cada um dos sistemas, bem como as tabelas presentes na base de dados, de forma a ir de encontro ao idealizado.

A arquitetura das aplicações web/móvel foi a primeira a ser concebida e posta em prática. De notar que durante todo o processo de conceção e execução da mesma, teve-se sempre em especial atenção o facto de esta ser a mais dinâmica, responsiva e intuitiva possível. Sabendo que há várias formas possíveis de chegar a uma solução, das quais existe sempre uma mais ótima, foram concebidas duas formas distintas de carregamento dos conteúdos no caso da aplicação móvel. A primeira prende-se com o facto da possível falta ou fraco sinal de WiFi no local a visitar. Nesta, toda a informação contida no ficheiro de configuração é lida pela aplicação de uma e apenas uma só vez durante toda a experiência de realidade aumentada. A segunda solução, ao contrário da primeira, não sobrecarrega o dispositivo do visitante com toda a informação essencial à experiência. Sabendo que nem todos os dispositivos possuem as mesmas capacidades em termos de *hardware*, nem a mesma quantidade de memória, esta solução acaba por ser mais leve. Os conteúdos necessários para a visualização total de um determinado artefacto são apenas carregados na hora da observação do mesmo. Contudo, esta solução requer uma cobertura completa de rede WiFi em todo o local a visitar, bem como a possibilidade de atrasos devido a *delays* entre o envio e a receção da informação, levando a uma eventual quebra na visita. Deste modo, optou-se por escolher a primeira abordagem.

Começou por ser feita a distinção entre dois tipos de utilizador. O Administrador e o Visitante. O primeiro possui como principal função a de fazer a gestão dos guias de realidade aumentada, ou seja, pode criar e editar visitas através da aplicação, na aba de criação de um novo guia. Este novo guia irá conter os dados relativos ao nome, descrição, data de criação, perfil de visitante e todos os artefactos com a respetiva multimédia associada, de acordo com o(s) perfil(s) definido(s). De notar que toda a informação necessária para a criação de um guia é armazenada numa base de dados global. Como sabemos, esta é o cerne de toda a aplicação visto que é a partir de cada uma das diferentes tabelas que se formam as muitas interações entre o *front-end* e o *back-end* e onde ficam armazenados os dados relativos à aplicação.

Tendo o Administrador criado o guia, a aplicação é descarregada para o dispositivo do Visitante. Optou-se por utilizar a leitura de um código QR na leitura do ficheiro de configuração que alimenta a aplicação de RA, ou seja, mal o Visitante entra na aplicação, terá a oportunidade de usar a câmara do seu dispositivo para importar a configuração do guia por si escolhido, passando deste modo a poder iniciar a visita. Na Figura 15 e Figura 16 encontra-se a representação gráfica da arquitetura do sistema acima descrito.

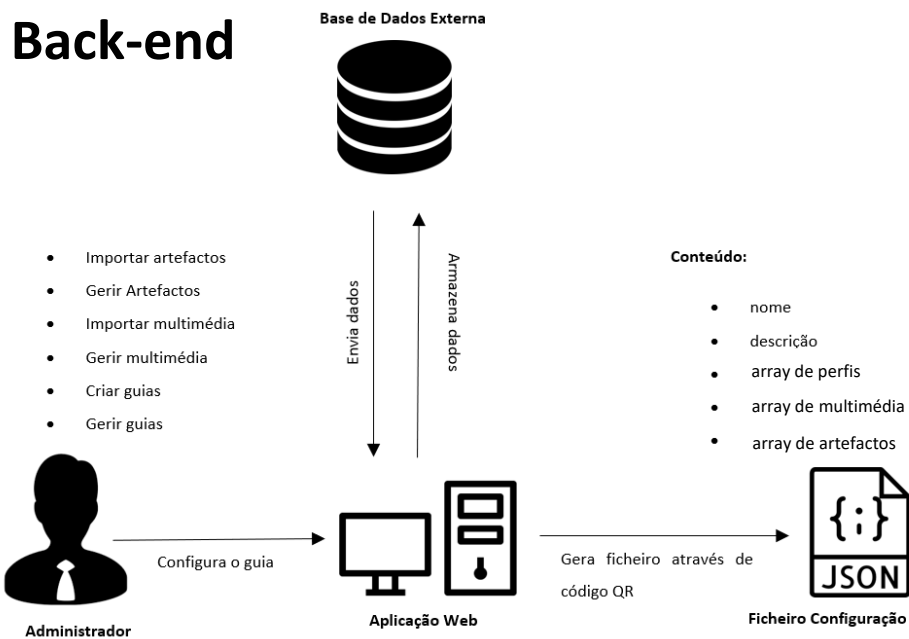


Figura 15 - Arquitetura da aplicação de Realidade Aumentada (Back-end)

Front-end



Figura 16 - Arquitetura da aplicação de Realidade Aumentada (Front-end)

3.4. Configuração da Visita

Antes de um Visitante iniciar a experiência de realidade aumentada, é necessário que haja uma visita para ele configurada com toda a informação relevante. Para que tal aconteça, foi criada uma aplicação de gestão das diversas visitas possíveis de existir, criadas e geradas pelo administrador do sistema. De notar que toda a informação referente a cada uma das visitas situa-se armazenada numa base de dados global, concebida para o efeito.

De uma forma geral, o administrador pode realizar cada uma das seguintes interações:

- Importar artefactos;
- Gerir Artefactos;
- Importar multimédia;
- Gerir multimédia;
- Criar guias;
- Gerir guias;

Inicialmente foi pensada a possibilidade de existir um sistema de login por visitante. No entanto, acabou por ser descartada uma vez que, tal como acontece numa visita guiada com a compra do bilhete, um visitante apenas quer visualizar cada um dos artefactos existentes sem ter de se prender a qualquer tipo de sistema de identificação. No final, caso o visitante deseje, pode deixar um comentário, anónimo ou não, acerca da experiência e do guia por si escolhido.





Capítulo 4. Implementação


Neste capítulo e consequentes subcapítulos irá ser esmiuçada a implementação da solução proposta, bem como os pensamentos e ideias que originaram essa mesma solução, tendo sendo como foco principal o dinamismo e a natural interação dos utilizadores com o sistema. Começará por ser apresentada a implementação da aplicação responsável pela configuração dos guias de RA, uma vez que é através desta que os guias são concebidos e, posteriormente, a implementação da aplicação de RA propriamente dita.

4.1. Ferramentas utilizadas

Antes de se pôr em prática o projeto, foi necessário escolher um conjunto de ferramentas de um vasto leque, capazes de dar uma resposta eficaz ao problema proposto e à solução pretendida com o mesmo. Desta forma, as ferramentas foram escolhidas de acordo com os requisitos do sistema, com a facilidade de utilização e com maior documentação, seja pelos próprios criadores das mesmas, ou de utilizadores experientes.

Tabela 1 - Ferramentas de trabalho

Logotipo	Designação	Descrição	Versão
	AR Foundation	SDK de RA	9.9.3
	Unity	Plataforma de desenvolvimento 3D	2020.3.17f1
	Rider	Editor de código	2021.3.2
	MySQL Workbench	Ferramenta de criação de bases de dados	8.0.23
	Google Chrome	Navegador web	96.0.4664.110

	Postman	Ambiente de desenvolvimento de APIs	9.9.3
-----------------------------------------------------------------------------------	---------	-------------------------------------	-------

4.2. Modelo de dados

Neste tópicos irão ser abordados os diferentes modelos de dados que compõe a base de dados. Primeiramente foi necessário realizar o planeamento cuidadoso de cada uma das tabelas que compõem a base de dados, bem como a respetiva relação entre elas com auxílio de um diagrama ER (ver Figura 17).

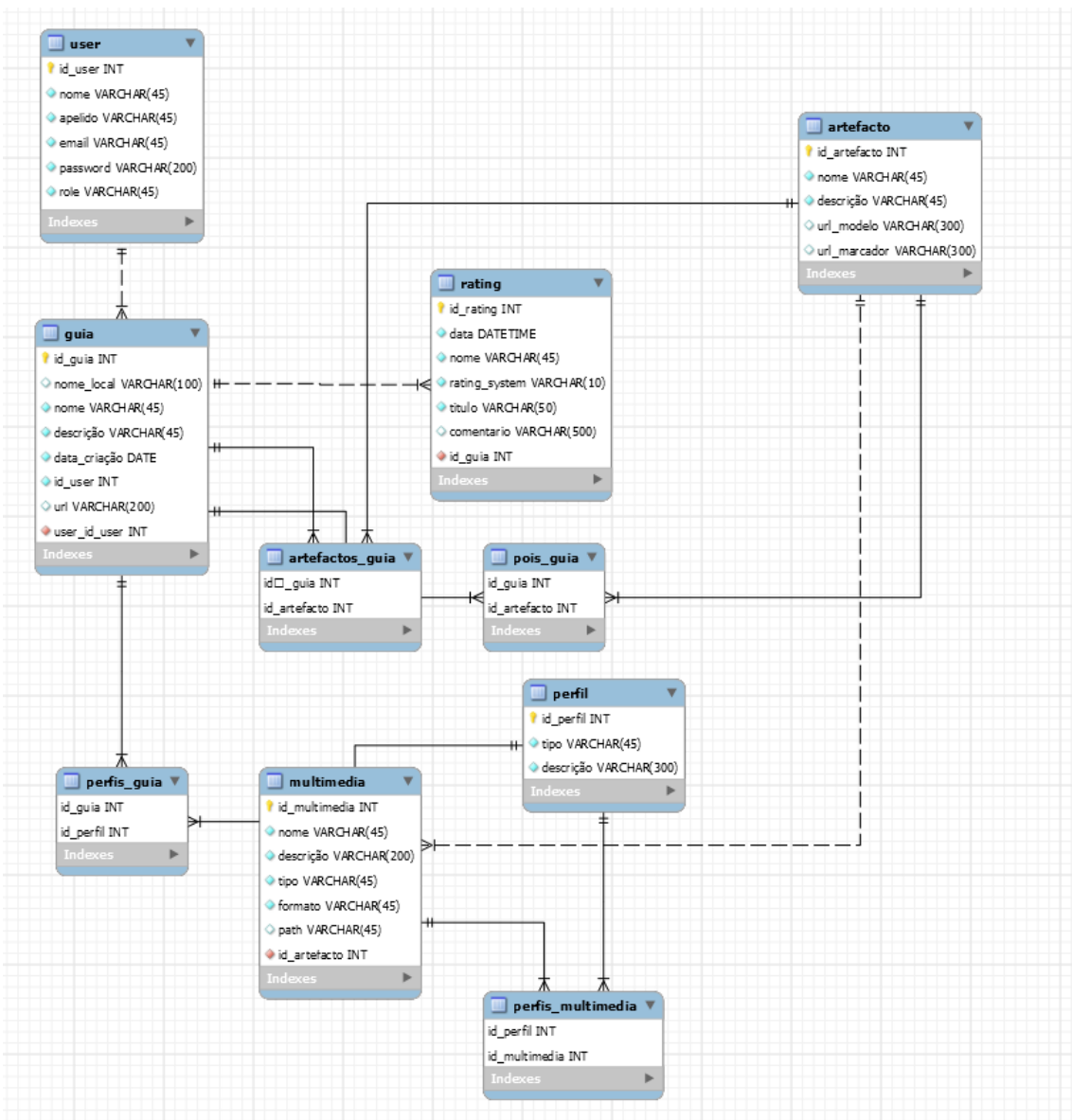


Figura 17 - Diagrama ER da Base de Dados

Inicialmente foi pensada a possibilidade de cada utilizador efetuar várias visitas em simultâneo. Tal facto faria com que cada visitante de um determinado museu tivesse a oportunidade de realizar múltiplas visitas, caso pretendesse. No entanto, achou-se por bem que cada visitante possuísse uma, e apenas uma só visita em cada momento. Caso pretendesse realizar outra, teria de se dirigir novamente aos gestores de guias de realidade aumentada de forma a poderem importar outra visita, podendo assim dar início à experiência de realidade aumentada. Desta forma, uma visita não está diretamente conectada a um visitante específico, aumentando a escalabilidade da solução. Por outro lado, uma vez que existe apenas uma base de dados central, uma visita criada por um administrador do sistema pode ser vista por todos os outros de maneira a contornar entradas duplicadas. Assim sendo, todos os itens necessários à criação de um guia podem ser vistos por todos os administradores do sistema do mesmo sítio arqueológico.

Em primeiro lugar, foi criada a tabela referente aos Utilizadores do Sistema, User, contemplando os seguintes parâmetros:

- **Id_user (PK)** – identificador único do utilizador;
- **nome** – nome do utilizador;
- **apelido** – apelido do utilizador;
- **email** – endereço de email do utilizador;
- **password** – password única do utilizador, convenientemente encriptada;
- **role** – papel do utilizador: normal ou administrador;

Apesar de se denominar administradores do sistema ao conjunto de pessoas responsáveis pela gestão e configuração dos guias de RA, realizou-se a separação destes mesmos em dois distintos grupos: utilizadores normais e administradores propriamente ditos. Enquanto os primeiros possuem apenas permissão para criar e editar novos guias de RA, bem como visualizar outros, os últimos possuem total permissão sobre cada um dos guias de RA, independentemente de quem os criou. Tais permissões vão desde editar guias, ficheiros multimédia, etc., a eliminar por completo qualquer guia que não se encontre adequado.

O modelo de dados concebido para guardar as diferentes visitas criadas denomina-se por Guia e possui os seguintes campos:

- **id_guia (PK)** – identificador único do guia;
- **nome_local** – nome do local onde a visita tem lugar;
- **nome** – nome da visita;
- **descrição** – descrição da visita;
- **data_criação** – data de criação da visita;
- **id_user (FK)** – utilizador que criou a visita;
- **url** – link do local onde se encontra o ficheiro de configuração da visita;

Através dos campos que compõem um Guia, conclui-se que não existe qualquer ponto de interesse diretamente associado ao mesmo. Sabendo que um Guia deve conter diversos pontos de interesse de forma a fazer sentido uma visita, criou-se uma tabela adicional responsável pela ligação de um guia aos pontos de interesse, visto que um Guia pode ter associados vários pontos de interesse, mas um ponto de interesse específico pode também estar contemplado em vários Guias. A tabela anteriormente descrita pode ser especificada pelos seguintes campos:

- **id_guia (PK)** – id do Guia que possui um determinado ponto de interesse;
- **id_poi (PK)** – id do ponto de interesse pertencente a um Guia específico;

Desta forma, a relação entre guias e pontos de interesse torna-se clara e evidente, sem a possibilidade de ocorrência de falhas no sistema.

Sabendo que não é possível a realização de uma visita sem os pontos de interesse que vão aparecendo ao longo da mesma, foi necessário conceber essa mesma tabela, na qual estão armazenados todos os pontos de interesse relevantes a todos os guias criados. Tal tabela contempla os seguintes campos:

- **id_poi (PK)** – identificador único de um ponto de interesse;
- **nome** – nome do ponto de interesse;
- **descrição** – descrição do ponto de interesse;
- **url_modelo** – link onde está guardado o objeto 3D, no servidor;
- **url_marcaador** – link onde está guardada a representação do marcaador, no servidor;

Um ponto de interesse por si só não é suficiente para proporcionar ao visitante uma visita imersiva e educacional. Apesar de auxiliar na visualização espacial através de

um objeto 3D à escala, há casos onde o ponto de interesse não se define como sendo um artefacto, mas sim um local outrora importante, como por exemplo um jardim. Surge então a extrema necessidade de se associar algum tipo de multimédia a cada um dos pontos de interesse, de forma que a vertente educacional seja mais pormenorizada e enfatizada.

Sabendo que um qualquer tipo de multimédia (vídeos, imagens, textos, áudio) se encontra diretamente ligado a um ponto de interesse particular, fica fácil a definição da tabela multimédia. Neste contexto, a relação entre um ficheiro multimédia e um ponto de interesse é de um para muitos, ou seja, um ponto de interesse pode e deve ter a si associados múltiplos ficheiros multimédia, mas um ficheiro multimédia específico apenas pertence a um único ponto de interesse. De modo a facilitar o que acabou de ser descrito, apresentam-se de seguida as entradas da tabela Multimédia:

- **id_multimedia (PK)** – identificador único de um ficheiro multimédia;
- **nome** – nome do ficheiro multimédia;
- **descrição** – descrição do ficheiro multimédia;
- **tipo** – tipo do ficheiro multimédia (imagem, vídeo, áudio);
- **formato** – formato do ficheiro multimédia (jpg, png, wav, etc...);
- **path** – link onde está guardado o ficheiro multimédia, no servidor;
- **id_artefacto (FK)** – id do artefacto correspondente ao ficheiro;

Neste momento, a caracterização de um guia de realidade aumentada encontra-se devidamente definida. A partir deste ponto é possível a um visitante começar uma visita com todos os seus elementos descritivos e informativos. No entanto, e de forma a melhorar ainda mais a experiência de realidade aumentada, foi considerada uma nova tabela, Perfil, na qual se faz o levantamento do perfil de visitante do local a visitar. Por meio desta tabela, o administrador do sistema consegue traçar uma visita ideal atentando o perfil de visitante em questão, pois sabemos que nem todos os visitantes de um sítio arqueológico possuem o mesmo conhecimento sobre os testemunhos materiais do homem e do seu entorno. Posto isto, a tabela Perfil contempla apenas três campos distintos:

- **id_perfil (PK)** – identificador único de um perfil de visitante;
- **tipo** – tipo de perfil de visitante;

- **descrição** – breve descrição do perfil de visitante;

Através da incorporação dos perfis de visitante, o administrador consegue então um maior controlo sobre todos os dados a inserir numa visita, aquando da criação dos guias, sabendo que nem todos os visitantes querem ver as mesmas peças, ou visitar os mesmos locais dentro de um sítio arqueológico. Para além disso, o nível de conhecimento acerca de determinados objetos também não é o mesmo. Um perito num certo assunto talvez queira uma visita mais pormenorizada do que um leigo, por exemplo.

Por fim, de modo a juntar guias com perfis, foi necessária uma tabela suplementar, uma vez que guias e perfis possuem uma relação de muitos para muitos. Tal tabela faz a correspondência entre um perfil específico e o respetivo guia de realidade aumentada e define-se por intermédio dos seguintes campos:

- **id_guia (PK)** – identificador do guia correspondente ao perfil;
- **id_perfil (PK)** – identificador do perfil de visitante presente num guia;

Construída esta tabela, dá-se por finalizado o modelo de dados no qual assenta toda a experiência de realidade aumentada.

4.3. Aplicação Web

A aplicação web tem como principal objetivo a criação dos diferentes guias de Realidade Aumentada que possam vir a existir. Desta forma, nada mais é do que uma ferramenta de gestão da experiência de RA. Esta deve ser prática, eficaz, dinâmica e intuitiva. Começaram por ser concebidas cada uma das páginas que se achou essenciais para criar os guias. Assim sendo, a aplicação web é composta por um total de 11 páginas: login, registo, página inicial, criar guia, listar guias, criar pontos de interesse, listar pontos de interesse, importar multimédia, listar multimédia, criar perfil de visitante, listar perfis de visitante. De modo a se alternar entre cada uma das páginas da maneira mais rápida possível, existe um menu lateral visível em todas as páginas. Tais estão melhor apresentadas no diagrama da Figura 18.

De notar que o desenvolvimento da aplicação web, não se focou na parte do UI propriamente dito, mas sim na facilidade e dinamismo com que se criam os diferentes guias.

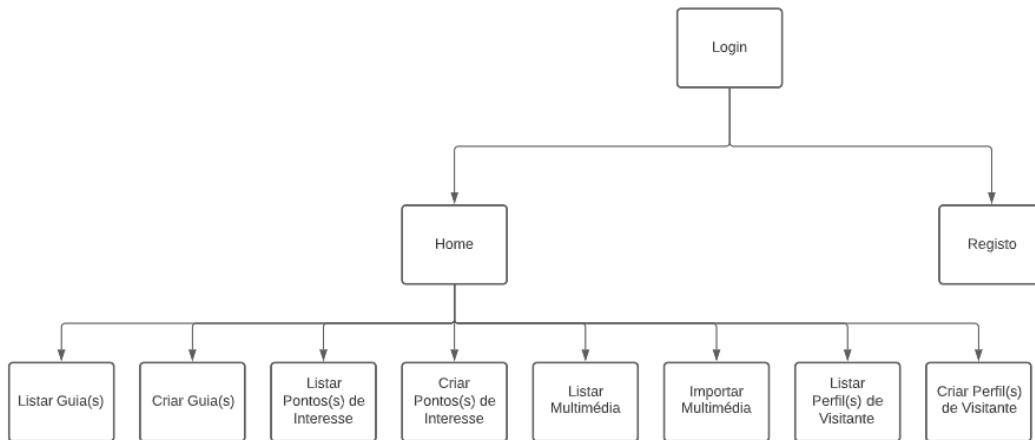


Figura 18 - Diagrama de navegação da aplicação Web

4.3.1. Páginas Web

Inicialmente o administrador e os responsáveis pela conceção dos guias de realidade aumentada são levados para a página inicial da interface. Esta apenas contém o menu de navegação, um título e um botão de início de sessão. Na Figura 19, encontra-se representada a página de início.



Figura 19 - Página inicial da aplicação Web

Caso os utilizadores não estejam previamente registados, são levados para um simples formulário de registo, Figura 20, onde inserem respetivo nome, apelido, email e password. Todos os campos do formulário são de carácter obrigatório.

REGISTO

👤	<input type="text" value="nome"/>
👤	<input type="text" value="apelido"/>
✉	<input type="text" value="email"/>
🔒	<input type="password" value="password"/> 👁
🔒	<input type="password" value="confirmar password"/> 👁

❗

Figura 20 - Página de registo da aplicação Web

Uma vez criada a conta, esta tem de ser aprovada pelo administrador do sistema de forma a poder ser utilizada. Após a conta ser devidamente autenticada, o utilizador é redirecionado para a página de login, podendo assim entrar na sua conta pessoal de criação e gestão de guias de realidade aumentada.

Quando o utilizador entra na interface, é-lhe apresentada a página de referência (ver Figura 21) acerca do que se pretende com a própria e o que cada um dos diferentes menus se propõe alcançar.

Uma vez que existe apenas uma única base de dados global, apesar de cada um dos utilizadores criar as suas próprias definições, estas aparecem visíveis para todos os outros, de forma a controlar possíveis duplicados na base de dados. No entanto, foi pensada a implementação de uma base de dados local para o efeito, que será mencionada num capítulo futuro.

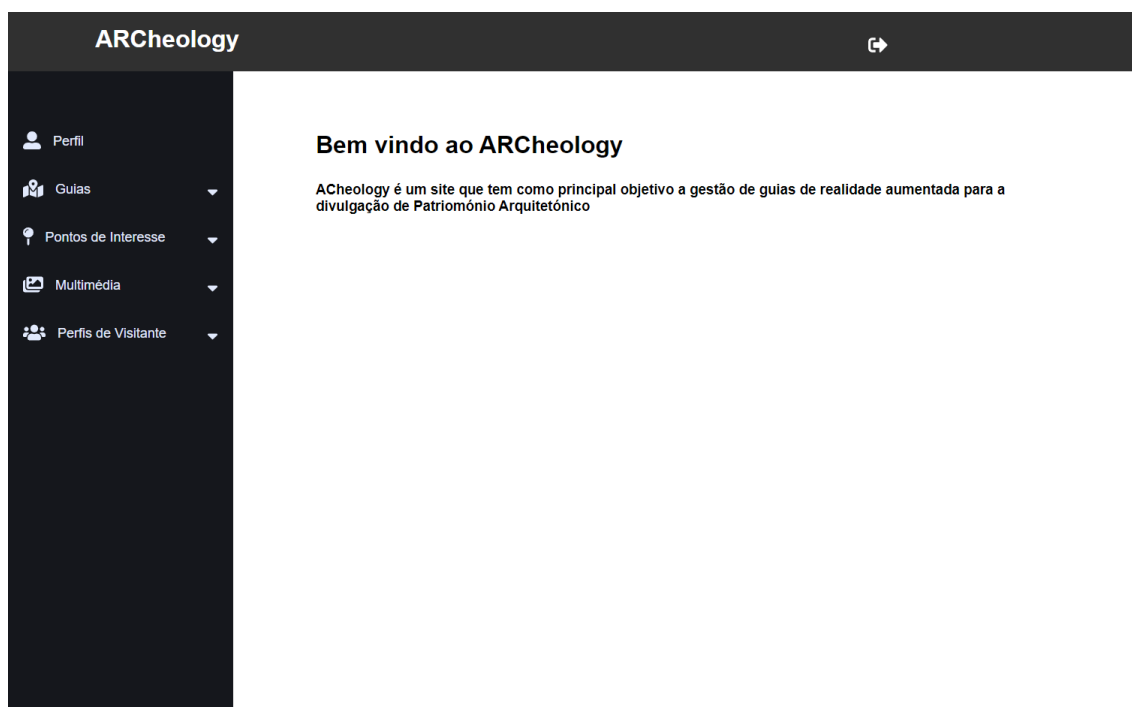


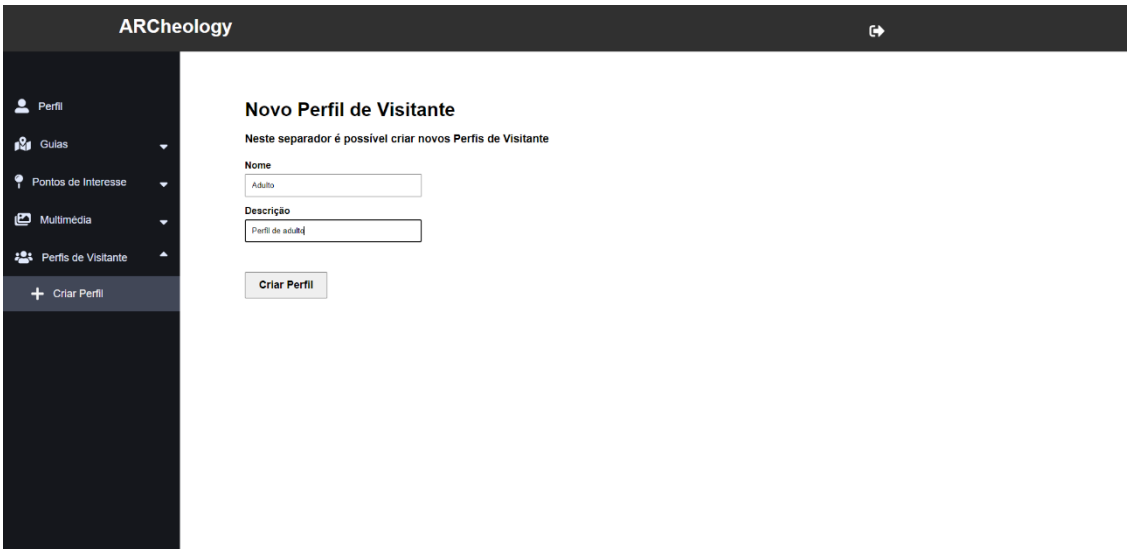
Figura 21 - Página principal da aplicação Web

Como é possível reparar, do lado esquerdo, existe um menu lateral sempre presente em qualquer uma das páginas acedidas. Tal facto, ajuda a uma rápida troca de página. Também conseguimos observar que a ordem em que os diferentes itens do menu aparecem se encontra pela ordem inversa, ou seja, primeiro deverão ser criados os perfis, sendo o resto posteriormente criado de forma ascendente. Deste modo,

quando o administrador chegar ao separador dos guias, possui todos os elementos necessários à conceção dos mesmos.

De seguida irá ser feita uma descrição pormenorizada de cada uma das páginas correspondentes a cada um dos itens constituintes do menu. Perceba-se que, quando se carrega em qualquer um deles o administrador consegue visualizar, através de uma tabela, toda a informação relevante a um determinado item. Possui ainda a opção de pesquisar por palavra-chave, o que pode ser útil caso as tabelas apresentem inúmeras entradas. Cada cabeçalho de cada uma das tabelas é clicável, o que faz com que as entradas fiquem ordenadas por ordem alfabética conforme o cabeçalho escolhido. Tal facto permite uma melhor organização e procura na hora da pesquisa, caso haja inúmeras entradas.

Como já foi explicado acima, cada visitante que visita um determinado museu possui um perfil associado, seja adulto, criança ou jovem. Sabemos ainda que existe uma diferença entre experientes num determinado tema, história romana por exemplo, e leigos nesse mesmo tema. Sabendo de tal facto o perfil de visitante ajuda a precisar o que cada um dos visitantes pretende aquando da criação do guia. Assim sendo, apenas é necessária a inserção do tipo de perfil e a respetiva descrição, como pode ser visto através da Figura 22.



The screenshot displays the 'ARCheology' web application interface. On the left is a dark sidebar menu with options: Perfil, Guias, Pontos de Interesse, Multimédia, Perfis de Visitante, and a 'Criar Perfil' button. The main content area is titled 'Novo Perfil de Visitante' and contains the text 'Neste separador é possível criar novos Perfis de Visitante'. Below this, there are two input fields: 'Nome' with the value 'Adulto' and 'Descrição' with the value 'Perfil de adult[...]', followed by a 'Criar Perfil' button.

Figura 22 - Página de criação de novo Perfil de Visitante na aplicação Web

Após a criação do perfil desejado, basta aceder à página dos Perfis de Visitante de forma a visualizar o novo perfil (ver Figura 23).

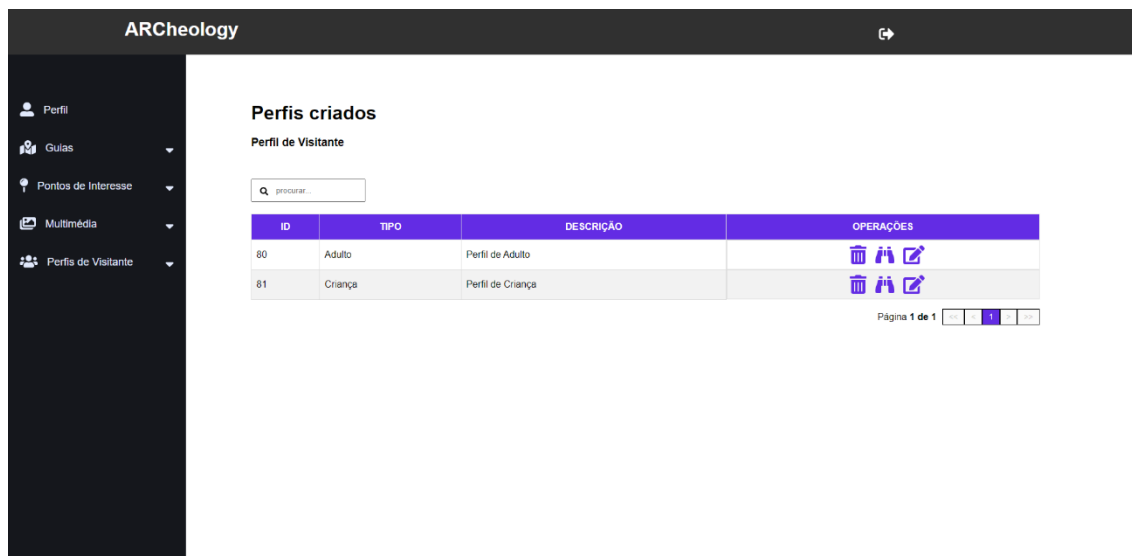


Figura 23 - Página de listagem de perfis de visitante na aplicação Web

Na Figura 23 pode ver-se que existem três distintos botões, denominados operações. O primeiro elimina a entrada pretendida, o segundo permite visualizar de uma forma mais detalhada o item em questão e o terceiro permite editar o mesmo. Contudo, tanto o segundo como o terceiro não se encontram funcionais para todas as tabelas existentes.

Seguindo o menu progressivamente, verifica-se que o próximo item é o conjunto de multimédias. Um item de multimédia pode adotar diferentes tipos e formatos, de acordo com o pretendido em cada visita. De uma forma geral, podemos afirmar que existem três diferentes tipos: imagens, vídeos e ficheiros áudio. Cada um destes ficheiros está diretamente relacionado com um objeto(artefacto), ao qual se deu o nome de Pol (*Point of Interest*). De notar que um determinado objeto não necessita de possuir todos os três tipos de multimédia, variando de acordo com as necessidades do visitante e de quem cria e gere as visitas.

Na aba de importação multimédia, o gestor de guias é capaz de importar ficheiros multimédia tendo em atenção o respetivo ponto de interesse, bem como o perfil a ele associado, de forma que, durante a visita e através da aplicação de RA seja possível visualizar cada um dos ficheiros no devido local. Um determinado ficheiro pode estar presente em vários tipos de perfil de visitante, mas apenas pertencer a um e só

um ponto de interesse. Na Figura 24 é possível ver a representação gráfica do que foi anteriormente explicado.



Figura 24 - Página de importação de ficheiros multimédia na aplicação Web

Tal como em todos os separadores, também o da multimédia possui uma tabela para rápida e fácil visualização de cada ficheiro já importado, com os campos de maior relevo para o gestor de visitas (ver Figura 25).

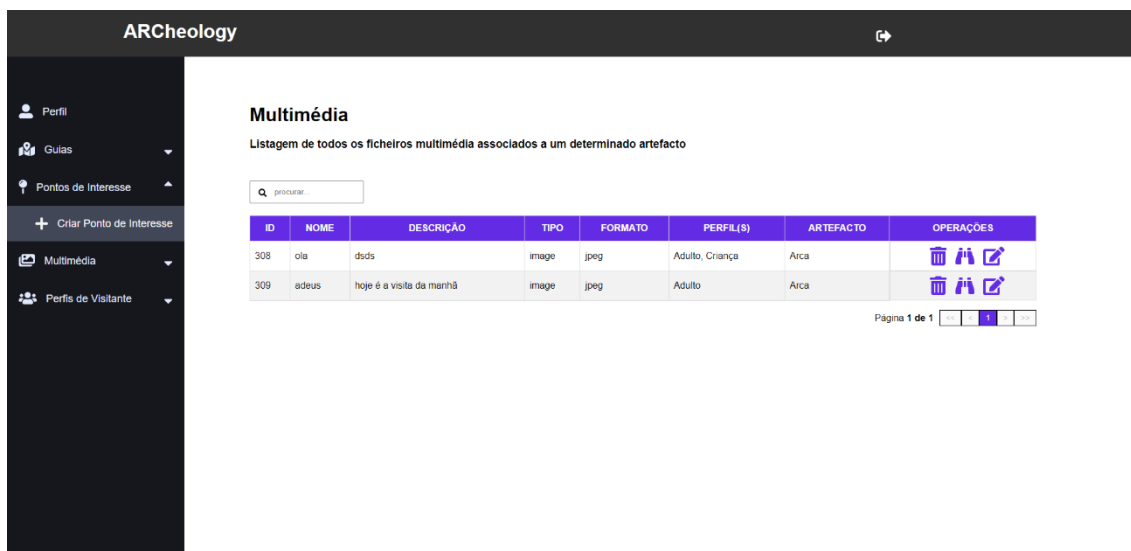


Figura 25 - Página de listagem de ficheiros multimédia na aplicação Web

Subindo na hierarquia do menu lateral, chegamos a um dos separadores essenciais ao sucesso da aplicação de realidade aumentada, o separador dos pontos de interesse.

Quando um visitante decide realizar uma visita guiada a um determinado museu, é-lhe indicado um guia específico que irá fazer a visita ao local, levando-o a ver diversos pontos de interesse de importância relativa num período da história, através de um percurso pré-definido. Acontece que esses pontos de interesse podem não ser artefactos ou objetos em si, mas locais outrora importantes, como por exemplo um jardim ou uma sala. Assumindo este facto, o separador de criação de pontos de interesse possui a particularidade de o gestor de guias poder gerar um ponto de interesse associado a um marcador sem que este esteja ligado a um modelo 3D. Tal facto é possível ser observado com melhor detalhe através da Figura 26.

The screenshot shows the ARCheology web application interface. On the left is a dark sidebar with navigation items: 'Perfil', 'Guias', 'Pontos de Interesse', '+ Criar Ponto de Interesse', 'Multimédia', and 'Perfis de Visitante'. The main content area is titled 'Novo Ponto de Interesse' and includes the following elements:

- Subtitle: 'Neste separador é possível criar novos POI (Points Of Interest). Artefactos, Locais, etc'
- Form fields: 'Nome' and 'Descrição'.
- Section: 'Caso o Ponto de Interesse seja um artefacto em concreto, o upload do marcador e do objeto 3D é realizado aqui'
- Section: 'Marcador' with a file selection button labeled 'Escolher ficheiro' and the text 'Nenhum ficheiro selecionado'.
- Section: 'Modelo 3D' with a file selection button labeled 'Escolher ficheiro' and the text 'Nenhum ficheiro selecionado'.
- Bottom button: 'Criar POI'.

Figura 26 - Página de criação de novos Pontos de Interesse na aplicação Web

Mais uma vez, e tal como acontece em cada um dos separadores, existe uma tabela com todos os pontos de interesse criados, de forma que a gestão do guia ocorra de uma forma prática e eficaz (ver Figura 27).

The screenshot shows the 'ARCheology' application interface. On the left is a dark sidebar menu with icons and text for 'Perfil', 'Guias', 'Pontos de Interesse', 'Criar Ponto de Interesse', 'Multimédia', and 'Perfis de Visitante'. The main area is titled 'Lista de Pontos de Interesse' and contains a search bar with the placeholder 'procurar...'. Below the search bar is a table with the following data:

ID	NOME	DESCRIÇÃO	OPERAÇÕES
146	Arca	Sou uma arca	[Icons for delete, edit, and share]

At the bottom right of the table area, it says 'Página 1 de 1' with navigation arrows.

Figura 27 - Página de listagem dos Pontos de Interesse

Por fim, chegamos ao último patamar do menu, a criação do guia de realidade aumentada (ver Figura 28). Visto que todos os elementos necessários à conceção de um guia de RA foram posteriormente criados, neste separador apenas é necessário definir o(s) perfil(s) de visitante, bem como o(s) ponto(s) de interesse. Não é necessário identificar os ficheiros multimédia referentes aos pontos de interesse que se pretendem visitar, uma vez que a relação entre estes e o respetivo ponto de interesse foi posteriormente conseguida no separador Multimédia.

The screenshot shows the 'ARCheology' application interface for creating a new guide. The sidebar menu is the same as in Figure 27. The main area is titled 'Novo Guia' and contains the following text: 'Neste separador pode criar novos guias temáticos personalizados. Se ainda não importou qualquer tipo de multimédia, sugere-se que a importe primeiro'. Below this are several form fields:

- Temática:** Input field with 'tema' as a placeholder.
- Descrição:** Input field with 'descrição' as a placeholder.
- Data Criação:** Input field with 'Tue2/29/21' as a placeholder.
- Perfil(s) de Visitante(s):** Input field with 'perfil(s) associado(s)' as a placeholder.
- Pontos de Interesse:** Input field with 'poi(s) associado(s)' as a placeholder.

At the bottom of the form is a button labeled 'CRIAR GUIA'.

Figura 28 - Página de criação de um novo Guia de RA na aplicação Web

Existe, uma vez mais, uma tabela com todos os guias criados, possuindo os cabeçalhos de maior importância para o gestor de guias, de forma que a procura por um determinado guia seja o mais breve e intuitiva possível.

Ao contrário do que acontece com todas as outras tabelas dos demais separadores, esta possui a particularidade de conter uma entrada a mais no cabeçalho das operações, como pode ser visto através da Figura 29.


ID	NOME	DESCRIÇÃO	DATA CRIAÇÃO	PERFIL(S)	PONTOS DE INTERESSE	OPERAÇÕES
152	Romanos	jdsjskd	2021-10-18	Adulto, Criança	Arca	

Figura 29 - Página de listagem de Guias de RA na aplicação Web

Tal símbolo permite que seja gerado um código QR que contém o link do ficheiro JSON com toda a informação relevante à visita, consumido subsequentemente pela aplicação de realidade aumentada (ver Figura 30), ou seja, quando a aplicação de RA lê o código QR através da própria câmara do dispositivo móvel. Desta forma, a leitura do ficheiro de configuração da aplicação web para a aplicação móvel é realizada de uma forma subtil, sem a necessidade de utilização de qualquer tipo de cabo para o efeito.

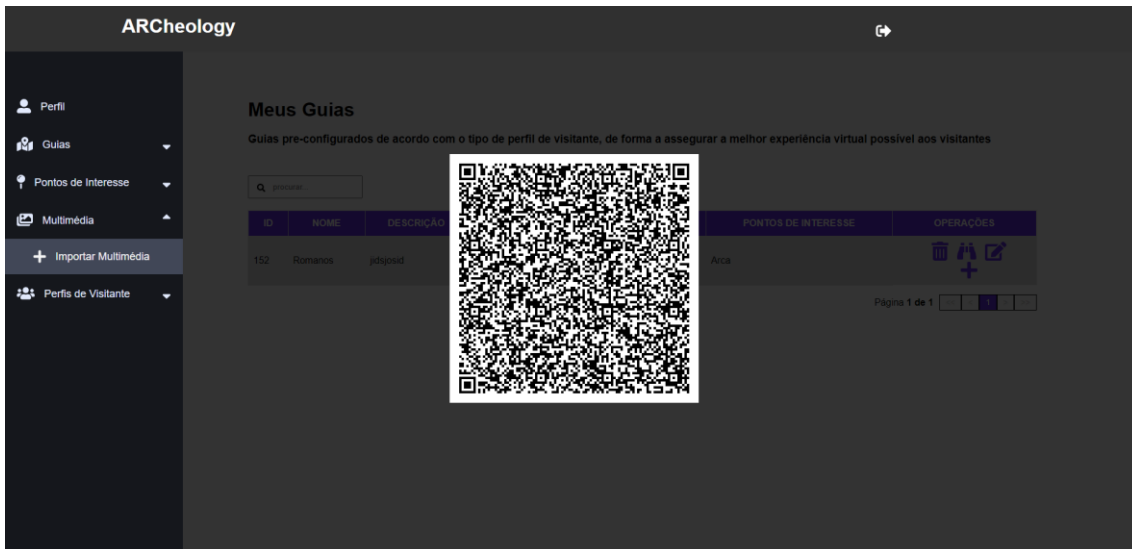


Figura 30 - Ficheiro de configuração no formato de código QR na aplicação Web

Ainda no separador dos guias, cada entrada na tabela pode ser visualizada de uma forma mais detalhada, de maneira que o gestor de guias consiga obter toda a informação de um determinado guia, caso necessário. Essa mesma informação encontra-se bem delimitada e compacta, como se pode observar através da Figura 31.



Figura 31 - Página de visualização de um guia específico na aplicação Web

Caso o gestor queira visualizar os ficheiros multimédia associados a um determinado ponto de interesse, apenas necessita de carregar no ponto de interesse em questão. Aparecerá um modal com todos os tipos de multimédia existentes, bem como o marcador e o modelo, caso o ponto de interesse em questão seja um artefacto. O gestor pode então navegar entre os diferentes separadores de forma a poder observar o(s) ficheiro(s) correspondente(s) como poder ser visualizado através da Figura 32.

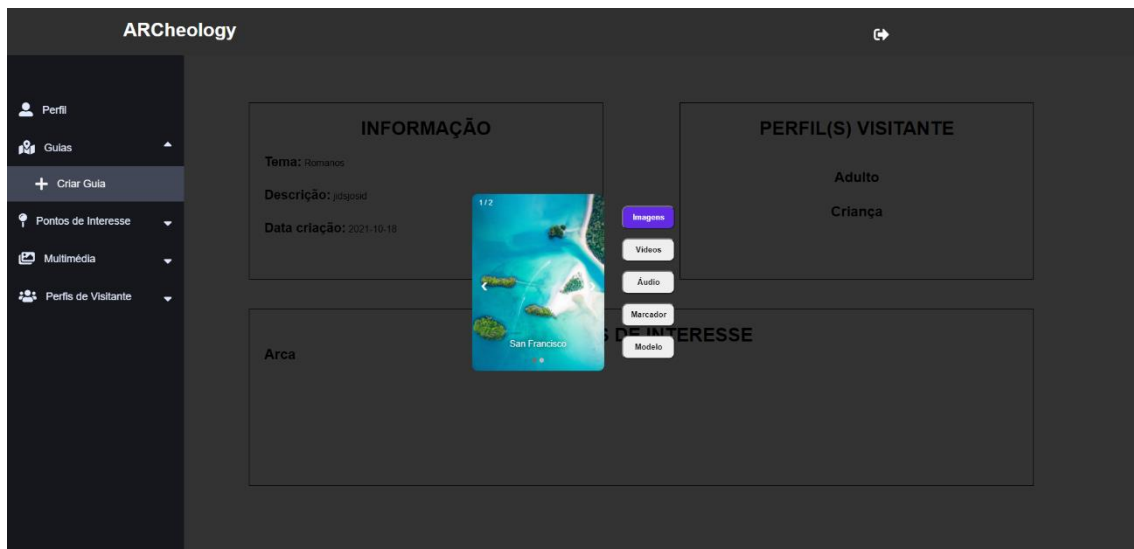


Figura 32 - Exemplo de imagem vista no modal

Se o ponto de interesse não possuir um determinado tipo de multimédia, imagens por exemplo, aparecerá uma mensagem a sinalizar tal acontecimento (ver Figura 33).



Figura 33 - Exemplo de Ponto de Interesse sem qualquer ficheiro de vídeo associado

4.4. Aplicação de Realidade Aumentada

Uma vez configurada a visita, a aplicação de realidade aumentada reúne os requisitos para poder começar a ser utilizada pelos Visitantes.

Como já foi explicado, após o descarregamento da aplicação e respetivo início da mesma, a única forma de iniciar a experiência é após a leitura de um código QR que contém informação acerca da visita em questão. No início deste capítulo, foram estudados dois modos diferentes de se proceder ao download da informação, optando-se por obter toda a informação de uma só vez, visto que não é dependente de WiFi no momento da visita em si, apenas da capacidade de armazenamento do dispositivo, extremamente elevada nos dias correntes. No final, caso o utilizador opte por desinstalar a aplicação do seu dispositivo, toda a informação presente em memória referente a uma visita é também eliminada.

Também é possível a um utilizador classificar a experiência realizada, bem como dar um breve *feedback* sobre a mesma, de forma que seja possível aos administradores do sistema ajustarem e melhorarem os guias de acordo com as sugestões e críticas dos visitantes.

4.4.1. Ficheiro de configuração

A escolha de um ficheiro de configuração responsável pelo armazenamento da informação referente a uma determinada visita deveu-se ao dinamismo conseguido pelo mesmo, ou seja, a partir de um único ficheiro consegue ser gerada toda a experiência de realidade aumentada, desde a localização dos pontos de interesse presentes até aos ficheiros multimédia associados a cada um. Assim sendo, o ficheiro de configuração destaca-se por ser o ponto chave no bom desempenho da solução proposta. Devido à familiaridade com ficheiros JSON e às inúmeras bibliotecas que o Unity possui de leitura deste tipo de ficheiros, optou-se por essa abordagem em detrimento de ficheiros XML.

De uma forma geral, a aplicação de RA tem de ser capaz de ler o conteúdo do ficheiro, procurando por palavras-chave de modo a dinamizar ao máximo a experiência.

De notar que a correta leitura do ficheiro é crucial para o funcionamento da aplicação, não podendo deixar ler ficheiros corrompidos ou inacabados.

Seguidamente, foi realizado um levantamento de todos os dados que deveriam constar no ficheiro de configuração, nomeadamente os pontos de interesse e todo o conjunto de ficheiros multimédia a eles associados.

No caso dos pontos de interesse, sabemos que a localização depende apenas da localização do seu respetivo marcador. Uma vez que a um ponto de interesse pode ou não estar associado um objeto 3D, foi indispensável criar dois pares no ficheiro de configuração relativamente a um marcador. O primeiro diz respeito à imagem utilizada para o *tracking* e o segundo ao próprio modelo. Visto que existem uma relação de dependência entre ambos, é essencial que o ficheiro de configuração contemple as duas entradas, mesmo que um ponto de interesse não possua um objeto 3D associado, ficando o respetivo valor vazio.

Sabemos também que cada ponto de interesse possui um ou vários ficheiros multimédia a si associados e que cada ficheiro multimédia tem o respetivo *url*, nome e descrição. Desta forma, o ficheiro de configuração necessita somente de um *array* de ficheiros multimédia para cada ponto de interesse, bem como do próprio *array* de pontos de interesse, como pode ser visualizado através da Figura 34.

```
{
  id_guia: 162,
  nome: "rerere",
  descrição: "ddfaffds",
  poi: [
    {
      nome: "Arca",
      descrição: "Sou uma arca",
      marker:
        "https://images.unsplash.com/photo-1534188753412-3e26d0d618d6?ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx86ixlib=rb-1.2.1&auto=format&fit=crop&w=3541&q=80",
      modelo:
        "https://raw.githubusercontent.com/AndreGoncaloLopes/Mybrary/main/cat.zip",
      media: {
        imagens: [
          {
            descrição: "ddsswe",
            url: "https://images.unsplash.com/photo-1611689342806-0863700ce1e4?ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx86ixlib=rb-1.2.1&auto=format&fit=crop&w=3541&q=80",
            fileName: "aguia.jpg",
          },
          {
            descrição: "jvlkvkjc",
            url: "https://images.unsplash.com/photo-1534188753412-3e26d0d618d6?ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx86ixlib=rb-1.2.1&auto=format&fit=crop&w=3541&q=80",
            fileName: "leao.jpg",
          },
        ],
        videos: [
          {
            descrição: "ddsafstrfd",
            url: "http://techslides.com/demos/sample-videos/small.mp4",
            fileName: "video.mp4",
          },
        ],
        audio: [
          {
            descrição: "gfjkgofg",
            url: "https://www.soundjay.com/bells/default01.wav",
          },
        ],
      },
    },
  ],
}
```

Figura 34 - Exemplo ficheiro de configuração

De notar que o mesmo ficheiro de configuração pode ser utilizado por diversos utilizadores, caso os interesses sejam os mesmos, uma vez que não existe qualquer

sistema de login por parte dos visitantes. Tal facto faz com que a experiência seja do tipo *plug and play*, consumida na hora, sem a necessidade de elementos intermédios entre a criação da visita e a própria visita.

4.4.2. Aplicação Móvel de Realidade Aumentada

A implementação da aplicação de RA propriamente dita no Unity será descrita de seguida. Para começar, toda a informação proveniente do ficheiro de configuração deve ser corretamente consumida pela aplicação, ou seja, antes mesmo de se realizar qualquer outra tarefa, a aplicação tem de ser capaz de reconhecer se o ficheiro se encontra corrompido.

De notar que todos os processos que ocorrem durante o tempo de vida útil da aplicação são realizados em *runtime*, ou seja, quaisquer ficheiros multimédia, marcadores, botões e modelos 3D existentes no ficheiro de configuração são gerados e consumidos na hora de utilização, o que faz com que a aplicação possa ser utilizada nos mais diversos museus e locais arqueológicos.

Durante as próximas secções referentes ao desenvolvimento da aplicação de realidade aumentada, irá ser primeiro exposto o fluxograma acerca do funcionamento das principais funções, seguido de partes de código explícitas. De acordo com a arquitetura pensada e planeada, dividiu-se a aplicação de RA em duas grandes cenas: A cena de rastreio de marcadores e a cena de visualização dos próprios marcadores.

Quando um visitante se encontra perante um marcador reconhecido pela aplicação, visto fazer parte da sua visita, o respetivo objeto 3D ou informação é exibido na tela, caso exista. De forma a dar um maior grau de autonomia aos visitantes, optou-se por se separar a informação sobre um objeto ou local do próprio modelo visualizado, uma vez que este último tem obrigatoriamente de se encontrar dentro do raio de ação da câmara do dispositivo. Tal facto faz com que o visitante não possua grande margem de manobra caso queira obter mais conhecimento acerca do ponto de interesse em questão. Com a separação da informação e do objeto 3D no mundo real é possível que um visitante veja o objeto dos mais variados ângulos com toda a estrutura que o rodeia e, caso queira saber mais sobre o mesmo, basta apertar um botão desenhado para o efeito, sempre visível durante o *tracking*.

4.4.2.1. Cena de Rastreo

Posto isto, será primeiro exibido o nó central do sistema (ver Figura 35) seguido pelas diferentes cenas individuais que compõem a aplicação e as suas respetivas funções.

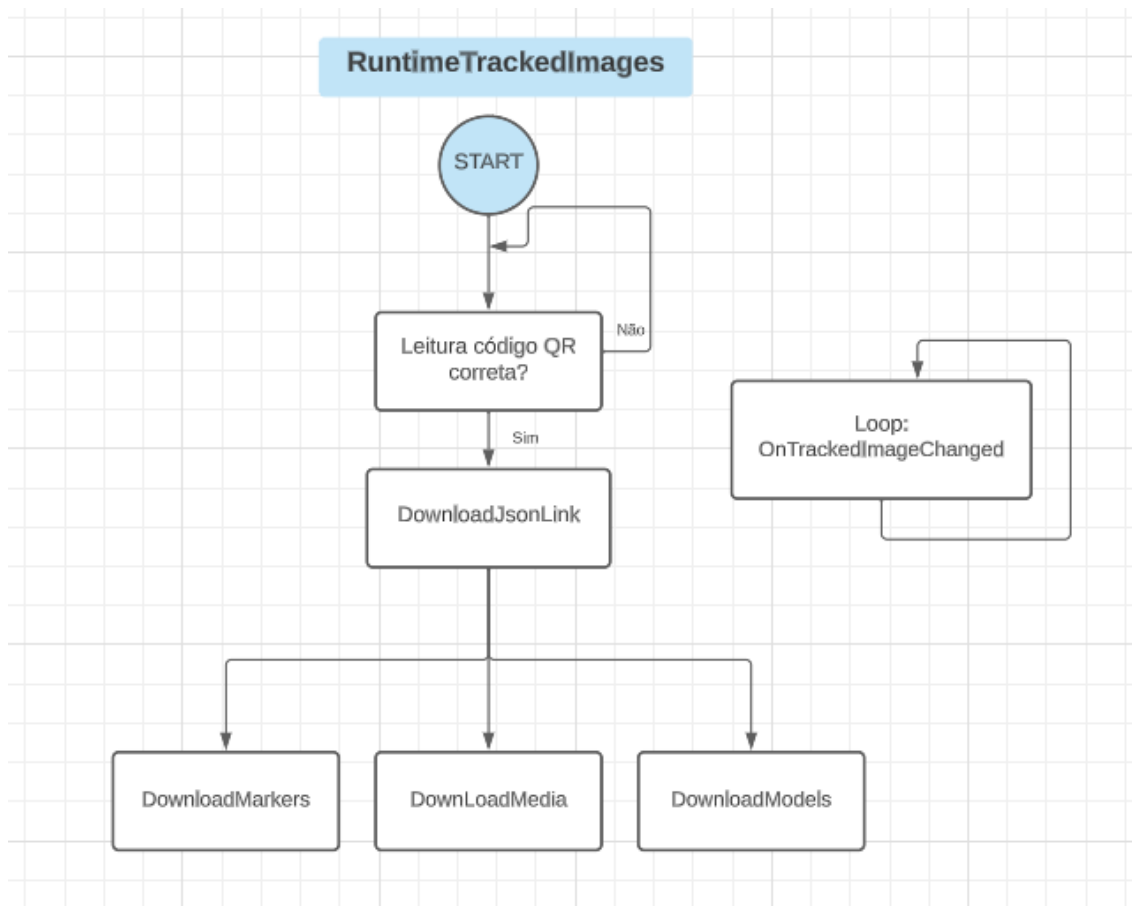


Figura 35 - Fluxograma da Cena de Rastreo

Através da visualização da Figura 35, de uma forma sucinta e numa primeira fase, a aplicação deve ser capaz de ler a informação proveniente do código QR de modo a processá-la mais tarde, fazer o download de todos os ficheiros multimédia escolhidos para o utilizador em questão, dos marcadores utilizados e dos respetivos objetos 3D, caso existam. Caso a leitura do código falhe é pedido novamente ao utilizador que o volte a reler até ser bem-sucedido. Este código QR contém apenas a *url* que fornece todo o ficheiro de configuração JSON posteriormente utilizado para a experiência em questão, encontrando-se armazenado no servidor. Desta forma, o utilizador encontra-se pronto para começar a experiência autonomamente.

Logo após a correta leitura do código o download do ficheiro de configuração é a próxima tarefa a ser executada, uma vez que toda a experiência de RA depende deste.

A aplicação começa por fazer o download do ficheiro através do *url* obtido da leitura do código, único parâmetro de entrada. Enquanto este processo não for terminado com sucesso, não é possível o avanço do código, visto tratar-se de uma operação síncrona (ver Figura 36).

```
private void DownloadJsonLink(string str)
{
    var www = new WWW(str);

    while (!www.isDone) { }

    var json:string = www.text;
    _guide          = JsonConvert.DeserializeObject<Guide>(json);
}
```

Figura 36 - Código responsável pelo download do ficheiro de configuração

Uma vez que o ficheiro se encontra no formato JSON, é necessário converter cada um dos pares chave-valor presentes na Figura 34 em funções autónomas recorrendo a classes de apoio, nas quais a própria classe corresponde à chave principal e as funções dentro de cada classe a cada um dos atributos. Para uma melhor compreensão, encontra-se representado graficamente na Figura 37 o parágrafo descrito.

```
public class Guide
{
    public int id_guia { get; set; }
    public string nome { get; set; }
    public string descricao { get; set; }
    Frequently called 10 usages
    public List<Poi> poi { get; set; }
}

1 usage AndreGoncaloLopes
public class Poi
{
    Frequently called 10 usages
    public string nome { get; set; }
    2 usages
    public string descricao { get; set; }
    1 usage
    public string marker { get; set; }
    3 usages
    public string modelo { get; set; }
    8 usages
    public Media media { get; set; }
}

1 usage AndreGoncaloLopes 2 exposing APIs
public class Media
{
    3 usages
    public List<Images> imagens { get; set; }
    3 usages
    public List<Videos> videos { get; set; }
    2 usages
    public List<Audio> audio { get; set; }
}
```

Figura 37 - classes responsáveis pela conversão dos pares chave-valor vindos do ficheiro de configuração

De seguida, a informação é totalmente carregada e armazenada, no caso dos ficheiros multimédia, marcadores e objetos 3D, em memória, de uma forma assíncrona. Tal abordagem pareceu a mais indicada, uma vez que nos dias correntes os dispositivos pessoais possuem cada vez mais uma elevada capacidade de armazenamento. Para além disso, no final de todos os *downloads*, o utilizador já não se encontra mais dependente de WiFi, mesmo que a aplicação vá abaixo, visto que todos os dados relativos à visita já se encontram guardados. Contudo, o tempo total de *download* pode ser um fator a considerar, caso o ficheiro de configuração possua uma enorme quantidade de ficheiros multimédia e modelos 3D.

No final da experiência, na eventualidade do utilizador querer uma nova visita, basta apagar a *cache* antes de ler um novo código de modo a eliminar da memória os ficheiros anteriores. Caso este não queira realizar outra visita após o término da anterior, eliminar a aplicação é suficiente para libertar da memória todos os ficheiros utilizados para tal.

O *download* de cada um dos diferentes ficheiros que compõem a visita possui, de uma certa forma, o mesmo fio condutor, ou seja, inicialmente é realizada a transferência destes recorrendo ao WiFi. Posteriormente, uma vez que os ficheiros já se encontram armazenados em memória, caso aconteça algum erro ou engano por parte do utilizador que force a aplicação a reiniciar, não é necessário que se volte a realizar o *download*, o que torna a aplicação por um lado mais rápida e por outro faz com que o possível problema do museu ou local não possuir WiFi seja mitigado e com que o utilizador não necessite de andar com os dados móveis ligados a todo o momento, gastando bateria e recursos. Para se compreender melhor cada uma destas, irá ser feita a separação entre elas, falando-se de cada uma individualmente.

4.4.2.1.1. *Download* dos marcadores

O diagrama da função responsável pelo *download* de todos os marcadores (ver Figura 38), bem como o código da própria função (ver Figura 39) serão apresentados de seguida.

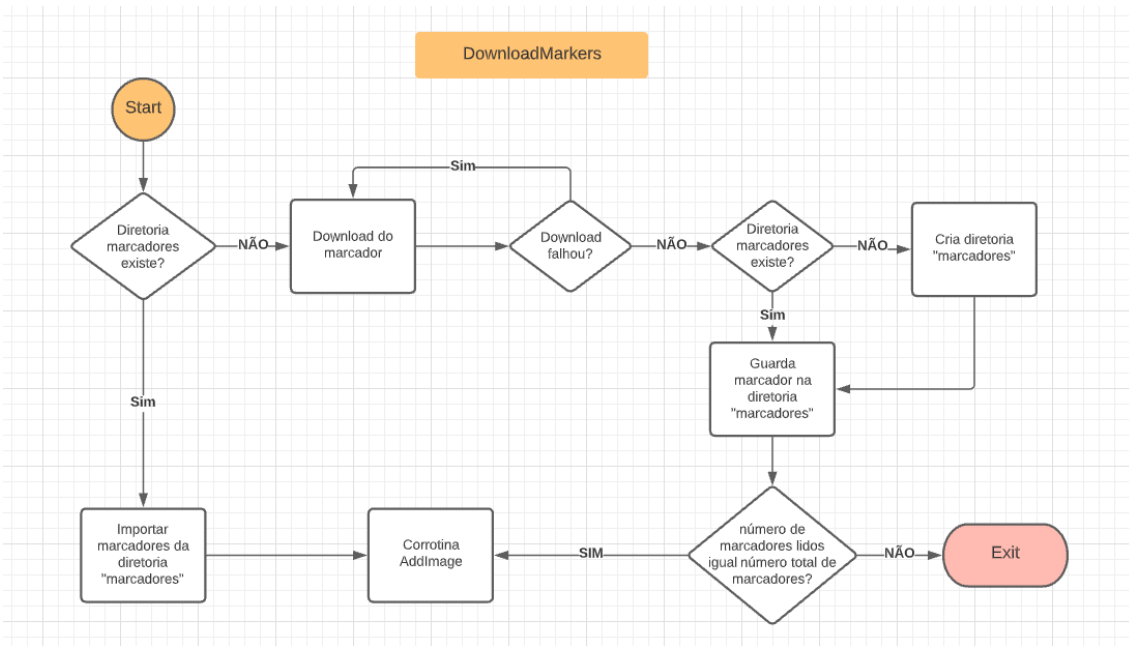


Figura 38 - Fluxograma da função DownloadMarkers

Na Figura 39 encontra-se apresentado parte do código responsável pelo download de todos os marcadores relativos a um guia de RA.

```

private void DownloadMarkers()
{
    if (Directory.Exists(path: Application.persistentDataPath + "/marcadores"))
    {
        foreach (var poi in _guide.poi)
        {
            _poiDesc.Add(poi.nome, poi.descricao);
        }
        var fileNames:string[] = Directory.GetFiles(path: Application.persistentDataPath + "/marcadores");
        foreach (var file:string in fileNames)
        {
            var fileSplit:string[] = file.Split(params separator: '/');
            var fileName:string = fileSplit[fileSplit.Length - 1];
            var uploadBytes:byte[] = File.ReadAllBytes(file);
            var texture = new Texture2D(width:0, height:0)
            {
                name = fileName.Split(params separator: '.')[0]
            };
            texture.LoadImage(uploadBytes);
            _texture2Ds.Add(texture);
        }
        StartCoroutine(routine: AddImage());
    }
    else
    {
        /*
         * download dos marcadores referentes a cada poi
         */
        foreach (var poi in _guide.poi)
        {
            _poiDesc.Add(poi.nome, poi.descricao);
            // Download de cada um dos markers
            StartCoroutine(routine: LoadMarkersFromWeb(url: poi.marker, fileName: $"{poi.nome}.jpg"));
        }
    }
}

```

Figura 39 - Código que carrega os marcadores de uma diretoria o que realiza o download dos mesmos

Caso a diretoria de marcadores já tenha sido criada e devidamente escrita em memória, cada um dos marcadores existentes é percorrido e criada uma textura com o nome do marcador. De seguida, inicia-se uma sub-rotina de forma a criar a biblioteca de marcadores em *runtime* e adicionar os marcadores lidos na mesma. Todo este processo executado em tempo real confere dinamismo à aplicação de RA. De notar que para um determinado modelo 3D aparecer na tela do dispositivo, tanto este como o respetivo marcador têm obrigatoriamente de possuir o mesmo nome.

No entanto, na primeira iteração do programa, mal o visitante inicia pela primeira vez a experiência, a diretoria de marcadores ainda não se encontra criada em memória. Neste caso, é inicialmente realizado o *download* de todos os marcadores presentes no ficheiro de configuração, criada a diretoria e, de seguida, executada a sub-rotina mencionada no parágrafo anterior aquando do *download* do último marcador, visualmente descrito na Figura 40.

```
private IEnumerator LoadMarkersFromWeb(string url, string fileName)
{
    var path = $"{_filePath}/marcadores"; // /marcadores/Arca

    var www:UnityWebRequest = UnityWebRequestTexture.GetTexture(url);
    yield return www.SendWebRequest();

    if (www.result == UnityWebRequest.Result.ConnectionError)
    {
        Debug.LogError(message: "Error: " + www.error);
        Debug.Log(message: "Por favor vá para uma área com cobertura Wifi");
        /* caso não haja wifi, volta-se a repetir o processo */
        StartCoroutine(routine: LoadMarkersFromWeb(url, fileName));
    }
    else
    {
        // Caso diretoria não exista no dispositivo é criada
        if (!Directory.Exists(path)) // /imagens"
        {
            Directory.CreateDirectory(path); // /imagens"
        }

        var loadedTexture = DownloadHandlerTexture.GetContent(www);
        // nome da imagem referência === nome do modelo 3D
        loadedTexture.name = fileName.Split(params separator: '.')[0];
        // guarda ficheiro numa pasta local, em memória
        var dataBytes:byte[] = loadedTexture.EncodeToJPG();
        File.WriteAllBytes(path: $"{path}/{fileName}", dataBytes);
        _texture2Ds.Add(loadedTexture);

        /* número de marcadores já lidos == número total de marcadores?
        Sim -> avança para a sub-rotina que cria a biblioteca em runtime
        Não -> continua o download dos marcadores em falta
        */
        if (_texture2Ds.Count != _guide.poi.Count) yield break;
        StartCoroutine(routine: AddImage());
    }
}
```

Figura 40 - Código responsável pelo Download dos marcadores vindos do servidor

4.4.2.1.2. Biblioteca de criação de marcadores em *runtime*

Logo após o *download* de todos os marcadores presentes no ficheiro de configuração, foi indispensável gerar a biblioteca em *runtime*, uma vez que só a partir desta é que as imagens operam efetivamente como marcadores. Estas são posteriormente lidas pela função de *tracking* que se encontra num *loop* infinito à procura de marcadores existentes na biblioteca de marcadores para ler. Através da criação da biblioteca de marcadores em tempo real consegue-se obter o dinamismo procurado ao longo do desenvolvimento deste projeto, uma vez que com isto a aplicação não se prende apenas a um único museu ou local arqueológico, mas pode ser utilizada em qualquer local sem restrições.

Através da Figura 41, verificamos que após a criação da biblioteca o código entra num *loop* que só termina quando todas as imagens tiverem sido lidas e interpretadas com sucesso. Dentro do ciclo, cada uma das imagens é transformada num marcador com o mesmo nome da imagem e com um tamanho fixo predefinido.



```
private IEnumerator AddImage()
{
    yield return null;

    var mutableRuntimeReferenceImageLibrary = _mTrackedImageManager.referenceLibrary as MutableRuntimeReferenceImageLibrary;

    foreach (var texture in _texture2Ds)
    {
        var job = mutableRuntimeReferenceImageLibrary.ScheduleAddImageWithValidationJob(texture, texture.name, widthInMeters: 0.22f);
        yield return new WaitUntil(() => job.jobHandle.IsCompleted);
    }

    if (mutableRuntimeReferenceImageLibrary != null)
    {
        Debug.Log(message: "LIBRARY COUNT: " + mutableRuntimeReferenceImageLibrary.count);
    }
}
```

Figura 41 - Código responsável por adicionar os marcadores à biblioteca de marcadores, criada e gerida em *runtime*

Por intermédio do fluxograma que se segue (ver Figura 42), compreendemos melhor o código visualizado anteriormente.

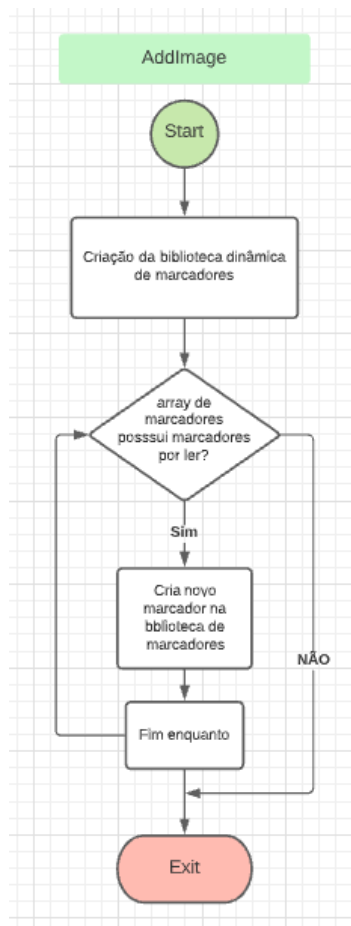


Figura 42 - Fluxograma da função AddImage

4.4.2.1.3. Download dos modelos 3D

O *download* dos modelos 3D que irão ser visualizados pelos visitantes apresenta a mesma estrutura do *download* de marcadores, ou seja, na primeira vez que o visitante entrar na aplicação a diretoria de modelos é criada e o *download* de todos os modelos é realizado recorrendo ao WiFi. Logo após, o utilizador não se encontra mais dependente de WiFi, podendo usufruir da experiência sem qualquer limitação. Este apresentou-se como um grande desafio, visto que para os modelos serem visualizados corretamente na tela do dispositivo foi necessário um conjunto enorme de operações realizadas em *back-end*, por trás dos panos. Sabendo que os modelos se encontram forçosamente no formato *obj*, recorreu-se ao uso de uma biblioteca disponível de forma completamente gratuita na própria loja do Unity para a leitura destes. Sabendo ainda que modelos em formato *obj* vêm normalmente acompanhados por um outro ficheiro de texturas, *mtl*, achou-se por bem que o conjunto modelo/texturas fosse lido do servidor como uma pasta zip maioritariamente devido ao tamanho que ocupa, dado que

um modelo 3D por muito minimalista que seja ocupa sempre uma quantidade razoável de espaço.

O fluxograma abaixo representado (ver Figura 43), representa de uma forma geral o fluxo dos acontecimentos na função encarregue de realizar o *download* de todos os modelos 3D contidos no ficheiro de configuração, *DownloadModels*. Como se consegue verificar, todas as operações realizam-se de uma forma síncrona, ou seja, para começar uma nova ação é necessário que a ação anterior tenha sido terminada.

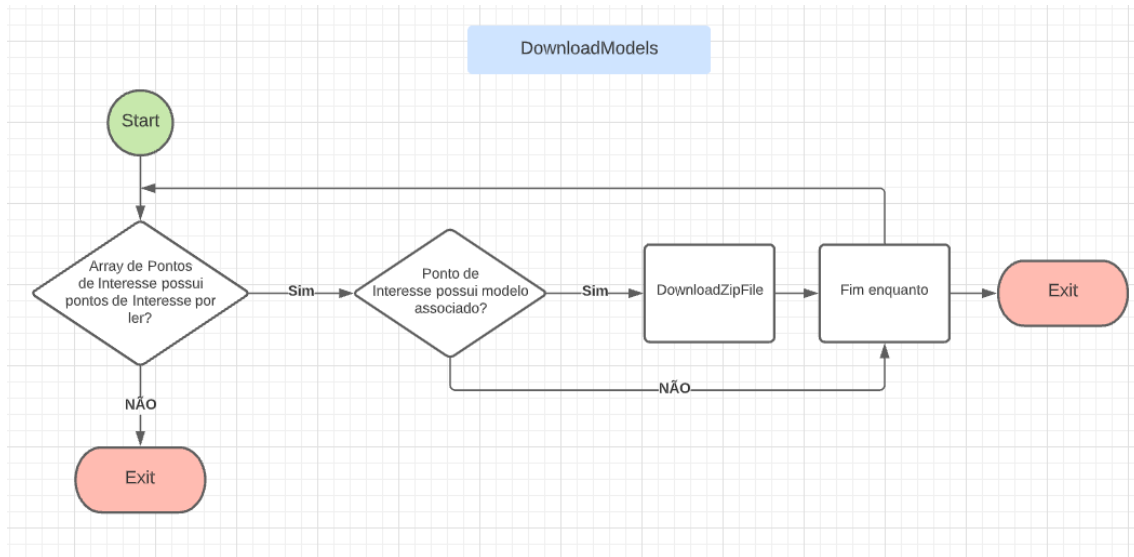


Figura 43 - Fluxograma da função *DownloadModels*

Como já foi explanado, um marcador pode ou não ter a si associado um modelo 3D. De forma a facilitar o *download* destes, a estrutura do ficheiro JSON relativamente aos modelos não se altera, independentemente de um marcador ter ou não um. Se tiver, a chave correspondente ao marcador vem com o *url* do ficheiro relacionado, caso contrário vem vazia. Tal facto pode ser confirmado através da visualização da Figura 44.

```

private void DownloadModels()
{
    foreach (var poi in _guide.poi)
    {
        // caso marcador não possua modelo associado
        if (string.IsNullOrEmpty(poi.modelo))
        {
            _pointsOfInterest.Add(poi.nome);
            continue;
        }
        var folderPath = $"{Application.persistentDataPath}/modelos/{poi.nome}";
        DownloadZipFile(url: poi.modelo, poi.nome, folderPath);
    }
}

```

Figura 44 - Código que itera cada um dos pontos de interesse e realiza o download dos mesmos

A partir da Figura 44 verificamos a existência de uma outra função que possui como parâmetros de entrada o *url* do modelo, o nome do modelo e a pasta onde o ficheiro zip irá ser guardado após download (ver Figura 45).

```

private void DownloadZipFile(string url, string poiName, string folderPath)
{
    // caso diretoria já exista em memória, apenas é realizado o load do modelo
    if (Directory.Exists(folderPath))
    {
        LoadModelFromZip(folderPath, poiName);
        return;
    }

    var www = new WWW(url);

    while(!www.isDone) {}

    // só entra aqui na primeira iteração dos Pontos de Interesse
    if (!Directory.Exists(folderPath))
    {
        Directory.CreateDirectory(folderPath);
    }

    // nome será dinâmico, de acordo com o nome do Ponto de Interesse
    var savePath = $"{folderPath}/{poiName}.zip";
    File.WriteAllBytes(savePath, www.bytes);
    Unzip(savePath, folderPath, poiName);
    LoadModelFromZip(folderPath, poiName);
}

```

Figura 45 - Código responsável por realizar o download dos modelos 3D

Esta opera de duas maneiras diferentes, tal como sucede com a diretoria de marcadores. Se esta já existir, significa que já foi posteriormente criada e foi realizado o

unzip do conteúdo do ficheiro *zip* vindo do servidor. Neste caso, é apenas efetuado o *load* do modelo por intermédio da biblioteca já mencionada. Por outro lado, se a diretoria de modelos ainda não existir, é primeiramente criada, o ficheiro *zip* é guardado em memória, realizado o *unzip* desse mesmo ficheiro e por último o *load* do modelo 3D. De modo a facilitar a compreensão deste último parágrafo, irá ser realizada a divisão do fluxo natural que ocorre durante o *download* dos modelos, expondo cada uma das funções intervenientes nesse processo.

O diagrama da Figura 46 auxilia na compreensão do parágrafo anterior.

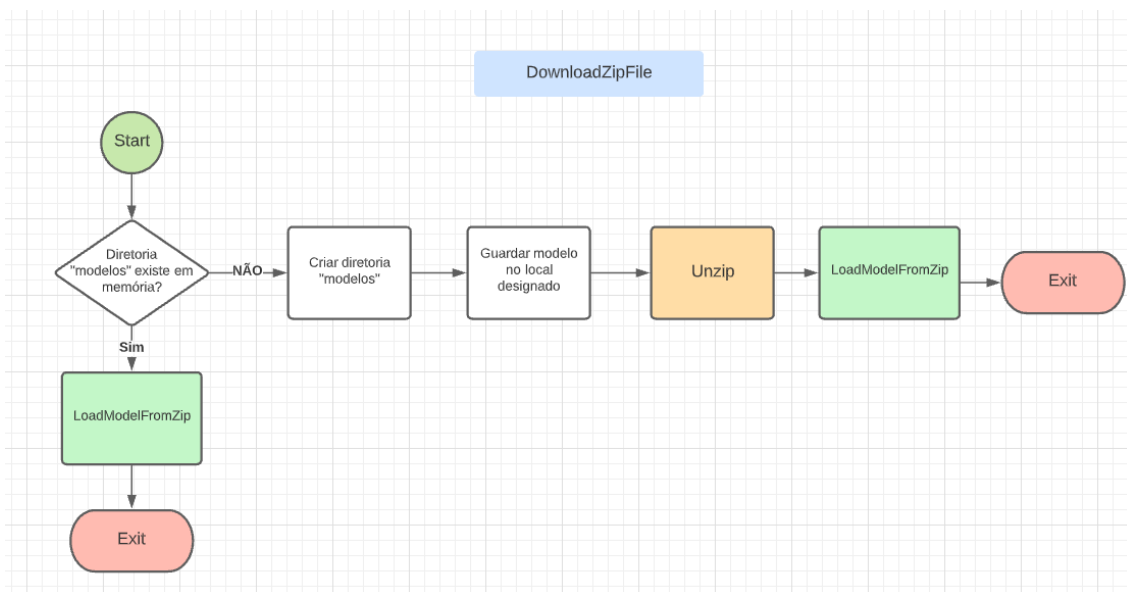


Figura 46 - Fluxograma da função *DownloadZipFile*

4.4.2.1.4. Carregamento dos modelos 3D

Na função anterior, constatamos que há duas formas de operar relativamente ao carregamento dos modelos. Tanto pode acontecer diretamente ou indiretamente através de funções de apoio que irão ser mencionadas mais à frente.

A função *LoadFromZip* aceita como parâmetros de entrada o caminho onde estão armazenados cada um dos ficheiros relativos aos modelos e o nome do ponto de interesse correspondente (ver Figura 47). Primeiramente é obtido o ficheiro *obj* da pasta que corresponde ao modelo 3D. Para isso é somente necessário obter todos os ficheiros contidos na pasta do modelo (*obj*, *mtl*, *jpg*, *png*) e procurar apenas o que possui a extensão *obj*, o próprio modelo. Logo após retorna-se o primeiro elemento encontrado visto que sabemos que cada pasta referente a cada objeto contém um e um só ficheiro do tipo *obj*. De seguida utiliza-se a biblioteca já existente de forma a processar o ficheiro, retornando o modelo como um *GameObject*. A partir deste momento o modelo já se encontra pronto para ser visualizado na tela do visitante, faltando só a correspondência modelo-marcador. Para isso, como já foi mencionado, o modelo tem de possuir obrigatoriamente o mesmo nome do seu marcador. De notar que tanto a posição, rotação e escala dos modelos não se encontram corretamente configuradas. Estas apresentaram-se como um enorme desafio relativo à visualização correta e adequada de cada um dos modelos.

```
private void LoadModelFromZip(string folderPath, string poiName)
{
    // retorna o ficheiro obj
    var objFilePath:string = Directory.GetFiles(folderPath, searchPattern: "*.obj")[0];

    /*
     * como o ficheiro obj e os ficheiros de textura se encontram na mesma pasta, apenas é
     * necessário passar como parâmetro o ficheiro obj, visto este fazer referência ao ficheiro
     * de texturas
     */
    var model:GameObject = new OBJLoader().Load(objFilePath);
    // modelo só irá ser ativo quanto se apontar a câmara para o respetivo marcador
    model.SetActive(false);
    model.name = poiName;
    model.transform.position = new Vector3(x:0, y:0, z:0);
    model.transform.Rotate(xAngle:180.0f, yAngle:180.0f, zAngle:180.0f);
    model.transform.localScale = new Vector3(x:0.007f, y:0.007f, z:0.007f);

    _placeablePrefabs.Add(model);
}
```

Figura 47 - Código responsável pelo carregamento para a tela de cada um dos modelos 3D

4.4.2.1.5. Unzip dos modelos 3D

Uma vez que os modelos vêm dentro de uma pasta *zipada*, foi necessário realizar o *unzip* da mesma antes de se poder realizar o carregamento dos modelos (ver Figura 48). A função responsável pelo *unzip* faz uso de uma biblioteca já existente de forma a copiar o conteúdo presente na pasta zip para a respetiva pasta normal. No final, após todos os ficheiros terem sido criados, a pasta *zip* é eliminada da memória. Após o *unzip* dos modelos, a função presente na Figura 47 é novamente invocada e termina assim a parte do código responsável pelo *download* de um modelo.

```
/*
 * savePath -> local do ficheiro zip específico de cada um dos modelos 3D
 * folderPath -> local onde se encontram armazenados cada um dos modelos 3D
 * poiName -> nome do Ponto de Interesse
 */
1 usage  AndreGoncaloLopes
private static void Unzip(string savePath, string folderPath, string poiName)
{
    using var s = new ZipInputStream(File.OpenRead(savePath));
    ZipEntry theEntry;
    while ((theEntry = s.GetNextEntry()) != null)
    {
        var fileName:string = Path.GetFileName(theEntry.Name);

        if (fileName == string.Empty) continue;

        using var streamWriter = File.Create(path: $"{folderPath}/{fileName}");
        var fData = new byte[2048];
        while (true)
        {
            var size:int = s.Read(fData, offset: 0, count: fData.Length);
            if (size > 0)
            {
                streamWriter.Write(fData, offset: 0, count: size);
            }
            else
            {
                break;
            }
        }
    }

    // nome dinâmico -> mesmo do ficheiro zip
    File.Delete(path: $"{folderPath}/{poiName}.zip");
}
```

Figura 48 - Código responsável pelo unzip do modelo 3D

4.4.2.1.6. Download dos ficheiros multimédia

Por último, mas não menos importante, foi necessário realizar o *download* de todos os ficheiros multimédia contidos no ficheiro de configuração. Mais uma vez, este

acontece concorrentemente com os outros dois mencionados nas secções 4.4.2.1.1. e 4.4.2.1.3.

Inicialmente, caso qualquer uma das diretorias de ficheiros multimédia já exista, significa que estas já foram criadas e o conteúdo presente em cada uma delas já se encontra disponível para utilização/visualização. Sabendo que tanto as imagens como os vídeos possuem descrições associadas de forma a dar a perceber ao visitante o que este se encontra a visualizar no momento, apenas é inserido num dicionário o par chave-valor, onde a chave corresponde ao nome do ficheiro sem a extensão e o valor à respetiva descrição (ver Figura 49).

```
private void DownloadMedia()
{
    if (Directory.Exists(path.Application.persistentDataPath + "/imagens") ||
        Directory.Exists(path.Application.persistentDataPath + "/videos") ||
        Directory.Exists(path.Application.persistentDataPath + "/audio"))
    {
        foreach (var poi in _guide.poi)
        {
            var poiName:string = poi.nome;

            var imagesDictionary = poi.media.imagens.ToDictionary(media:Images => media.fileName.Split(params separator:'.')[0], media:Images => media.descricao);
            _imagesDesc.Add(poiName, imagesDictionary);
            var videosDictionary = poi.media.videos.ToDictionary(video:Videos => video.fileName.Split(params separator:'.')[0], video:Videos => video.descricao);
            _videosDesc.Add(poiName, videosDictionary);
        }
    }
    else
    {
        DownloadPoiMedia();
    }
}
```

Figura 49 – Código responsável pelo mapeamento ou Download de cada um dos ficheiros multimédia presente no guia

De modo a facilitar a visualização e compreensão do código acima descrito, apresenta-se de seguida o fluxograma da própria função na Figura 50.

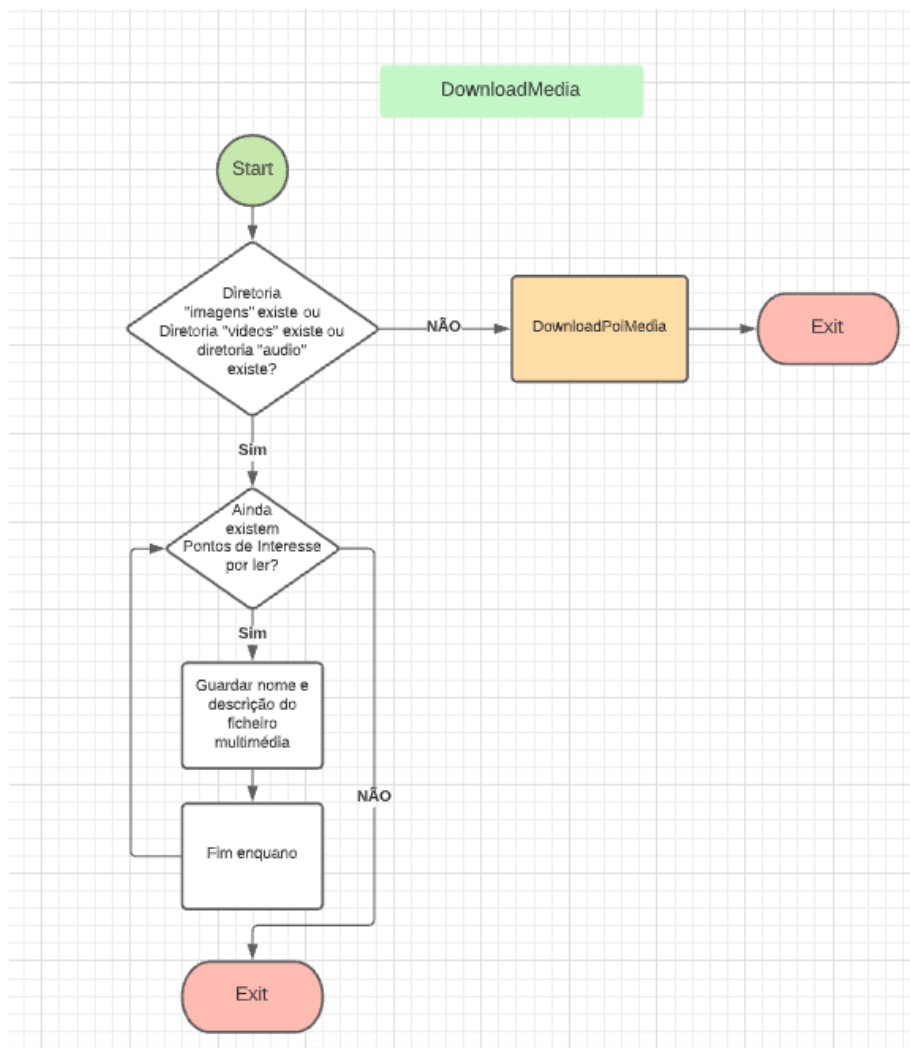


Figura 50 - Fluxograma da função DownloadMedia

Caso as diretorias de multimédia ainda não tenham sido criadas em memória é necessário realizar o *download* de cada uma destas, de forma assíncrono. De modo a aumentar a velocidade de processamento realiza-se apenas um único ciclo que itera cada um dos pontos de interesse presentes no ficheiro de configuração. De seguida, para cada um dos tipos de multimédia (imagens, vídeos, áudio), verifica-se se possuem de facto os respetivos tipos de multimédia. Desta forma o código torna-se dinâmico, conseguindo ler corretamente qualquer tipo de ficheiro de configuração, sem limitações. Tal facto pode ser visualizado e comprovada na Figura 51.

```

private void DownloadPoiMedia()
{
    foreach (var poi in _guide.poi)
    {
        var poiName:string = poi.nome; // nome do POI

        // Verificar se guia possui imagens referentes a um POI para mostrar
        if (poi.media.imagens.Count > 0)
        {
            var dictionary = new Dictionary<string, string>();
            // Download de todas as imagens referentes a um POI
            foreach (var media:Images in poi.media.imagens)
            {
                dictionary.Add(media.fileName.Split( params separator: '.' )[0], media.descricao);
                StartCoroutine(routine: LoadImageFromWeb(media.url, media.fileName, type: $"/imagens/{poiName}"));
            }
            _imagesDesc.Add(poiName, dictionary);
        }

        // Verificar se guia possui vídeos referentes a um POI
        if (poi.media.videos.Count > 0)
        {
            var dictionary = new Dictionary<string, string>();
            foreach (var video:Videos in poi.media.videos)
            {
                dictionary.Add(video.fileName.Split( params separator: '.' )[0], video.descricao);
                DownloadVideosFromWeb(video.url, video.fileName, type: $"/videos/{poiName}");
            }
            _videosDesc.Add(poiName, dictionary);
        }

        // Verificar se guia possui ficheiros audio referentes a um POI
        if (poi.media.audio.Count > 0)
        {
            foreach(var aud:Audio in poi.media.audio) {
                DownloadAudio(aud.url, aud.fileName, type: $"/audio/{poiName}");
            }
        }
    }
}

```

Figura 51 - Código responsável pelo download de cada um dos diferentes tipos de multimédia

Por meio do diagrama da Figura 52, observa-se ainda que é realizado o *download* de cada um dos ficheiros pertencentes ao tipo de multimédia correspondente. Todos estes são concorrentes e assíncronos, independentes uns dos outros.

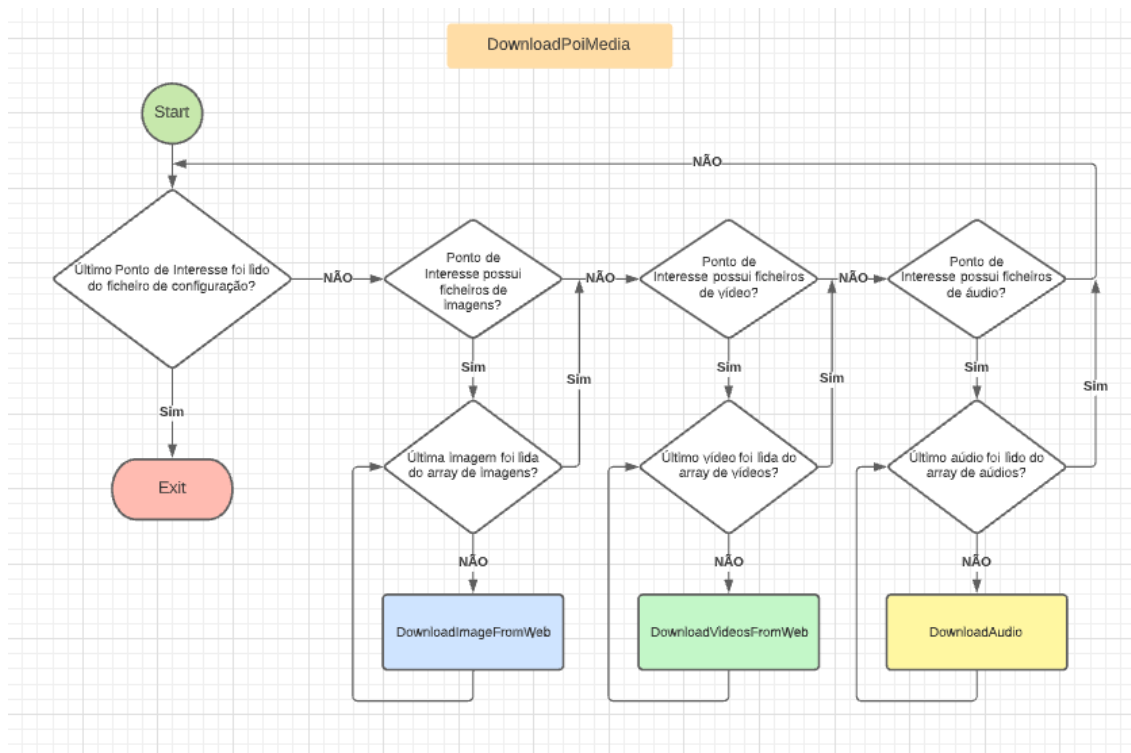


Figura 52 - Fluxograma da função DownloadPoiMedia

De seguida irão ser especificadas cada uma das funções próprias criadas para o *download* do conteúdo multimédia, presentes no fluxograma acima.

4.4.2.1.7. Download de imagens

O *download* das imagens existentes em cada ponto de interesse funciona de forma muito similar ao dos marcadores. Primeiramente cria-se uma sub-rotina de forma a não ter de se esperar que cada *download* termine antes que se comece o próximo. Deste modo a velocidade de *download* aumenta substancialmente.

A partir da Figura 53, verifica-se que a função aceita três parâmetros: o *url* da imagem, o nome da imagem e o tipo de ficheiro. O primeiro nada mais é do que o sítio onde se encontra armazenada a imagem, o segundo é o próprio nome da imagem sem a extensão e o último é o tipo de ficheiro, neste caso “imagem”, mais o nome do ponto de interesse em questão.

Primeiramente, obtém-se o local onde irão ser armazenadas todas as imagens, de acordo com o respetivo ponto de interesse. De seguida, na primeira vez que a aplicação é rodada, a diretoria de imagens é criada caso ainda não exista. Após o término do *download*, o nome da textura fica com o próprio nome do ficheiro de modo a facilitar posteriormente a sua visualização na tela. Após isso, a imagem é escrita no respetivo local da memória na forma de *array* de *bytes*. Este processo realiza-se tantas vezes quanto o número de imagens total presentes em todos os pontos de interesse.

```
private IEnumerator LoadImageFromWeb(string url, string fileName, string type)
{
    yield return StartCoroutine(routine: LoadTextureFromWeb(url, fileName, type));
}

Frequently called 2 usages AndreGoncalolopes +
private IEnumerator LoadTextureFromWeb(string url, string fileName, string type)
{
    var path = $"{_filePath}{type}/{fileName}"; // /imagens/Arca

    var www:UnityWebRequest = UnityWebRequestTexture.GetTexture(url);
    yield return www.SendWebRequest();

    if (www.result == UnityWebRequest.Result.ConnectionError)
    {
        Debug.LogError(message: "Error: " + www.error);
        Debug.Log(message: "Por favor vá para uma área com cobertura WiFi");
        /* caso não haja wifi, volta-se a repetir o processo */
        StartCoroutine(routine: LoadTextureFromWeb(url, fileName, type));
    }
    else
    {
        // Caso diretoria não exista no dispositivo
        if (!Directory.Exists(path: Application.persistentDataPath + type)) // /imagens"
        {
            Directory.CreateDirectory(path: Application.persistentDataPath + type); // /imagens"
        }
        var loadedTexture = DownloadHandlerTexture.GetContent(www);
        loadedTexture.name = fileName;
        var dataBytes:byte[] = loadedTexture.EncodeToJPG();
        File.WriteAllBytes(path, dataBytes);
    }
}
```

Figura 53 - Código responsável pelo download dos ficheiros de imagem

4.4.2.1.8. Download de vídeos

A função encarregue pelo *download* de ficheiros de vídeo apresenta a mesma estrutura das funções encarregues pelos *downloads* de imagens e áudios, ou seja, os parâmetros de entrada são exatamente os mesmos e representam os mesmos conceitos (ver Figura 54).

Inicialmente, caso a diretoria ainda não exista em memória, é criada. De seguida, o vídeo é guardado dentro da pasta apropriada e dá-se por terminada a função que realiza o *download* de cada um dos vídeos presentes em cada um dos pontos de interesse.

```
private void DownloadVideosFromWeb(string url, string fileName, string type)
{
    StartCoroutine(routine: DownloadVideo(url, fileName, type));
}

Frequently called 1 usage AndreGoncaloLopes *
private IEnumerator DownloadVideo(string url, string fileName, string type)
{
    var path = $"{_filePath}{type}/{fileName}"; // /marcadores/Arca/nome do poi sem extensão

    var www:UnityWebRequest = UnityWebRequest.Get(url);
    yield return www.SendWebRequest();

    if (www.result == UnityWebRequest.Result.ConnectionError)
    {
        Debug.Log(www.error);
    }
    else
    {
        // /videos/nome do poi
        if (!Directory.Exists(path:_filePath + type))
        {
            Directory.CreateDirectory(path:_filePath + type); // /videos/Arca
        }

        var data:byte[] = www.downloadHandler.data;
        File.WriteAllBytes(path, data);
    }
}
```

Figura 54 - Código responsável pelo download dos ficheiros de vídeo

4.4.2.1.9. Download de áudio

Por último são realizados os *downloads* relativos aos ficheiros de áudio. Mais uma vez, o fluxo natural do código segue o mesmo padrão dos anteriores. Inicialmente realiza-se o *download* de cada um dos ficheiros relativos aos ficheiros de áudio. De seguida, caso a diretoria de ficheiros não exista, esta é criada. Por fim o ficheiro resultante do *download* é guardado dentro da diretoria, com o respetivo nome (ver Figura 55).


```

private void DownloadAudio(string url, string fileName, string type)
{
    StartCoroutine(routine: DownloadAudioFromWeb(url, fileName, type));
}

Frequently called 1 usage AndreGoncaloLopes
private IEnumerator DownloadAudioFromWeb(string url, string fileName, string type)
{
    var www:UnityWebRequest = UnityWebRequestMultimedia.GetAudioClip(url, AudioType.MPEG);
    yield return www.SendWebRequest();

    if (www.result == UnityWebRequest.Result.ConnectionError)
    {
        Debug.LogError(message: "Error: " + www.error);
        Debug.Log(message: "Por favor vá para uma área com cobertura WiFi");
    }
    else
    {
        if (Directory.Exists(path: $"{_filePath}{type}")) yield break;
        Directory.CreateDirectory(path: $"{_filePath}{type}"); // /audio/Arca

        var clip:byte[] = www.downloadHandler.data;
        File.WriteAllBytes(path: $"{_filePath}{type}/{fileName}", clip);
    }
}

```

Figura 55 - Código responsável pelo download dos ficheiros de áudio

4.4.2.1.10. Leitura de marcadores

Durante todo o processo de *download* dos ficheiros multimédia e dos objetos 3D presentes no ficheiro de configuração, existe uma função que se encontra num ciclo infinito, à espera de marcadores para ler, *OnTrackedImageChanged*. Tais marcadores são lidos a partir da biblioteca criada de forma dinâmica, em *runtime*, o que significa que só é possível ler de forma eficaz um marcador após o *download* de todos os marcadores estar concluído. Desta forma, quando o utilizador se encontrar perante um qualquer marcador, esta função irá fazer o reconhecimento do mesmo recorrendo à biblioteca de marcadores para tal. Com isto, é possível a um museu ou local arqueológico possuir inúmeros marcadores, mas apenas os contemplados no ficheiro de configuração especificado para cada utilizador serem corretamente identificados.

A função *OnTrackedImageChanged* opera de três formas distintas, dependendo do tipo de operação realizada: adicionar, atualizar ou eliminar (ver Figura 56).

```
// loop infinito -> fica à espera da leitura de marcadores presentes na biblioteca de marcadores
Frequently called 2 usages AndreGoncaloLopes *
private void OnTrackedImageChanged(ARTrackedImagesChangedEventArgs eventArgs)
{
    // apenas invocada uma única vez quando uma nova imagem rastreada for encontrada
    foreach (var trackedImage in eventArgs.added)
    {
        ActivateTrackedObject(trackedImage.referenceImage.name);
    }

    /*
     * Verifica o estado de rastreamento e esconde o modelo 3D quando necessário.
     * Se o estado for limitado, significa que o rastreamento é fraco ou
     * a imagem de referência não se encontra na tela de visualização
     */
    foreach (var trackedImage in eventArgs.updated)
    {
        UpdateTrackedObject(trackedImage);
    }

    // esconde ou elimina o modelo 3D
    foreach (var trackedImage in eventArgs.removed)
    {
        Destroy(trackedImage.gameObject);
    }
}
```

Figura 56 - Modos de operação da função *OnTrackedImageChanged*

Na primeira vez que um novo marcador é rastreado com sucesso, ou seja, quando o utilizador aponta a câmara para um determinado marcador pela primeira vez durante o tempo de vida da aplicação, a função *ActivateTracked* é invocada, aceitando como único parâmetro o nome do marcador. Sabendo que um marcador pode ou não ter a si associado um modelo 3D, tal verificação é indispensável. Caso o marcador possua um modelo, este é mostrado na tela do visitante juntamente com o respetivo nome (ver Figura 57).

```
private void ActivateTrackedObject(string imageName)
{
    // dicionário nome-objeto contém chave = imageName?
    if (_spawnedPrefabs.ContainsKey(imageName))
    {
        _spawnedPrefabs[imageName].SetActive(true);
    }

    // nome do modelo/local
    _poiName[imageName].gameObject.SetActive(true);
}
```

Figura 57 - Código responsável por mostrar o modelo e o nome do mesmo

Após o primeiro reconhecimento do marcador com sucesso, todas as próximas vezes que o visitante apontar a câmara para esse mesmo marcador, a função *UpdateTrackedObject* é invocada (ver Figura 58).

```
private void UpdateTrackedObject(ARTrackedImage trackedImage)
{
    // nome do marcador em visualização
    var markerName:string = trackedImage.referenceImage.name;
    // caso câmara esteja a apontar corretamente para um marcador presente na biblioteca de marcadores
    if (trackedImage.trackingState == TrackingState.Tracking)
    {
        if (_spawnedPrefabs.ContainsKey(markerName))
        {
            _spawnedPrefabs[markerName].SetActive(true);

            var transform1 = trackedImage.transform;
            _spawnedPrefabs[markerName].transform.position = transform1.position;
            _spawnedPrefabs[markerName].transform.rotation = transform1.rotation;
        }
        _buttons[markerName].gameObject.SetActive(true);
        _poiName[markerName].gameObject.SetActive(true);

        _buttons[markerName].SetClickBehaviour(someMethodFromSomewhereElse: () => ViewDetails(markerName));
    }
    else
    {
        if (_spawnedPrefabs.ContainsKey(markerName))
        {
            _spawnedPrefabs[markerName].SetActive(false);
        }
        _buttons[markerName].gameObject.SetActive(false);
        _poiName[markerName].gameObject.SetActive(false);
    }
}
}
```

Figura 58 - Código responsável por mostrar ou omitir o modelo, nome e botão de detalhes

Através da visualização da figura, observamos que esta função possui dois modos de operação. Primeiramente, realiza-se o levantamento do nome do marcador em questão visto que será utilizado inúmeras vezes no decorrer da função. De seguida, verifica-se o estado de rastreo. Este pode assumir três valores distintos:

- Limitado – rastreo é fraco ou imagem de referência não se encontra na tela de visualização do dispositivo
- Inexistente – não existe qualquer rastreo
- Tracking – rastreo funciona de forma normal, mostrando modelo 3D caso exista

Tal verificação tem de ser obrigatoriamente realizada, visto que mal o marcador saia do raio de visão da câmara do dispositivo, tanto o modelo 3D como todos os componentes a ele associados devem desaparecer. Em suma, se o marcador estiver a ser corretamente interpretado pela câmara, o modelo aparecerá na tela, caso exista, assim como os respetivos componentes, à semelhança do que acontece na função *ActivateTrackedObject* (ver Figura 57). Mal o marcador deixe de aparecer na tela, tanto o modelo como os seus componentes deixam de ser visíveis. De notar que a passagem de visível para invisível é realizada de uma forma subtil e quase automática, tornando a experiência ainda mais agradável. Ainda nesta função, verifica-se o aparecimento de um botão de visualização que leva o visitante a uma nova cena, quando carregado (ver Figura 59). Tal botão leva o utilizador da aplicação para uma nova cena na qual é possível visualizar o objeto 3D, caso exista, com um maior nível de detalhe por meio de rotações e ampliações. Por outro lado, esta nova cena contém todos os ficheiros multimédia necessários para que o visitante ganhe conhecimento sobre o objeto ou local, com recurso à interatividade. Todos os botões gerados, bem como cada uma das imagens, vídeos, áudios ou texto são criados em *runtime* para que a experiência de RA seja o mais imersiva e real possível.

```
private void ViewDetails(string poiName)
{
    PoiInfo.PoiName = poiName;
    PoiInfo.PoiDesc = _poiDesc[poiName];
    PoiInfo.ImagesDesc = _imagesDesc;
    PoiInfo.VideosDesc = _videosDesc;
    PoiInfo.VideosDesc = _videosDesc;

    // primeira vez que a aplicação roda
    if (StaticGuideDetails.FirstTime)
    {
        StaticGuideDetails.FirstTime = false;

        StaticGuideDetails.PlaceablePrefabs = _placeablePrefabs;
        foreach (var model:GameObject in StaticGuideDetails.PlaceablePrefabs)
        {
            DontDestroyOnLoad(model);
        }
        StaticGuideDetails.PointsOfInterest = _pointsOfInterest;
        StaticGuideDetails.SpawnedPrefabs = _spawnedPrefabs;
        StaticGuideDetails.Buttons = _buttons;
        StaticGuideDetails.PoiName = _poiName;
        StaticGuideDetails.PoiDesc = _poiDesc;
        StaticGuideDetails.Texture2Ds = _texture2Ds;
        StaticGuideDetails.ImagesDesc = _imagesDesc;
        StaticGuideDetails.VideosDesc = _videosDesc;
    }

    SceneManager.LoadScene("PoiDetail");
}
```

Figura 59 - Código responsável por levar o utilizador para uma nova cena, após carregar no botão de detalhes

A função apresentada na Figura 59 aceita como único parâmetro de entrada o nome do ponto de interesse que se encontra a ser visualizado no momento. De seguida, toda a informação relevante relativa ao ponto de interesse é armazenada num conjunto de variáveis públicas, de forma a poderem ser acedidas mais tarde numa outra cena independente. Sabendo que existem duas cenas e que a segunda depende obrigatoriamente da primeira, de forma a acelerar o processo de transição entre ambas foi criada uma classe de apoio com variáveis públicas e estáticas. Deste modo, quando se volta da cena de visualização para a cena de rastreio não é necessário voltar a preencher qualquer tipo de mapa ou *array*, uma vez que estes já se encontram disponíveis para uso direto. Tal acontecimento diminui consideravelmente o tempo de processamento entre a mudança de cena, contribuindo para uma melhor experiência de utilização. Com o auxílio do comentário presente na Figura 60, constatamos tal afirmação.

```
private void Start()
{
    _mTrackedImageManager = gameObject.AddComponent<ARTrackedImageManager>();
    _mTrackedImageManager.referenceLibrary = _mTrackedImageManager.CreateRuntimeLibrary();
    _mTrackedImageManager.requestedMaxNumberOfMovingImages = _guide.poi.Count;

    _mTrackedImageManager.enabled = true;

    //DownloadJsonLink(_str);

    /*
     * quando volta a esta cena, após mudar de cena, não é necessário voltar a verificar
     * cada um dos ficheiros multimédia, marcadores e modelos, visto que eles foram posteriormente
     * armazenados em variáveis estáticas, o que aumenta a velocidade da aplicação
     */
    if (!StaticGuideDetails.FirstTime)
    {
        _placeablePrefabs = StaticGuideDetails.PlaceablePrefabs;
        _pointsOfInterest = StaticGuideDetails.PointsOfInterest;
        _spawnedPrefabs = StaticGuideDetails.SpawnedPrefabs;
        _buttons = StaticGuideDetails.Buttons;
        _poiName = StaticGuideDetails.PoiName;
        _poiDesc = StaticGuideDetails.PoiDesc;
        _texture2Ds = StaticGuideDetails.Texture2Ds;
        _imagesDesc = StaticGuideDetails.ImagesDesc;
        _videosDesc = StaticGuideDetails.VideosDesc;

        StartCoroutine(routine: AddImage());
        PopulateSpawnedPrefabs();
    }
    else {...}

    _mTrackedImageManager.trackedImagesChanged += OnTrackedImageChanged;
}
```

Figura 60 - Código responsável por verificar se utilizador trocou de cena

Termina assim a cena de rastreio. No próximo capítulo e subsequentes subcapítulos irá ser realizado o levantamento das funções mais importantes para o

correto funcionamento da cena de visualização, acompanhadas dos respetivos fluxogramas quando necessário.

4.4.2.2. Cena de Visualização

Para além da cena principal, na qual o visitante interage com os diferentes marcadores espalhados pelo local, é possível visualizar cada um dos artefactos ou locais com maior detalhe, bem como informação útil e relevante acerca deles recorrendo a uma outra cena especialmente pensada para o efeito. Tal cena dá ao visitante margem de manobra adicional, visto que não necessita de estar constantemente enquadrado com o marcador. Desta forma, o utilizador da aplicação consegue andar livremente de marcador em marcador enquanto ganha conhecimento relativo a um determinado ponto de interesse. Quando o visitante quiser terminar a visita por vontade própria terá a oportunidade de deixar uma crítica ou reclamação sobre o guia para si escolhido, de modo a proporcionar aos administradores do sistema *feedback* sobre o guia em questão, fazendo os respetivos ajustes caso necessário.

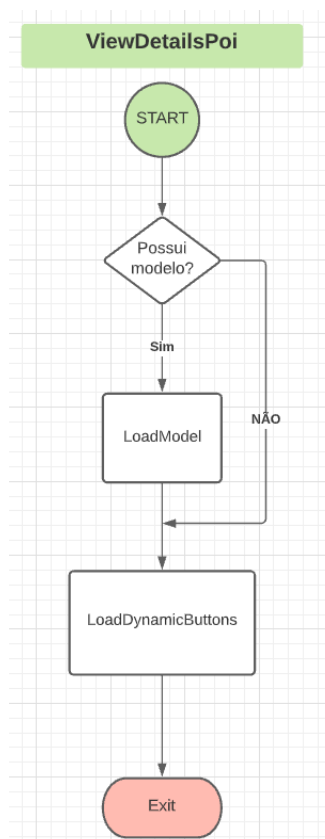


Figura 61 - Fluxograma da classe ViewDetailsPoi

Através do fluxograma representado na Figura 61 e da respetiva função, Figura 62, verificamos que a cena de visualização apresenta fundamentalmente duas funções bem delineadas. A primeira mostra o modelo 3D na tela do visitante caso o ponto de interesse o possua. A segunda diz respeito aos botões multimédia que irão ser exibidos na tela, de acordo com os ficheiros multimédia associados a um relativo ponto de interesse. Mais uma vez, um ponto de interesse não necessita de possuir todos os tipos de multimédia. Posto isto, cada uma destas duas funções irá ser caracterizada nas próximas secções.

```
private void Start()
{
    _menuButtonToggler = false;

    _name = PoiInfo.PoiName;
    _desc = PoiInfo.PoiDesc;

    _filePath = $"{Application.persistentDataPath}/modelos/{_name}";

    // se modelo existir
    if (Directory.Exists(_filePath))
    {
        LoadModel();
    }

    LoadDynamicButtons();
}
```

Figura 62 - Código inicial da Cena de Visualização

4.4.2.2.1. Carregamento do modelo 3D

Na secção 4.4.2.1.5. verificamos que cada um dos modelos é armazenado numa pasta própria com o respetivo nome, em memória. Através desta premissa fica fácil de se saber qual o modelo correto a carregar na cena de visualização. O carregamento do modelo é realizado recorrendo a uma simples função, *LoadModel*. Primeiramente, uma vez que já se sabe a diretoria certa onde se encontra o modelo, obtém-se o *url* completo deste e de seguida faz-se o *load* direto do modelo para a tela (ver Figura 63).

```

private void LoadModel()
{
    // url completo do modelo a ser visualizado
    var objFilePath :string = Directory.GetFiles(_filePath, searchPattern: "*.obj")[0];
    ResetWrapper();
    var loadedObject = new OBJLoader().Load(objFilePath);
    loadedObject.name = _name;
    loadedObject.transform.SetParent(_wrapper.transform);
}

```

Figura 63 - Código responsável pelo carregamento do modelo na Cena de Visualização

4.4.2.2.2. Carregamento dos botões multimédia

O carregamento dos botões multimédia realiza-se dinamicamente, visto que pontos de interesse não necessitam de possuir todos os tipos de multimédia. Desta forma, só se apresentam os botões que realmente fornecem informação útil ao utilizador (ver Figura 64).

```

private void LoadDynamicButtons()
{
    var media :Dictionary<string,bool> = CheckMedia();

    foreach (var type :KeyValuePair<string,bool> in media.Where(type :KeyValuePair<string,bool> => type.Value))
    {
        InstantiatePrefab(type.Key);
    }
}

```

Figura 64 - Código responsável pelo carregamento dinâmico dos botões de multimédia

Primeiramente faz-se a verificação dos tipos de multimédia que um ponto de interesse tem associado através de uma função auxiliar (ver Figura 65).


```

private Dictionary<string, bool> CheckMedia()
{
    // Imagens | Videos | Audio
    var dictionary = new Dictionary<string, bool>
    {
        {"imagens", false},
        {"videos", false},
        {"audio", false}
    };

    // Verificar imagens
    if (Directory.Exists(path: Application.persistentDataPath + "/imagens/" + _name))
    {
        var fileNames<string[]> = Directory.GetFiles(path: Application.persistentDataPath + "/imagens/" + _name);
        if (fileNames.Length > 0)
        {
            dictionary["imagens"] = true;
        }
    }

    // Verificar videos
    if (Directory.Exists(path: Application.persistentDataPath + "/videos/" + _name))
    {
        var videoNames<string[]> = Directory.GetFiles(path: Application.persistentDataPath + "/videos/" + _name);
        if (videoNames.Length > 0)
        {
            dictionary["videos"] = true;
        }
    }

    // Verificar áudio
    if (Directory.Exists(path: Application.persistentDataPath + "/audio/" + _name))
    {
        var audioNames<string[]> = Directory.GetFiles(path: Application.persistentDataPath + "/audio/" + _name);
        if (audioNames.Length > 0)
        {
            dictionary["audio"] = true;
        }
    }

    return dictionary;
}

```

Figura 65 - Código responsável por indicar os tipos de multimédia que um determinado ponto de interesse possui

Esta função retorna um dicionário que indica quais os três tipos de multimédia (imagens, vídeos, áudio) que fazem parte do ponto de interesse, de forma independente. Na hipótese de um tipo estar de facto associado, o botão correspondente a este é instanciado e mostrado na tela do dispositivo com o auxílio da função *InstantiatePrefab* (ver Figura 66).

```

private void InstantiarPrefab(string key)
{
    switch (key)
    {
        case "imagens":
            _imageButton = Instantiate(Images, _parentCanvas.transform);
            _imageButton.gameObject.SetActive(false);
            _imageButton.SetClickBehaviour(OnClickImageButton);
            _instanciados[0] = true;
            break;
        case "videos":
            _videoButton = Instantiate(Videos, _parentCanvas.transform);
            _videoButton.gameObject.SetActive(false);
            _videoButton.SetClickBehaviour(OnClickVideoButton);
            _instanciados[1] = true;
            break;
        case "audio":
            _audioButton = Instantiate(Audio, _parentCanvas.transform);
            _audioButton.gameObject.SetActive(false);
            _audioButton.SetClickBehaviour(OnClickAudioButton);
            _instanciados[2] = true;
            break;
    }
}

```

Figura 66 - Código responsável por instanciar os botões de multimédia na tela de visualização

A função representada na Figura 66 apenas instancia os botões de multimédia. No entanto, estes só serão realmente visíveis quando o utilizador premir o botão de visualização de todos os botões multimédia. Este botão apenas torna visíveis os restantes quando carregado, de forma a tornar mais limpa a cena de visualização.

De seguida, caso o visitante carregue num dos botões de multimédia, este irá despoletar um conjunto diferente de ações. Para que tal aconteça existem três funções independentes diretamente ligadas ao respetivo botão: *OnClickImageButton*, *OnClickVideoButton*, *OnClickAudioButton*.

Nas secções seguintes irão ser descritas cada uma destas funções mais detalhadamente.

4.4.2.2.3. Botão de imagens

Caso o visitante pretenda visualizar as imagens referentes ao ponto de interesse em questão, bem como uma breve descrição de cada, aparecerá na tela tanto a imagem como a descrição, por baixo. De forma a se conseguir visualizar a imagem seguinte é necessário apenas deslizar o dedo sobre a imagem para a direita. Para voltar para a imagem anterior, realiza-se o processo contrário. A descrição associada à imagem serve apenas para dar informação adicional acerca da imagem em questão.

De forma a dar a perceber a função responsável pela visualização das imagens e descrições na tela do visitante, *OnClickImageButton*, o diagrama apresentado na Figura 67 foi concebido.

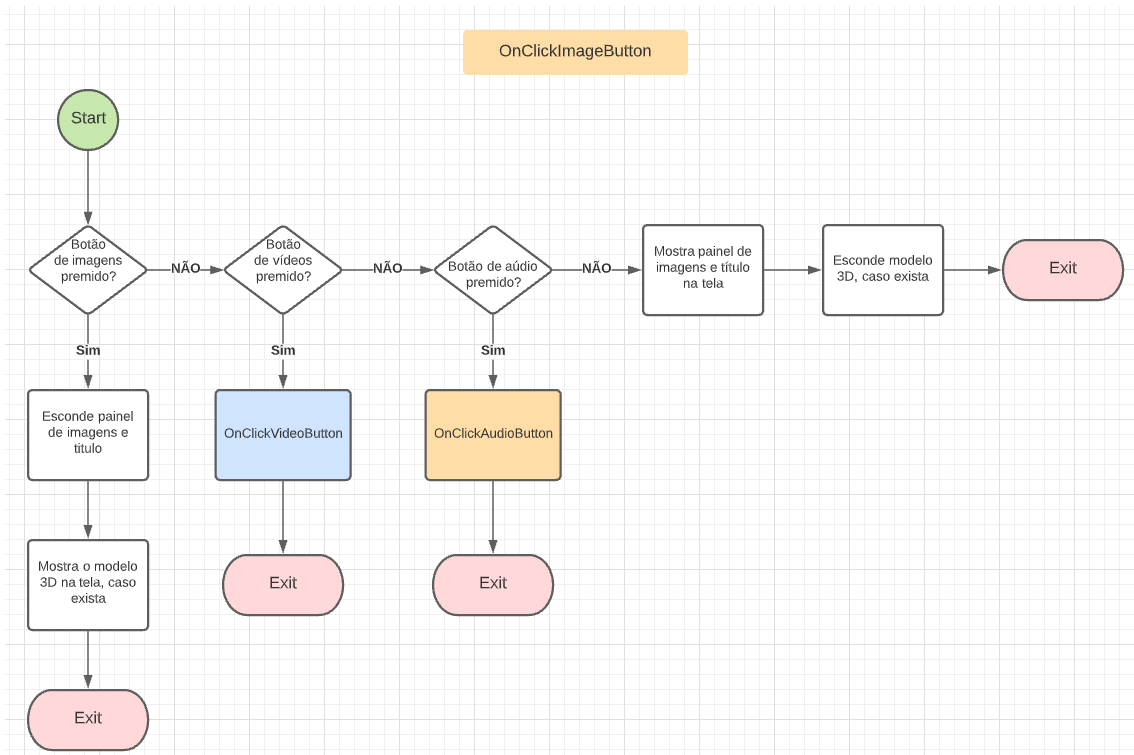


Figura 67 - Fluxograma da função *OnClickImageButton*

Através da Figura 68 podemos visualizar o código relativo à função *OnClickImageButton*.

```

private void OnClickImageButton()
{
    /*
     *   _toggler[0] -> imagens
     *   _toggler[1] -> videos
     *   _toggler[2] -> áudio
     *
     */

    /*
     *   caso utilizador já se encontre na tela de imagens e carregue nesta novamente,
     *   o painel desaparece e volta a reaparecer o modelo 3D, caso exista
     */
    if (_toggler[0])
    {
        _mainCamera.GetComponent<ImageSelector>()._levelHolder.SetActive(false);
        _mediaTitle.gameObject.SetActive(false);
        _wrapper.SetActive(true);
        _toggler[0] = false;
    }
    else
    {
        /*
         *   caso utilizador carregue num dos outros botões (video, áudio), painel muda
         *   para tipo de multimédia carregado
         */
        if (_toggler[1])
        {
            OnClickVideoButton();
        }

        if (_toggler[2])
        {
            OnClickAudioButton();
        }

        // ativa o painel de visualização de imagens
        _mainCamera.GetComponent<ImageSelector>()._levelHolder.SetActive(true);
        _mainCamera.GetComponent<ImageSelector>().enabled = true;
        _mediaTitle.text = "Imagens";
        _mediaTitle.gameObject.SetActive(true);
        // esconde o modelo 3D durante a visualização das imagens
        _wrapper.SetActive(false);
        _toggler[0] = true;
    }
}

```

Figura 68 - Código responsável por mostrar na tela do utilizador as imagens e a respetiva descrição relativas a um ponto de interesse

Inicialmente, quando o utilizador faz a mudança da cena de rastreio para a cena de visualização, o modelo 3D é imediatamente exibido, na eventualidade do ponto de interesse apresentar um. Quando o utilizador carrega no menu de multimédias, os diversos ícones associados ao respetivo tipo de multimédia são exibidos na tela.

No caso dos ficheiros de imagem, quando o utilizador decide carregar no botão relativo aos mesmos, é necessário verificar primeiramente se o utilizador já se encontra no painel de visualização das imagens, uma vez que quando carregado de novo este terá de desaparecer e o modelo 3D terá de voltar a aparecer em cena. Tal facto pode ser comprovado por intermédio do segundo comentário presente na Figura 68. Seguidamente, caso o utilizador decida carregar em qualquer um dos outros botões

disponíveis (vídeos e áudio), será mostrada uma nova *interface* de acordo com o botão premido. Por último, foi apenas necessário ativar o *script* referente aos ficheiros multimédia e logo após esconder da cena o modelo 3D por forma a não se sobrepor ao painel de imagens.

4.4.2.2.4. Classe responsável pelo painel de imagens

De forma a tornar o código mais legível e limpo optou-se pela separação entre a parte que controla o painel de imagens/descrições da classe geral da cena de visualização. Assim existe um maior controlo sobre cada uma das classes e caso seja necessário modificar código é mais fácil de se saber qual classe modificar.

Posto isto, a função inicial (*Start*) da classe responsável pelo carregamento das imagens e das respetivas descrições apenas inicializa algumas variáveis que irão ser necessárias futuramente (ver Figura 69).

```
// Start is called before the first frame update
Event function AndreGoncaloLopes *
private void Start()
{
    _levelHolder.SetActive(true);
    _textures = new List<Texture2D>();
    _teste = new Dictionary<int, string>();
    /*
     * número de imagens existentes para um determinado
     * Ponto de Interesse -> dinâmico
     */
    _numberOfImages = NumberOfImages();

    // carregamento dos diferentes painéis, de acordo com o número de imagens
    LoadPanels();
}
```

Figura 69 - Função inicial da classe que carrega as imagens e as respetivas descrições

Através da visualização da figura observa-se que existe uma variável, *_numberOfImages*, que armazena o número total de imagens que um determinado ponto de interesse possui recorrendo a uma função de apoio, *NumberOfImages*. Com isto, o código torna-se totalmente dinâmico.

O código da função de apoio pode ser visualizado através da Figura 70.

```
/*  
 * função que retorna o número total de um Ponto de Interesse  
 */  
#usage AndreGoncaloLopes *  
private int NumberOfImages()  
{  
    // totalidade das imagens referentes a um Ponto de Interesse  
    var fileNames:string[] = Directory.GetFiles(path: Application.persistentDataPath + "/imagens/" + _poiName);  
    var i = 0;  
  
    // para cada ficheiro de imagem presente na directoria de imagens  
    foreach (var file:string in fileNames)  
    {  
        var fileSplit:string[] = file.Split(params separator: '/');  
        var fileName:string = fileSplit[fileSplit.Length - 1].Split(params separator: '.')[0];  
        var uploadBytes:byte[] = File.ReadAllBytes(file);  
        var texture = new Texture2D(width: 0, height: 0);  
        texture.LoadImage(uploadBytes);  
        // armazena textura (imagem) lida num array de texturas  
        _textures.Add(texture);  
        // array que armazena a descrição de cada uma das imagens iteradas  
        _teste.Add(i++, fileName);  
    }  
    return fileNames.Length;  
}
```

Figura 70 - Função de apoio que retorna o número total de imagens relativas a um Ponto de Interesse específico

De seguida, seguindo o fluxo do código da Figura 69, deparámo-nos com uma outra função, *LoadPanels*. Esta é responsável por instanciar cada um dos diferentes painéis que compõe a cena de visualização de imagens. Um painel nada mais é do que o conjunto da imagem e da respetiva descrição. Quantas mais imagens existirem, mais painéis serão instanciados. De forma a se perceber melhor o que foi escrito, apresentase na Figura 71 a respetiva representação visual.

```
private void LoadPanels()  
{  
    var panelClone:GameObject = Instantiate(_levelHolder);  
    /*  
     * PageSwiper -> classe de apoio que permite ao utilizador realizar o swipe tanto  
     * para a direita, como para a esquerda. Dependente do número total de imagens  
     */  
    var pageSwiper = _levelHolder.AddComponent<PageSwiper>();  
    pageSwiper.TotalPages = _numberOfImages;  
  
    // número total de painéis == número total de imagens  
    for (var i = 1; i <= _numberOfImages; i++)  
    {  
        var panel:GameObject = Instantiate(panelClone, _canvas.transform, worldPositionStays: false);  
        var scrollArea:GameObject = Instantiate(_scrollArea, _canvas.transform, worldPositionStays: false);  
        panel.transform.SetParent(_levelHolder.transform);  
        panel.name = "Panel -" + i;  
        scrollArea.transform.SetParent(panel.transform);  
        scrollArea.name = "ScrollArea -" + i;  
        panel.GetComponent<RectTransform>().localPosition = (Vector3)new Vector2(Screen.width * (i - 1), 0);  
        // child -> componente que possui a descrição acerca de cada imagem visualizada na tela  
        var child = scrollArea.transform.Find("TextContainer").Find("Text").gameObject.GetComponent<Text>();  
  
        LoadImages(panel, i, child);  
    }  
    Destroy(panelClone);  
}
```

Figura 71 - Código responsável por instanciar cada um dos painéis

Com recurso à Figura 71 constatamos que a função *LoadPanels* invoca uma outra função, *LoadImages*. Decidiu-se separar o carregamento das imagens propriamente ditas do restante código pois o conteúdo presente na última função diz somente respeito às imagens. Desta forma, não se misturam duas ideias distintas.

A função *LoadImages* aceita três parâmetros de entrada: o painel no qual a imagem irá assentar, o índice do painel e o campo de texto para a descrição (ver Figura 72).

```
private void LoadImages(GameObject parentObject, int index, Text child)
{
    /* NOME ASSOCIADO À TEXTURA */
    var imageName:string = _teste[index - 1];
    /* DESCRIÇÃO ASSOCIADA À TEXTURA */
    var imageDesc:string = _poiImagesDesc[imageName];

    var image:GameObject = Instantiate(_image, _canvas.transform, worldPositionStays:false);
    image.GetComponent<Image>().sprite = Sprite.Create(_textures[index - 1], new Rect(0f, 0f, _textures[index - 1].width, _textures[index - 1].height), Vector2.zero);
    image.transform.SetParent(parentObject.transform, worldPositionStays:false);
    child.text = imageDesc;
    image.name = "Image -" + imageName;
}
```

Figura 72 - Código responsável por instanciar as imagens e a respetiva descrição

Para cada painel criado, é necessário primeiramente obter o nome da imagem, utilizado posteriormente para obter a descrição da mesma, uma vez que cada imagem possui uma descrição associada. De seguida, procedeu-se à instanciação da imagem propriamente dita recorrendo ao *array* de texturas presente na Figura 70. Este possui todas as imagens que fazem referência ao ponto de interesse que se encontra a ser visualizado. Foi apenas necessário encontrar nesse mesmo *array* a imagem correta através do índice passado como parâmetro de entrada. A descrição alusiva à textura (já obtida), foi posteriormente associada à componente de texto já existente, mas outrora em branco.

De modo a se compreender melhor o parágrafo acima retratado, irão ser expostas referências visuais no capítulo de Testes.

4.4.2.2.5. Botão de vídeos

A visualização de cada um dos vídeos referentes a um ponto de interesse acontece quase da mesma forma da visualização das imagens. Estes também podem apresentar uma breve descrição sobre o que o visitante está a assistir no momento, sendo o visitante capaz de deslizar para o próximo ou anterior com um simples toque de dedo.

Por meio do diagrama da Figura 73 percebemos que a função responsável por mostrar as imagens na tela (ver Figura 67) é idêntica em todos os aspetos.

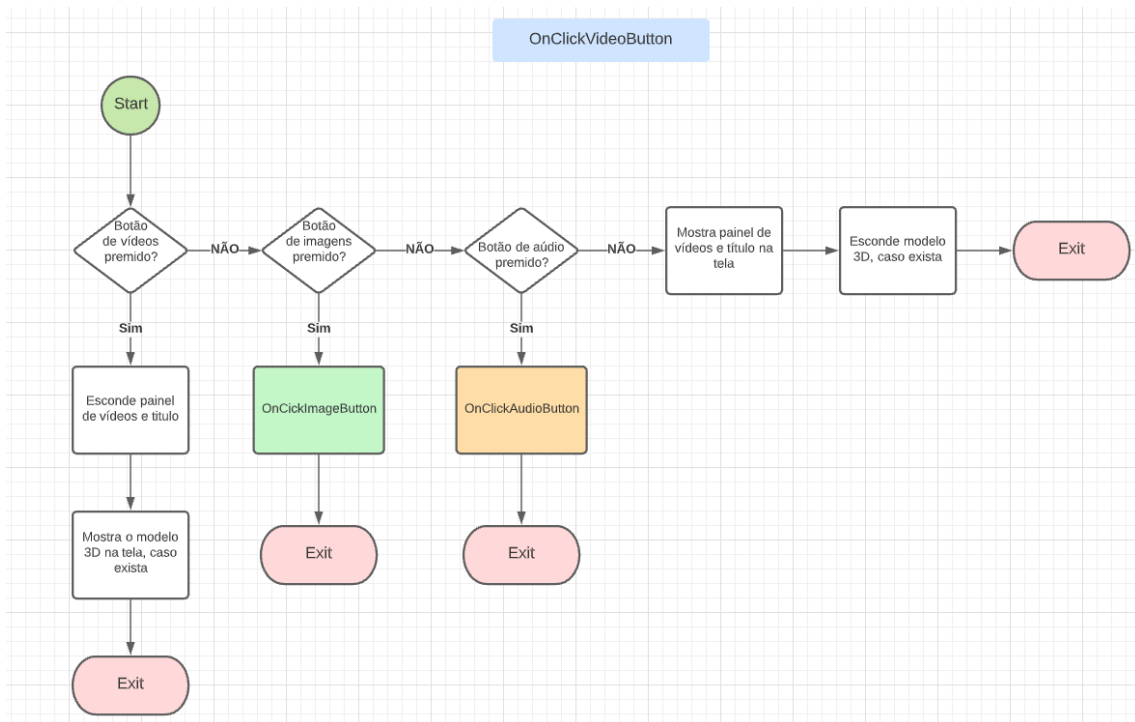


Figura 73 - Fluxograma da função OnClickVideoButton

Passando o fluxograma para código (ver Figura 74), percebemos que em relação à função da Figura 68 não existem, mais uma vez, diferenças significativas, alterando-se apenas campos referentes a imagens por campos referentes a vídeos. No caso de ficheiros de vídeo o painel de visualização dos ficheiros e das respetivas descrições é ativo, sobrepondo-se assim uma nova camada de sobre a cena de visualização.


```

private void OnClickVideoButton()
{
    /*
    *   _toggler[0] -> imagens
    *   _toggler[1] -> videos
    *   _toggler[2] -> áudio
    *
    */

    /*
    *   caso utilizador já se encontre na tela de vídeos e carregue nesta novamente,
    *   o painel desaparece e volta a reaparecer o modelo 3D, caso exista
    */
    if (_toggler[1])
    {
        _mainCamera.GetComponent<VideoSelector>()._videosPanel.SetActive(false);
        _mediaTitle.gameObject.SetActive(false);
        _wrapper.SetActive(true);
        _toggler[1] = false;
    }
    else
    {
        /*
        *   caso utilizador carregue num dos outros botões (imagens, áudio), painel muda
        *   para tipo de multimédia carregado
        */
        if (_toggler[0])
        {
            OnClickImageButton();
        }
        if(_toggler[2]) {
            OnClickAudioButton();
        }

        // ativa o painel de visualização de vídeos
        _mainCamera.GetComponent<VideoSelector>()._videosPanel.SetActive(true);
        _mainCamera.GetComponent<VideoSelector>().enabled = true;
        _mediaTitle.text = "Vídeos";
        _mediaTitle.gameObject.SetActive(true);
        // esconde o modelo 3D durante a visualização dos vídeos
        _wrapper.SetActive(false);
        _toggler[1] = true;
    }
}

```

Figura 74 - Código responsável por mostrar na tela do utilizador os vídeos e a respetiva descrição relativos a um ponto de interesse

4.4.2.2.6. Classe responsável pelo painel de vídeos

Quando um utilizador decide visualizar os ficheiros de vídeo que dizem respeito ao ponto de interesse que se encontra a visualizar através de um botão criado para o efeito, um novo painel entra em cena. Este encontra-se dividido em dois componentes: o próprio vídeo e a descrição associada ao mesmo. Na eventualidade do vídeo não possuir descrição, o campo desta aparecerá vazio. O botão de visualização de ficheiros de vídeo apresenta como função principal a de ativar o *script* relativo ao carregamento dos vídeos e descrições.

Inicialmente foi necessário recolher informação acerca de algumas variáveis essenciais de forma a ser possível atingir o tão desejado dinamismo, nomeadamente o número total de ficheiros de vídeo que um determinado ponto de interesse possui (ver Figura 75).

```
// Start is called before the first frame update
Event function AndreGoncaloLopes *
private void Start()
{
    // lista que contém o url completo dos vídeos
    _videos = new List<string>();
    /* dicionário que contém o index e o respetivo nome do vídeo, por ordem
       de aparição na diretoria
    */
    _videosName = new Dictionary<int, string>();
    // número total de vídeos presentes na diretoria
    _numberOfVideos = NumberOfVideos();

    // carregamento dos painéis de visualização do conjunto vídeo/descrição
    LoadPanels();
}
```

Figura 75 - Primeira função a ser invocada mal o utilizador carrega no botão de visualização de vídeos

Analisando o código da figura acima constata-se que a função *Start*, invocada antes de qualquer outra, cria um *array* que guarda o *url* completo de cada um dos vídeos e um dicionário que faz a associação entre o *index* e o nome do vídeo. Por último verifica-se ainda a existência de duas funções, *NumberOfVideos* e *LoadPanels*. A primeira é responsável por retornar o número total de vídeos existentes na diretoria, bem como preencher tanto o *array* como o dicionário. A segunda é responsável por realizar o carregamento de cada um dos painéis que contempla o vídeo e a respetiva descrição. Existirão tantos painéis quanto o número de vídeos.

A função *NumberOfVideos* para além de retornar a totalidade dos vídeos presentes na diretoria específica, preenche também o *array* com os *url* de cada um dos vídeos e o dicionário que associa um índice ao nome do vídeo de acordo com a ordem de leitura dos vídeos. A partir da Figura 76 obtemos uma representação visual do que foi anteriormente escrito.

```
private int NumberOfVideos()
{
    // array que contém todos os ficheiros de vídeo presentes na diretoria específica
    var fileNames :string[] = Directory.GetFiles(path: Application.persistentDataPath + "/videos/" + _poiName);
    var i = 0;

    // iteração sobre cada um dos ficheiros
    foreach (var file:string in fileNames)
    {
        var fileSplit :string[] = file.Split( params separator: '/' );
        var fileName :string = fileSplit[fileSplit.Length - 1].Split( params separator: '.' )[0];
        _videos.Add(file);
        _videosName.Add(i++, fileName);
    }
    return fileNames.Length;
}
```

Figura 76 - Função que retorna o número total de vídeos presentes numa diretoria específica

A outra função mencionada, *LoadPanels* (ver Figura 77), possui como finalidade instanciar os painéis que por sua vez possuem um ficheiro de vídeo acompanhado da descrição, caso exista. Assim sendo, o número total de painéis é diretamente proporcional ao número de ficheiros de vídeo contidos na diretoria. Como já foi referido, um painel nada mais é do que o conjunto do vídeo e da devida descrição.

```

private void LoadPanels()
{
    var panelClone:GameObject = Instantiate(_videosPanel);
    /*
    * PageSwiper -> classe de apoio que permite ao utilizador realizar o swipe tanto
    * para a direita, como para a esquerda. Depende do número total de imagens
    */
    var pageSwiper = _videosPanel.AddComponent<PageSwiper>();
    pageSwiper.TotalPages = _numberOfVideos;

    // número total de painéis == número total de vídeos
    for (var i = 1; i <= _numberOfVideos; i++)
    {
        var panel:GameObject = Instantiate(panelClone, _canvas.transform, worldPositionStays: false);
        var scrollArea:GameObject = Instantiate(_scrollArea, _canvas.transform, worldPositionStays: false);
        panel.transform.SetParent(_videosPanel.transform);
        panel.GetComponent<RectTransform>().localPosition = (Vector3) new Vector2(x:Screen.width * (i - 1), y:0);
        panel.name = "Panel-" + i;
        scrollArea.transform.SetParent(panel.transform);
        scrollArea.name = "ScrollArea -" + i;
        var renderTexture = Instantiate(_custom);
        var image = Instantiate(_rawImage, panel.transform, worldPositionStays: false);
        image.texture = renderTexture;
        // child -> componente que possui a descrição acerca de cada vídeo visualizado na tela
        var child = scrollArea.transform.Find("TextContainer").Find("Text").gameObject.GetComponent<Text>();
        LoadVideos(panel, i, renderTexture, child);
    }
    Destroy(panelClone);
}

```

Figura 77 - Código responsável pelo carregamento dos painéis de visualização dos vídeos

No final do código presente na figura repare-se que é invocada uma outra função, *LoadVideos* (ver Figura 78), encarregue de carregar cada um dos vídeos e de associar a referente descrição para a tela do dispositivo, de modo individual.

```

private void LoadVideos(GameObject parentObject, int index, RenderTexture renderTexture, Text child)
{
    /* nome do vídeo */
    var videoName:string = _videosName[index - 1];
    /* Descrição associada ao vídeo */
    var videoDesc:string = _poiVideosDesc[videoName];

    var video:GameObject = Instantiate(_video, parentObject.transform, worldPositionStays: false);
    video.GetComponent<VideoPlayer>().url = _videos[index - 1];
    video.GetComponent<VideoPlayer>().targetTexture = renderTexture;

    video.transform.SetParent(_canvas.transform, worldPositionStays: false);
    video.transform.SetParent(parentObject.transform);
    child.text = videoDesc;
    video.name = "Video-" + videoName;
}

```

Figura 78 - Código responsável por carregar vídeo e respetiva descrição para a tela do dispositivo

4.4.2.2.7. Botão de áudio

Por último um ponto de interesse pode possuir na sua constituição ficheiros de áudio. No entanto, de forma a facilitar a implementação destes mesmos ficheiros decidiu-se que um ponto de interesse disporá apenas de um único ficheiro de áudio.

Uma vez mais, o fluxo da função responsável por mostrar o ficheiro de áudio na tela, *OnClickAudioButton*, segue exatamente o mesmo princípio dos botões de imagens e vídeos (ver Figura 79).

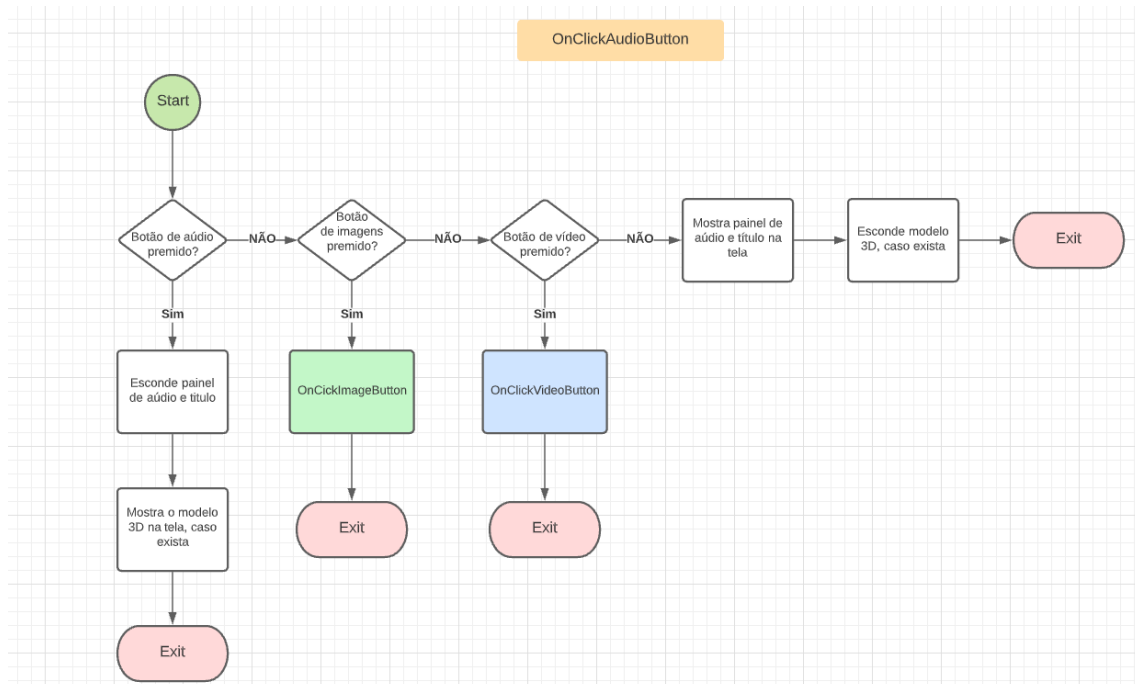


Figura 79 - Fluxograma da função *OnClickAudioButton*

Sabendo que o código é praticamente o mesmo no que diz respeito ao funcionamento da função apresenta-se de seguida apenas a função do diagrama da figura acima representada em formato de código (ver Figura 80).

```

private void OnClickAudioButton()
{
    /*
    *   _toggler[0] -> imagens
    *   _toggler[1] -> videos
    *   _toggler[2] -> audio
    *
    */

    /*
    *   caso utilizador já se encontre na tela de áudio e carregue nesta novamente,
    *   o painel desaparece e volta a reaparecer o modelo 3D, caso exista
    */
    if (_toggler[2])
    {
        _audioPanel.SetActive(false);
        _mediaTitle.gameObject.SetActive(false);
        _wrapper.SetActive(true);
        _toggler[2] = false;
    }
    else
    {
        /*
        *   caso utilizador carregue num dos outros botões (imagens, videos), painel muda
        *   para tipo de multimédia carregada
        */
        if (_toggler[0])
        {
            OnClickImageButton();
        }
        if (_toggler[1])
        {
            OnClickVideoButton();
        }

        // ativa o painel de visualização de áudio
        _audioPanel.SetActive(true);
        _mediaTitle.text = "Audio";
        _mediaTitle.gameObject.SetActive(true);
        // esconde o modelo 3D enquanto no painel de áudio
        _wrapper.SetActive(false);
        _toggler[2] = true;
    }
}

```

Figura 80 - Código responsável por mostrar na tela do utilizador o áudio relativo ao ponto de interesse em visualização

No caso dos ficheiros áudio a classe de apoio apresenta um papel diferente das demais. Enquanto as outras duas se prendem na instanciação dos painéis com as imagens/vídeos e as descrições, esta realiza apenas o *download* do ficheiro áudio. Assim, foi criado um simples painel com três botões distintos que agem conforme o *input* do utilizador: Play, Pause e Stop. Estes três botões encontram-se presentes em qualquer vídeo / áudio que se veja ou oiça, tornando-os universais.

Logo após o visitante carregar no botão de áudio, aparecerão na tela os três botões acima referidos podendo o visitante interagir com os mesmos. Primeiramente é obtido o *url* do *clip* de áudio associado ao ponto de interesse através da diretoria com o nome do ponto de interesse (ver Figura 81).

```
private void LoadAudioClip()
{
    // url do ficheiro de áudio
    // exemplo -> Application.persistentDataPath/audio/Arca/teste.mp3
    var objFilePath :string = Directory.GetFiles( path: Application.persistentDataPath + "/audio/" + _poiName)[0];
    StartCoroutine( routine: LoadMusic(objFilePath));
}
```

Figura 81 - Código responsável por obter o clip de áudio e invocar a sub-rotina de carregamento do mesmo

De seguida, com recurso à sub-rotina *LoadMusic* realiza-se o carregamento do ficheiro áudio para o *clip* de áudio através de um conjunto de ações e funções predefinidas, passando como único parâmetro de entrada o *url* do ficheiro de áudio (ver Figura 82).

```
private IEnumerator LoadMusic(string songPath)
{
    using var uwr :UnityWebRequest = UnityWebRequestMultimedia.GetAudioClip(uri: "file://" +songPath, AudioType.MPEG6);
    ((DownloadHandlerAudioClip)uwr.downloadHandler).streamAudio = true;

    yield return uwr.SendWebRequest();

    if (uwr.result == UnityWebRequest.Result.ConnectionError)
    {
        Debug.LogError(uwr.error);
        yield break;
    }

    var dlHandler = (DownloadHandlerAudioClip)uwr.downloadHandler;

    if (dlHandler.isDone)
    {
        var audioClip = dlHandler.audioClip;

        if (audioClip != null)
        {
            _audioSource.clip = DownloadHandlerAudioClip.GetContent(uwr);
        }
    }
}
```

Figura 82 - Código responsável pelo carregamento do ficheiro de áudio para o clip de áudio

Por último, de forma a terminar o painel de áudio, atribui-se a cada um dos botões funções próprias que despoletam a respetiva ação associada (ver Figura 83).

```

1 asset usage AndreGoncaloLopes
public void OnPlay()
{
    _audioSource.Play();
}

1 asset usage AndreGoncaloLopes
public void OnPause()
{
    _audioSource.Pause();
}

1 asset usage AndreGoncaloLopes
public void OnStop()
{
    _audioSource.Stop();
}

```

Figura 83 - Funções de controlo do painel de áudio

Para finalizar a cena de visualização, foi criado um botão de retorno para a cena de rastreio (ver Figura 84).

```

public void OnClickBack()
{
    StartCoroutine( routine: WaitBeforeLeave());
}

Frequently called 1 usage AndreGoncaloLopes *
private static IEnumerator WaitBeforeLeave()
{
    yield return new WaitForSeconds(.3f);
    // volta para a cena de rastreio
    SceneManager.LoadSceneAsync("RealTimeImageTracking");
}

```

Figura 84 - Código responsável por trocar a cena de visualização pela cena de rastreio

Capítulo 5. Avaliação do Sistema

No presente capítulo será realizada a avaliação do sistema proposto. Primeiramente será efetuado um levantamento do *hardware* empregue durante todo o período de execução dos testes práticos no Convento de São Francisco de Real. Estes serviram para testar de uma forma prática as funcionalidades do sistema criado.

5.1. *Hardware* utilizado

Nesta secção irá ser identificado todo o *hardware* utilizado no momento da elaboração dos testes práticos de avaliação ao sistema proposto. Assim sendo, os dispositivos utilizados foram:

- Computador portátil: LENOVO 81FV com processador i7-8750H, 16 GB de memória RAM e placa gráfica Nvidia GEFORCE GTX 1050.
- *Smartphone*: Moto g9 Power com processador Snapdragon 662, 4 GB de memória RAM e câmara com 64MP.

O Unity apresenta-se como uma aplicação que consome uma quantidade significativa de memória. Visto que o trabalho desenvolvido manifesta algum grau de complexidade foi necessário possuir *hardware* compatível com este de forma a proporcionar uma experiência de programação suave e agradável. O dispositivo Android por sua vez foi escolhido conforme a lista de dispositivos suportados pelo AR Core.

5.2. Preparação do teste

A aplicação de realidade aumenta propriamente dita necessita apenas dos diferentes marcadores para funcionar corretamente. Inicialmente foi realizada a recolha dos tamanhos reais dos modelos e da melhor posição para assentar os marcadores durante a realização dos testes práticos.

Os diferentes marcadores que compõe um sítio arqueológico nada mais são do que códigos QR relativos ao modelo em questão, ou seja, caso o modelo seja uma arca o código QR correspondente será o da arca. Uma vez que estes são quadrados, a simetria e orientação do modelo nos mesmos fica mais estável (ver Figura 85). A rotação e

direção do modelo 3D está diretamente ligada com a rotação e orientação do próprio marcador pelo que foi necessário ajustar o marcador até que a representação espacial do modelo fosse a mais próxima da realidade.



Figura 85 - Exemplo de marcador

Após a escolha dos marcadores, procedeu-se com a leitura do código QR. Uma vez que a aplicação web não se encontrava *online*, o código QR utilizado como caso de estudo veio diretamente das imagens do google (ver Figura 86). De lembrar que o código apenas revela informação sobre o local onde se encontra armazenado o próprio ficheiro de configuração.

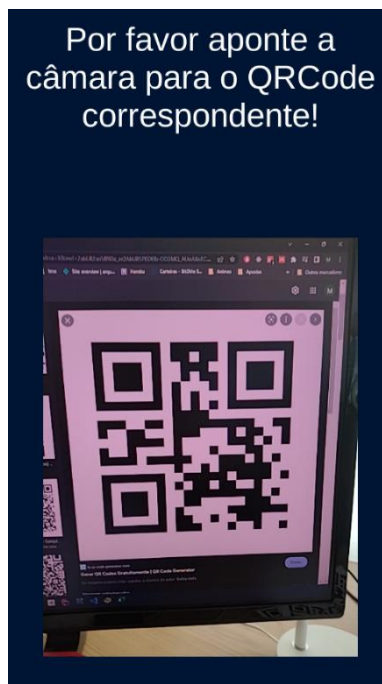


Figura 86 - Leitura código QR na aplicação de RA

De seguida o utilizador depara-se com uma *loading screen* enquanto espera pelo *download* dos marcadores, modelos e ficheiros multimédia (ver Figura 87). O tempo que este processo demora a concluir varia de acordo com os parâmetros já referidos.

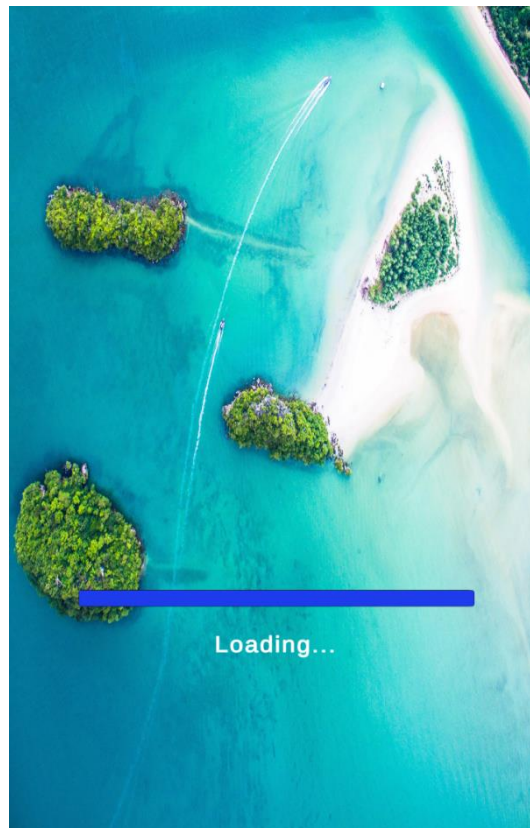


Figura 87 - Loading screen na aplicação de RA

Como forma de testar a aplicação de RA, esta possui o ficheiro de configuração *hardcoded*, ou seja, o ficheiro já se encontra preenchido manualmente *a priori*, tornando o código rígido e estático. Tendo tal facto em consideração, todos os *url* dos modelos, marcadores e ficheiros multimédia já se encontram embutidos diretamente na aplicação, sem qualquer *input* externo (ver Figura 88).

```

// Exemplo JSON retornado através da leitura de um QRCode
private const string Str = @"
{
  'id_guia': 162,
  'nome': 'rerere',
  'descricao': 'ddfgffds',
  'poi': [
    {
      'nome': 'Arca',
      'descricao': 'Sou uma arca',
      'marker': 'https://raw.githubusercontent.com/AndreGoncaloLopes/Mybrery/main/arca_qrcode.png',
      'modelo': 'https://raw.githubusercontent.com/AndreGoncaloLopes/Mybrery/main/Arca_v5.zip',
      'media': {
        'imagens': [
          {
            'descricao': 'ddsswe',
            'url': 'https://images.unsplash.com/photo-1611689342806-0863700ce1e4?ixid=MnwMjA3F0B8MHxwa690by1wYWd1fHx8fGVuZD88fHx86ixlib=rb-1.2.1&auto=format&fit=crop&w=1935&q=80',
            'fileName': 'aguia.jpg'
          },
          {
            'descricao': 'jvlkvkjc',
            'url': 'https://images.unsplash.com/photo-1534188753412-3e26dd618d6?ixid=MnwMjA3F0B8MHxwa690by1wYWd1fHx8fGVuZD88fHx86ixlib=rb-1.2.1&auto=format&fit=crop&w=687&q=80',
            'fileName': 'leao.jpg'
          }
        ],
        'videos': [
          {
            'descricao': 'ddsafstrfd',
            'url': 'http://techsLides.com/demos/sample-videos/small.mp4',
            'fileName': 'video.mp4'
          }
        ],
        'audio': [
          {
            'descricao': 'audio da arca',
            'url': 'https://raw.githubusercontent.com/AndreGoncaloLopes/Mybrery/main/beep.mp3',
            'fileName': 'teste.mp3'
          }
        ]
      }
    }
  ]
}
";

```

Figura 88 - Exemplo ficheiro de configuração retornado pela leitura do código QR

Na Figura 88 verificamos, a partir do ficheiro de configuração, que a visita possui apenas um único ponto de interesse (Arca) e que este por sua vez contempla os três tipos de multimédia: imagens, vídeos e áudio. Observando agora estes três tipos verificamos que existem dois ficheiros de imagens, um ficheiro de vídeo e um ficheiro de áudio associados ao ponto de interesse.

A aplicação de RA encontra-se assim pronta a ser utilizada em contexto real.

5.3. Realização do teste

Para a realização dos testes no Convento foi necessário numa primeira fase estudar o local, escolhendo a melhor localização para cada um dos diferentes marcadores. No caso prático desta dissertação, o teste foi realizado com recurso a dois modelos, chafariz e arca. No entanto, numa situação da vida real, todos os modelos pertencentes a um determinado local arqueológico terão o respetivo marcador colocado no local onde outrora se encontrava a peça original de forma a dar ao visitante uma representação mais precisa do passado.

De notar que durante a realização dos testes os marcadores não foram pousados num local, mas sim segurados. Tal aconteceu devido a um problema de escala que será mencionado numa secção mais adiante.

Posto isto, o primeiro local escolhido para teste foi um claustro que continha no seu centro um chafariz. Este foi um dos pontos destacados como local para um dos marcadores e, conseqüentemente, para respetiva visualização do chafariz com recurso à tecnologia de RA (ver Figura 89).



Figura 89 - Modelo do chafariz visualizado na aplicação de RA

Visualizando a Figura 89, observa-se que o chafariz não se encontra bem no centro do marcador. Tal problema será falado mais adiante na discussão dos resultados e futuras melhorias. Observam-se ainda três botões distintos: dois do lado esquerdo e um no canto superior direito. O botão mais acima do lado esquerdo possibilita ao utilizador sair da aplicação caso queira. O botão logo abaixo apresenta informação acerca do local visitado, quando premido. Por último, o terceiro botão realiza a troca de cenas, da Cena de Rastreio (4.4.2.1.) para a Cena de Visualização (4.4.2.2.).

Na cena de visualização o utilizador é capaz de estudar o modelo ao pormenor, caso exista. Pode ainda aplicar rotações, *zooms* e translações sobre o mesmo de forma a conseguir ver mais detalhadamente a textura e os pormenores adjacentes (ver Figura 90 e ver Figura 91).



Figura 90 - Chafariz visto de frente na aplicação de RA

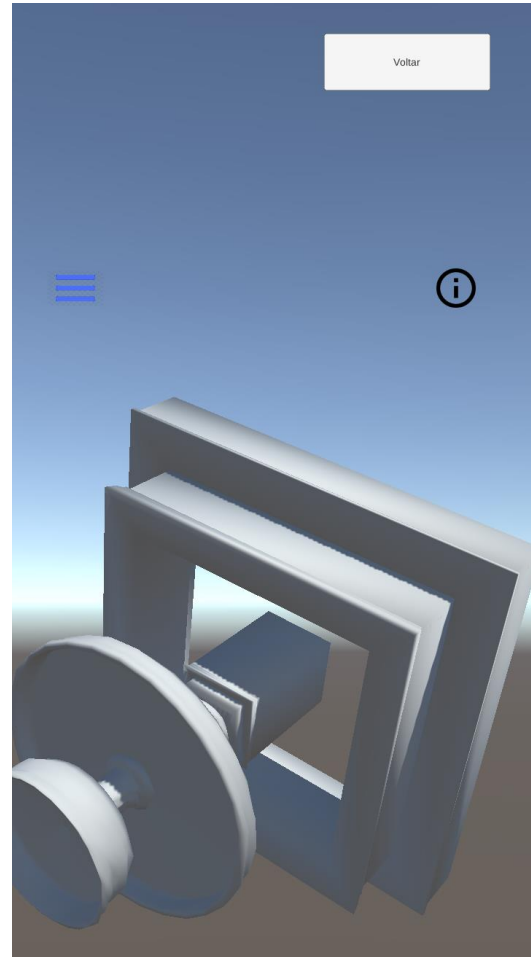


Figura 91 - Chafariz rodado na aplicação de RA

Para além disso, possui também a opção de interagir com os diversos tipos de multimédia que o ponto de interesse oferece através do botão de menu (símbolo *hamburger*). Uma vez premido, aparecerão na tela os diferentes tipos de multimédia dinamicamente, de acordo com o ficheiro de configuração lido. No exemplo da Figura 92 verificamos que o chafariz contempla apenas ficheiros de imagem.

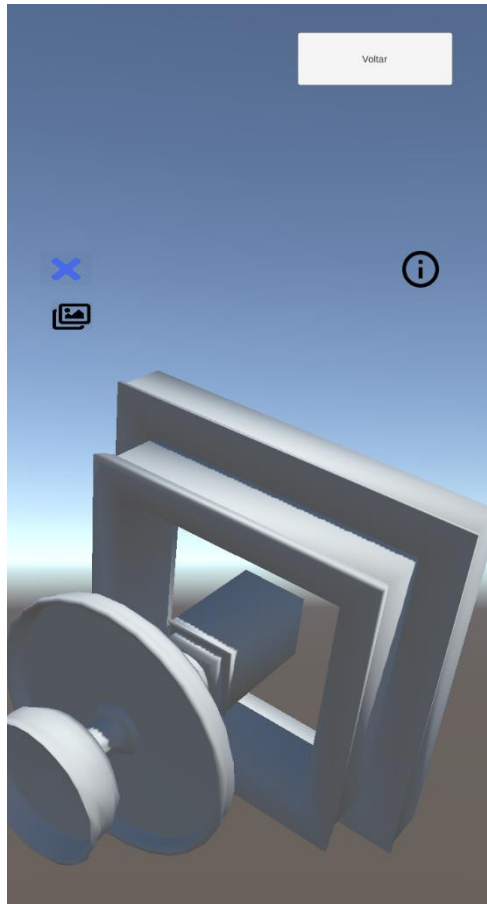


Figura 92 - Menu de multimédia aberto na aplicação de RA

Caso o utilizador carregue no botão de visualização de imagens, aparecerá então o painel mencionado em 4.4.2.2.4. Irão ser mostradas várias imagens, na eventualidade de existir mais do que uma, acompanhadas da respetiva descrição (ver Figura 93).

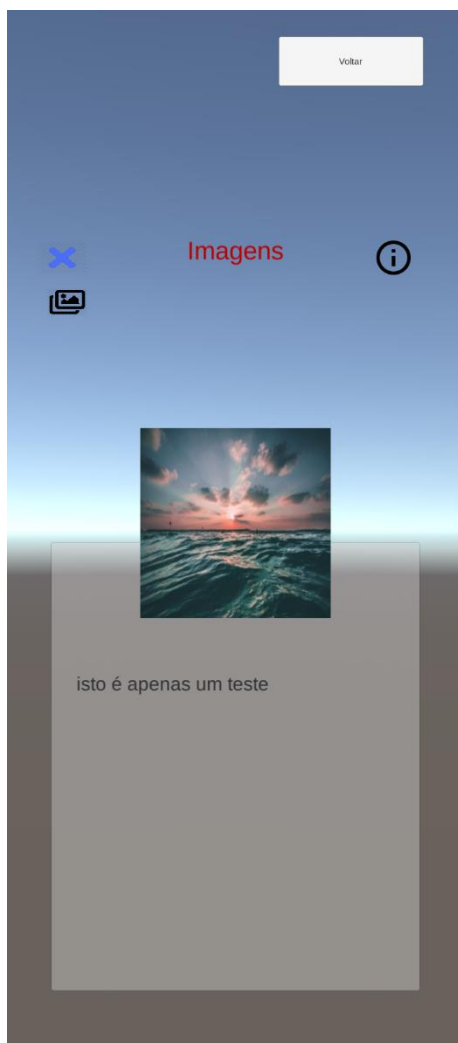


Figura 93 - Exemplo painel de imagens na aplicação de RA

Caso existam mais imagens para serem mostradas, basta que o utilizador deslize o dedo para a direita. Um novo painel surgirá com a nova imagem e a nova descrição.

O símbolo oposto ao menu de multimédia presente na Figura 93 apresenta uma breve informação sobre o modelo/local que o visitante se encontra a visualizar de modo a o contextualizar no tempo e espaço (ver Figura 94).



Figura 94 - Exemplo de informação acerca de um Ponto de Interesse na aplicação de RA

A partir do claustro era possível aceder a um átrio onde se encontrava uma arca, atual Portaria. A visualização do modelo da arca possui o mesmo princípio de visualização do modelo do chafariz. O Utilizador após chegar ao local desejado aponta a dispositivo pessoal para o marcador, aguardando que a aplicação reconheça o mesmo. Logo após o modelo aparecerá na tela do dispositivo (ver Figura 95).

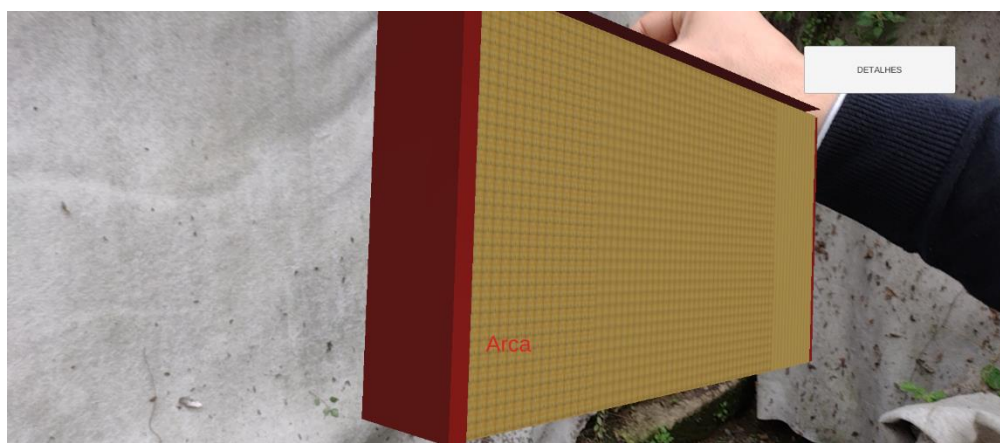


Figura 95 - Modelo da arca visualizado na aplicação de RA

Mais uma vez o marcador teve de ser suspenso devido a problemas com escalas e rotações que irão ser referidos mais adiante.

Carregando no botão de “DETALHES” o utilizador é levado para a cena de visualização da arca. Tal como acontece com o chafariz, também no caso da arca é possível visualizar o modelo em todas as direções e ângulos através de rotações, zooms e translações (ver Figura 96).

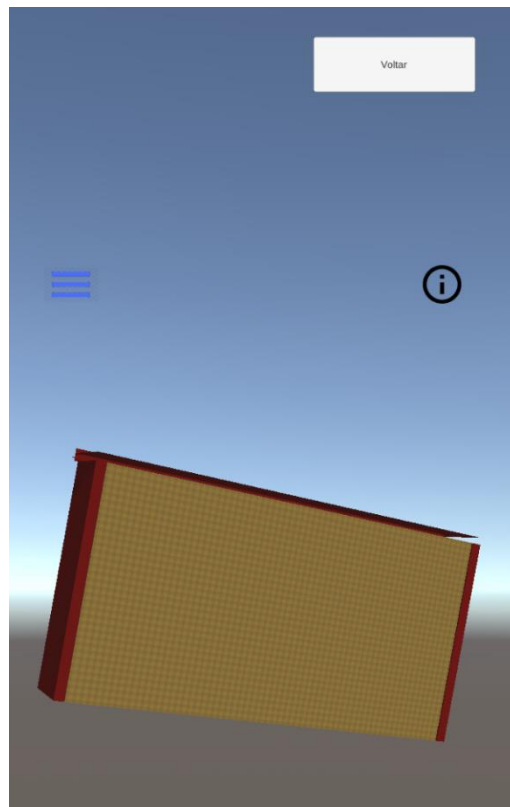


Figura 96 - Arca vista de frente na aplicação de RA

No teste prático realizado atribuíram-se ao modelo da arca todos os tipos de multimédia, de modo a testar cada um destes (ver Figura 97).

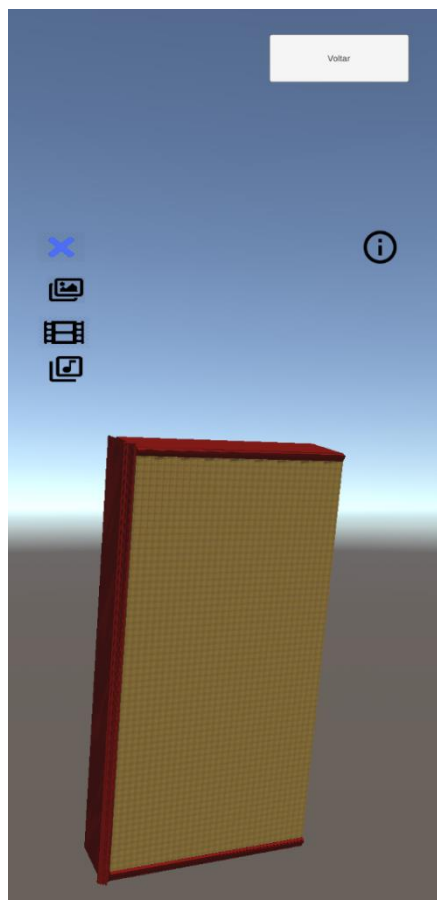


Figura 97 - Menu de multimédia aberto do modelo da arca na aplicação de RA

No caso da arca foram atribuídas duas imagens de forma a verificar o sistema de deslize para o próximo painel. Como era de se prever, este aconteceu de forma suave tanto para a direita, como para a esquerda (ver Figura 98 e Figura 99). A descrição também alterou de acordo com a imagem projetada na tela.

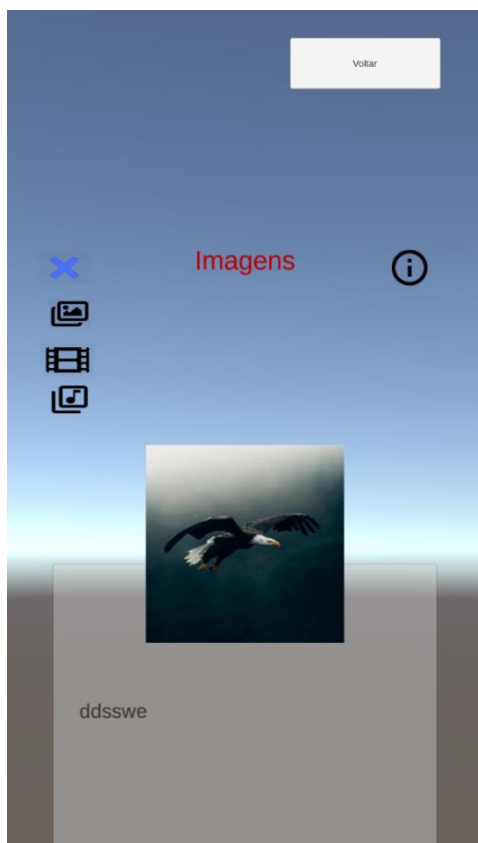


Figura 98 - Exemplo de imagem no primeiro painel na aplicação de RA

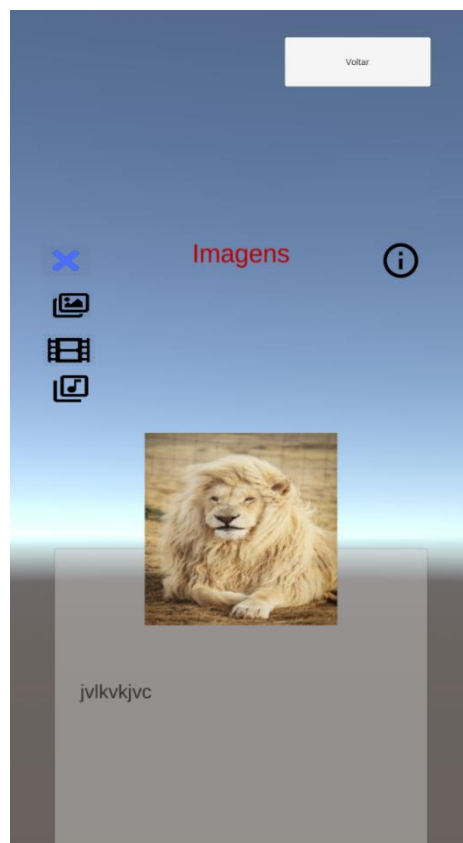


Figura 99 - Exemplo de imagem no segundo painel na aplicação de RA

A partir da Figura 97 observamos que a arca possui tanto ficheiros de vídeo, como ficheiros de áudio ao contrário do chafariz que apenas possui ficheiros de imagem. Como já foi referido, os ficheiros de vídeo apresentam exatamente a mesma estrutura dos ficheiros de imagem (ver Figura 100).

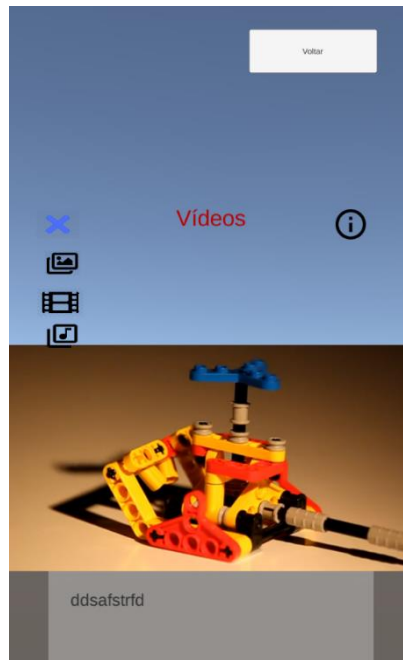


Figura 100 - Exemplo de vídeo no primeiro painel na aplicação de RA

Por fim, o painel de áudio contém somente três botões por si só descritivos (ver Figura 101). No projeto em concreto optou-se apenas por colocar um único ficheiro de áudio por ponto de interesse, uma vez que o Unity não suporta ficheiros muitos longos. Assim, é possível numa melhoria à própria aplicação acrescentar mais ficheiros de áudio por ponto de interesse à semelhança do que acontece com os outros dois tipos de multimédia.

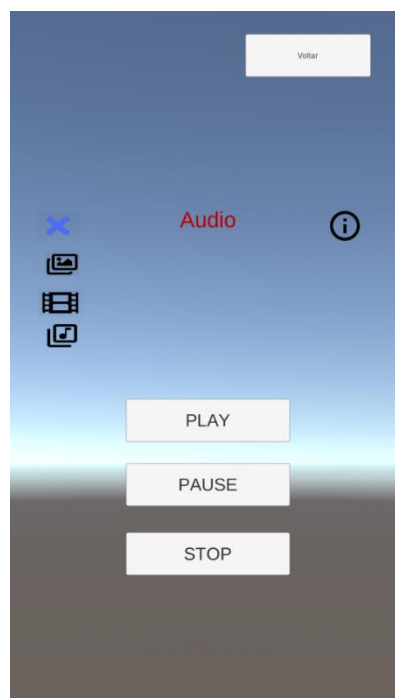


Figura 101 - Exemplo painel de áudio na aplicação de RA

Capítulo 6. Conclusões

6.1. Sumário e discussão dos resultados

Durante todo o processo de conceção e implementação da aplicação de RA, não se pensou na parte gráfica da mesma (UI/UX), tendo-se focado único e exclusivamente na funcionalidade e dinamismo da mesma, bem como na robustez geral.

Posto isto, no início do projeto foi sugerido o desenvolvimento de uma aplicação Web que auxiliaria na conceção e manutenção de guias de realidade aumentada, posteriormente lidos por uma aplicação para Android que guiaria os visitantes pelos vários pontos de interesse espalhados pelo Convento. De forma a testar o conceito criado, foram realizados testes no local provando que a partir de um único ficheiro de controlo (ficheiro de configuração) era possível um completo manuseamento de toda a experiência de realidade aumentada.

Os testes práticos realizados no Convento de São Francisco de Real apresentaram-se como sendo o culminar do projeto, sendo os mais importantes na avaliação do produto proposto. Estes revelaram-se bem-sucedidos devido ao contributo e disponibilidade de todos os intervenientes.

De modo a ser possível criar um ficheiro de configuração robusto, prático e dinâmico foi necessário projetar uma interface simples, com um baixo grau de complexidade para o administrador do sistema operar, criando e gerindo visitas. Deste modo qualquer tipo de complexidade adicional é abstraída. Para tal, optou-se pelo desenvolvimento de uma aplicação Web capaz de importar e exportar os modelos 3D, bem como todos os ficheiros pertinentes a uma determinada visita.

Ao longo de todo o desenvolvimento do projeto o conceito de dinamismo esteve sempre presente. Foi-se sempre tentando criar uma aplicação que pudesse ser usada em qualquer local, independentemente no número de pontos de interesse e ficheiros multimédia. Assim, a aplicação não se prende somente ao caso de estudo da dissertação, Convento de São Francisco de Real.

A nível de *software*, uma vez que o SDK AR Foundation ainda se encontra numa fase precoce de desenvolvimento, apresenta inúmeros *bugs* e problemas,

especialmente quando utilizado para desenvolver aplicações em android. O facto de se tratar de uma tecnologia relativamente recente faz com que não exista uma grande documentação acerca da mesma, tanto a nível oficial como *open source*. Tal facto fez com que determinadas tarefas fossem difíceis de ultrapassar. A documentação existente é confusa e os fóruns dedicados a esta tecnologia não conseguem dar resposta aos problemas, na maior parte das vezes. No entanto, vejo este SDK como sendo o futuro da RA, visto que suporta todos os sistemas operativos.

Como conclusão, gostaria de frisar que o conceito apresentado é de extrema importância nos tempos correntes uma vez que tanto a realidade aumentada como a realidade virtual se estão a tornar na tecnologia do futuro. No âmbito da herança cultural, a realidade aumentada permite abrir novas portas para a forma como os locais de culto e aprendizagem são vistos.

6.2. Trabalho futuro

Apesar da aplicação proposta cumprir o seu propósito em termos de funcionalidade é possível implementar melhorias na mesma de forma a proporcionar ao visitante uma visita ainda mais agradável e imersiva.

Relativamente à aplicação Web, o fluxo de como o guia de realidade aumentada é criado não apresenta uma ordem natural devido à quantidade excessiva de menus que apesar de estarem conectados entre si, é fácil perder-se nos mesmos. Assim sendo, de forma que seja possível aos utilizadores do sistema a criação de guias de uma forma ainda mais rápida, seria apenas necessário criar uma aba no menu lateral, sendo que a partir desta se ia construindo o guia um passo de cada vez.

No caso da aplicação móvel, optou-se por realizar o *download* de todos os recursos (marcadores, modelos, elementos multimédia) de uma só vez, aquando da correta leitura do código QR. Apesar dos dispositivos correntes possuírem uma enorme capacidade de memória, podendo assim acomodar uma enorme quantidade de modelos e ficheiros multimédia associados, existe sempre o tempo de espera pelo *download* de todos os componentes e a eventualidade de um determinado visitante possuir pouco espaço de armazenamento no momento do *download* da visita. Para tal, poderia ser utilizado um método híbrido, *download* de alguns componentes *a priori* e

durante a visita o carregamento dos restantes quando necessário. No entanto o local visitado necessita de possuir total cobertura de WiFi ou o utilizador teria de ter os dados móveis ligados durante todo o período da visita.

Foi ainda considerada a existência de um mapa do local visitado com os pontos de interesse dinamicamente inseridos no mesmo de acordo com a visita escolhida. Este seria uma melhoria na qualidade de vida da aplicação, visto que um local pode possuir dezenas de pontos de interesse espalhados. O mapa ajudaria o visitante a deslocar-se apenas para os pontos de interesse, indicando sempre o melhor caminho para tal.

Uma outra melhoria notória seria melhorar a escala e rotação dos pontos de interesse quando visualizados. Neste momento, estes aparecem na tela do dispositivo sem estarem à escala e não posicionados no sítio onde outrora se encontravam. Ainda neste tópico, uma outra melhoria significativa seria aumentar a distância de *tracking* ao ponto de interesse. Na implementação atual, o visitante tem de estar relativamente perto do ponto de interesse para que seja possível a correta leitura. Uma das opções para contornar o problema seria aumentar o tamanho do marcador, o que não faria qualquer sentido visto que este iria ficar excessivamente grande.

Outra melhoria na aplicação de RA seria tornar a UI mais moderna e atraente, enquanto se melhorava também a UX, tornando a aplicação ainda mais responsiva e com uma interface bem cuidada e profissional.

Por último, uma outra melhoria significativa seria abandonar os marcadores físicos e focar apenas nos marcadores por localização ou *tracking* sem recurso a marcadores. Com o rápido avanço da tecnologia existem outras alternativas aos marcadores físicos, mais flexíveis e inovadoras.

Em suma, o *software* desenvolvido ao longo do projeto obteve um resultado positivo após testar o conceito em ambiente real. Contudo, possui ainda lacunas consideráveis para que se possa fazer dele um produto fidedigno e rentável. No entanto, algumas das melhorias citadas anteriormente não apresentam um elevado grau de complexidade a nível de codificação sendo por isso possível tornar este projeto algo rentável.

Referências Bibliográficas

- [1] Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., E. Damiani, Ivkovic, M. (2011). *Augmented reality technologies, systems and applications* (vol. 51, no. 1, pp. 341–377). *Multimed. Tools Appl.*
- [2] Zheng, J., Chan, K., Gibson, I. (1998). *Virtual reality* (vol. 17, no. 2, pp. 20–23). *IEEE Potentials.*
- [3] Kim, S., Kang, J., Choi, J., Choi, H., Hong, M (2017). *Augmented-reality survey: From concept to application* (vol. 11, no. 2, pp. 982–1004). *KSII Trans. Internet Inf. Syst.*
- [4] Rabbi, I., Ullah, S. (2013). *A Survey on Augmented Reality Challenges and Tracking* (vol. 24, no. 1–2, pp. 29–46). *Acta Graph. Znan. časopis za Tisk. i Graf. Komun.*
- [5] Krueger, W., Gionfriddo, T., Hinrichsen, K. (1985). *Videoplace - an Artificial Reality* (pp. 35–43). *University of Connecticut Storrs.*
- [6] Caudell, P., Mizell, W. (1992). *Augmented Reality : An Application of Heads-Up Display Technology to Manual Manufacturing Processes.*
- [7] Feiner, S., Macintyre, B., Seligmann, D. (1993). *Knowledge - based augmented reality. Communications of the ACM.*
- [8] Milgram, P. (2011). *A Taxonomy of Mixed Reality Visual Displays* (pp. 1–14). *University of Toronto, Ontario, Canada.*
- [9] Amin, D., Govilkar, S. (2015). *Comparative Study of Augmented Reality Sdk's* (vol. 5, no. 1, pp. 11–26). *University of Mumbai, New Panvel, India.*
- [10] Fuhrmann, A., Encarnac, M. (2002). *The Studierstube Augmented* (pp. 33–54). *Vienna University of Technology.*
- [11] Azuma ,R. (1997). *A Survey of Augmented Reality* (vol. 6, no .4, pp.355-385). *Hughes Research Laboratories, Malibu.*
- [12] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B. (2001). *Recent Advances in Augmented Reality. IEEE Computer Graphics and Applications.*
- [13] Arth, C., Grasset, R., Gruber, L., Langlotz, T., Mulloni, A., Wagner, D. (2015). *The History of Mobile Augmented Reality.* <http://arxiv.org/abs/1505.01319>.
- [14] D’Orazi, G. (2021). *Recent advances in P53* (vol. 11, no. 2, pp. 1–4).
- [15] Piekarski, W., Thomas, B. (2002). *ARQuake: The outdoor augmented reality gaming system* (vol. 45, no. 1, pp. 36–38). *Communications of the ACM.*

- [16] Wagner, D., Pintaric, T., Schmalstieg, D. (2004). *The invisible train* (p. 12). *Vienna University of Technology*
- [17] "Introduction to Augmented Reality." <https://www.se.rit.edu/~jrv/research/ar/introduction.html> (accessed Jan. 20, 2022).
- [18] Pereira, B., Pereira, M. (2008). *TARCAST: Uma Taxonomia para Sistemas de Realidade Aumentada*.
- [19] Livingston, A., State A. (1997). *Magnetic tracker calibration for improved augmented reality registration* (vol. 6, no. 5, pp. 532–546). *University of North Carolina*.
- [20] Azuma, R., Lee, J., Jiang, B., Park, J., You, S., Neumann, U. (1999). *Tracking in unprepared environments for augmented reality systems* (vol. 23, no. 6, pp. 787–793). *University of Southern California, Loas Angeles*.
- [21] Zhou, F., Duh, L., Billingham, M. (2008). *Trends in Augmented Reality Tracking, Interaction and Display*.
- [22] ARToolworks. (2022, January 20). *FLARToolKit*. <https://www.artoolworks.com/products/web/flartoolkit-2.html>
- [23] Vuforia. (2022, January 20). *Recommended Devices*. <https://library.vuforia.com/platform-support/vuforia-engine-recommended-devices.html>
- [24] Circuit Stream. (2022, January 20). *Ultimate AR Comparison Guide*. <https://circuitstream.com/blog/augmented-reality-guide/>
- [25] Unity. (2022, January 20). *About AR Foundation* <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html>
- [26] Makarov, Andrew. (2022, January 20). *10 Augmented Reality Trends of 2022: A Vision of Future*. <https://mobidev.biz/blog/augmented-reality-trends-future-ar-technologies>
- [27] Paine, J. (2022, January 20). *10 Real Use Cases for Augmented Reality*. <https://www.inc.com/james-paine/10-real-use-cases-for-augmented-reality.html>
- [28] Morgan, K. (2022, January 20). *The Role of Augmented Reality in Medicine*. <https://www.webmd.com/a-to-z-guides/features/augmented-reality-medicine>

- [29] Layar. (2022, January 20). *Geo Layer: AED4EU*. <https://www.layar.com/layers/sander>
- [30] Pokemon Go. (2022, January 20). *Pokémon GO*. https://pokemongolive.com/pt_br/#learn
- [31] NarraSoft. (2022, January 20). *What is AR in Gaming? A Brief Guide*. <https://narrasoft.com/augmented-reality-gaming/>
- [32] eLearning Industry. (2022, January 20). *Augmented Reality In Education*. <https://elearningindustry.com/augmented-reality-in-education-staggering-insight-into-future>
- [33] ViewSonic. (2022, January 20). *6 Benefits and 5 Examples of Augmented Reality in Education*. <https://www.viewsonic.com/library/education/6-benefits-and-5-examples-of-augmented-reality-in-education/>
- [34] Maryville Online, (2022, January 20). *Augmented Reality in Education: Interactive Classrooms*. <https://online.maryville.edu/blog/augmented-reality-in-education/>
- [35] Active Learning. (2022, January 20). *Chem101 – Transform Your Chemistry Students with Active Learning*. <https://101edu.co/>
- [36] Photomath. (2022, January 20). <https://photomath.com/en/>
- [37] MergeEdu. (2022, January 20). <https://mergeedu.com/cube>
- [38] Tynker. (2022, January 20). <https://www.tynker.com/courses/home-augmented-reality>
- [39] Timelooper. (2022, January 20). *TimeLooper - Story Building Platform*. <https://www.timelooper.com/>
- [40] 360Cities. (2022, January 20). *Stock 360° Panoramic Images and Videos for VR and More*. <https://www.360cities.net/>
- [41] Mortice, Z. (2022, January 20). *What Is Augmented Reality in Construction and Architecture?*. <https://redshift.autodesk.com/what-is-augmented-reality/>
- [42] Souza, E. (2022, January 20). *9 Augmented Reality Technologies for Architecture and Construction*. *ArchDaily*. <https://www.archdaily.com/914501/9-augmented-reality-technologies-for-architecture-and-construction>
- [43] Virtualist. (2022, January 20). *Augmented Reality (AR) in Architecture*. <https://virtualist.app/augmented-reality-ar-in-architecture/>
- [44] Morpholio Trace. (2022, January 20). <https://www.morpholioapps.com/trace/>

- [45] Carson. B. (2022, January 20). *I tried on the smart helmet from the future, and it gave me super powers*. <https://www.businessinsider.com/what-is-the-daqri-smart-helmet-2016-4>
- [46] GAMMA AR. (2022, January 20). <https://gamma-ar.com/>
- [47] Aliprantis, J., Caridakis, G. (2019). *A Survey of Augmented Reality Applications in Cultural Heritage* (vol. 3, no. 2, pp. 118–147). University of the Aegean, Mytilene, Greece.
- [48] Stricker, D., Karigiannis, J., Christou, T., Gleue, T., Ioannidis, N. (2001). *Augmented Reality for Visitors of Cultural Heritage Sites* (pp. 89–93).
- [49] Papagiannakis, G. et al. (2002). *LIFEPLUS: revival of life in ancient Pompeii, virtual systems and multimedia* (pp. 25–27). University of Geneva, Switzerland.
- [50] Noh, Z., Sunar, S., Pan, Z. (2009). *A review on augmented reality for virtual heritage system* (pp. 50–61). Universiti Teknologi Malaysia.