UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

SERVIÇOS WEB PARA UMA APLICAÇÂO DE REALIDADE AUMENTADA

Daniel Onofre Nunes Soares

Mestrado em Engenharia Informática
Especialização em Sistemas de Informação

Trabalho de projeto orientado por:
Prof. Doutor António Manuel Silva Ferreira
Prof.ª Doutora Maria Beatriz Duarte Pereira do Carmo

2016

Resumo

Este relatório documenta a evolução da arquitetura da SolAR, passando de uma aplicação móvel de realidade aumentada com cópia local de dados sobre radiação solar em fachadas de edifícios, escolhidos manualmente pelo utilizador, para uma aplicação que faz uso de serviços *Web* para obtenção automática de dados em função da posição geográfica.

Assim, o primeiro objetivo do trabalho foi a concretização de serviços *Web* para fornecer dados sobre radiação solar armazenados numa base de dados espacial. Este tipo de base de dados permite realizar pesquisas geográficas, devolvendo informação de acordo com a localização. Os principais resultados da avaliação mostraram que os serviços *Web* conseguem responder em tempo útil, num cenário de requisição de dados.

O desafio seguinte passou pela implementação de uma componente de *middleware* que fosse capaz de comunicar com os serviços *Web*, e que funcione qualquer que seja a aplicação cliente. Neste segundo objetivo, foi necessário que o *middleware* utilizasse os instrumentos de localização do dispositivo e que se criasse um mecanismo de armazenamento temporário, ou *cache*, para guardar os pedidos de dados mais recentes ou relevantes, permitindo poupar a rede de comunicação e os serviços *Web*. Os resultados da avaliação comprovaram o funcionamento correto da *cache* e sugeriram uma configuração de acordo com as condições em que se realizaram os testes.

O terceiro objetivo prendeu-se pela necessidade de adaptar a aplicação SolAR para que esta funcionasse corretamente com o *middleware*, iniciando e terminando a sua execução, e implementando uma política de comunicação com este. A responsabilidade do utilizador selecionar e armazenar manualmente os dados deixou de existir, sendo assumida pelo *middleware*, bastando que exista uma ligação de rede ativa.

A conclusão destes objetivos torna possível que sejam criadas aplicações concorrentes e alternativas à aplicação SolAR, utilizando o mesmo *middleware* e os mesmos serviços *Web*, aumentando o valor dos dados armazenados.

Palavras-chave: serviços *Web*, computação móvel, bases de dados espaciais

Abstract

This report documents the evolution of the SolAR architecture, from a mobile application of augmented reality with a local copy of solar radiation data on the facades of buildings, manually selected by the user, to an application that uses Web services to retrieve data depending on the geographical location.

Thus, the first objective was the implementation of a set of Web services to provide data on solar radiation stored in a spatial database. This type of database allows users to perform spatial queries, returning information according to the given location. The evaluation results showed that Web services are able to respond in time, given the scenario where the nearby data were requested.

The next challenge included the implementation of a middleware component that was able to communicate with Web services, and that works whatever the client application. In this second objective, it was necessary that the middleware used the device location instruments and created a temporary storage system, or cache, to store the more recent and relevant requests, saving the use of the network and the server. The evaluation results confirmed the correct operation of the cache and suggested a configuration according to the conditions in which the tests were performed.

The third goal was the need to adapt the SolAR application for it to function properly with the middleware, with the ability to start and shut down its execution, and implementing a communication policy with the middleware. The user's responsibility to manually select and store the data no longer exists, being taken by the middleware, as long as there is an active network connection.

The completion of these objectives makes it possible for competing applications to be created, alternative SolAR application, using the same middleware and the same Web services, increasing the value of the data stored.

Keywords: web services, mobile computing, spatial databases

Conteúdo

Lista de figuras

Lista de tabelas

Abreviaturas

API	*Application Programming Interface*
FIFO	*First In First Out*
GPS	*Global Positioning System*
GPX	*GPS Exchange Format*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
JSON	*JavaScript Object Notation*
LFU	*Least Frequently Used*
MFU	*Most Frequently Used*
RA	Realidade Aumentada
ReST	*Representational State Transfer*
RPC	*Remote Procedure Call*
SGBD	Sistema de Gestão de Base de Dados
SQL	*Structured Query Language*
SRID	*Spatial Reference System Identifier*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
WKT	*Well-known Text*
XML	*eXtensible Markup Language*

Capítulo 1

Introdução

No âmbito da disciplina de Projeto de Engenharia Informática (PEI) do Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa (FCUL), surgiu este projeto de evolução da aplicação SolAR, para visualização em realidade aumentada de radiação solar em fachadas de edifícios, que inicialmente estava assente no armazenamento de dados e processamento local num dispositivo móvel, para passar a basear-se numa arquitetura de serviços *Web*.

Este capítulo explicita o enquadramento do trabalho realizado e as suas motivações. Enumera e descreve, também, os objetivos propostos e apresenta as suas principais contribuições. São ainda referidos detalhes sobre este documento, nomeadamente a notação adotada e a estrutura dos capítulos.

1.1 Motivação

Nos dias de hoje existe uma grande preocupação com o meio ambiente e a eficiência energética. A produção e utilização de recursos energéticos renováveis e de energias limpas aumentou consideravelmente nos últimos anos [1], sendo um setor que tem verificado um crescimento económico e de popularidade, trazendo esses benefícios também para o setor das Tecnologias de Informação (TI). Dada a grande popularidade desta tendência, é importante oferecer as melhores e variadas soluções às pessoas, e que lhes deem a possibilidade de tomar as melhores decisões e suprir as suas necessidades.

Neste contexto surgiu a SolAR: uma aplicação móvel para o sistema operativo Android que permite visualizar a radiação solar incidente nas fachadas dos edifícios recorrendo a técnicas de realidade aumentada (RA).

Até ao início deste PEI, todos os dados que a aplicação utilizava eram armazenados localmente no dispositivo. Esta solução não é prática quando se tem muitos dados pois utiliza muita memória do dispositivo e requer o processamento de uma maior quantidade de dados além da estritamente necessária. Como consequência, os dados disponíveis eram apenas referentes ao edifício C6 da FCUL, ou seja, apenas a um edifício de tamanho médio embora com um número razoável de fachadas.

Existia ainda a questão de os dados estarem apenas disponíveis para a própria aplicação SolAR, sendo que para outro sistema ou aplicação cliente que necessite dos mesmos dados, teria que criar uma cópia. A criação e propagação de cópias seria da responsabilidade dos utilizadores e possibilita o aparecimento de incoerências na informação, que se deseja ser a mesma para os vários cenários de utilização.

Pretendia-se com este trabalho que o acesso aos dados passasse a ser feito através de pedidos a um servidor remoto, permitindo retirar a responsabilidade do utilizador fornecer os dados à aplicação, e aumentando a mobilidade desta. Esse servidor remoto seria o ponto de acesso à informação permitindo que vários clientes, sejam eles múltiplos utilizadores do SolAR, ou outra aplicação, acedam concorrentemente aos dados.

Era também desejável que a aplicação SolAR conseguisse ter armazenada uma quantidade de dados adequada a um uso satisfatório por parte do utilizador, nomeadamente os elementos referentes às fachadas para as quais se encontra direcionado. Contudo, era suposto que se oferecesse um nível de transparência em relação aos pedidos efetuados ao servidor através da rede, bem como a falhas eventuais na cobertura de rede ou força de sinal.

Esperava-se também oferecer uma maior qualidade de informação, visto esta se encontrar num ponto central, facilitando a sua atualização, e permitindo que esta abranja áreas físicas mais vastas, aumentando a mobilidade do utilizador, e dando a possibilidade de melhorar a aplicação SolAR existente e que sejam criadas outras aplicações que façam uso da mesma informação.

1.2 Objetivos

Tendo em conta os problemas descritos e o rumo apresentado, enumeram-se os três principais objetivos deste trabalho:

1. Serviços *Web* para aceder a uma base de dados relativos a radiação solar e painéis solares.
2. Componente *middleware* para facilitar a ligação entre os serviços *Web* e as aplicações.
3. Adaptação da aplicação SolAR existente para que tire proveito dos serviços *Web*.

Nas seguintes secções, existe uma breve descrição de cada um dos três principais objetivos.

Objetivo 1: Serviços *Web*

Tendo um servidor e uma base de dados, com a informação sobre radiação solar e painéis solares, é necessário criar métodos para que seja possível a uma entidade externa aceder aos dados. Pretende-se que os métodos permitam que vários tipos de utilizadores – como uma pessoa comum, um investigador, um técnico de instalação de painéis solares, um administrador de sistema – possam criar, ler, editar ou apagar informação no sistema. Cada uma destas tarefas será correspondente à tipologia do método invocado e tendo em conta as permissões dadas a cada tipo de utilizador esperado.

Objetivo 2: *Middleware*

O principal requisito do *middleware* é que este, para além de interagir com os serviços *Web*, deve poder ser usado em qualquer tipo de aplicação. Isso retira a responsabilidade de uma aplicação aceder diretamente aos serviços, mas também permite manter o máximo de mecanismos já existentes na aplicação móvel SolAR.

Anteriormente, a aplicação recorria a um ficheiro local para acesso a dados, mas passando os dados a ser alojados num servidor, ou seja, fora do domínio da aplicação cliente, o *middleware* terá que utilizar uma rede de comunicação para enviar e receber dados. É também necessário criar um mecanismo que funcione como armazenamento temporário, ou *cache*, que armazena os dados recebidos, minimizando tráfego de rede e carga no servidor, e também editá-los, permitindo ao utilizador ter informação disponível no seu dispositivo.

Objetivo 3: Adaptações

Devido à existência do *middleware*, é necessário que a aplicação SolAR funcione corretamente em conjunto com este. Para isso, é necessário modificar a aplicação para que esta inicie o *middleware* quando arranca, e termine a execução do mesmo quando a aplicação é encerrada. Existe a preocupação de realizar o mínimo de alterações nas funcionalidades da aplicação SolAR, de modo a que melhor se aproveite o trabalho realizado, e que o esforço deste projeto seja em concluir os dois primeiros objetivos.

1.3 Contribuições

A conclusão deste projeto traz benefícios à aplicação SolAR mas também permite criar novas aplicações que façam uso dos serviços *Web* disponibilizados.

Existindo dados referentes a vários edifícios, estes foram organizados e armazenados numa base de dados espacial. Uma base de dados espacial possibilita que, através de pesquisas geográficas, sejam devolvidos dados referentes à zona envolvente do utilizador. Por consequência, os dados devolvidos pela pesquisa podem ter uma maior relevância pois encontram-se perto do utilizador e permite que se utilize o espaço ocupado pelos dados de forma mais eficiente.

Com a centralização dos dados, e permitindo que estes sejam acedidos através de uma chamada a um serviço *Web*, abre-se a possibilidade que sejam criadas outras aplicações móveis que utilizem os mesmos dados, sem ter que os copiar manualmente. Essas novas aplicações podem ser criadas de forma independente dos serviços, visto que apenas necessitam de integrar o *middleware* na sua estrutura e operar de acordo com o protocolo que este exige. Ou seja, o *middleware* introduz uma camada que abstrai a utilização dos serviços e providencia à aplicação cliente o ficheiro de dados que esta necessita para operar.

O *middleware* permite que os dados sejam requisitados de acordo com a posição geográfica do utilizador, fazendo com que a aplicação não esteja restringida espacialmente. Se existirem dados referentes à zona em que o utilizador se encontra, através da ligação ao servidor, estes serão transferidos de forma transparente para a aplicação. Através de um armazenamento temporário, *cache*, os dados dos últimos pedidos são mantidos no dispositivo. Isso evita a multiplicação de pedidos ao serviço *Web*, quando, por exemplo, o utilizador se vai deslocando nas proximidades de um mesmo edifício.

Foram realizados testes sobre o *middleware*, e analisados aos valores obtidos. A análise dos resultados incidiu, entre outros aspetos, sobre o tamanho dos últimos pedidos armazenados, tempo despendido pelos pedidos de dados, posições onde ocorreram ou repetição dos dados recebidos. Os valores dos resultados permitiram que se sugerisse uma configuração adequada, tendo em conta as condições em que os testes ocorreram.

1.4 Notação adotada

O texto que se encontre em outros idiomas, que não o Português, encontram-se escrito em itálico. Os excertos de ficheiros de texto ou de código utilizam o tipo de letra Courier New. Para o desenho de diagramas foi adotada a notação *Unified Modeling Language* (UML).

1.5 Organização do documento

Este documento está organizado da seguinte forma:

Capitulo 1 – expõe as motivações para a realização do projeto bem como os objetivos pretendidos para o mesmo. Exibe as principais contribuições dadas pelos resultados do projeto. Define, também, a notação adotada no documento e a sua estrutura dos capítulos.

Capítulo 2 – apresenta conceitos teóricos importantes neste âmbito. Mostra também alguns sistemas produzidos por outros investigadores e o trabalho que serviu de base a este projeto.

Capítulo 3 – descreve o trabalho realizado, explicando a implementação de *software* e decisões tomadas. Analisa os resultados da avaliação de cada objetivo e tira conclusões sobre os mesmos.

Capítulo 4 – resume as principais contribuições dadas. Enumera ainda as competências adquiridas fruto do desenvolvimento do trabalho, os problemas encontrados e também propõe algumas melhorias e funcionalidades como trabalho futuro.

Capítulo 2

Conceitos e trabalho relacionado

2.1 Conceitos

Neste capítulo são abordados vários conceitos úteis para a tipologia deste trabalho, bem como tecnologias concorrentes ou alternativas consideradas durante a fase de análise do problema.

2.1.1 Realidade aumentada

A realidade aumentada (RA), em inglês *Augmented Reality* (AR) é uma tecnologia que apresenta ao utilizador um ambiente real sobreposto com elementos virtuais [2]. Deste modo, o utilizador continua a ter noção exata do espaço real, ao contrário da realidade virtual onde todo o ambiente é gerado computacionalmente, conseguindo ver o ambiente real acrescido de informação gerada pelo computador. Esta informação acrescenta valor ao nosso mundo e é aplicada em várias áreas, como por exemplo, a visualização de dados científicos e aplicações de engenharia ou médicas.

2.1.2 Serviços *Web* e formatos de dados

Segundo o *Web Services Architecture Working Group* do W3C, um serviço *Web* é um sistema de *software* concebido para suportar interações pela rede entre máquinas [3]. As máquinas interagem com o serviço seguindo o prescrito por um *Web Service Description Language* (WSDL) ou uma *Application Programming Interface* (API). Estes elementos definem o método protocolar de requerer e receber dados através do serviço. A comunicação entre a aplicação cliente e a aplicação servidora é suportada através do protocolo *Hypertext Transfer Protocol* (HTTP), seguindo a informação tipicamente em formato XML, JSON ou outros meios de representação de informação.

ReST e SOAP

Representational State Transfer (ReST) é um estilo arquitetural de serviço *Web*. Segue uma arquitetura Cliente-Servidor em que o seu funcionamento se assemelha a um *Remote Procedure Call* (RPC). Ou seja, são invocados, através de um *Uniform Resource Identifier* (URI), métodos remotos que são ativados recorrendo a verbos HTTP. O serviço *Web* processa o pedido e retorna uma resposta, podendo esta conter dados encapsulados ou apenas códigos de confirmação. Os métodos URI são considerados recursos, podendo estes ser simples ou mais complexos, resultantes da composição de vários recursos. As aplicações desenhadas nesta arquitetura são, por norma, focadas na simplicidade, escalabilidade e usabilidade.

Simple Object Access Protocol (SOAP) é um estilo arquitetural concorrente ao ReST combinando comunicação RPC com comunicação por mensagens [4]. Este estilo recorre à troca de mensagens em formato XML em vez de verbos HTTP.

Um sistema baseado em ReST é menos complexo, devido ao protocolo de uso e simplicidade de comunicação entre serviços e clientes, e envolve menos decisões arquiteturais, poupando tempo e trabalho na fase de análise. No entanto introduz uma maior necessidade de criação de código [4]. Este ponto é mitigado pelo facto de ser mais flexível e dar mais poder de decisão e controlo a quem define o sistema. Visto basear-se em URI é facilmente considerado *ad-hoc* pois o seu esquema tem grandes semelhanças à *Web* e pode assim ser utilizado por mais clientes.

SOA

Arquitetura orientada a serviços, em inglês *service-oriented architecture* (SOA), define um modelo arquitetural de construção de *software* em que é defendido que as funcionalidades existentes nas aplicações devem ser disponibilizadas primeiramente em forma de serviços. Esses serviços ficam acessíveis geralmente através da *Web* ou outras formas de comunicação entre aplicações. Posicionando os serviços como componente primário da arquitetura permite melhorar a eficiência, agilidade, e produtividade da equipa de desenvolvimento [5][6].

JSON

JavaScript Notation Object (JSON) é uma representação de dados em formato par chave:valor [7]. Este formato é uma norma livre e de fácil leitura por qualquer pessoa

ou por computadores, ganhando popularidade devido a estes aspetos e ainda por ser menos verboso em comparação com XML (referido mais à frente).

GeoJSON

O *Geometry JavaScript Notation Object* (GeoJSON) é um formato baseado em JSON e dedicado à representação de dados espaciais e deste modo proceder ao transporte desde tipo de informação [8].

XML

A *Extensible Markup Language* (XML) é baseada num sistema de anotações para representar a informação [9]. Tal como o JSON, o XML é de entendimento para humanos, ou seja, não é só facilmente entendido por máquinas, mas as suas anotações criam mais texto. Qualquer documento neste formato segue o seguinte esquema: abre anotação, informação, fecha anotação, tornando-se mais verboso.

KML e GML

Keyhole Markup Language (KML) é um formato para dados espaciais/geográficos baseado em XML utilizado pela Google [10]. Este formato foi popularizado pelas aplicações Google Maps e Google Earth.

O *Geography Markup Language* (GML) é um formato baseado em XML, tal como o KML, e desenvolvido pelo Open Geospatial Consortium (OGC) [11] e serve o mesmo propósito do KML, representar informação geográfica.

WKT

O *Well-known Text* (WKT) é um formato que recorre a uma codificação para tornar a representação de formas geométricas mais compacta [12]. Esse processo é reversível, facilitando a leitura por parte de máquinas e pessoas. A sua representação habitual é geometria(coordenadas) sendo que as coordenadas são longitude e latitude, por esta ordem. Por exemplo, a representação de um ponto no espaço seria: `ST_Point-(38.755649,-9.1583982)`. Seguindo este formato de escrita, os SGBD espaciais, referidos na Secção seguinte, interpretam a *string* e armazenam-na em binário.

GPX

O *GPS Exchange Format* (GPX) é um formato baseado em XML, que permite representar dados, como pontos ou rotas, oriundos de um dispositivo *Global Positioning System* (GPS) [13]. Feita a representação, torna-se possível que os dados possam ser transferidos pela *Web* ou representados e visualizados num mapa.

2.1.3 Informação espacial e radiação solar

Para guardarmos dados de informação geográfica, que conhecemos do mundo em nosso redor, precisamos de manipular as suas estruturas para que possam ser representados em formato computacional. O formato convencional que foi adotado são formas geométricas [14], tais como: pontos, linhas, polígonos, cubos, circunferências, entre outros. Utilizando analogias, conseguimos então guardar a localização e limites de, por exemplo, pontos de interesse, estradas, rios, cidades e até áreas de pesquisa.

Descrevem-se em seguida, conceitos geográficos relacionados com o trabalho.

Radiação solar

A radiação solar, vulgarmente designada de luz solar, é composta por uma parte visível e outra não visível, como as radiações ultravioleta e infravermelha. O mais importante é que toda a radiação é energia, podendo ser aproveitada e utilizada no dia-a-dia através de células fotovoltaicas, componente elementar dos painéis solares.

Cota de um ponto

Cota é um termo utilizado na topografia e representa a altura de um ponto em relação a um plano horizontal de referência, como é o nível médio da água do mar.

Azimute

Azimute é o ângulo formado por uma qualquer direção e o norte geográfico. Seja A o ponto da nossa posição, B um ponto para onde olhamos e N o norte geográfico. Dá-se o nome de azimute ao ângulo interno formado pelas retas AB e AN.

Sistema referencial e geodésico

Para se conseguir identificar posições no espaço são utilizados sistemas de eixos. Neste projeto são utilizadas as coordenadas esféricas World Geodetic System (WGS84), correspondentes ao Spatial Reference System Identifier (SRID 4326), que são encontra-

das no sistema de coordenadas GPS. Também se faz uso de um sistema de coordenadas cartesianas ETRS89-PT-TM06 específico do território português pois os dados existentes encontram-se representados neste sistema.

Bases de dados espaciais

Tendo a necessidade de armazenar dados espaciais, temos a solução de recorrer a uma base de dados relacional tradicional, estendida para suportar este tipo de dados, também conhecida como base de dados espacial. Essas extensões são possíveis graças a uma implementação de tipos espaciais presente em SQL/MM- Part 3: Spatial. Esta norma adiciona capacidades ao SQL comum e está definida, mais recentemente, no ISO/IEC 13249-3/2011 [15].

Alguns SGBD mais conhecidos e utilizados fornecem suporte à informação espacial: o PostgreSQL recorre à extensão PostGIS [16], o Microsoft SQL Server tem suporte nativo desde a versão 2008 [17] e o MySQL também tem uma extensão espacial dedicada. De considerar também o SQLite e a sua extensão espacial SpatiaLite [18]. O SQLite é um SGBD relacional mais leve que os anteriormente referidos, mas largamente utilizado em aplicações utilizadas no dia-a-dia como navegadores *Web*, Skype ou sistemas operativos Android ou Windows.

Com estas extensões, temos as vantagens de um SGBD convencional – tais como: desempenho, escalabilidade, segurança, tolerância a falhas – e as vantagens de ter tipos de dados dedicados e otimizados para realizar operações espaciais. Akbari e Peikar apresentam uma comparação entre vários SGBD [19], referindo que em MySQL nem todas as funções espaciais se encontram implementadas. Em contraponto, o PostGIS oferece compatibilidade com SQL/MM, suportando tipos geométricos e geográficos, este último utiliza medições geodésicas em vez de medições cartesianas, funções e indexação espacial.

Sistema de informação geográfico

Um sistema de informação geográfico (SIG) é um conjunto de infraestruturas, aplicações e procedimentos dispostos para albergar o armazenamento, gestão, acesso e processamento de dados espaciais [20]. Desde modo é possível que a manipulação e visualização desde tipo de dados seja eficaz, acessível e eficiente.

2.2 Trabalho relacionado

Nesta secção são abordadas soluções apresentadas em trabalhos semelhantes. Enumeram-se ainda os pontos fortes e fracos e as diferenças para este trabalho.

Alguns exemplos de construção de sistemas compostos por serviços *Web* utilizados por aplicações móveis de RA são descritos por Selonen, Belimpasakis, You, Pylvanainen e Uusitalo [21]. As duas aplicações apresentadas recorrem a um *backend* comum, composto por serviços *Web* ReSTful, que lhes fornecem os dados. No artigo são exemplificados vários pedidos e respostas que dão suporte ao que foi inicialmente pensado para o sistema desenvolvido neste PEI. A aplicação cliente requer os dados construindo um *link* e executando o pedido e o sistema encapsula os dados num formato específico e estruturado em JSON enviando-os através da rede de comunicação para o cliente. Estando este a par da organização dos dados e do protocolo dos dados, resta-lhe ler e organizar os dados recebidos de forma a responder ao seu propósito.

Os exemplos descritos por Selonen mostram que serviços *Web* capazes de para armazenar e fornecer informação alojada num servidor. Esta necessidade é igual à existente neste projeto SolAR. Este caso serviu de inspiração e exemplo prático para a adoção de uma arquitetura semelhante, centrada em serviços, recorrendo a JSON para minimizar os custos de comunicação de informação.

Outro exemplo de utilização de serviços *Web* e localização numa aplicação Android é o TrafficPulse [22]. Este serviço pretende ajudar os utilizadores automobilistas a perceber as vias que se encontram mais congestionadas, recorrendo ao *feedback* dos utilizadores. Para armazenar essa informação de várias origens, os autores precisaram de ter uma base de dados central, à qual acedem através de um serviço *Web*. Dado que as aplicações móveis nem sempre têm cobertura de rede, recorreu-se a um armazenamento de dados local utilizando a tecnologia SQLite. Esta característica permite retirar alguma carga ao servidor, visto que o número de pedidos diminui.

O Traffic Pulse mostra a utilização de serviços *Web* para disponibilizar uma grande quantidade de dados a um cliente móvel. Mostra também as vantagens de, aliado ao recurso de um servidor remoto, ter dados armazenados numa *cache* local recorrendo ao SQLite. Neste projeto foi também necessário guardar localmente alguns dados, para que não se sobrecarreguem os serviços *Web* e para mitigar eventuais falhas de rede.

2.3 A aplicação móvel SolAR

Este trabalho teve como ponto de partida uma aplicação de RA, denominada de SolAR. Esta aplicação foi sendo sucessivamente desenvolvida ao longo dos anos por várias pessoas, nomeadamente José Pedrosa [23], Silvana Silva [24] e Carolina Meireles, no âmbito de trabalhos de mestrado. Esta aplicação recorre a dados de radiação solar incidente sobre o pavilhão C6 da FCUL, fornecidos por elementos do Departamento de Engenharia Geofísica, Geográfica e Energia (DEGGE) da FCUL. A aplicação mostra ao utilizador uma visualização dos dados sobreposta às paredes do edifício, permitindo uma perceção mais fiel de quais as fachadas que recebem mais ou menos radiação. Uma das funcionalidades em desenvolvimento na aplicação é o apoio à colocação de painéis fotovoltaicos nas fachadas, auxiliando no ato da decisão do local onde será mais rentável a sua colocação.

No momento em que este PEI se iniciou, era possível utilizar um dispositivo móvel com a aplicação, apontá-lo a uma fachada do C6 e, recorrendo à RA, visualizar os respetivos níveis de radiação solar, com base num ficheiro pré-carregado. Na Figura 1 é apresentada uma captura de ecrã da SolAR. Nessa imagem podemos verificar como é feita a apresentação dos dados.

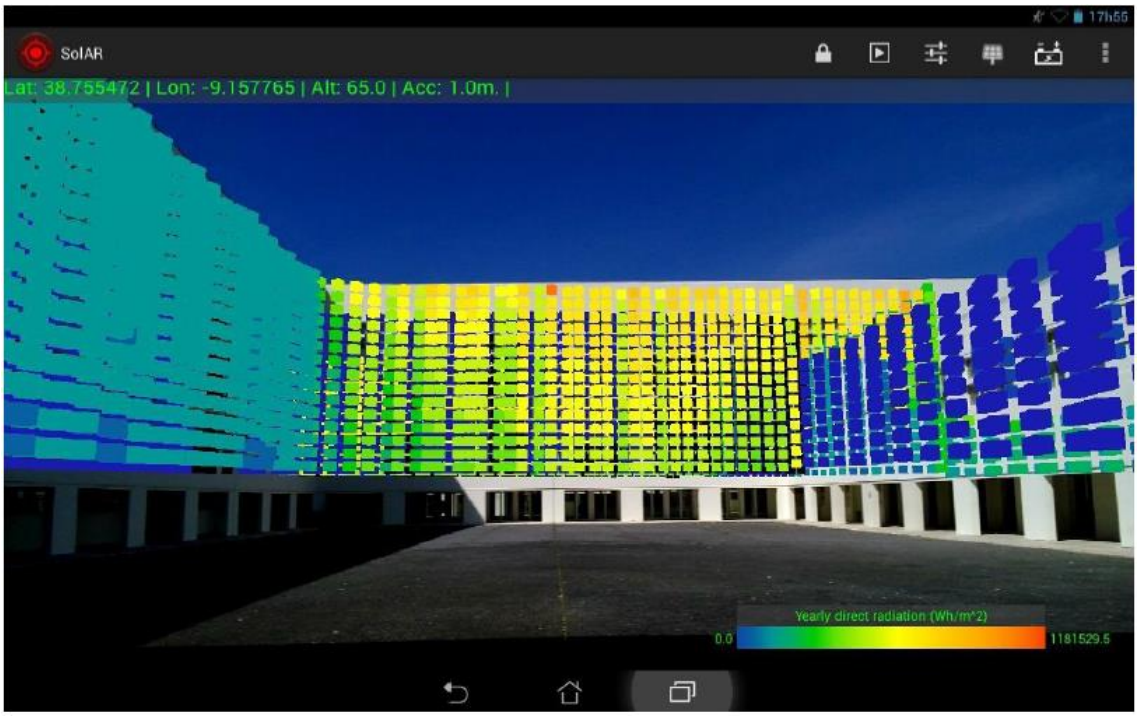

Figura 1. Aplicação móvel de realidade aumentada SolAR mostrando dados de radiação solar no edifício C6 da FCUL [24].

Na aplicação é possível selecionar a opção de painel solar, permitindo ao utilizador escolher um ponto para colocação de um painel e deslocá-lo ao longo da fachada. Contudo, quando se iniciou este projeto, não existiam ainda dados sobre os painéis e dados estatísticos sobre a sua utilização no ponto escolhido.

Assim, um dos aspetos que este trabalho pretendeu concretizar, o armazenamento de dados sobre painéis solares existentes no mercado e disponibilização desta informação a uma aplicação como a SolAR.

A aplicação recorre a um ficheiro local onde armazena toda a coleção de dados sobre os quais trabalha. A análise detalhada do ficheiro é feita na Secção 3.4 .

Eis um exemplo da primeira e segunda linha do ficheiro de dados, correspondentes aos rótulos dos campos e a uma linha de dados:

```
"FID_";"ID";"X";"Y";"Zdem";"svf";"cota_ponto";"a_Rdir";"a_R
dif";"a_Rglobal";"a_nhsombra";"a_nhtotal";"Edifico";"cod_fa
ch";"lat";"long_";"Altitude"
;178.000000;-89132.989000;-100755.304000;97.830000;0.306200
;82.830000;1358.334800;185932.825000;187291.159800;4027.000
000;4030.000000;"C6";155.000000;38.756198;-9.158519;136.229
995
```

Ao iniciar a aplicação, esta carrega o conteúdo do ficheiro para memória principal onde depois trabalha sobre eles. Do ponto de vista de otimização, estão a ser utilizados mais recursos do que os necessários, pois não é necessário carregar o sistema com dados sobre edifícios que estão longe do dispositivo, ou que estão tapados por obstáculos.

Neste projeto, pretendeu-se que a informação de radiação solar fosse armazenada numa base de dados espacial, minimizando o carregamento de dados desnecessários, e também retirando a responsabilidade do utilizador ter de saber onde e como estão armazenados os dados, e também que dados tem disponíveis no momento. A aplicação solicitará ao serviço *Web* os dados, indicando a sua posição no espaço, e o serviço fica encarregue de devolver a informação necessária para o correto funcionamento do SolAR.

2.4 Sumário do capítulo

Neste capítulo foram introduzidos alguns conceitos que surgem no âmbito deste trabalho. Foi também enunciado algum trabalho realizado por outras equipas de investigação. Ambos serviram para confirmar que exequibilidade do trabalho proposto, sugerir

abordagens e evitar erros já apontados. Por fim, apresentou-se a aplicação SolAR, a aplicação móvel de realidade móvel que surge de base ao trabalho que se vai detalhar no capítulo seguinte.

No capítulo seguinte descreve-se o trabalho realizado, explicando a concretização dos três principais objetivos propostos. Por fim é apresentada uma análise aos resultados gerados pelos testes aos serviços *Web* e ao *middleware*.

Capítulo 3

Trabalho realizado

Neste capítulo é exposto o trabalho realizado para a concretização dos três principais objetivos propostos. Em primeiro lugar é apresentada uma visão de alto nível de todos os componentes desenvolvidos e como eles se relacionam. Em seguida, são detalhadas as concretizações dos serviços *Web* e a componente de *middleware*, bem como as alterações efetuadas à aplicação SolAR para que esta passasse a fazer uso do *middleware* construído. Por fim são apresentados os resultados da avaliação dos objetivos e tiradas conclusões acerca dos valores obtidos.

3.1 Visão geral

O desenvolvimento deste projeto foi constituído por três fases. Numa primeira fase, foi desenhada uma base de dados, carregados os dados sobre radiação solar e gerados os campos geográficos necessários a uma pesquisa baseada na posição do utilizador. É possível aceder aos dados através da interface gráfica do PostgreSQL, utilizando a consola SQL ou criando uma ligação ao PostGIS utilizando o QGIS. Com o QGIS é possível gerar visualizações 2D ou 3D dos pontos existentes na base de dados, permitindo visualizar todos os dados, selecionar partes dos mesmos, e, se desejável, editar incorreções encontradas.

A segunda fase passou pela para construção de múltiplos serviços *Web* para dar acesso aos dados armazenados. Concluídas as duas primeiras fases, torna-se possível a qualquer aplicação cliente consiga consultar os dados do projeto SolAR.

A realização da terceira fase do projeto prendeu-se com a adaptação da aplicação móvel SolAR para que execute pedidos dos dados através da rede no momento em que

precise deles. Caso não exista ligação à internet, a aplicação mantém o uso dos dados armazenados em *cache*.

Cadeia de valor

Uma cadeia de valor representa um processo de negócio em que são modelados todos os instrumentos e intervenientes da conceção de um produto final. A análise de todos os componentes de uma cadeia de valor permite identificar elementos-chave no sucesso de um produto ou os pontos que determinaram o insucesso deste.

Analisando a cadeia de valor de uma *Application Programming Interface* (API), para que os serviços prestados tenham sucesso e acrescentem valor, é necessário ter em atenção os seguintes aspetos: a utilidade da informação e serviços disponibilizados e a qualidade das aplicações que façam uso destes [25].

Em primeiro lugar, a informação disponibilizada pelos serviços deve ser rica, de forma a atrair a maior quantidade possível de profissionais que criem aplicações que façam uso destes serviços. Conforme aumente a visibilidade dos serviços, maior será o proveito gerado pela informação possuída pelo detentor do ativo.

Em segundo lugar, mas igualmente importante, são os programadores e as aplicações que estes constroem. A qualidade das aplicações que fazem uso da API, fazem com que os ativos, ou seja, a informação fornecida, tenham utilidade para os utilizadores finais. Sem aplicações que permitam que a informação seja utilizada e que traga benefícios ao utilizador final, não importa ter um ativo muito forte pois este não chega a ter visibilidade e utilidade para os clientes finais. Caso as aplicações criadas não possuam qualidade suficiente, os utilizadores dificilmente as utilizam e a cadeia quebra-se.

No entanto, de nada vale que se tenha muitas aplicações que experimentem os serviços se estes não acrescentarem valor a estas e as tornem úteis. Rapidamente os programadores migram para outra API se esta for mais apelativa e trouxer mais benefícios para o utilizador final e rentabilidade para a entidade que desenvolve a aplicação cliente. Esta migração é facilitada por uma enorme concorrência no setor das TI. E também, porque as API são desenvolvidas para que sejam fáceis de experimentar e incorporar nas aplicações, devido a uma grande padronização de uso.

A Figura 2 ilustra a cadeia de valor em que este trabalho de PEI tem suporte, adaptada de um diagrama em Jacobson [25].

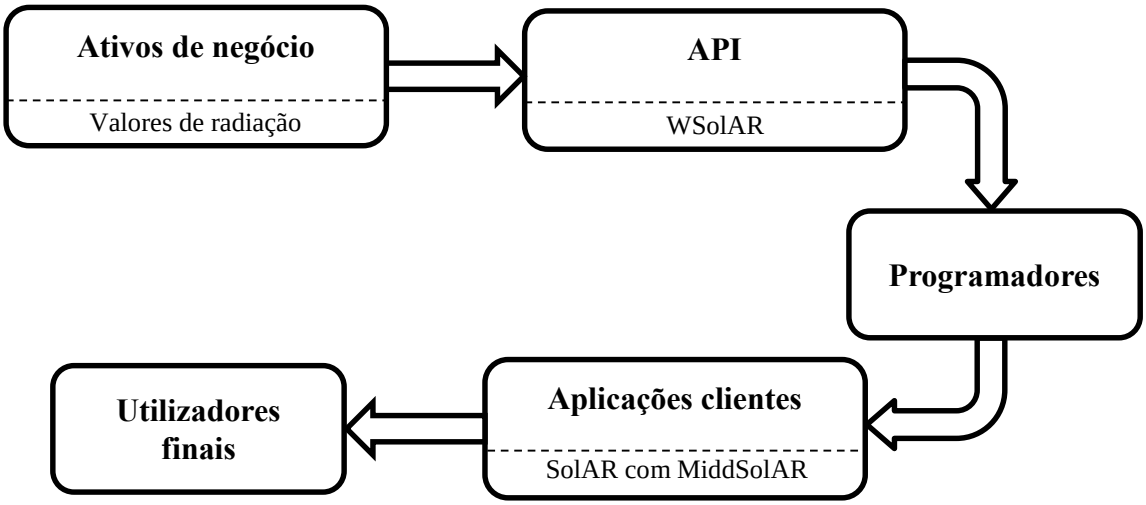

Figura 2. Cadeia de valor que dá suporte a este projeto.

A cadeia começa pelos ativos de negócio que, neste projeto, são os valores de radiação solar incidente em fachadas de edifícios. Esta continua para o método de disponibilização dos ativos, aqui fornecidos através de uma API de serviços *Web*.

Nem sempre quem desenvolve a API de serviços, está envolvido no desenvolvimento de uma aplicação cliente. No entanto, neste projeto de PEI, existiu também o desenvolvimento do MiddSolAR, *middleware* para que a aplicação cliente SolAR tenha usufruto da API WSolAR.

Finalizando a cadeia, o utilizador final é a pessoa que interage com o SolAR, que pode ser um técnico de painéis solares, um investigador ou um utilizador comum.

3.2 Arquitetura do sistema

O sistema construído baseou-se numa arquitetura com três camadas: cliente, onde se inserem todo o tipo de aplicações que utilizam dos serviços; servidor aplicacional, onde se encontram os serviços *Web*; e dados, correspondente à base de dados espacial.

A Figura 3 mostra um diagrama de alto nível da arquitetura.

Figura 3. Arquitetura de sistema baseada em três camadas principais: cliente, servidor aplicacional e dados.

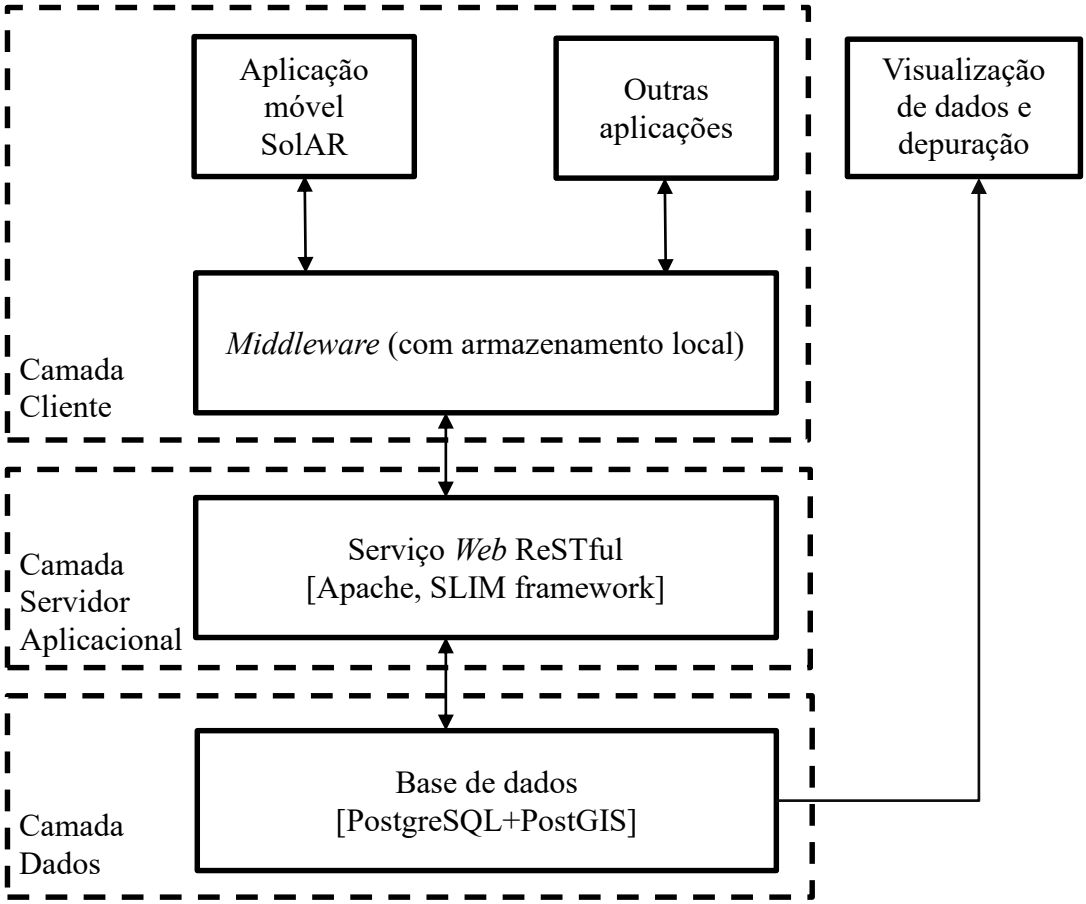

Figura 4. Arquitetura do sistema e interações no ecossistema desenvolvido.

De cima para baixo, podem observar-se na Figura 4 as mesmas três camadas presentes na Figura 3, agora mais detalhadas. Na camada cliente estão as aplicações que fazem uso da camada de servidor aplicacional. Nesta camada está a aplicação móvel SolAR, que faz uso do *middleware* MiddSolAR e o próprio *middleware*, que se liga à camada de servidor aplicacional para obter dados. Considera-se também a visualização de dados e depuração como pertencente à camada cliente pois esta faz uso dos dados, no entanto esta ferramenta interliga-se diretamente ao SGDB espacial, não precisando dos

serviços *Web*. Em seguida tem-se a camada de servidor aplicacional, onde se encontra definida a lógica funcional dos serviços *Web* que fornecem dados às aplicações cliente. Por último, a camada de dados, onde se encontra a base de dados espacial e toda a informação disponibilizada.

Na Figura 5 são expostas as interações entre alguns dos componentes deste trabalho. Nomeadamente, da camada cliente, a aplicação cliente e o *middleware* que serve de interface aos serviços e o serviço *Web* utilizado pelo *middleware*, referente à camada de servidor aplicacional.

3.3 Ambiente de desenvolvimento

Para a camada de dados foi preciso um SGBD com boa escalabilidade de armazenamento, robusto em termos de desempenho e com a capacidade de armazenar e trabalhar com dados espaciais. O PostgreSQL [26] e a sua extensão espacial PostGIS [26] foi o SGBD que melhor se adaptou às necessidades existentes pois a sua utilização é gratuita e é uma tecnologia já conhecida por parte de quem desenvolveu o projeto.

Para a camada de servidor aplicacional escolheu-se o Apache [27] e a linguagem PHP para executar a lógica no servidor [28]. Ambas as tecnologias são largamente utilizadas no mercado e contam com uma boa comunidade de apoio. A biblioteca Slim [29], para uso no código PHP, foi adotada para auxiliar a implementação dos serviços *Web*. Para encapsulamento dos dados a serem transmitidos escolheu-se o formato JSON.

Na camada de *middleware* foi utilizada o Java [30], que é a linguagem de programação para construção de aplicações nativas em Android e, por questões de compatibilidade com a estrutura do projeto SolAR já existente, o IDE utilizado foi o Eclipse ADT (*Android Development Tools*).

Para auxiliar a visualização e depuração dos dados existentes na base de dados utilizou-se o QGIS [31] e a sua extensão Qgis2threejs [32]. Com o propósito de analisar a informação gerada num ficheiro de registo de eventos, foi utilizada a linguagem R [33]. De referir ainda o uso de ficheiros em formato GPX, para depuração e análise das posições registadas durante a execução do *middleware*.

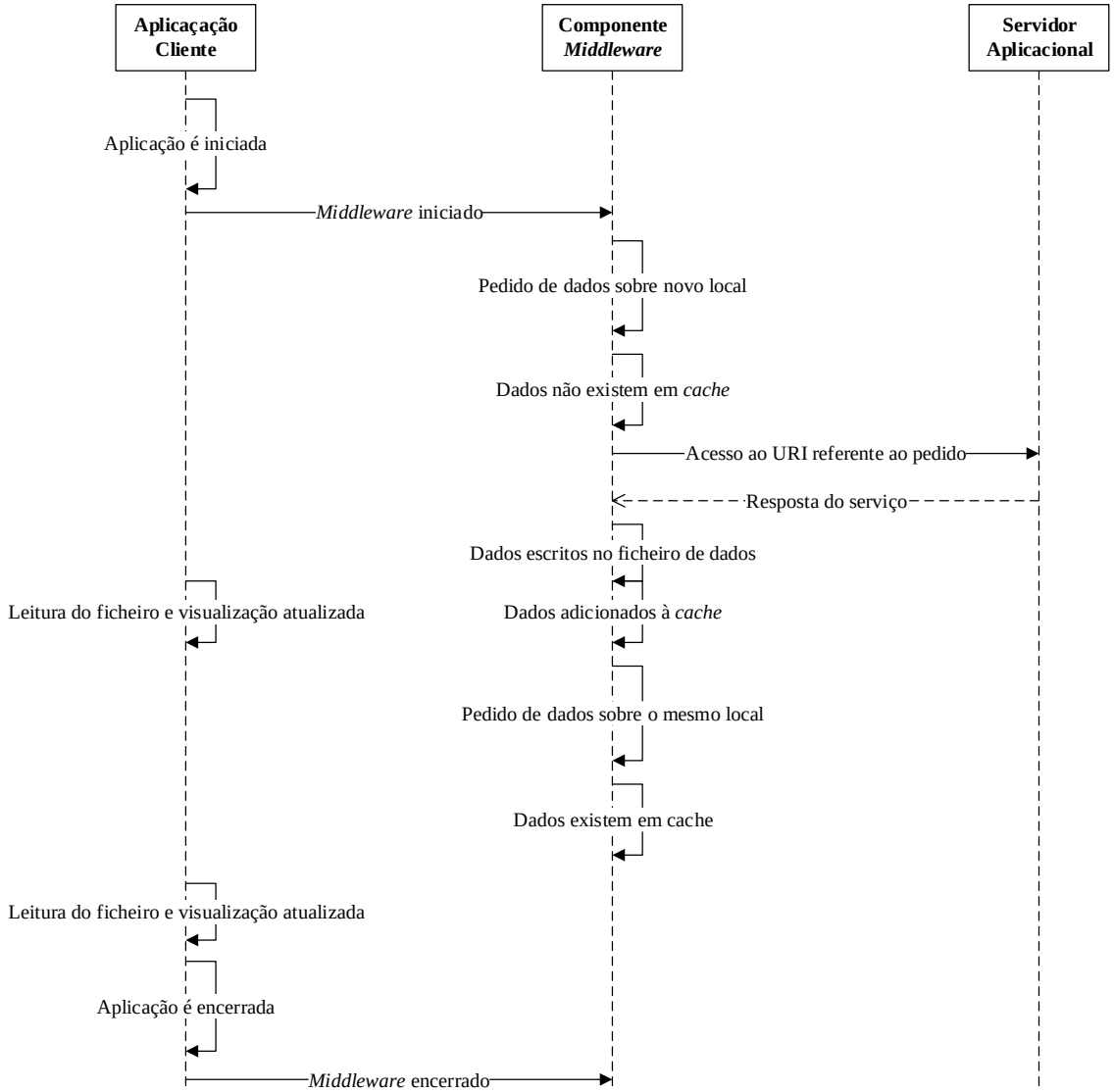

Figura 5. Diagrama de interação entre dois componentes da camada cliente, a aplicação SolAR e o *middleware* MiddSolAR, e a camada de servidor aplicacional.

3.4 Descrição, análise e visualização dos dados

Inicialmente trabalhava-se apenas com dados referentes ao C6 que estavam a ser usados pela aplicação SolAR. Tendo em vista o aumento do raio da ação do sistema e da escala do serviço, foram agendadas reuniões com elementos do DEGGE da FCUL que forneceram dados referentes às fachadas de todos os edifícios da FCUL, tendo também sido incluídos novos campos, ou seja, novos atributos em cada ponto.

Na Tabela 1 podemos verificar as variáveis existentes numa primeira fase, incluídas no ficheiro inicial da aplicação SolAR. As novas variáveis recebidas, integram um novo ficheiro. O ficheiro agora fornecido, inclui todos os pontos incluídos no ficheiro inicial, ou seja, inclui os dados do edifício C6 juntamente com os outros edifícios da FCUL. A coluna "em uso?" indica se a aplicação SolAR requer a variável para executar a sua atividade.

Tabela 1. Comparação entre os campos presentes no ficheiro inicial da aplicação SolAR e os novos dados disponibilizados pelo DEGGE.

Variável	**Presente em:**	**Em uso?**
FID_	Ambos	Não
Join_Count	Novo ficheiro	Não
ID	Ambos	Não
X	Ambos	Sim
Y	Ambos	Sim
Zdem	Ambos	Não
svf	Ambos	Não
cota_ponto	Ambos	Não
a_Rdir	Ambos	Sim
a_Rdif	Ambos	Sim
a_Rglobal	Ambos	Sim
a_nhsombra	Ambos	Sim
a_nhtotal	Ambos	Sim
a_Rglo_kW	Novo ficheiro	Não
Join_Cou_1	Novo ficheiro	Não
LEFT_FID	Novo ficheiro	Não
RIGHT_FID	Novo ficheiro	Não
Azimuth	Novo ficheiro	Não

x_start	Novo ficheiro	Não
y_start	Novo ficheiro	Não
x_end	Novo ficheiro	Não
y_end	Novo ficheiro	Não
Edifício	Ambos	Sim
cod_fach	Ambos	Sim
cota_min	Novo ficheiro	Não
x_reb	Novo ficheiro	Não
z_reb	Novo ficheiro	Não
Lat	Ficheiro inicial	Não
Long_	Ficheiro inicial	Não
Altitude	Ficheiro inicial	Sim

Numa reunião com a Prof. Paula Redweik do DEGGE, o significado de cada campo foi explicado, tendo produzido o seguinte dicionário onde podemos ver o nome do campo, o seu significado e exemplo, e a unidade em que se encontra, caso aplicável:

- **FID_** é um campo que se encontra vazio;
- **ID** corresponde a FeatureID. Este identifica um hiper ponto que identifica uma coluna de pontos. Exemplo: 178.000000;
- **X** é a coordenada cartesiana no eixo dos xx e encontra-se no formato estabelecido em ETRS89-PT-TM06. Exemplo: -89132.989000 metros;
- **Y** é a coordenada cartesiana no eixo dos yy e encontra-se no formato estabelecido em ETRS89-PT-TM06. Exemplo: -100755.304000 metros;
- **Zdem** é a cota de pontos LIDAR z do telhado. Corresponde à distância entre o nível médio da água do mar e o telhado. Exemplo: 97.830000 metros;
- **svf** é uma sigla para *sky view factor* e refere a percentagem de céu visível a partir do ponto. Exemplo: 0.306200;
- **cota_ponto** é um campo relacionado com a altitude em que o ponto se encontra. Este valor é medido a partir do nível médio da água do mar. Esse nível encontra o seu ponto de referência na cota zero, medida por um marégrafo localizado na baía de Cascais. Exemplo: 82.830000 metros;

- **a_Rdir** corresponde à radiação solar direta anual, proveniente da sua exposição à fonte de radiação. Tem como unidade de medida watt-hora por metro quadrado (Wh/m^2). Exemplo: 1358.334800;
- **a_Rdif** corresponde à radiação solar difusa anual, contabilizando possíveis reflexões incidentes. Tem como unidade de medida watt-hora por metro quadrado (Wh/m^2). Exemplo: 185932.825000;
- **a_Rglobal** soma dos valores de a_Rdir e a_Rdif e denomina-se de radiação global anual. Exemplo: 187291.159800;
- **a_nhsombra** representa o número de horas de sombra anual. Exemplo: 4027.000000;
- **a_nhtotal** representa o número de horas potenciais de iluminação. Ou seja, o máximo de horas de luz natural possível durante o ano. Exemplo: 4030.000000;
- **a_Rglo_kW** corresponde ao valor encontrado em a_Rglobal convertido para Kilowatt (kW). Exemplo: 258.708169;
- **Azimuth**, ou azimute, refere a orientação da fachada em relação ao norte geográfico. Este valor é medido em graus. Exemplo 61.666675;
- **x_start**, **y_start**, **x_end** e **y_end** definem coordenadas cartesianas de referência sobre a fachada onde se encontra o ponto em questão. Exemplo: -89172.508800, -100710.446100, -89110.662900, -100677.099100;
- **Edificio** identifica o nome da infraestrutura onde se inclui o ponto referido. Exemplo: C6, C8, C3C4C5, C1, Museu, Outros;
- **cod_fach** serve de identificador único para a fachada onde se encontra o ponto. Exemplo: 155.000000;
- **cota_min** refere-se ao valor mínimo de cota que se encontra na constituição dos pontos da fachada. Exemplo: 80.849998 metros;
- **x_reb** e **z_reb** representam uma transformação dos valores de coordenadas rebatidos para um plano. Exemplo: 1.526430 e 0.100001;
- **Lat**, **Long_** e **Altitude** correspondem, respetivamente, à latitude, longitude e altitude em metros do sistema geodésico WGS84. Exemplos: 38.756198, -9.158519 e 136.229995;

- Os campos **FID_**, **Join_Count**, **Join_Cou_1**, **LEFT_FID**, **RIGHT_FID** não são relevantes para este trabalho.

Visto que o número de pontos e as variáveis que lhes estão associadas serem em grande quantidade, procedeu-se a uma análise estatística utilizando a linguagem R, com o objetivo de facilitar a análise dos dados. O resultado dessa análise é apresentado na Tabela 2, onde foram omitidas variáveis vazias – FID_ – e valores não numéricos como o caso do campo Edificio.

Tabela 2. Análise estatística dos dados sobre radiação. Neste novo ficheiro existem novas variáveis, comparativamente às inicialmente em uso pela aplicação SolAR. Existem também novos dados, referentes aos restantes edifícios da FCUL.

Variável	**Mínimo**	**1º Quartil**	**Mediana**	**Média**	**3º Quartil**	**Máximo**
Join_Count	0.0000	1.0000	1.0000	0.9972	1.0000	1.0000
ID	21	1132	2965	2779	4222	6278
X	-89171	-89071	-88992	-89000	-88946	-88801
Y	-100889	-100790	-100739	-100726	-100666	-100478
Zdem	77.17	88.12	95.67	93.23	97.86	108.81
svf	0.1008	0.3821	0.4191	0.4225	0.4570	0.9991
cota_ponto	75.70	82.56	85.88	86.80	90.83	108.70
a_Rdir	0	1024	56314	211724	401334	1223010
a_Rdif	61456	231921	254318	256481	277357	606371
a_Rglobal	61456	260741	335315	468207	652979	1698586
a_nhsombra	184	2194	3472	3048	3988	4030
a_nhtotal	4030	4030	4030	4030	4030	4030
a_Rglo_kW	61.46	260.74	335.31	468.21	652.98	1698.59
Azimuth	23.68	69.04	248.68	199.71	249.23	343.33
x_start	-89173	-89072	-88989	-89989	-88947	-88801
y_start	-100878	-100799	-100743	-100725	-100667	-100477

x_end	-89173	-89070	-88997	-89001	-88954	-88801
y_end	-100878	-100781	-100745	-100722	-100669	-100477
cod_fach	0.0	102.0	156.0	168.9	222.0	345.0
cota_min	75.70	77.30	77.76	79.46	82.35	87.10
x_reb	0.08674	6.76816	14.92990	23.45932	34.19040	101.30800
z_reb	0.000	2.970	6.040	7.336	11.000	33.000

Neste projeto, recorreu-se ao software QGIS e à sua extensão Qgis2threejs para visualizar em 3D os pontos com dados de radiação solar. A Figura 6 mostra o ambiente de trabalho deste *software* após carregado o ficheiro de dados.

Para gerar esta visualização, foi instalada a extensão Qgis2threejs. Esta extensão utiliza a biblioteca de Javascript three.js e, recorrendo a um navegador *Web* com suporte para WebGL, permite visualizar os dados numa vista a três dimensões. Após a sua instalação, fica disponível um atalho para esta extensão na barra de ferramentas superior do QGIS, ilustrado na Figura 7.

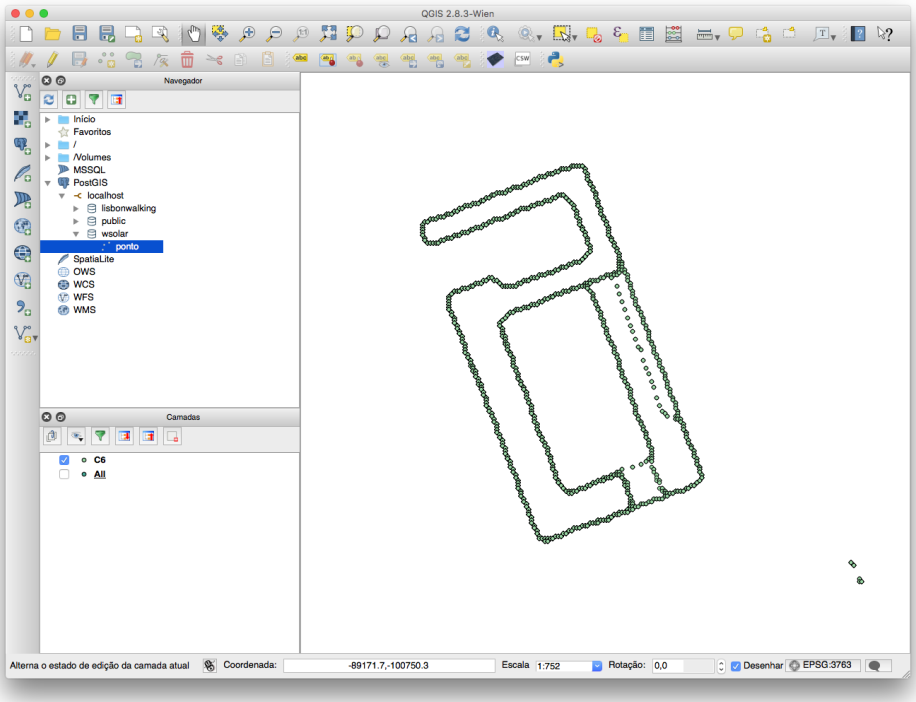

Figura 6. Ambiente de trabalho do QGIS mostrando a camada em que se filtraram os pontos referentes ao edifício C6 da FCUL.

Figura 7. Barra de ferramentas referente à extensão Qgis2threejs.

Ao carregar no ícone da esquerda surge uma caixa de diálogo como apresentada na Figura 8. Nesta caixa, é configurado o nome do ficheiro de dados a visualizar, é também definida uma personalização dos objetos 3D, e é também especificado o caminho de destino dos ficheiros gerados. São gerados vários ficheiros, sendo que para abrir a visualização, utilizamos o ficheiro ao qual demos o nome.

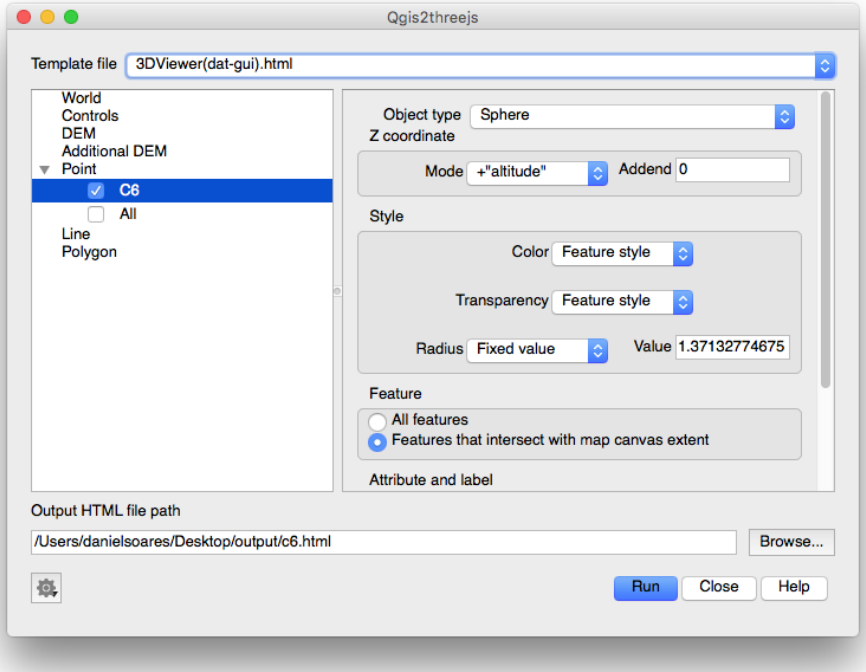

Figura 8. Caixa de diálogo em que se efetuam as configurações para a representação 3D dos dados.

Ao clicar em Run, dá-se início à criação de ficheiros que permitem a visualização e o resultado é mostrado no navegador *Web* pré-definido. A Figura 9 mostra esse resultado final. Esta simulação tende a consumir uma qualidade relevante de recursos e é útil restringir a quantidade de pontos a visualizar.

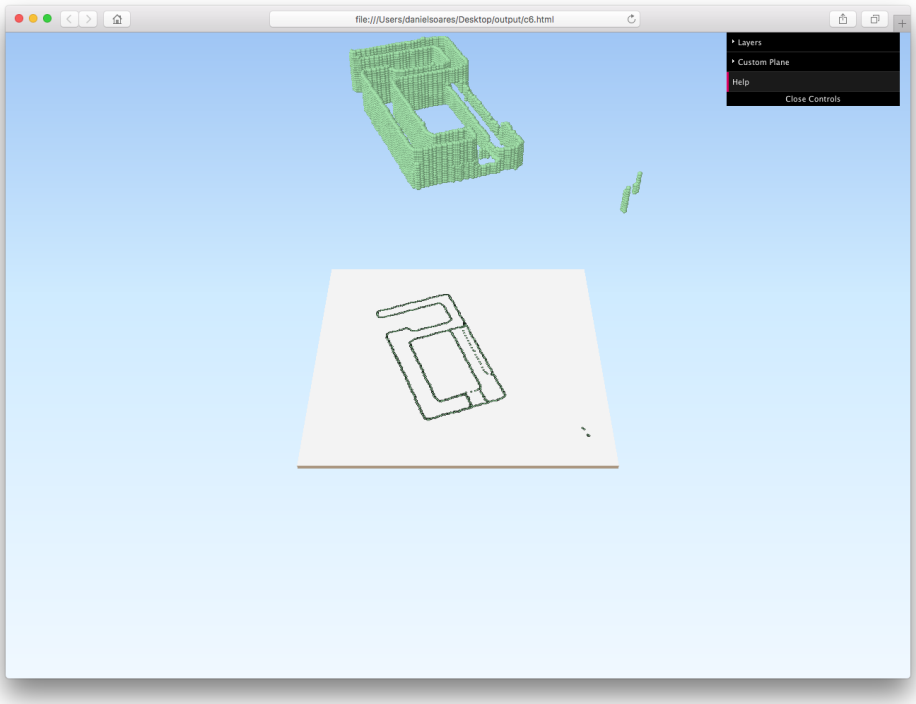

Figura 9. Visualização, gerada pelo QGIS, do edifício C6 da FCUL a partir dos dados de radiação solar.

3.5 Serviços *Web* WSolAR

A concretização deste objetivo compreendeu construir uma base de dados relacional com os dados sobre radiação solar, bem como sobre painéis solares, e o acesso a estes dados através de serviços *Web*. Os dados incluem a sua localização geográfica de modo a permitir pesquisas de dados existentes numa dada área. Os serviços *Web* incluem a criação de novos dados, leitura, alteração e remoção de dados existentes.

3.5.1 Esquema da base de dados

Para a modelação dos dados de radiação e informação sobre painéis solares, foi elaborado o modelo de dados apresentado na Figura 10.

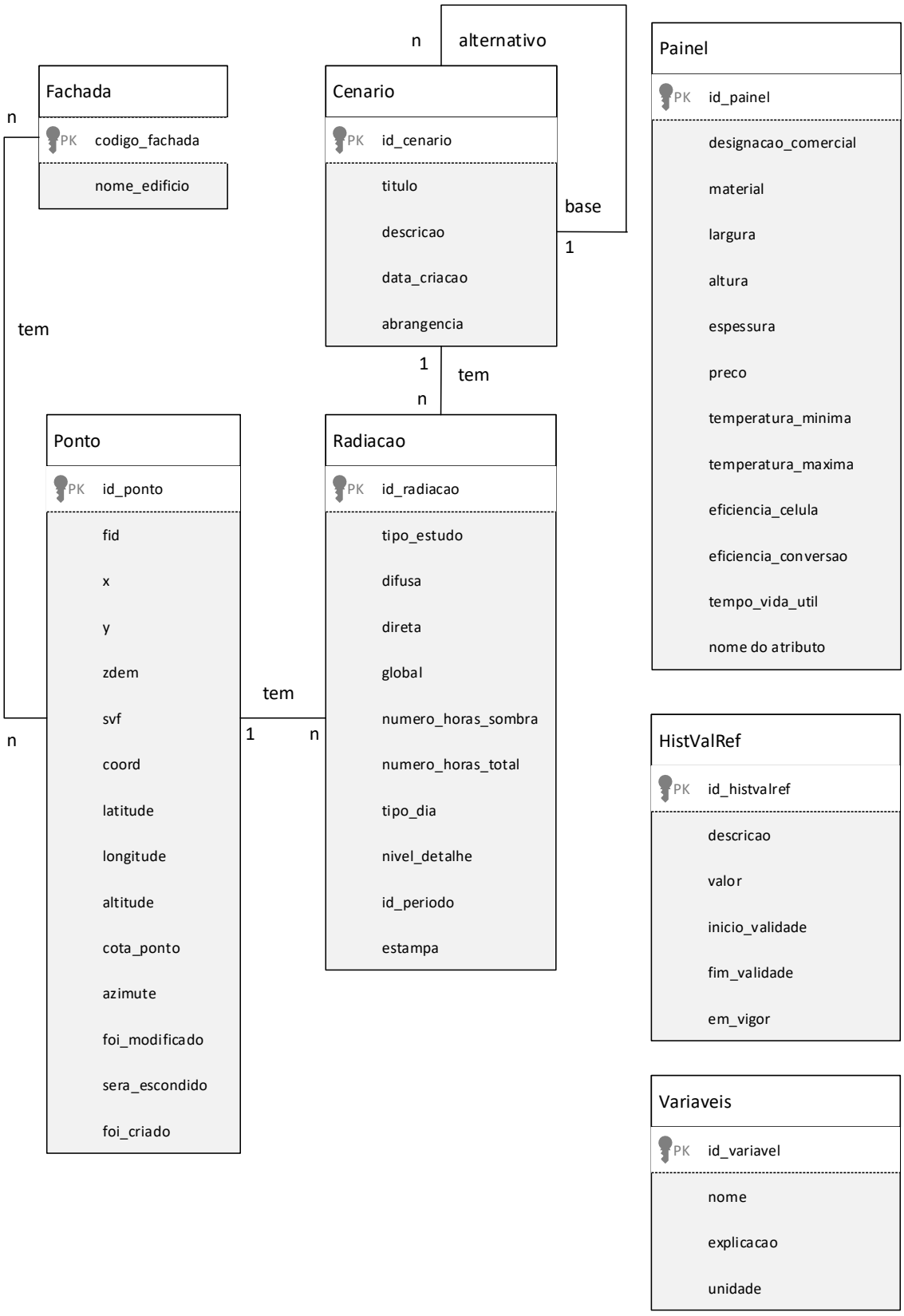

Figura 10. Modelo de dados sobre radiação, cenários e painéis solares.

Este modelo de dados compreende as seguintes entidades:

- Fachada: corresponde a uma parede completa ou apenas a uma porção. Está identificada através de um identificador que existia nos dados fornecidos pelo DEGGE. A fachada tem também identificado o edifício a que pertence.
- Ponto: representa um ponto dentro da fachada e inclui dados sobre radiação solar. A entidade Ponto é responsável por armazenar a posição geográfica e geométrica do ponto. Guarda também a indicação se o ponto foi originado fora do contexto dos dados fornecidos ou se teve os seus valores corrigidos.
- Radiação: armazena os valores de radiação referentes a um ponto associado. Guarda ainda o tipo de estudo que gerou a informação registada.
- Cenário: pretende modelar a existência de valores de radiação gerados através de cálculos de situações simuladas ou futuras.
- Painel: modela as especificações de um painel fotovoltaico.
- HistValRef: representa um histórico de valores de referência que se espera ser inalterados durante um intervalo de tempo.
- Variáveis: é um dicionário de variáveis presentes em campos de outras entidades, como o Ponto ou a Radiação.

3.5.2 Serviços *Web*

Para que se consiga obter a informação armazenada na base de dados espacial foram implementados múltiplos serviços *Web* destinados a fachadas, radiação, pontos, painéis e variáveis armazenadas, recorrendo-se a uma biblioteca de forma a agilizar o processo de identificação do URL e verbo HTTP do pedido. Essa biblioteca foi a SLIM Framework, a qual introduz uma camada de abstração entre o pedido e a sua lógica operacional, recolhendo os pedidos HTTP recebidos para uma lógica de alto nível.

Em seguida são apresentados os serviços desenvolvidos, separados de acordo com o seu predicado de forma a facilitar a sua leitura por parte do leitor.

Apresentação dos serviços *Web*

Na Tabela 3 apresentam-se os serviços referentes à consulta e inserção de dados relativos a fachadas de edifícios. Com o serviço em que se pede informação, é possível identificar as fachadas existentes, bem como os edifícios sobre os quais existe informação. Existe também a possibilidade de adicionar fachadas, e como consequência disso, adicionar de novos edifícios.

Tabela 3. Serviços *Web* referentes a fachadas.

[Verbo HTTP] URI **Descrição da funcionalidade do serviço**
[GET] `/fachadas` Devolve as fachadas presentes no sistema.
[POST] `/fachadas` Insere uma fachada no sistema.

Na Tabela 4 apresentam-se os serviços referentes a pontos e cenários. Neste conjunto de serviços sobre pontos encontram-se os utilizados pela aplicação SolAR. Os serviços de edição coletiva permitem que um técnico recolha os pontos existentes, os edite, e corrija posições numa aplicação como o QGIS, e os consiga recarregar na base de dados. Neste projeto não se chegou a concretizar na aplicação móvel a noção de vários cenários, no entanto, os serviços *Web* e o modelo de dados para os implementar foram especificados.

Tabela 4. Serviços *Web* referentes a pontos.

[Verbo HTTP] URI **Descrição da funcionalidade do serviço**
[GET] `/pontos` Devolve todos os pontos presentes no sistema.
[GET] `/pontos/{variaveis}` Devolve as variáveis, indicadas como parâmetro, de todos os pontos presentes no sistema.
[GET] `/pontos/edifício/{nome}` Devolve todos os pontos referentes a um dado edifício.

[GET] `/pontos/{latitude}/{longitude}/{raio}` Devolve todos os pontos existentes dentro do raio indicado e tendo em conta a posição atual referida.
[GET] `/pontos/{latitude}/{longitude}/{raio}/contagem` Devolve a quantidade de pontos existentes dentro da área indicada.
[GET] `/pontos/{latitude}/{longitude}/{raio}/momento/{data}/{hora}` Devolve a quantidade de pontos numa dada área que foram alterados depois da data e hora indicadas.
[POST] `/pontos` Insere um ponto no sistema.
[GET] `/pontosparaeditar` Devolve a posição de todos os pontos presentes no sistema para facilitar a correção.
[POST] `/pontosparaeditar` Insere no sistema os pontos corrigidos.
[GET] `/cenarios` Devolve todos os cenários presentes no sistema.
[GET] `/cenarios/{id}/{lat}/{long}/{raio}` Devolve a informação de radiação calculada de uma dada área.
[GET] `/cenarios/{id}/{lat}/{long}/{raio}/{variaveis}` Devolve as variáveis escolhidas de radiação calculada de uma dada área.
[POST] `/cenarios` Insere um cenário no sistema.

Na Tabela 5 é apresentado o serviço *Web* sobre radiação, que permite carregar informação sobre radiação solar associada a um ponto existente no sistema.

Tabela 5. Serviços *Web* referentes a radiação.

[Verbo HTTP] URI **Descrição da funcionalidade do serviço**
[POST] `/radiacao` Insere informações referentes à entidade radiação no sistema.

Na Tabela 6 estão presentes os serviços *Web* que permitem visualizar todos os atributos sobre os painéis solares e filtrar os painéis conforme as restrições passadas em argumentos. Existem também serviços para editar individualmente cada atributo de um painel.

Tabela 6. Serviços *Web* referentes a painéis solares.

[Verbo HTTP] URI **Descrição da funcionalidade do serviço**
[GET] `/paineis` Devolve todos os painéis solares presentes no sistema.
[GET] `/paineis/{id}` Devolve informação sobre o painel solar indicado.
[GET] `/paineis/dimensoes/{largura}/{altura}/{espessura}` Devolve todos os painéis solares presentes no sistema com as dimensões indicadas.
[GET] `/paineis/preco/inferior/{valor}` Devolve todos os painéis solares presentes no sistema com um preço inferior ao indicado.
[GET] `/paineis/preco/superior/{valor}` Devolve todos os painéis solares presentes no sistema com um preço superior ao indicado.
[GET] `/paineis/preco/{minimo}/{maximo}` Devolve todos os painéis solares presentes no sistema com um preço entre os valores indicados.
[GET] `/paineis/temperatura/inferior/{valor}` Devolve todos os painéis solares presentes no sistema com uma temperatura de funcionamento inferior ao indicado.
[GET] `/paineis/temperatura/superior/{valor}` Devolve todos os painéis solares presentes no sistema com uma temperatura de funcionamento superior ao indicado.
[GET] `/paineis/temperatura/{minimo}/{maximo}` Devolve todos os painéis solares presentes no sistema com uma temperatura de funcionamento entre os valores indicados.
[GET] `/paineis/eficienciaCelula/{minimo}/{maximo}` Devolve todos os painéis solares presentes no sistema com uma eficiência de célula entre os valores indicados.
[GET] `/paineis/eficienciaConversao/{minimo}/{maximo}` Devolve todos os painéis solares presentes no sistema com uma eficiência de conversão energética entre os valores indicados.
[GET] `/paineis/validade/{minimo}/{maximo}` Devolve todos os painéis solares presentes no sistema com uma validade entre os valores indicados.
[POST] `/paineis` Insere um painel solar no sistema.
[PUT] `/paineis/{id}/designacaoComercial/{designacao}` Altera a designação comercial de um painel solar existente no sistema.
[PUT] `/paineis/{id}/material/{material}` Altera o material de um painel solar existente no sistema.

[PUT] `/paineis/{id}/largura/{largura}` Altera a largura de um painel solar existente no sistema.
[PUT] `/paineis/{id}/altura/{altura}` Altera a altura de um painel solar existente no sistema.
[PUT] `/paineis/{id}/espessura/{espessura}` Altera a espessura de um painel solar existente no sistema.
[PUT] `/paineis/{id}/preço/{preco}` Altera o preço de um painel solar existente no sistema.
[PUT] `/paineis/{id}/temperaturaMinima/{temperatura_minima}` Altera a temperatura mínima de um painel solar existente no sistema.
[PUT] `/paineis/{id}/temperaturaMaxima/{temperatura_maxima}` Altera a temperatura máxima de um painel solar existente no sistema.
[PUT] `/paineis/{id}/eficienciaCelula/{eficiencia}` Altera a eficiência de célula de um painel solar existente no sistema.
[PUT] `/paineis/{id}/eficienciaConversao/{eficiencia}` Altera a eficiência de conversão de um painel solar existente no sistema.
[PUT] `/paineis/{id}/tempoVidaUtil/{tempo_vida_util}` Altera o tempo de vida útil de um painel solar existente no sistema.
[DELETE] `/paineis/{id}` Elimina um painel do sistema.

Na Tabela 7 são apresentados os serviços *Web* que permitem a um utilizador consultar os atributos que existem sobre os pontos e verificar a sua explicação e unidade de medida. São ainda disponibilizados serviços para que seja possível editar, quer individualmente quer em conjunto, cada campo associado.

Tabela 7. Serviços *Web* referentes a variáveis de sistema.

[Verbo HTTP] URI **Descrição da funcionalidade do serviço**
[GET] `/variaveis` Devolve todas as variáveis presentes no sistema.
[GET] `/variaveis/{id}` Devolve os valores da variável com o identificador fornecido.
[POST] `/variaveis` Insere informações sobre uma variável no sistema.
[PUT] `/variaveis/{id}/nome/{nome}` Altera o nome de uma variável do sistema.
[PUT] `/variaveis/{id}/explicacao/{explicacao}` Altera a explicação de uma variável do sistema.
[PUT] `/variaveis/{id}/unidade/{unidade}` Altera a unidade de uma variável do sistema.

[DELETE] `/variaveis/{id}` Elimina uma variável do sistema.

Casos especiais de uso e interação dos serviços

Nas tabelas apresentadas, enumeram-se os vários serviços disponibilizados ao cliente. Praticamente todos são de interação imediata, ou seja, a aplicação cliente define o que necessita, faz o pedido e recebe a resposta. No caso específico de não ter a possibilidade de saber as variáveis que precisa, tem então que elaborar um pedido para que receba a lista de campos disponíveis. Só aí é que pode formular o pedido, consoante a informação que deseja. Esta interação é exemplificada no diagrama da Figura 11.

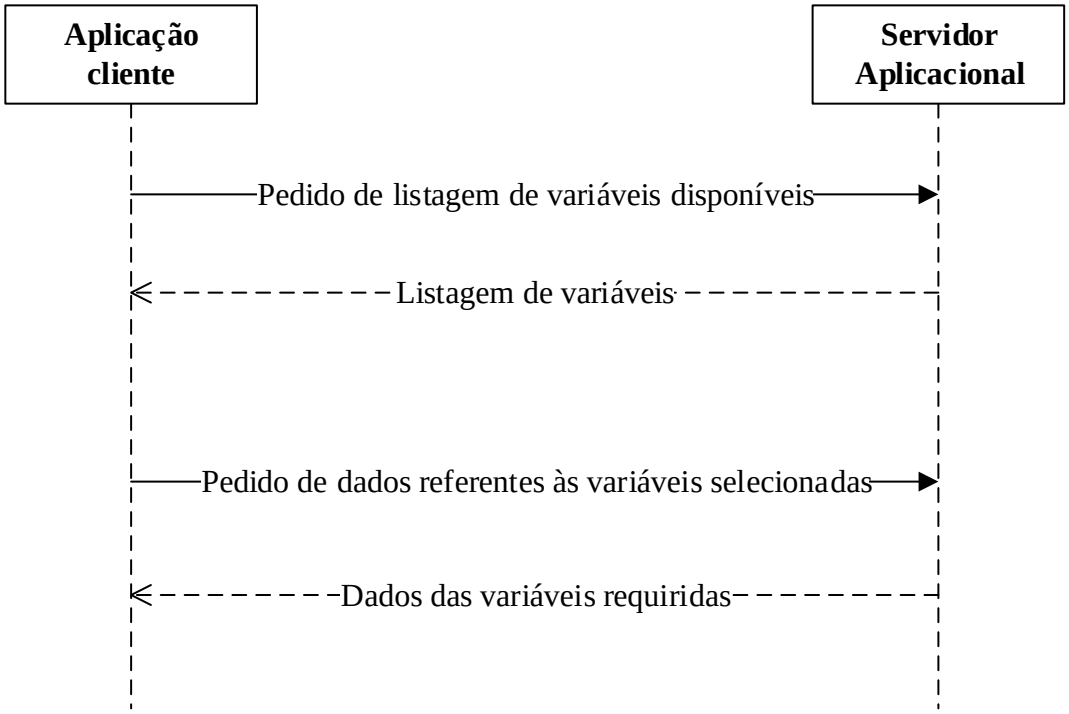

Figura 11. Exemplo de caso especial onde ocorrem interações entre a aplicação cliente e servidor. Neste caso o segundo pedido depende a resposta ao primeiro.

Para os outros serviços, o protocolo de comunicação é simples, bastando invocar um pedido e receber a resposta.

No caso de ocorrer um erro no processo do pedido ou interno no sistema, a resposta tentará informar o utilizador do sucedido. O diagrama presente na Figura 12 exemplifica a informação de erro cometido, enviado na resposta, por parte do cliente.

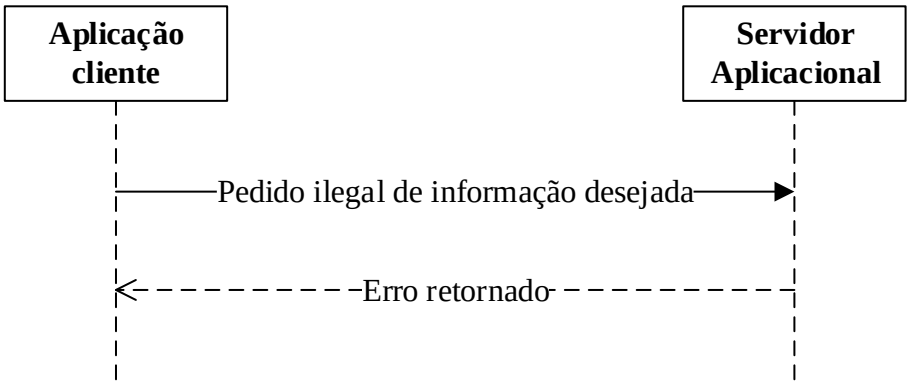

Figura 12. Exemplo de um pedido ilegal ao serviço *Web* e retorno do erro.

Deste modo, a camada cliente recebe informação explícita do tipo de erro de forma a conseguir ter dados para corrigir o pedido.

Página informativa sobre os serviços *Web*

Para facilitar a consulta, por parte de um programador, dos serviços criados para este projeto, foi criada uma página *Web* onde se apresentam todos os serviços. Ao aceder a esta página informativa, o programador pode ter a noção de toda a informação disponibilizada no WSolAR e rapidamente experimentar aceder a um serviço que lhe devolva dados presentes no serviço. Esta página pode ser consultada através do seguinte endereço:

`http:// solar.campus.ciencias.ulisboa.pt/info/`

A Figura 13 mostra a página inicial de exposição dos serviços *Web* na qual é possível consultar a descrição do projeto e a equipa de investigação envolvida no seu desenvolvimento.

Figura 13. Página inicial de apoio ao uso dos serviços *Web*.

Os serviços *Web* existentes podem ser consultados navegando pelos vários separadores, cada um de acordo com o seu tema correspondente. A Figura 14 mostra a página onde se encontram os serviços *Web* referentes a painéis solares. De forma análoga, podem ser consultados os serviços de outros temas.

Figura 14. Página de exposição dos serviços *Web* referentes a painéis solares.

Exemplificação de um pedido bem-sucedido

Para que a aplicação SolAR obtenha dados de radiação solar, precisa de executar um pedido ao serviço *Web*, utilizando o verbo GET, através do seguinte endereço:

`http://solar.campus.ciencias.ulisboa.pt/api/pontos`

O serviço irá devolver uma resposta, em formato JSON. Supondo que existe apenas informação referente a dois pontos, a resposta teria o seguinte aspeto:


```
{"pontos":
     [
          {
               "FID_":"3298",
               "X":"-88982.989",
               "Y":"-100741.304",
               "Zdem":"80.57",
               "svf":"0.43663",
               "cota_ponto":"80.57",
               "a_Rdir":"811362.8208",
               "a_Rdif":"265006.0475",
               "a_Rglobal":"1076368.856",
               "a_nhsombra":"1878",
               "a_nhtotal":"4030",
               "Edifico":"C1",
               "cod_fach":"199",
               "lat":"0",
               "long_":"0",
               "Altitude":"133.97"
          },
          {
               "FID_":"3299",
               "X":"-88982.989",
               "Y":"-100767.304",
               "Zdem":"84.34",
               "svf":"0.30805",
               "cota_ponto":"77.34",
               "a_Rdir":"33.98872",
               "a_Rdif":"186997.2404",
               "a_Rglobal":"187031.2293",
               "a_nhsombra":"4020",
               "a_nhtotal":"4030",
               "Edifico":"C3C4C5",
               "cod_fach":"113",
               "lat":"0",
               "long_":"0",
               "Altitude":"130.74"
          }
     ]
}
```

Neste caso, como se simula a existência de informação referente a apenas dois pontos, a resposta contém um vetor com dois objetos.

Após a recebida a resposta na sua totalidade, a aplicação SolAR processa a informação consoante as suas necessidades.

3.5.3 Cenário passo-a-passo de carregamento e correção de dados

Os dados existentes na base de dados tendem a não se alterar ao longo do tempo. No entanto, podem ocorrer alterações imprevistas, bem como erros na informação já carregada. De forma a evitar usar comandos SQL ou recarregar todos os dados novamente, é útil que existam formas práticas de atualizar a informação armazenada.

Foram criados *scripts* em PHP para facilitar a edição de dados recorrendo ao QGIS e às suas ferramentas de edição e visualização de dados. Em seguida, são enumerados os passos necessários para realizar esta tarefa:

1. Usar o ficheiro retrieveFileToEdition.php para gerar o ficheiro para ser editado.
2. O ficheiro vai ser gerado em ./processing/ficheiroPontosParaEditar.txt
3. Abrir o QGIS, carregar o ficheiro gerado e editar os dados. Os dados editados devem ser guardados no mesmo ficheiro.
4. Por fim, usar o loadCSVeditados.php para carregar os dados editados novamente para a base de dados.

Em alternativa, o técnico pode criar uma ligação direta ao PostGIS o QGIS e aceder à tabela espacial, neste caso a tabela Ponto, através deste modo. No entanto, só ficará disponível a tabela que contém o campo geográfico.

3.5.4 Exemplificação de uma *query* geográfica

Para fazer uso do campo geográfico `coord` introduzido na tabela de pontos (ver Figura 10), existem funções incluídas na extensão espacial de SQL que executam sobre esse campo. Eis um exemplo de uma *query* SQL, que devolve os pontos existentes dentro de uma circunferência:

```
// Latitude dada pelo sistema de posicionamento
// Longitude dada pelo sistema de posicionamento
// SRID dos dados fornecidos como argumento = 4326
// SRID dos dados armazenados na base de dados = 3763
SELECT *
FROM wsolar.ponto p, wsolar.fachada f, wsolar.radiacao r
```

```
WHERE p.pertence = f.codigo_fachada AND
p.tem = r.id_radiacao AND
ST_Point_Inside_Circle(p.coord,
ST_X(ST_Transform(ST_SetSRID(ST_Point(".$longitude.",".$lat
itude."), ".$received_srid."), ".$encoded_srid.")),
ST_Y(ST_Transform(ST_SetSRID(ST_Point(".$longitude.",".$lat
itude."), ".$received_srid."),
".$encoded_srid.")),".$raio.");
```

Como argumentos são necessárias a latitude e longitude do ponto central da circunferência e o raio desta. A latitude e longitude devem encontrar-se no sistema de coordenadas padrão utilizado pela maioria de dispositivos GPS, o WGS84 (SRID 4326). Internamente, os pontos encontram-se codificados no sistema de coordenadas cartesianas português, o ETRS89-PT-TM06 (SRID 3763).

3.5.5 Instalação dos serviços numa máquina virtual

De modo a disponibilizar os serviços *Web* ao público em geral e com uma capacidade de resposta que permita atender múltiplos pedidos simultaneamente, estes foram transferidos para os servidores do Centro de Informática da FCUL. A informação prestada pelos serviços pode ser acedida através do seguinte domínio:

```
http:// solar.campus.ciencias.ulisboa.pt/
```

Para que os serviços *Web* fossem colocados à disposição dos utilizadores, todo o *software* que os suporta teve de ser instalado e configurado numa máquina virtual com o sistema operativo CentOS.

Antes de iniciar a instalação, os seguintes comandos podem ser úteis:

```
nmap -sT -O localhost
ip addr show
rpm -qa
```

O primeiro comando visa auditar os portos e os serviços que se encontram ativos. O segundo permite a visualização dos endereços de rede. O terceiro, e último, apresenta os pacotes que se encontram instalados no sistema operativo.

Servidor Apache e linguagem PHP

O processo de instalação do *software* necessário inicia-se com a instalação do servidor Apache, sendo necessário definir que este seja deve ser executado sempre que a máquina arranca. Estes passos foram concretizados através dos seguintes comandos:

```
yum -y install httpd
```

```
systemctl start httpd.service
systemctl enable httpd.service
```

Após a execução dos comandos anteriores, é necessário que a *firewall* do sistema permita a escuta nos portos padrão usados pelo Apache. Após alterada essa permissão, é necessário reiniciar a proteção. Isto é conseguido através dos seguintes comandos:

```
firewall-cmd --state
firewall-cmd --list-all
firewall-cmd --permanent --zone=public --add-service=http
firewall-cmd --permanent --zone=public --add-service=https
firewall-cmd –reload
```

A linguagem utilizada no servidor é a PHP. Portanto, necessita-se instalar também os respetivos pacotes. Como referido na Secção 3.3 , o SGBD utilizado é o PostgreSQL, sendo também necessário instalar o pacote que fornece uma biblioteca de funções PHP específica para o uso desse SGBD. Por fim, é necessário reiniciar o servidor Apache. Estas operações são realizadas com os seguintes comandos:

```
yum -y install php
yum -y install php-pgsql
systemctl restart httpd.service
```

Neste momento, resta configurar alguns aspetos, opcionais a este trabalho, como o uso de ficheiros de configuração htaccess e eventuais erros de PHP serem apresentados na página *Web* que execute uma função problemática. Para isso, é necessário alterar o ficheiro httpd.conf e o php.ini respetivamente.

SGBD PostgreSQL e extensão espacial PostGIS

Em primeiro lugar, precisamos instalar o PostgreSQL e definir o porto 5432 como ativo na *firewall* do sistema. Realizamos estes dois passos com os seguintes comandos:

```
yum -y install
https://download.postgresql.org/pub/repos/yum/9.5/redhat/rh
el-7-x86_64/pgdg-centos95-9.5-2.noarch.rpm
yum -y install postgresql95*
firewall-cmd --permanent --add-port=5432/tcp
service httpd stop
setsebool -P httpd_can_network_connect 1
service httpd start
```

A escuta do porto e o endereço de onde são permitidas ligações devem também ser definidos no ficheiro de configuração postgresql.conf.

É aconselhável que seja definida uma palavra-passe para o utilizador criado pela instalação. Esse utilizador é denominado de postgres. A sua definição é conseguida através dos seguintes comandos:

```
su -l postgres
psql
\password
```

Em seguida, é pedido ao utilizador que digite a nova palavra-passe e que a confirme. Os métodos de autenticação também deverão ser alterados, para MD5, no ficheiro pg_hba.conf, para que não se permitam ligação indevidas.

Por último, resta a instalação e ativação do PostGIS. Os comandos que permitem a sua instalação são os seguintes:

```
yum -y install epel-release
yum install postgis2_95 postgis2_95-client
```

Finalizada a instalação da extensão espacial, esta tem que ser ativada durante a execução do PostgreSQL. Os seguintes comando permitem que isso seja conseguido:

```
su -l postgres
psql
CREATE EXTENSION postgis;
SELECT postgis_full_version();
```

Estes comandos finalizam a instalação dos pacotes necessários à execução dos serviços *Web*.

Carregamento de dados

Visto que todo o *software* necessário já se encontra instalado, a base de dados, descrita na Secção 3.5.1 pode ser criada executando um ficheiro SQL criado para esse efeito. É ainda necessário que o *schema* seja adicionado ao caminho de pesquisa do PostgreSQL. Os seguintes comandos possibilitam essas tarefas:

```
su -l postgres
psql
\i /var/www/html/wsolar/data/wsolar.sql
SHOW search_path;
SET search_path TO wsolar,public,"$user";
```

A execução destes comandos permite que seja armazenada informação na base de dados. O carregamento de dados é efetuado executando o *script* loadFile.php, o qual lê

o ficheiro de dados disponibilizado pelo DEGGE (ver secção 3.4 e, utilizando os serviços *Web*, insere os dados na base de dados. A execução do ficheiro é feita através do seguinte endereço:

```
http://solar.campus.ciencias.ulisboa.pt/loadFile.php
```

Em alternativa, é possível executar o ficheiro PHP através da linha de comandos, digitando no terminal:

```
php /var/www/html/wsolar/loadFile.php
```

A confirmação da completude da inserção dos dados pode ser feita verificando a quantidade de linhas armazenadas na entidade Ponto.

3.5.6 Avaliação e discussão dos resultados obtidos

A avaliação da correção deste objetivo está ligada à implementação da aplicação cliente. Se o WSolAR não devolve informação corretamente, isso será detetado pelo utilizador que esteja a fazer uso da aplicação móvel de realidade aumentada.

Resta então avaliar o desempenho temporal das respostas prestadas pela base de dados espacial, de forma a aferir se os serviços respondem em tempo útil às necessidades das aplicações clientes.

Protocolo de teste

Para este teste, foram definidas algumas *queries* representativas do trabalho realizado no momento em que a aplicação cliente requer dados aos serviços *Web*. Interrogações como a requisição de todos os pontos, ou todos os pontos limitando a quantidade de dados devolvidos, servem para estimar o tempo de resposta consoante o número de pontos devolvidos. O teste principal utiliza uma interrogação espacial com centro no edifício C6 da FCUL e com raio variável. Este teste simula um caso em que uma aplicação cliente solicita dados ao serviço *Web*, enviando a sua posição e o raio de pesquisa desejado.

O teste realizou-se executando cinco repetições das interrogações na consola SQL do PostGIS, estando este instalado numa máquina com as seguintes configurações: Intel Core i7 920M, 6GB RAM DDR3 1333MHz, Samsung SSD Evo 850 e sistema operativo Microsoft Windows 10.

Obtenção de todos os pontos

Em primeiro lugar, testou-se a capacidade de resposta do sistema ao devolver todos os pontos presentes na base de dados. A Tabela 8 apresenta a análise estatística do tempo de execução de uma interrogação que selecione todos os pontos da tabela Ponto:

Tabela 8. Tempo para obtenção de todos os pontos.

	Média	**Mediana**	**Mínimo**	**Máximo**
Tempo (ms)	5952	5845	5835	6131

A *query* executada devolve 39365 linhas, que representam a totalidade de pontos sobre os quais existe informação no sistema. Podemos observar que não existe uma grande variação nos tempos de execução registados, sendo previsível que esta *query* demore entre 5,8 a 6 segundos.

No entanto, a aplicação SolAR necessita de um conjunto de variáveis bem especificado. Pelo que o seguinte teste requer ao sistema que devolva esse conjunto de variáveis para todos os pontos armazenados. A análise estatística do resultado do teste encontra-se na Tabela 9.

Tabela 9. Tempo para obtenção de todos os pontos, selecionando apenas as variáveis necessárias ao funcionamento da aplicação SolAR.

	Média	**Mediana**	**Mínimo**	**Máximo**
Tempo (ms)	4473	4461	4457	4505

Os tempos de execução foram quase constantes, aproximando-se sempre dos 4,5 segundos.

Desempenho das interrogações limitando o número de pontos retornados

O objeto principal de estudo dos testes seguintes é o edifício C6. Sendo assim, é importante perceber quantos pontos estão associados a este edifício. Ao todo são 9134 os pontos pertencentes ao edifício C6.

O teste seguinte visa perceber o tempo que o sistema demora a devolver uma certa quantidade de pontos. A Tabela 10 mostra o número limite definido e o respetivo tempo de execução.

Tabela 10. Tempo de execução em função do número de pontos devolvido.

Número de pontos limite	**Tempo (ms) de execução**
1	31
5	16
10	31
50	31
100	53
500	100
1000	200
5000	932
9134	1671
10000	1802
20000	3581
30000	5367
40000	7036

A análise do tempo de execução em função do número de pontos mostra que este cresce linearmente a um ritmo de aproximadamente 0,18 ms/ponto. Pelo gráfico da figura realizou-se uma aproximação linear com elevado grau de confiança.

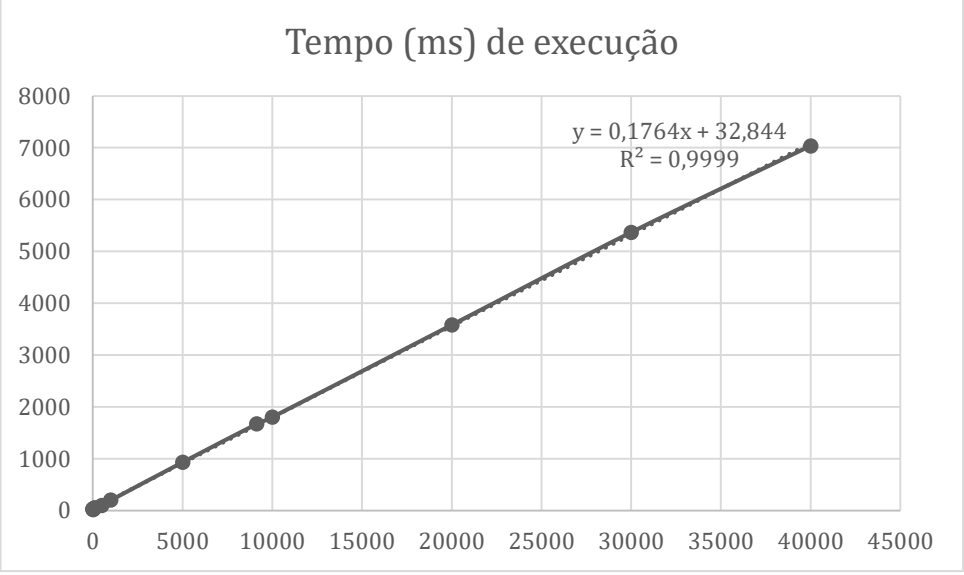

Figura 15. Tempo, em milissegundos, de execução em função do número de pontos retornado.

Simulação de um pedido requisitado pela aplicação SolAR

O último teste pretendia simular um pedido que se espera ser efetuado de forma mais frequente por parte da aplicação SolAR. Foi definido um ponto que simula a posição de um utilizador e este requer informação à sua volta. A Figura 16 mostra esse ponto no mapa. Essa requisição é executada cinco vezes por cada raio definido. Os raios são 50, 75 e 100 metros.

Após a execução dos testes, reuniram-se os resultados e realizou-se uma análise estatística, presente na Tabela 11.

Figura 16. Ponto que simula a posição do dispositivo móvel no terreno. É possível verificar o contexto espacial, bem como as coordenadas do ponto.

Tabela 11. Tempo para obtenção de todos os pontos centrados na posição dada e com um raio de pesquisa igual a 50, 75 ou 100 metros.

Raio (m)	Tempo (ms)				# Pontos
	Média	Mediana	Mínimo	Máximo	
50	827	817	815	869	5081
75	1156	1155	1151	1165	8019
100	1409	1409	1397	1419	10202

Com o aumento do raio, é visível a evolução crescente de pontos retornados e do tempo de execução da interrogação espacial.

De notar também que número de pontos retornados, do caso do raio de pesquisa igual a 100 metros, é superior ao número de pontos do edifício C6. Conclui-se que este facto se deve a que o raio alcançou pontos referentes a outros objetos que não o C6.

Conclusão da avaliação

Ao analisar os dados originados pelos testes, percebe-se que a devolução de grandes quantidades de pontos pode ser um processo demoroso. A acrescer ao tempo de resposta do serviço *Web*, os dados ainda vão ter de navegar pela rede até chegar ao requerente, o que aumenta ainda mais o tempo de resposta.

Escolhendo um raio sensato, que cubra uma boa parte da fachada de edifícios, igual ou inferior a 100 metros, o resultado é obtido num intervalo de cerca de 1 a 1,5 segundos, que é um tempo razoável.

3.6 *Middleware* MiddSolAR

O MiddSolAR é a componente destinada a fornecer à aplicação SolAR o ficheiro com os dados que esta deve consultar em função do posicionamento do utilizador e do raio de pesquisa selecionado. Tendo em conta que já se encontrava operacional o servidor central onde esses dados estão alojados, o WSolAR, este objetivo teve como pressuposto fazer uso dessa informação.

Deste modo, foi criado um requisito adicional na aplicação SolAR. A aplicação passa agora a necessitar de ligação à Internet para carregar novos dados. Caso essa ligação não exista, a aplicação continuará a funcionar com os dados já carregados.

Aplicação SolAR e *middleware* MiddSolAR

Durante o processo de desenvolvimento, pretendeu-se na medida do possível separar ao máximo o código novo, referente ao *middleware* MiddSolAR, do já existente na aplicação SolAR e minimizar o número de dependências e interações entre antigos e novos componentes. O esforço concentrou-se no desenvolvimento do MiddSolAR, embora tenha sido necessário realizar algumas adaptações à aplicação SolAR para que esta tirasse partido do *middleware*.

Protocolo de comunicação entre a aplicação e o *middleware*

Dado que existe uma separação entre as duas componentes, foi necessário encontrar um meio para que estas consigam trabalhar em conjunto. Foram consideradas duas formas:

i. Fazer uso de métodos fornecidos pelo *middleware*; ou
ii. Observação e interação com um ficheiro de *lock*.

O recurso aos métodos da API é de interação direta e a descrição da API pode ser consultada na subsecção 3.6.4. Estes métodos servem para receber informação pontual e de depuração de objetos.

Como as alterações a introduzir no SolAR sempre necessárias, tentou-se manter ao máximo as mecânicas já existentes. Assim optou-se por a aplicação continua a ler a informação sobre radiação a partir de um ficheiro bem especificado, estando este presente no armazenamento interno do dispositivo móvel. O seu caminho é:

```
/[armazenamento interno]/middsolar/ficheiro_dados.txt
```

Deste modo, a aplicação mantém o processo de leitura, agora repetindo-o sempre que existam novos dados disponibilizados pelo *middleware*. No entanto, apesar de se minimizar o impacto da introdução do *middleware*, algumas alterações foram indispensáveis.

A aplicação tem noção da atualização do ficheiro de dados devido a uma política de comunicação entre a aplicação e o *middleware*. Esta política, ilustrada na Figura 17, recorre ao uso de um ficheiro de *lock* e a uma implementação do FileObserver sobre o mesmo.

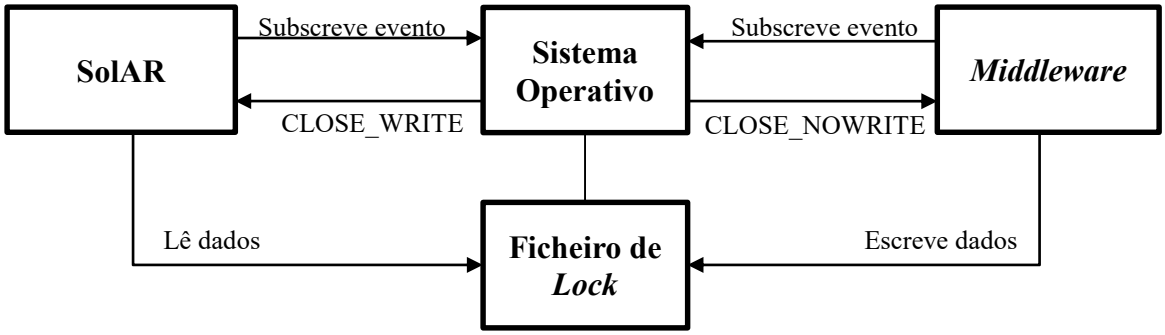

Figura 17. Política de interações e subscrições necessárias ao uso do ficheiro de *lock.*

O SolAR regista-se como subscritor de eventos de alteração do ficheiro de *lock*. Após a subscrição, aguarda um evento CLOSE_WRITE para perceber quando são escritos novos dados pelo *middleware* e iniciar a leitura do ficheiro de dados.

Por sua vez, também o *middleware* se regista como subscritor de eventos de alteração sobre o ficheiro de *lock*. Este aguarda um evento CLOSE_NOWRITE para saber que as leituras do SolAR sobre o ficheiro de dados terminaram.

O sistema operativo notifica os subscritores sobre operações executadas sobre o ficheiro de *lock*, conforme os eventos indicados no momento da subscrição por parte do subscritor.

Limitações do FileObserver do Android

Utilizando um objeto observador que implemente a interface FileObserver, é possível receber notificações sobre eventos de acesso a um ficheiro, incluindo a leitura ou modificação de um ficheiro.

Para este projeto, foram necessários dois observadores sobre o mesmo ficheiro, um na aplicação cliente, o SolAR, e outro presente no *middleware*. No entanto, e devido à implementação do FileObserver no sistema Android, tal não é possível [34]. Acontece que a observação do ficheiro é sempre atribuída ao subscritor mais recente. Consequentemente, apenas um dos observadores recebe notificações, e não dois como pretendido.

Ao pesquisar o problema, encontrou-se uma solução, que utiliza um conjunto de observadores [35], com a qual podem existir vários subscritores de um ficheiro e todos recebem os eventos solicitados. No entanto, tanto a aplicação como o *middleware* partilham este código, gerando assim um ponto de contacto e dependência entre os dois.

3.6.1 Organização do *middleware*

Para o desenvolvimento do MiddSolAR tentou-se separar ao máximo as responsabilidades de cada pacote do Java. A Figura 18 expõe a organização escolhida.

Nela podemos perceber que existem dois pontos de contacto entre a aplicação cliente e o *middleware*, que são a utilização da API e a observação do ficheiro de *lock*. Podemos também constatar a presença dos serviços *Web* no diagrama, que permitem receber os dados dada a posição atual e um raio de pesquisa.

No diagrama, ao centro, são apresentados os vários componentes constituintes do *middleware*, seguindo-se a descrição de cada um. Cada gestor apresentado, corresponde a um *package* na linguagem Java.

- **API** – neste pacote são disponibilizados os métodos da API que serve de interface a uma aplicação cliente, como é o caso da aplicação SolAR. Este é o ponto de contacto para iniciar e encerrar os serviços do *middleware* e aceder a funções de depuração.
- **Lógica e decisão** – este pacote é responsável pelo controlo de fluxo de pedidos entre todos os componentes e é o elemento central. Aqui também se encontra o gestor de escrita do ficheiro de *log*, para registar todas as ocorrências em ficheiro.
- **Gestor de escrita e leitura em ficheiro** – aqui encontram-se as ações de escrita nos ficheiros de dados, de *log* e de *lock*, abstraindo os restantes pacotes de lidar diretamente com a escrita em ficheiro. Neste pacote encontra-se um serviço responsável pela observação de um ficheiro *lock*.**Gestor de configurações** – responsável pelo carregamento do ficheiro de configuração e pelo fornecimento dos valores nele contidos aos serviços de posicionamento e de *cache*.
- **Gestor de armazenamento local** – pacote onde se encontra a lógica operacional do serviço de *cache*. Aqui encontra-se o serviço de *cache* que é arrancado quando o *middleware* é ativado
- **Gestor de localização** – pacote responsável pela deteção do posicionamento do dispositivo. É neste pacote que se encontra definido o serviço de localização que é executado quando o *middleware* é ativado.
- **Gestor de comunicação** – responsável pela execução das ligações à rede por parte do *middleware*.

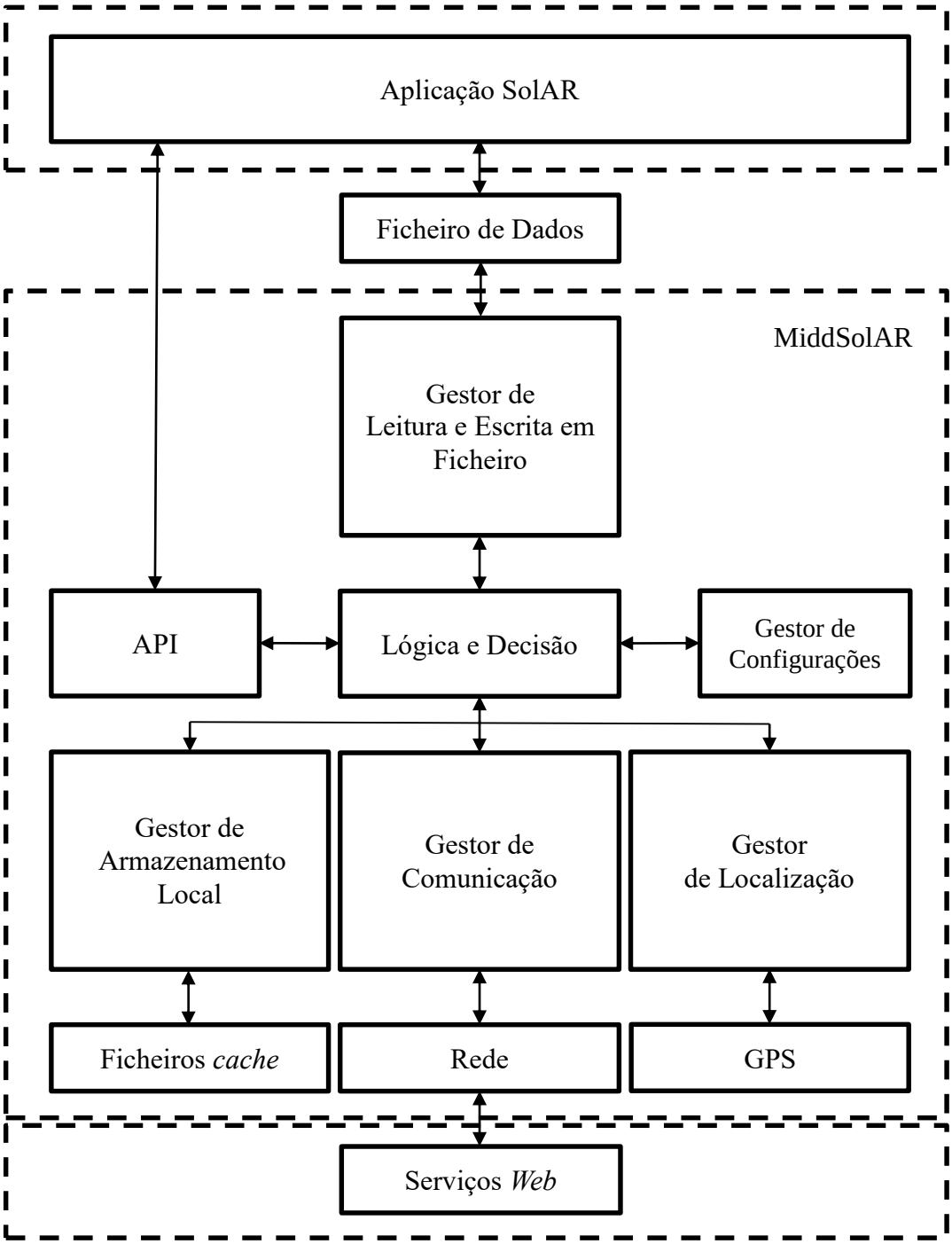

Figura 18. Organização do código do *middleware* e as interações com uma aplicação cliente e os serviços.

Diagrama de interação

Como descrito na composição do *middleware*, existem três serviços Android que são executados durante todo o ciclo de vida da aplicação: o responsável pela observação de ficheiros, o responsável pelo posicionamento do utilizador e o responsável pela *ca-*

che. Eles são os elementos principais do MiddSolAR pois cumprem as principais responsabilidades para quais o *middleware* está destinado a cumprir: atualizar o ficheiro de dados consoante a posição atual e armazenar localmente dados previamente requisitados ao serviço *Web*.

O diagrama da Figura 19 exemplifica a interação entre dois destes serviços no momento em que o serviço de posicionamento deteta que necessita de obter novos dados.

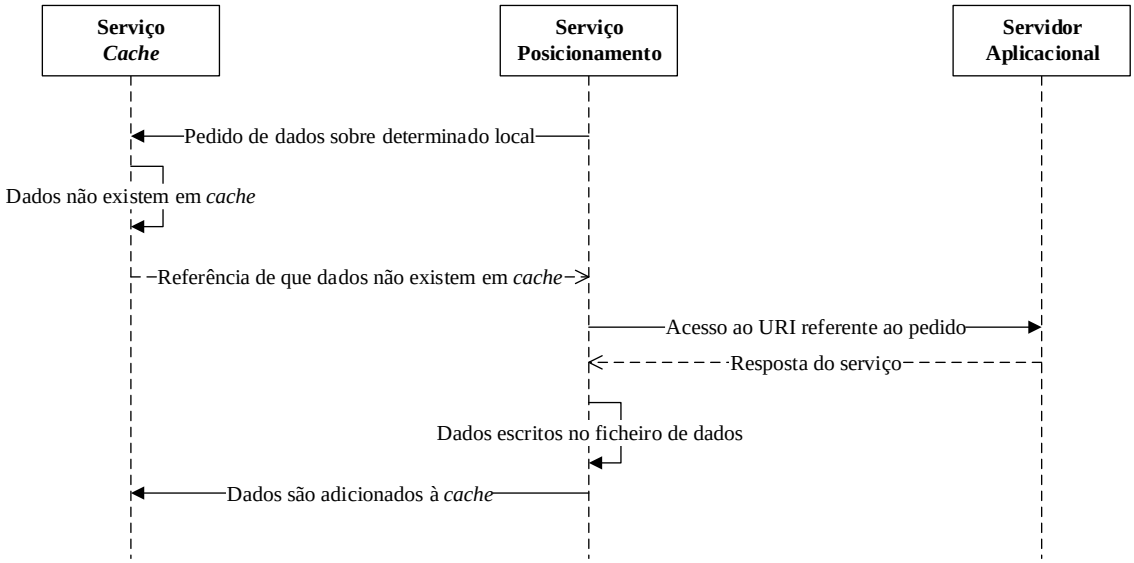

Figura 19. Diagrama de interação principal entre serviços do *middleware*, no cenário de acesso a dados que não estão em *cache*.

Ficheiro de configuração

Relativamente ao Gestor de Configurações, este controla um ficheiro de configuração, onde são guardadas definições com que o *middleware* deve operar.

Eis o aspeto do ficheiro de configuração:

```
{
    "radius": 50.0,
    "accuracy": 32.0,
    "factor": 2,
    "protocol": FIFO,
    "size": 20
}
```

No ficheiro são guardados valores referentes a várias componentes do *middleware*. Esses valores correspondem a:

- O raio de pesquisa em uso, definido em metros;
- A correção mínima, em metros, que as posições registadas pelo serviço de localização devem ter;
- O fator de divisão que delimita o momento em se deve proceder ao carregamento de novos dados. Como o raio definido é igual a 50 metros, dividindo por 2, resulta na distância a percorrer para que o middlware peça novos dados;
- A política de substituição de ficheiros em *cache*;
- A quantidade máxima de ficheiros em *cache*.

A existência deste ficheiro é obrigatória para o uso do *middleware* e este deve estar armazenado na seguinte diretoria e com o seguinte nome:

`/[armazenamento interno]/middsolar/middsolar.config`

Ficheiro de registo

Durante a execução do *middleware*, são escritas em ficheiro as diversas operações que este vai executando ao longo do tempo. As operações são discriminadas através de um código de operação, nome de operação e informação adicional. Essa informação adicional podem ser as coordenadas geográficas e grau de erro de uma posição registada ou o URL da execução de um pedido ao servidor e o tempo que este demorou a ser executado.

A análise deste ficheiro permite, por exemplo:

- Extrair as posições registadas pelos instrumentos de deteção de posicionamento, em geral GPS, ou tempos de execução de pedidos;
- Avaliar o estado da *cache*, contabilizando o número de casos em que esta providencia um ficheiro ou que não contém a informação requisitada;
- Identificar situações erróneas.

Em suma, a análise deste ficheiro permite depurar a execução do *middleware* e ter uma fonte de dados que sirva para uma avaliação de desempenho. O ficheiro de log é armazenado na seguinte localização:

`/[armazenamento interno]/middsolar/middsolar.log`

3.6.2 API fornecida

A API do *middleware* foi desenvolvida por dois motivos. Primeiro, para permitir que uma aplicação de realidade aumentada tenha uma rápida adaptação de código e consiga iniciar e encerrar os serviços do *middleware*. Deste modo, fica facilitado o acoplamento do novo código e funcionalidades.

Em segundo lugar, foi desenvolvida para que seja possível disponibilizar um conjunto de métodos para depuração do estado do *middleware*. Essa depuração pode ser feita editando o código fonte da aplicação SolAR ou criando uma nova aplicação cliente que faça uso desses métodos.

Na Tabela 12 são apresentados os métodos para iniciar e encerrar o *middleware*, designados `startServices` e `stopServices`, respetivamente. Como argumento em todos os métodos da tabela, é passado o objeto de contexto da aplicação cliente.

Os restantes métodos são usados pelos próprios métodos de inicialização e encerramento, pelo que não devem ser utilizados em separado. No entanto, são aqui referidos a título de explicação, sendo percetível pelo seu nome, que o *middleware* é composto principalmente por três serviços Android:

i. Um serviço responsável pelo armazenamento temporário de dados recebidos do serviço *Web* e relacionado com o gestor de armazenamento local;
ii. Um serviço de deteção do posicionamento do utilizador que está relacionado com o gestor de localização;
iii. Um serviço de observação do ficheiro de *lock*, relacionado com o Gestor de leitura e escrita de ficheiros.

Tabela 12. Métodos de controlo dos serviços que compõem o MiddSolAR.

Visibilidade do método, propriedades, nome e argumentos **Descrição da funcionalidade do método**
`public static void startServices(Context appContext)` Inicia os serviços referentes à execução do *middleware*.
`public static void stopServices(Context appContext)` Termina os serviços referentes à execução do *middleware*.
`public static void startCacheService(Context appContext)` Inicia o serviço referente à *cache* do sistema.

`public static void startNewLocationService(Context appContext)` Inicia o serviço de localização e atualização automática do ficheiro de dados.
`public static void startObservationService(Context appContext)` Inicia o serviço de observação do ficheiro de *lock*.
`public static void stopCacheService(Context appContext)` Termina o serviço referente à *cache* do sistema.
`public static void stopNewLocationService(Context appContext)` Termina o serviço de localização e atualização automática do ficheiro de dados.
`public static void stopObservationService(Context appContext)` Termina o serviço de observação do ficheiro de *lock*.

Na Tabela 13, expõem-se os métodos responsáveis por fornecer meios de depuração do estado de alguns objetos do *middleware*. Nomeadamente, objetos referentes à deteção do posicionamento do utilizador e os cálculos que o *middleware* executa ao analisar as distâncias entre essas mesmas posições. É ainda possível confirmar tempos de transferência de pontos e a localização absoluta dos ficheiros do *middleware*.

Tabela 13. Métodos de depuração de estado de objetos.

Visibilidade do método, propriedades, nome e argumentos **Descrição da funcionalidade do método**
`public static void getPointsBenchmark(int limit)` Permite requisitar a quantidade de pontos referida em argumento para calcular tempos de transferência e escrita em ficheiro.
`public static String getFileAbsolutePath()` Retorna o caminho onde se encontra o ficheiro de dados em uso.
`public static String getFileAbsolutePathLog()` Retorna o caminho onde se encontra o ficheiro de log.
`public static String getFileAbsolutePathLock()` Retorna o caminho onde se encontra o ficheiro de *lock*.
`public static Location getUltimaPosicaoAtualizacao()` Retorna a última posição onde ocorreu um pedido ao serviço *Web*.
`public static Location getAntigaPosicaoConhecida()` Retorna a posição imediatamente anterior à última registada pelos instrumentos do dispositivo.
`public static Location getNovaPosicaoConhecida()` Retorna a última posição registada pelos instrumentos do dispositivo.
`public static boolean getAtualizou()` Retorna verdadeiro ou falso caso o ficheiro de dados tenha sido ou não modificado

tendo em conta a nova posição conhecida.
`public static double getDistanceToLastKnownLocation()` Retorna a distancia, em metros, entre a antiga e a nova posição conhecida.
`public static double getDistanceToLastLocationOfUpdate()` Retorna a distancia, em metros, entre a nova e a última posição de atualização

`public static long getTime()`
`public static long getMilliseconds()`
`public static double getSeconds()`
`public static double getMinutes()`
`public static double getHours()`
Retorna o tempo que passou desde que o pedido foi efetuado até que a resposta foi recebida na totalidade.

`public static long getTimeForBenchmark()`
`public static long getMillisecondsForBenchmark()`
`public static double getSecondsForBenchmark()`
`public static double getMinutesForBenchmark()`
`public static double getHoursForBenchmark()`
Retorna o tempo que passou desde que o pedido de avaliação do serviço foi efetuado até que a resposta foi recebida na totalidade.

3.6.3 Serviço de localização

Iniciado este serviço, o *middleware* torna-se consciente da posição atual do utilizador. Isto permite aferir a necessidade de carregar novos dados para a aplicação SolAR, estejam eles já armazenados na *cache*, ou não, sendo neste último caso iniciada a ação de execução de um pedido ao serviço *Web*.

Estados do serviço

A Figura 20 ilustra os vários estados que este serviço pode tomar. No momento em que o serviço é criado, este regista-se imediatamente perante os serviços de localização do Google Play Services, requerendo atualizações sobre a posição do dispositivo. Após o registo, o serviço está à escuta de novas posições fornecidas pelo sistema, tratando-as segundo os critérios de qualidade de posição e distância perante a posição referente aos dados atualmente carregados para visualização.

No diagrama apresentado, podemos observar quatro estados. Quando o sistema operativo Android não deteta nenhuma posição, ficando em espera, e o seu oposto. Quando tem posição, esta é avaliada e pode desencadear um de dois subestados. Um que permite o tratamento de pedidos e um segundo estado em que não são permitidos pedidos por não estar disponível uma ligação à Internet.

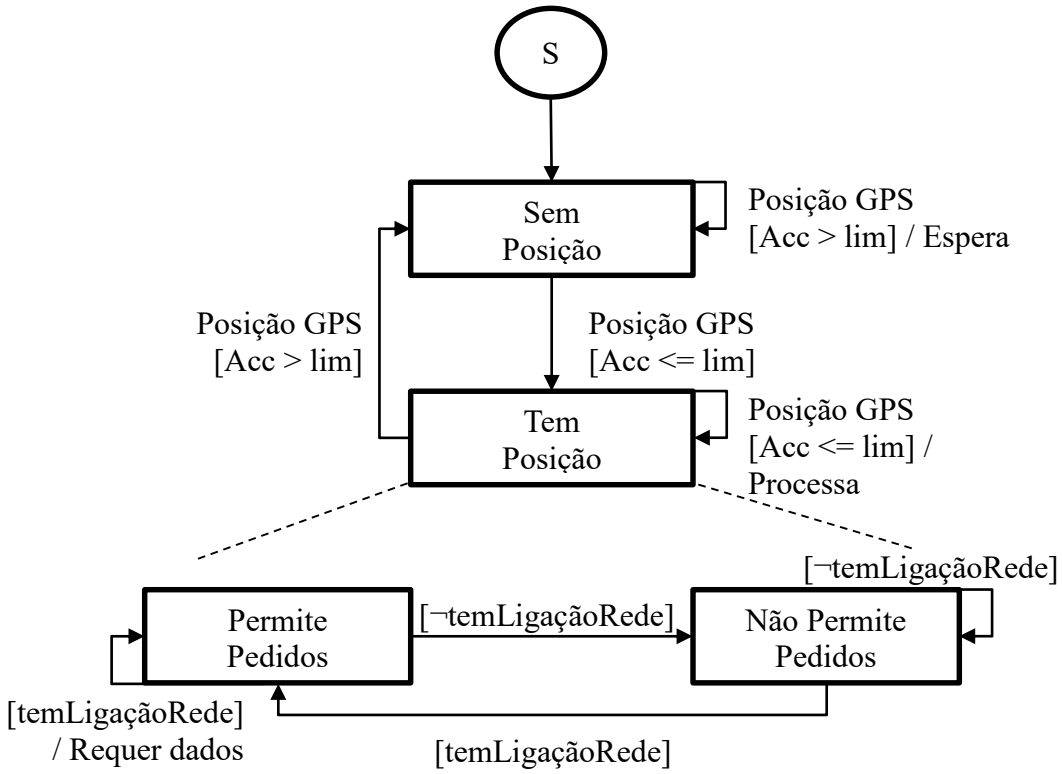

Figura 20. Diagrama de estados referente ao serviço de localização.

Surgem ainda os conceitos de correção (Acc) e limite (lim). O primeiro refere-se à correção do posicionamento obtido pelo dispositivo. O segundo é um limite de referência para avaliar a fiabilidade da posição geográfica detetada.

Grau de correção do posicionamento

Uma correta visualização em realidade aumentada dos dados de radiação solar depende da correção da posição obtida pelos instrumentos de localização. Esse grau de correção é fornecido em conjunto com o objeto de posição, tendo como nome de atributo *accuracy*. De forma a refinar as posições detetadas, o *middleware* descarta as deslocações instantâneas cuja correção esteja acima de um determinado valor limite. Este valor por predefinição é 31 metros, sendo possível modificá-lo, editando o ficheiro de configuração do *middleware*.

Normalmente, o grau de correção do GPS ronda 3 a 5 metros, e espera-se que a utilização desta aplicação ocorra no exterior de edifícios. No entanto, numa cidade na proximidade das fachadas dos edifícios, a qualidade do sinal pode não ser a ideal. Nesses casos, pode ser benéfico recorrer ao auxilio das redes Wi-Fi obter uma posição caso o GPS não forneça nenhuma. Ao utilizar essas redes, a posição obtida não será tão pre-

cisa quanto desejável. Por isso é conveniente não ter um critério de seleção de posições demasiado restrito. A transição entre uma fonte que fornece o posicionamento ao *middleware*, GPS ou Wi-Fi, é gerida pelo Google Play Services de forma transparente.

3.6.4 Serviço de *cache* de pedidos de dados

Ao iniciar o serviço de *cache*, este carrega todos os ficheiros de dados locais que existam no armazenamento interno, respeitando o limite de tamanho indicado no ficheiro de configuração do *middleware*. Após o carregamento, o serviço fica disponível para responder a pedidos de dados consoante uma posição ou para inserir em *cache* novos dados vindos do serviço *Web*.

Mecanismos e ficheiros gerados

A aplicação cliente, neste caso a SolAR, lê a informação que depois apresenta ao utilizador a partir de um ficheiro de dados. O *middleware*, nomeadamente, aquando do momento da escrita de dados oriundos do servidor, escreve um conjunto de informação pertinente num ficheiro de metadados. Esse ficheiro acompanha sempre o ficheiro de dados, sendo que é indispensável, para a lógica aplicacional, que exista em simultâneo.

Eis um exemplo de um ficheiro de metadados:

```
{
  "mills": 1465985701834,
  "date": "2016-06-15",
  "hour": "11:15:01",
  "latitude": 38.7554103,
  "longitude": -9.1577178,
  "accuracy": 3.0,
  "radius": 100.0,
  "points": 8148,
  "size": 910735,
  "size KB": 889,
  "used": "1"
}
```

No ficheiro de metadados encontram-se dados como o momento em que a informação foi obtida, correção da posição registada e o raio de pesquisa que se encontrava em uso. São ainda registados o número de pontos existentes no ficheiro de dados e o espaço por ele ocupado. Por fim, é contabilizado o número de vezes que o ficheiro de dados foi utilizado pela aplicação cliente.

Sempre que o *middleware* recebe dados do servidor, esses dados são escritos num ficheiro denominado ficheiro_dados_1465985701834.txt onde 1465985701834 é o nu-

mero exemplificativo de milissegundos passados após 1970 e referentes ao momento da escrita. Esse valor também se encontra descriminado no ficheiro de meta dados. Utilizando esta nomenclatura, é garantido que cada pedido dá origem a um ficheiro com nome único. Após este ficheiro ser inserido na *cache*, o seu conteúdo é copiado para o ficheiro de dados.

Com o acumular de pedidos, e se o limite de tamanho da *cache* o permitir, são gerados vários ficheiros de dados, identificados univocamente com os milissegundos, e os ficheiros de metadados respetivos a cada um. A *cache* mantém em memória informação sobre todos eles. Quando existe um pedido, a *cache* procura sobre os objetos em memória, representativos dos ficheiros de meta dados. A pesquisa executada visa encontrar um ficheiro de dados que tenha sido gerado numa posição próxima da atual, e com raio de pesquisa maior ou igual ao definido no momento atual. Caso exista um *hit*, ou seja, a *cache* contém a informação pretendida, os dados do ficheiro requisitado são copiados para o ficheiro de dados. Em seguida, o protocolo de comunicação entre a aplicação e o *middleware* é seguido, conforme ilustrado na Figura 17.

Estados do serviço

Devido a questões de concorrência, é importante que o serviço de *cache* bloqueie o atendimento de novos pedidos enquanto executa uma tarefa anterior. A Figura 21 exemplifica os dois estados que este serviço pode tomar.

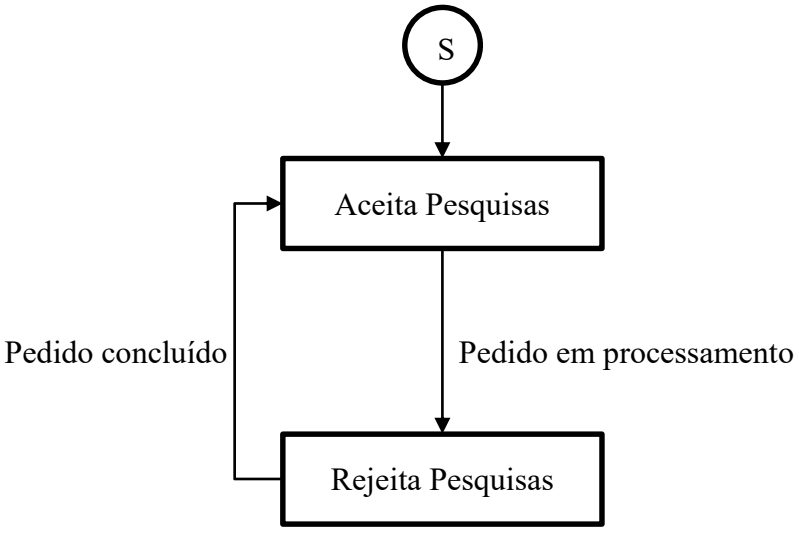

Figura 21. Diagrama de estados do serviço de *cache*.

De referir ainda que não existe uma fila de pedidos. Ou seja, se a *cache* está num estado ocupado, todos os pedidos que lhe forem feitos durante esse estado serão descartados. Esta perda de pedidos acaba por ter um impacto reduzido visto que as atualizações de posições são constantes, o que origina a verificação da necessidade, ou não, de atualizar os dados.

Políticas de substituição de ficheiros em *cache*

Para ordenar e decidir que ficheiros se devem manter em *cache*, é necessário que exista uma política de substituição de ficheiros. Em seguida, são apresentados os algoritmos implementados pela *cache* do *middleware*.

- FIFO (*First In First Out*)
- MFU (*Most Frequently Used*)
- LFU (*Least Frequently Used*)

O algoritmo definido por omissão é o FIFO, podendo ser alterado através das configurações do *middleware*.

Existe também a possibilidade de, através da programação do serviço de *cache*, adicionar novas políticas de substituição de ficheiros. Para isso, é necessário adicionar o caso específico da nova política nos métodos onde se inserem ou removem ficheiros da *cache*.

3.6.5 Alternativas consideradas

Durante o projeto foram surgindo múltiplas formas de implementar os componentes deste trabalho. No entanto, foi necessário proceder a escolhas baseadas na qualidade das soluções e na sua exequibilidade. Nesta secção são apresentadas as principais decisões tomadas e a respetiva justificação.

***Broadcast* de eventos para interação entre SolAR e MiddSolAR**

Apesar de ter sido adotada a política de comunicação utilizando o ficheiro de *lock* e a sua observação, esta não foi a implementação pensada no início deste trabalho. Outro método de comunicação é o lançamento de eventos bem especificados por parte do emissor. Os recetores subscrevem eventos desse emissor e definem uma ação em reação ao evento esperado.

Logo desde o início deste projeto definiu-se que o *middleware* e a aplicação cliente de realidade aumentada deviam ser, na medida do possível, independentes uma da

outra. Esta subscrição de eventos gerados pelo *middleware* iria criar uma dependência para este *middleware* específico.

Assim, optou-se pela solução recorrendo a um ficheiro de *lock* que qualquer *middleware* pode manipular. Caso seja necessário substituir o *middleware*, este novo apenas necessita implementar essa política.

Método de deteção de posicionamento

No sistema operativo Android existem dois meios de obter um posicionamento do utilizador. O primeiro utilizando os métodos incluídos no *software development kit* (SDK) [36] e o segundo recorrendo aos métodos fornecidos pela API do Google Play Services.

Neste projeto foi adotado o segundo método pois este oferece uma posição mais refinada, fazendo uso dos vários sensores presentes no dispositivo. Apesar de se perder controlo sobre o instrumento que fornece o posicionamento do utilizador, a transição entre sensores ocorre de forma transparente para a lógica aplicacional. Esta transparência facilita a escrita de código, tornando-o menos confuso, e libertando o desenvolvimento da lógica de negócio de responsabilidades desnecessárias.

De referir ainda que o dispositivo Android tem que ter o Google Play Services instalado e ativo, sendo que este é mantido e atualizado em ciclos mais rápidos comparativamente ao SDK do sistema operativo.

Implementação da *cache*

Durante o planeamento do projeto, tinha sido pensado implementar a *cache* utilizando uma base de dados relacional SQLite. Tinha ainda sido dado conta da existência de uma extensão espacial para esta tecnologia, o SpatiaLite.

No entanto, sempre que fosse necessário ler e escrever dados, seria necessário aceder a uma base de dados, podendo ser necessárias mais leituras e escritas de dados em tabelas do que as desejadas.

Foi decidido manter a informação em ficheiros individuais, identificados de forma unívoca através do número de milissegundos após 1970. Recorrendo a ficheiros, existe uma escrita inicial e quando um ficheiro for necessário, primeiro são consultados os seus meta dados e, se for o caso, o conteúdo do ficheiro é copiado para o ficheiro de dados da aplicação cliente.

Outra questão levantada durante a implementação foi a tipologia do ficheiro de dados. Visto que o ficheiro de dados é uma cópia de um ficheiro em *cache*, significa que a informação contida nele está duplicada. Com a motivação de poupar este espaço ocupado, considerou-se o uso de uma ligação simbólica.

Essa solução foi descartada pois durante a sua implementação percebeu-se que para um processo criar uma ligação simbólica em Android, este necessita de privilégios *root*. Visto que dar este tipo de permissão não estava inicialmente previsto e não faz sentido num *middleware* que é para ser usado em aplicações cliente, esta solução foi cancelada.

Processamento de respostas

A informação sobre radiação solar incidente nas fachadas encontra-se organizada numa malha de pontos com um espaçamento de aproximadamente um metro entre cada par de pontos. Esta configuração dá origem a uma malha com uma grande quantidade de pontos que pode ocupar uma parte significativa da memória RAM do dispositivo móvel. Variando o raio de pesquisa para diminuir a quantidade de pontos a serem recebidos e processados, pode levar à falta de pontos visualizados nas fachadas, pelo que não é uma solução sempre eficaz.

A informação proveniente do serviço *Web* é recebida em formato JSON e inicialmente era toda carregada para um JSONObject, fornecido pela API do Java em Android. Contudo, esta metodologia sobrecarregava a memória RAM do dispositivo pois não a libertava à medida que o objeto era processado.

A solução encontrada foi processar a resposta do servidor através de um *stream*. Deste modo, um cursor avança pela resposta evitando o acumular de dados desnecessários em memória. No entanto, a biblioteca de Java que efetua este tipo de processamento não está disponível em Android, pelo que teve que se encontrar uma solução externa. Foram encontradas duas bibliotecas para o efeito, a Jackson Streaming API e a GSON. A primeira produzia um código mais simples, mas a segunda foi criada e ainda é mantida pela Google. Por motivos de longevidade e suporte, foi adotada a biblioteca GSON.

Alternativas consideradas a nível conceptual

Atendendo a uma comparação de desempenho entre o armazenamento e processamento na nuvem e numa base de dados espacial [37], verifica-se que, tendo em conta

as restrições de infraestrutura e orçamentais, optar por uma base de dados espacial é aceitável para casos que não sejam de larga escala e que não têm grande complexidade.

Para a formulação dos tipos de serviços a oferecer foi ainda pensado oferecer serviços para conversão de dados geográficos para outras normas de representação, como é o caso do GeoHash [38]. No entanto, esta ideia foi posta de parte devido a restrições de confidencialidade no acesso aos dados disponibilizados e também porque não acrescentaria valor à aplicação SolAR.

Outras tecnologias consideradas

Na fase de concretização do projeto, foram ainda considerados outros componentes e decisões arquiteturais, tais como: Microsoft SQL Server, Web Map Server (WMS), GeoServer, GeoJSON, XML, GML e KML. Os motivos pelos quais se decidiu pela sua não inclusão foram:

- **SQL Server** é um SBGD proprietário, ao contrário do PostgreSQL;
- **WMS** e **GeoServer** são tecnologias que se aplicam mais ao uso de mapas por parte das aplicações cliente. Ao se optar por um servidor convencional e por serviços *Web*, está a adotar-se uma solução mais simples, respondendo às questões levantadas neste projeto e indo de encontro dos objetivos;
- **XML**, **GML** e **KML** foram preteridos por serem mais verbosos comparativamente com o JSON e isso ser um ponto a ter em consideração perante a necessidade de transportar dados através da rede;
- **GeoJSON** foi excluído por se pretender um formato de envio de dados o mais geral possível abrangendo o maior número de tipos de serviços *Web*.

3.6.6 Avaliação e discussão dos resultados obtidos

Para produzir resultados que permitam aferir se o objetivo foi concluído com sucesso, foi definido um método de avaliação.

Protocolo de teste

Utilizando a aplicação SolAR com a *cache* limpa e tamanho ilimitado, percorreu-se o perímetro do edifício C6. Este foi o edifício escolhido pois encontra-se relativamente isolado dos outros e pelo seu tamanho ser adequado a testes de visualização pois

é muito comprido e largo. Foi escolhido também porque foi o primeiro edifício objeto de estudo por parte do projeto SolAR.

A Figura 22 mostra uma comparação entre o caminho definido previamente em torno do C6 e o caminho registado pelos instrumentos de deteção de posicionamento do dispositivo utilizado. Esta comparação exemplifica o grau de erro que pode existir na deteção obtida. Podendo traduzir-se em posições menos precisas em que se peçam dados e que a resposta a esses pedidos traga menos ou mais dados do que o esperado.

Ao longo do percurso, a aplicação deteta a necessidade de carregamento de novos dados. Em primeiro lugar, verifica se os dados em falta existem em *cache*. Caso a resposta seja positiva, esses dados são carregados e visualizados. Caso contrário, o MiddSolAR encarrega-se de contactar os serviços *Web* do WSolAR e requisita a nova informação.

De modo a que o teste ocorresse o máximo possível nas mesmas condições, o caminho percorrido foi sempre o mesmo, e a escolha do raio foi previamente definida. Nomeadamente, recolheram-se dados com um raio de 50, 75 e 100 metros. A aplicação arranca com um ficheiro de dados com informação referente ao ponto inicial do teste.

Existem, no entanto, condições que podem condicionar os resultados, como a qualidade do sinal e quantidade de carga na rede de comunicação celular, e a correção da posição obtida pelo sistema de posicionamento.

Figura 22. Comparação entre o caminho percorrido e o registado. À esquerda está desenhado o caminho definido para a execução do teste. Este caminho foi o mesmo em todos os percursos de teste e começa no centro do átrio do edifício C6, sendo percorrido no sentido dos ponteiros de relógio. À direita é ilustrado um exemplo de um caminho registado pelo sensor de posicionamento do dispositivo, durante a execução de um teste.

Os testes à aplicação prolongaram-se por três dias. Em cada um desses dias, foram executadas três recolhas de dados por cada raio escolhido nas definições da aplicação. Como resultado, foram produzidos nove registos por cada raio definido.

Esses registos foram analisados de forma a extrair os seguintes valores:

- Espaço ocupado pelos dados em *cache*;
- Tempo de execução de pedidos ao serviço *Web*;
- Número de pedidos ao serviço *Web*;
- Tipo de resposta fornecida pela *cache* (*hit* ou *miss*);
- Redundância dos dados armazenados em *cache*;
- Posições dos pedidos.

Os testes foram executados num *tablet* Nvidia Shield K1 [39]. Como este dispositivo não possui ligação de dados celular, recorreu-se a um *smartphone* iPhone 6S [40] para fornecer o acesso à internet durante o percurso de teste como um *hotspot*.

Durante os testes, o WSolAR ainda não se encontrava instalado nos servidores da FCUL. Pois era mais prático introduzir alterações ao mesmo se este se encontrasse ainda num computador pessoal. As configurações do computador com o WSolAR são as seguintes: Intel Core i7 920M, 6GB RAM DDR3 1333MHz, Samsung SSD Evo 250 e Windows 10.

A ligação de rede foi estabelecida por 4G e fibra ótica (100Mbps/10Mbps), ambas do fornecedor Meo.

Em seguida, são apresentados dados e gráficos da avaliação do MiddSolAR. Por fim, são tiradas conclusões sobre os resultados obtidos e a melhor configuração testada.

Espaço ocupado pelos dados em *cache*

Em primeiro lugar, existiu a preocupação de analisar a evolução do espaço ocupado pela *cache*, conforme fosse incrementado o valor do raio de pesquisa. A Tabela 14 e o gráfico da Figura 23 apresentam a análise estatística dos valores em KBytes do tamanho da *cache* no fim de cada percurso.

Tabela 14. Espaço ocupado, em KB, pela *cache* no fim dos percursos de teste.

Raio (m)	**50**	**75**	**100**
Média	6408	6324	6839
Mediana	6089	6661	6513
Mínimo	5819	5180	5701
Máximo	7773	7390	8991

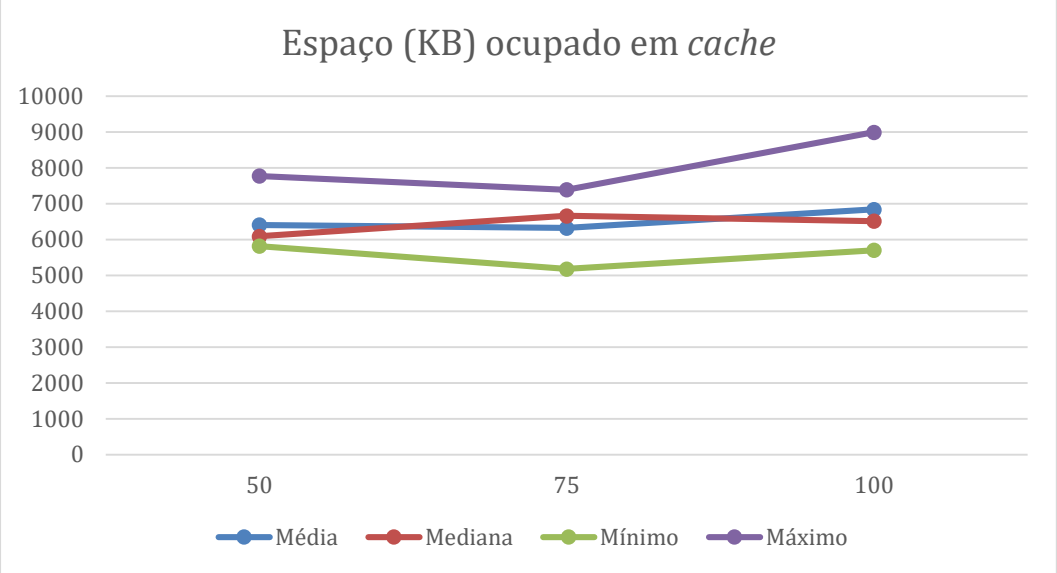

Figura 23. Espaço total ocupado pela *cache*, em KB, de acordo com os diferentes raios, no fim dos percursos de teste.

Os resultados mostram que: a) ao passar de um raio de 50 para 75 metros, ocorre uma ligeira descida no espaço ocupado; b) aumentando o raio de 75 para 100 metros, existe um ligeiro incremento no espaço total ocupado pela *cache*.

Tempo de execução de pedidos ao serviço *Web*

Existiu também a preocupação sobre o custo temporal que existe na transferência de dados oriundos do servidor. Na Tabela 15 e gráfico da Figura 24 podemos observar a análise estatística sobre o custo de tempos somados após o fim de cada percurso.

Tabela 15. Tempo somado, em segundos, de transferência dos dados de todos os pedidos realizados em cada percurso.

Raio (m)	**50**	**75**	**100**
Média	101,39	69,05	50,38
Mediana	83,47	45,07	52,45
Mínimo	56,12	27,82	27,94
Máximo	195,54	142,80	87,57

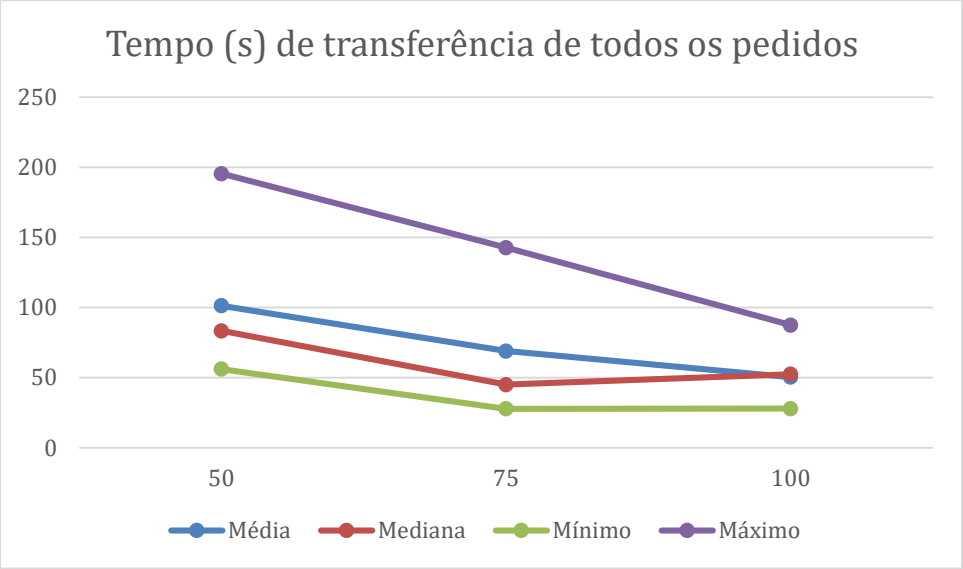

Figura 24. Tempo acumulado, em segundos, dos vários pedidos conforme o raio selecionado.

É possível verificar que existe um decréscimo do tempo total de espera pelos dados requeridos consoante o aumento do raio de pesquisa. Este resultado, está diretamente ligado ao ponto de análise seguinte, o número de pedidos realizados.

Número de pedidos ao serviço *Web*

O *middleware* está implementado de forma a que procure novos dados sempre que a distância percorrida desde o último carregamento ultrapasse metade do raio de pesquisa em vigor. Por isso, seria espectável que existissem menos carregamentos de dados caso se aumentasse o raio de pesquisa. A Tabela 16 e gráfico da Figura 25 mostram a análise estatística do número de pedidos efetuados em cada percurso.

Tabela 16. Quantidade de pedidos aos serviços realizados em cada percurso de teste realizado.

Raio (m)	**50**	**75**	**100**
Média	18	9	6
Mediana	18	10	6
Mínimo	16	8	5
Máximo	20	11	8

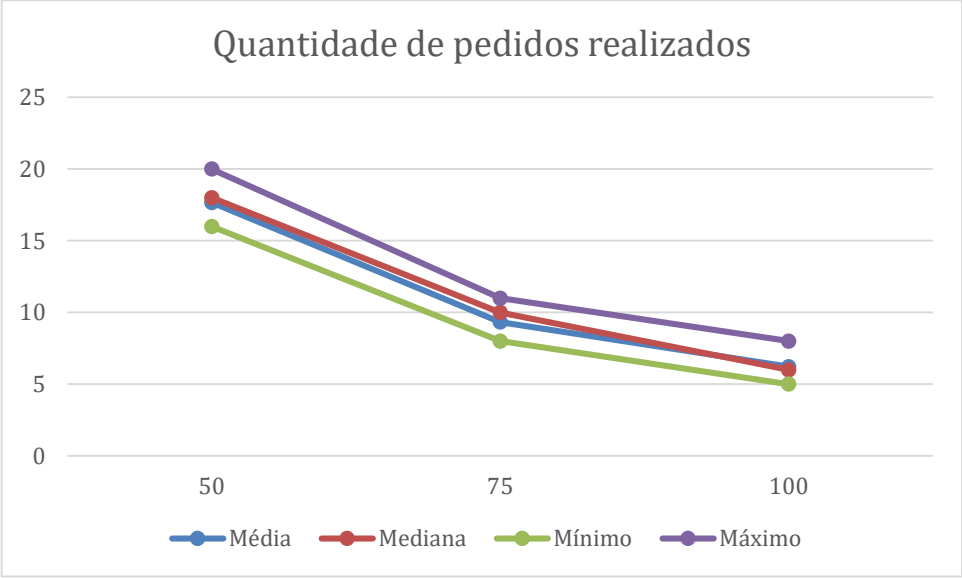

Figura 25. Quantidade de pedidos realizados aos serviços conforme o raio selecionado.

Verificando os dados obtidos, confirmou-se que existe um decréscimo do número de pedidos realizados ao servidor caso se aumente o raio, embora em cada pedido sejam retornados, tipicamente, um maior número de pontos. No entanto, o facto de serem menos pedidos efetuados, resulta em menos tempo gasto na espera total por nova informação.

Tipo de resposta fornecida pela *cache*

Quando o *middleware* tem a necessidade de obter nova informação para apresentar ao utilizador, este contacta a *cache*. Caso não exista nenhum ficheiro de dados em *cache* que seja correspondente ao requisito do *middleware*, a *cache* informa-o com uma resposta negativa. Em seguida o *middleware* executa um pedido ao serviço *Web*. Este evento corresponde a um *miss* na *cache*. Visto que existe uma implicação direta entre

número de *misses* e número de pedidos aos serviços *Web*, a Tabela 16 e o gráfico da Figura 25 referem-se à quantidade de *misses* ocorridos.

A Tabela 17 e o gráfico da Figura 26 apresentam a análise estatística da percentagem de misses ocorridos, tendo em conta o número total de necessidade de informação por parte do *middleware*.

A percentagem de *misses* com raio definido a 75 metros obteve o melhor resultado. No entanto, apesar de apresentar melhores valores de média, mediana e mínimo, a sua amplitude é a maior de entre os três raios estudados.

Tabela 17. Percentagem de *misses* ocorridos para cada percurso.

Raio (m)	**50**	**75**	**100**
Média	78%	67%	70%
Mediana	78%	63%	70%
Mínimo	69%	56%	60%
Máximo	87%	91%	83%

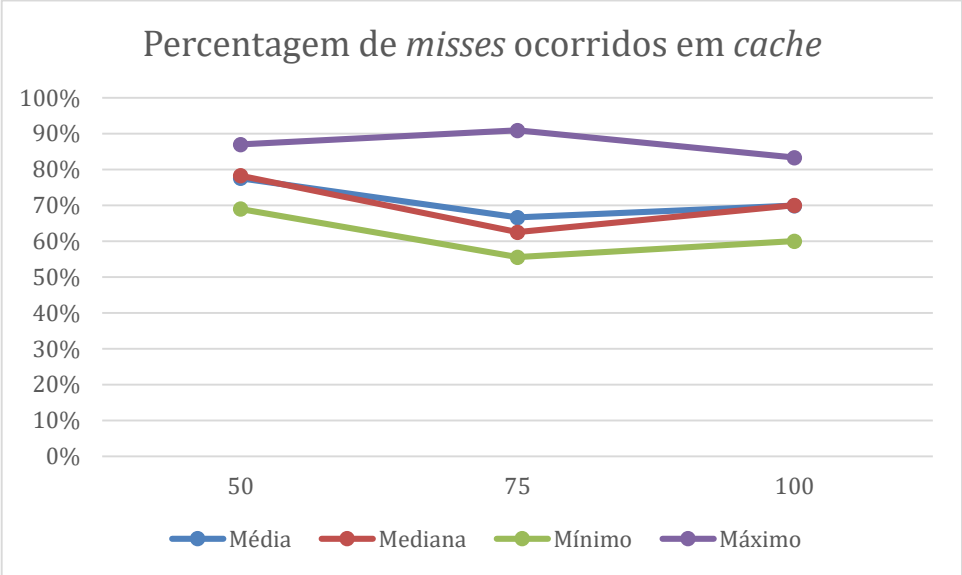

Figura 26. Percentagem de *misses* obtida pela *cache* consoante o raio definido.

Caso exista um ficheiro de dados em *cache* que seja correspondente ao requisito do *middleware*, esta responde positivamente, colocando a informação necessária no ficheiro de dados. Este evento corresponde a um *hit*, a *cache* tem a informação requerida.

Na Tabela 18 e gráfico da Figura 27 é apresentada a análise estatística dos hits ocorridos em cada percurso.

Em termos absolutos, o número de *hits* decresceu conforme o raio aumentou. No entanto, esses valores não têm em conta o número de situações em que o *middleware* necessitou de carregar novos dados.

Tabela 18. Quantidade de *hits* na *cache* para cada percurso realizado.

Raio (m)	**50**	**75**	**100**
Média	5	5	3
Mediana	5	5	3
Mínimo	3	1	1
Máximo	9	8	4

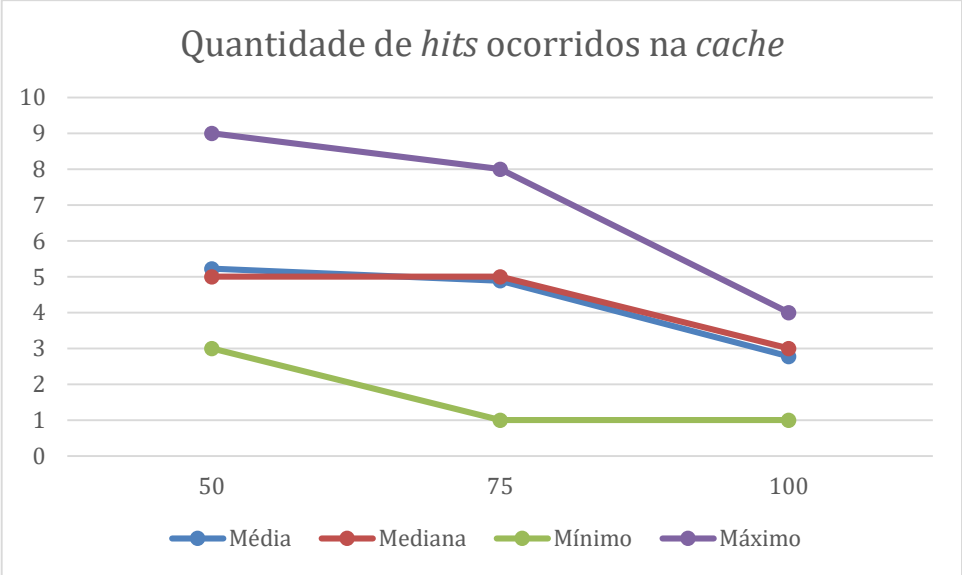

Figura 27. Quantidade de *hits* obtidos conforme o raio definido.

Na Tabela 19 e gráfico da Figura 28 é apresentada a análise estatística da percentagem de hits ocorridos nos percursos.

Ao analisar os dados, percebeu-se que o raio que obteve melhores resultados para a *cache* foi o raio de pesquisa com 75 metros. De notar ainda que o raio com 100 metros, embora tendo pior média e mediana, apresenta uma menor amplitude entre o valor máximo e valor mínimo.

Tabela 19. Percentagem de *hits* ocorridos durante cada percurso de teste.

Raio (m)	**50**	**75**	**100**
Média	22%	33%	30%
Mediana	22%	38%	30%
Mínimo	13%	9%	17%
Máximo	31%	44%	40%

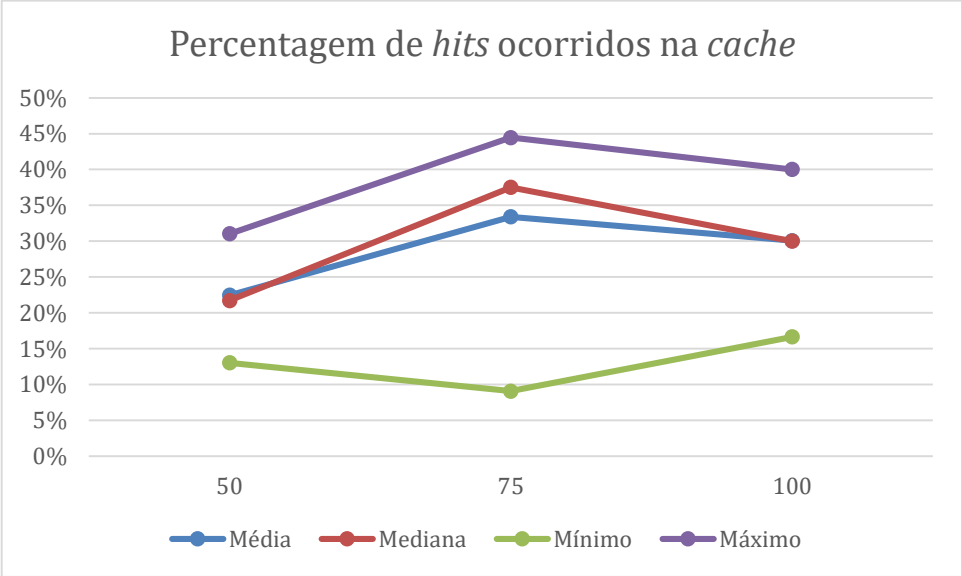

Figura 28. Percentagem de *hits* ocorridos na *cache* conforme o raio escolhido.

A Tabela 20 e gráfico da Figura 29 mostram a quantidade de vezes que a *cache* foi acionada durante cada percurso gerando *hits* ou *misses*. É possível perceber que, conforme o aumento do raio de pesquisa, a necessidade de obter novos dados para visualização diminui. Essa diminuição é menos acentuada quando se aumenta o raio de 75 para 100 metros.

Tabela 20. Quantidade de respostas fornecidas pela *cache* em cada percurso.

Raio (m)	**50**	**75**	**100**
Média	23	14	9
Mediana	23	14	9
Mínimo	20	11	6
Máximo	29	18	11

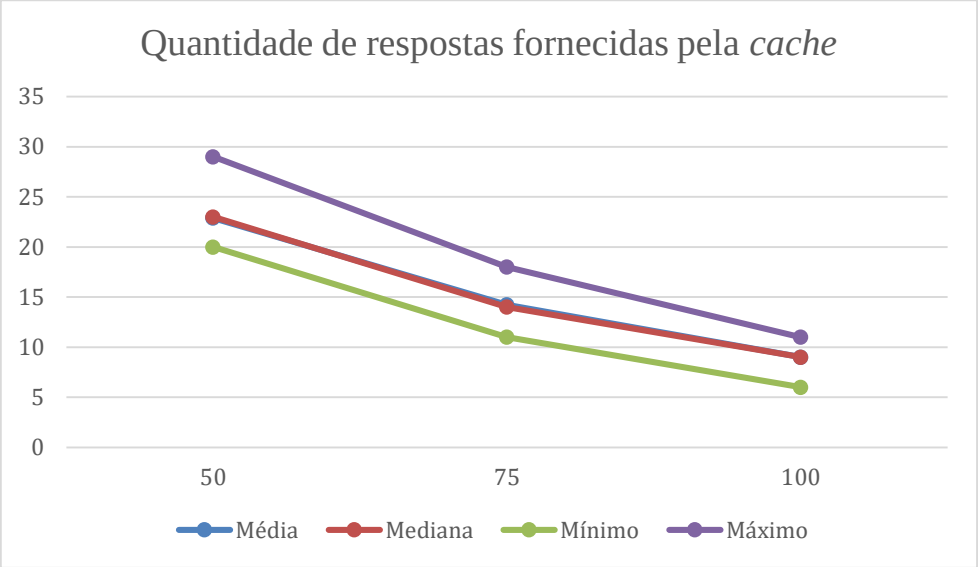

Figura 29. Quantidade de respostas dadas pela *cache* conforme o raio selecionado.

Redundância dos dados armazenados em *cache*

Na realização deste trabalho, não foi tida em conta a requisição de dados em que a resposta dada pelo servidor pudesse já estar parcialmente contida na *cache*. A *query* espacial criada no servidor, procura simplesmente todos os pontos existentes dentro de uma circunferência com um raio dado, e como a forma da figura geométrica de pesquisa é imutável, é possível que exista informação redundante devolvida pelo serviço.

Tomando como exemplo um caminho uma linha reta com comprimento ligeiramente inferior a 75 metros e um raio de pesquisa de dados igual a 50 metros, e *cache* vazia, aconteceria o que vem ilustrado na Figura 30.

Sempre que o utilizador caminhar e se distanciar o mínimo correspondente a metade do raio definido, neste caso de 25 metros, o *middleware* requer novos dados ao serviço *Web*. Esta distância é calculada tendo em conta a posição atual do utilizador e a posição onde os dados que estão carregados no momento foram pedidos. Esse ponto de atualização encontra-se representado na figura por pontos encarnados.

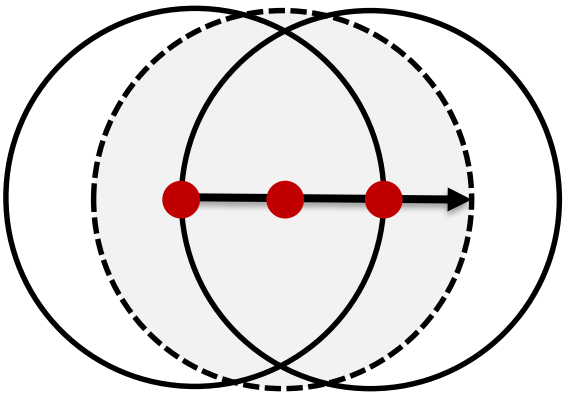

Figura 30. Sobreposição dos raios de pesquisa centrados na posição do dispositivo. A seta representa o deslocamento do utilizador. A sobreposição dos raios de pesquisa faz com que alguns dados devolvidos pelo servidor já existam em *cache*.

Por sua vez, o serviço *Web* devolve dados correspondentes a uma circunferência de raio igual a 50 metros com centro na posição de requerimento. A contagem do distanciamento é reiniciada, e o novo ponto de atualização de referência é atualizado.

O exemplo da Figura 30 expõe três pontos de atualização, marcados a encarnado. O primeiro ocorre no momento do arranque da aplicação. Os outros resultam de alterações de posição do utilizador. Este exemplo serve para ilustrar os dados que vão sendo requisitados.

Como podemos observar no exemplo teórico, a segunda circunferência, marcada a cinzento e com contorno a tracejado, está quase toda contida nas outras duas. Ou seja, no momento do segundo pedido, parte da informação recebida já existia em *cache* originada pelo primeiro pedido. E aquando do terceiro pedido, uma parte da sua resposta estava já contida na resposta anterior do servidor, bem como na resposta ao primeiro pedido.

Este exemplo tem cálculos e formas definidas perfeitamente. No caso prático do uso da aplicação SolAR, existe uma margem de erro do posicionamento, nomeadamente a correção da posição obtida. Essa correção varia consoante vários fatores e fazem com que as circunferências não se sobreponham exatamente como ilustrado na Figura 30. Quando o utilizador caminha, a sua nova posição é determinada com intervalos de tempo e de distância tendo em conta a última posição detetada, fazendo com que os 25 metros do exemplo possam ser 25,6 metros ou 27,68 metros. Em suma, o exemplo dado expõe o pior caso.

De forma a analisar a quantidade de informação redundante em *cache*, reuniu-se o conteúdo dos ficheiros de *cache* após trajetos em redor do edifício C6 da FCUL e contaram-se repetidos. Os gráficos da Figura 31 mostram os resultados obtidos para os três raios de pesquisa geográfica estudados.

Relativamente a 50 metros, nota-se que a regra é que exista repetição de dados, sendo que são poucos os casos onde um ponto aparece em apenas um ficheiro. A repetição de pontos concentra-se, principalmente, entre os quatro e os sete ficheiros podendo, no trajeto estudado, ir até onze ficheiros.

No caso de raio igual a 75 metros, a repetição de pontos concentra-se maioritariamente entre um a oito ficheiros, sendo que 30% dos pontos já só aparece num ficheiro, situação mais favorável que a com o caso anterior.

Por último, com raio igual a 100 metros, existe uma grande quantidade de pontos que aparecem uma única vez. Existe ainda um número considerável de pontos que se repetem ao longo de dois a seis ficheiros, mas quase nunca mais que isso.

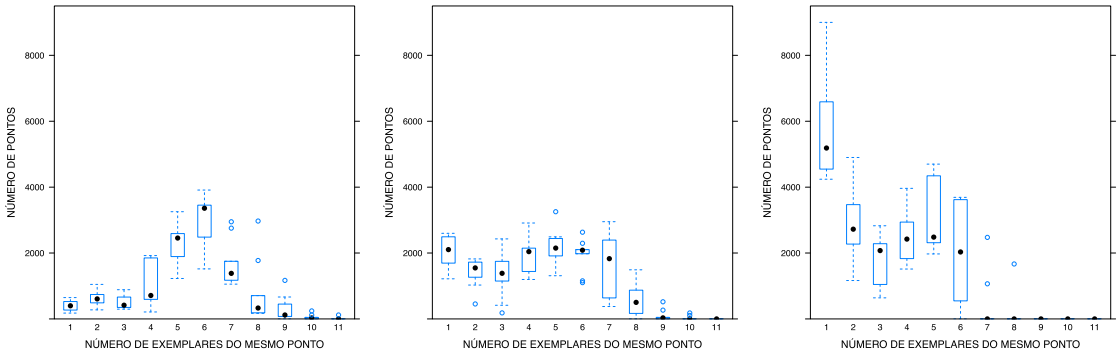

Figura 31. Quantidade de pontos que se repete numa *cache* com um raio de, da esquerda para a direita, 50, 75 e 100 metros.

Posições de pedidos de dados

Por fim, analisaram-se as posições ao longo do percurso que originavam, com mais frequência, a requisição de dados. Nesta análise, criaram-se ficheiros GPX com os pontos referentes a *misses*, ou seja, pedidos ao serviço *Web* de todas as experiências.

Nas figuras seguintes, cada cor dos pontos representa um dia de experiência e não apenas um percurso. Assim, na Figura 32 observamos os pontos em que se realizaram pedidos referentes a um raio de 50, 75 e 100 metros respetivamente.

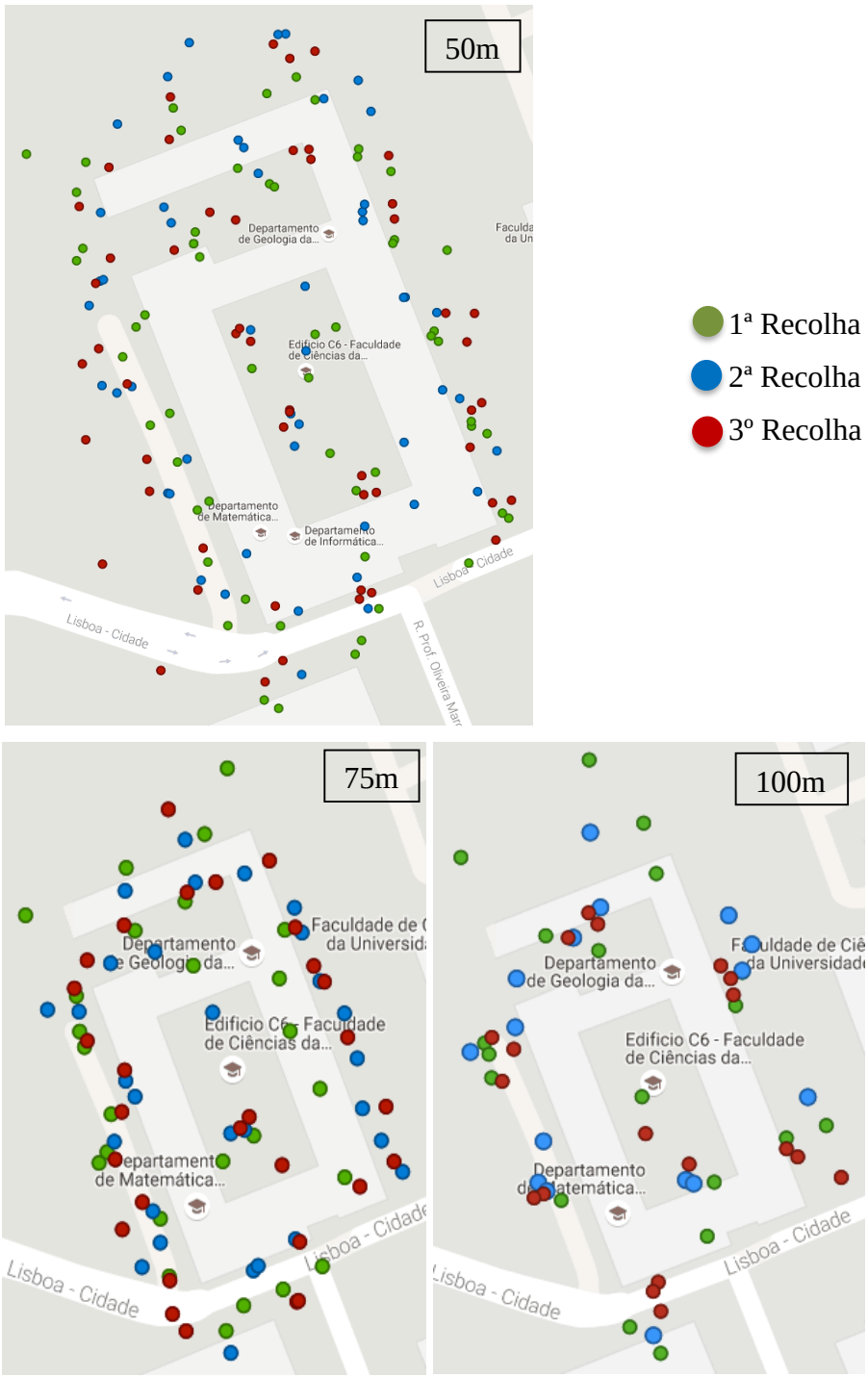

Figura 32. Aglomerado de pontos referentes a pontos onde ocorreram pedidos ao serviço *Web*.

Na Figura 33 podemos ver uma comparação entre as posições onde ocorreram mais frequentemente *hits* e *misses*, desta vez reunindo os dados de todos os trajetos, sem distinguir dias da experiência.

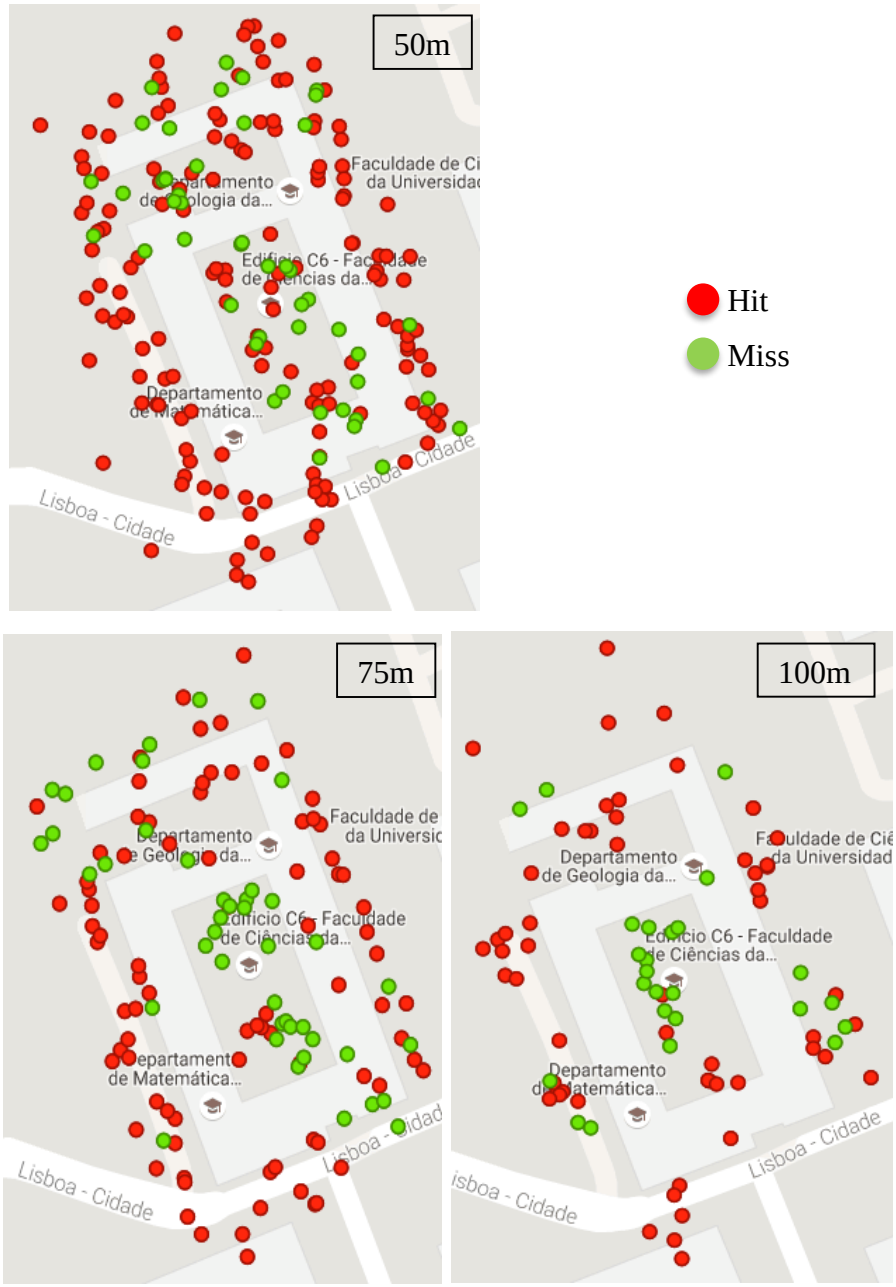

Figura 33. Posições onde ocorreram *hits* e *misses*.

Excetuando os saltos, posicionamentos detetados incorretamente pelo instrumento de posicionamento, é possível visualizar vários agrupamentos de pontos, que permitem perceber o correto funcionamento do algoritmo de requisição de dados. De notar que os agrupamentos de pedidos se tornam evidentes conforme se aumenta o raio de pesquisa.

Conclusão da avaliação

Tendo em conta todos os resultados obtidos, percebem-se os benefícios do incremento do raio de pesquisa, permitindo reduzir a quantidade de pedidos de dados ao serviço *Web* e respetivos, tempos de requisição, embora à custa do espaço ocupado.

No caso especifico do edifício C6, verificou-se que o raio de pesquisa que obteve uma melhor relação entre espaço ocupado, tempo despendido, número de pedidos ao serviço *Web* e percentagem de *hits* na *cache*, foi o raio igual a 75 metros.

3.7 Adaptações à aplicação SolAR

Durante a análise do código da aplicação SolAR, tentou-se perceber os pontos em que seria necessária uma futura intervenção aquando da existência dos serviços *Web*. Pretendeu-se minimizar a introdução de alterações, bem como manter o mais possível as interações entre os componentes já existentes.

Posto isto, ficou estabelecido que a estrutura de dados e lógica aplicacional se deveria manter. Em particular, continuaria a ser usado um ficheiro de dados, o qual deixaria de ser carregado só durante o carregamento da aplicação, mas sempre que fosse necessário obter novos dados.

Seguem-se as principais alterações realizadas na aplicação.

3.7.1 Alterações ao ficheiro de manifesto

No manifesto da aplicação Android foram introduzidas três modificações:

- Alteração do tamanho da *heap* para se poder obter uma maior quantidade de memória principal alocada. Esta modificação foi efetuada da seguinte forma:

  ```
  <application android:largeHeap="true">
  ```

- Declaração de novas intenções por parte da aplicação como a escrita de ficheiros, verificação do posicionamento, estado de rede e uso da Internet. Eis as permissões utilizadas:

  ```
  <uses-permission
  android:name="android.permission.WRITE_EXTERNAL_—
  STORAGE"
  />
  <uses-permission
  android:name="android.permission.ACCESS_FINE_—
  LOCATION
  ```

```
/>
<uses-permission
android:name="android.permission.ACCESS_NETWORK_—
STATE
/>
<uses-permission
android:name="android.permission.ACCESS_WIFI_—
STATE"
/>
<uses-permission
android:name="android.permission.INTERNET"
/>
```

- Declaração dos novos serviços Android utilizados pelo MiddSolAR.

3.7.2 Observação de ficheiros e recarregamento de dados

Devido ao protocolo em torno do ficheiro de *lock*, ver Figura 17, a aplicação SolAR passou a subscrever a observação desse ficheiro e a esperar eventos de escrita sobre ele. Foi criando um serviço para executar a tarefa constantemente e uma implementação do FixedFileObserver. O caminho do ficheiro de *lock* foi obtido através da API do MiddSolAR.

Quando o evento é acionado, a aplicação passou a proceder à nova leitura do ficheiro de dados e mostrar ao utilizador a nova informação. Estes novos procedimentos implicaram a alteração da classe SolAR_Activity.

3.7.3 Alteração da interface com o utilizador e definições

Para o utilizador é importante que tenha a noção que estão a ser carregados novos dados para visualização. Por isso foi introduzida uma indicação visual de sincronização de dados, na *action bar* da aplicação SolAR, como acontece habitualmente com outras aplicações quando efetuam a atualização de informação.

Foi ainda definido um parâmetro nas definições da aplicação que permite restringir ou alargar a correção do posicionamento utilizado para a visualização dos dados.

3.8 Sumário do capítulo

Neste capítulo abordaram-se todas as etapas da execução deste projeto. Primeiro, analisando o desafio pretendido e, em seguida, descrevendo a fase de desenvolvimento que procurou concluir os objetivos propostos. Por fim, foram apresentados os testes

exercidos sobre os componentes desenvolvidos e realizada uma avaliação dos resultados obtidos.

No capítulo seguinte são resumidas as principais contribuições dadas e competências adquiridas neste projeto, e os problemas encontrados. Por fim, são sugeridas melhorias e funcionalidades como trabalho futuro.

Capítulo 4 Conclusões

Este capítulo apresenta as principais contribuições deste projeto e as competências adquiridas durante a execução do mesmo. Enumera ainda os problemas encontrados ao longo da fase de desenvolvimento e de testes. Por fim, enuncia alguns melhoramentos possíveis e refere trabalho que pode ser feito em seguimento deste projeto.

4.1 Principais contribuições

Este trabalho contribuiu para a evolução da arquitetura da aplicação SolAR, oferecendo uma solução cliente-servidor em vez de todo o processamento ser feito numa aplicação móvel.

Assim, os dados sobre radiação solar agora encontram-se armazenados num servidor, em vez de terem que ser copiados manualmente para a aplicação móvel de cada utilizador. Para isso, foi implementada uma **base de dados espacial** que permite pesquisa de dados relativamente a uma posição geográfica, sendo possível obter automaticamente apenas os dados à volta do utilizador e não uma quantidade manualmente pré-selecionada pelo utilizador.

Para facilitar o acesso à informação armazenada no servidor, foram construídos vários **serviços *Web***, WSolAR, que servem de interface para leitura e escrita entre o cliente e a base de dados espacial, cobrindo fachadas de edifícios, pontos, radiação solar e painéis solares.

Foi desenvolvido um ***middleware*** MiddSolAR para aplicações cliente que serve vários propósitos: abstrair o uso dos serviços *Web*; atualizar o ficheiro de dados conforme a posição geográfica atual; e ter um mecanismo de armazenamento temporário, *cache*, evitando repetir os mesmos pedidos ao servidor.

A **aplicação SolAR foi adaptada** para fazer uso do *middleware*, podendo ser utilizada em vários locais sem que o utilizador se preocupe com o carregamento de dados, bastando para isso que tenha uma ligação de rede ativa. A aplicação, além de iniciar e

terminar o *middleware*, está também a executar em conformidade com o protocolo de interação com o ficheiro de *lock*, para saber quando estão disponíveis novos dados.

Foram executados testes para aferir a exequibilidade do sistema construído para este tipo de aplicação de realidade aumentada. Permitiram também perceber qual é uma **configuração adequada** para o *middleware* no contexto de visualização de dados de radiação em torno do edifício C6 da FCUL.

Com este trabalhado concluído, torna-se possível que sejam criadas aplicações concorrentes e alternativas à aplicação SolAR, utilizando o mesmo *middleware* e os mesmos serviços *Web*, aumentando o valor dos dados sobre radiação solar.

4.2 Competências adquiridas

Ao desenvolver este projeto, encontraram-se novos procedimentos e foram utilizados novos programas. Houve um primeiro contacto com base de dados espaciais, com tipos geográficos em SQL, com a aplicação QGIS para visualização de dados geográficos, e ainda pude criar e visualizar ficheiros GPX. Exercitou-se a construção de serviços *Web* e respostas em JSON, o desenvolvimento nativo de aplicações e serviços Android, e a análise estatística de dados. Estas novas experiências foram enriquecedoras e permitiram aumentar e exercitar conhecimentos.

4.3 Problemas encontrados

Durante a execução deste trabalho de PEI, foram encontrados vários problemas, que foram ultrapassados através de correções encontradas, ou que motivaram alterações na abordagem em curso. Os principais problemas encontrados foram os seguintes:

Excesso de memória ocupada pela aplicação móvel e necessidade de aumentar o valor de memória alocada. Numa abordagem inicial, a resposta JSON recebida após executar um pedido a um serviço *Web* era guardada em objetos e processada através da API de JSON nativa do SDK do Android. O processamento era efetuado por *parsing* dos dados, guardando em memória todos os dados, estivessem já processados ou não. Quando se recebe um grande número de pontos, o tamanho do objeto JSON em memória era também grande. Acrescendo a isso, a memória utilizada durante o processamento da resposta ia incrementado. Rapidamente se excedia a quantidade de memória disponi-

bilizada à aplicação pelo sistema, mesmo definindo o valor de memória alocada para o máximo possível.

A solução encontrada for utilizar uma biblioteca externa que processa a resposta JSON através de um *stream* de leitura. Deste modo, um cursor avança pelos dados da resposta, analisando-se o necessário e não aumentando desnecessariamente a quantidade de memória utilizada. Foram testadas duas bibliotecas para este efeito: a Jackson Streaming API e a GSON. Durante a seleção da biblioteca, verificou-se que o seu desempenho era semelhante, pelo que se optou pela biblioteca GSON devido ao seu desenvolvimento ser executado pela Google, esperando-se maior suporte e longevidade.

Limitações de uso do observador de ficheiros. De forma a perceber se um ficheiro era escrito por outra aplicação, foi necessário observar um ficheiro e registar, perante o sistema operativo, sobre que operações se deseja ser notificado. A classe que implementa essas operações é a FileObserver. No entanto, esta classe tem uma falha, levando a que apenas uma aplicação de cada vez consiga subscrever notificações sobre o ficheiro desejado. Para este projeto, foi necessário que a aplicação SolAR e o *middleware* MiddSolAR observem o mesmo ficheiro de *lock*. Utilizando o FileObserver, apenas o último processo a subscrever a observação é que recebia as notificações de eventos. A solução adotada, FixedFileObserver, garante que vários subscritores recebem notificações de eventos de leitura e escrita sobre um ficheiro.

Permissões necessárias para a utilização de ligações simbólicas. De modo a evitar criar uma cópia do ficheiro de *cache* quando este deve ser utilizado como ficheiro de dados, considerou-se utilizar uma ligação simbólica. No entanto, a criação desta ligação, requer permissão de administrador. Ou seja, o dispositivo teria que executar em modo *root* para que esta operação fosse concluída. Devido a esta necessidade, a utilização de ligações simbólicas foi descartada.

Qualidade de sinal de rede em certas zonas do edifício de teste. Durante a execução dos testes, verificou-se que, por vezes, o tempo entre a envio do pedido e a obtenção da resposta era grande comparado com o habitual. Verificou-se que estes casos aconteciam no momento em que o dispositivo móvel alterava entre rede 4G e 3G, ou perdia completamente o acesso à ligação de dados.

4.4 Trabalho futuro

No âmbito deste trabalho, foram introduzidos novos elementos ao projeto da aplicação SolAR, nomeadamente, uma base de dados espacial, serviços *Web* que fornecem os dados armazenados, e um *middleware* que interliga a aplicação cliente aos serviços. Em todos estes novos componentes podem ser exploradas alternativas e concebidas melhorias. Eis algumas sugestões que podem ser analisadas em trabalho futuro:

- Definição de índices na base de dados, que tornem as interrogações espaciais mais eficientes;
- Melhoramento da pesquisa geográfica, contemplando um critério de pesquisa de pontos que tenha em conta a orientação do dispositivo móvel ou a direção do caminho percorrido;
- Implementação do conceito de cenários na aplicação SolAR;
- Adição de novas políticas de substituição de ficheiros na *cache* e respetivos testes de desempenho;
- Uma melhor gestão dos dados que já existem em *cache*, nomeadamente, tendo em conta os pontos que se encontram repetidos e replicados por vários ficheiros.

Bibliografia

[1] U.S Energy Information Administration, «International Energy Statistics». [Em linha]. Disponível em: http://www.eia.gov/cfapps/ipdbproject/IEDIndex3.cfm?-tid=6&pid=29&aid=12. [Acedido: 25-Nov-2015].

[2] R. Azuma, «A survey of augmented reality», *Presence Teleoperators Virtual Environ.*, vol. 6, n. 4, pp. 355–385, 1997.

[3] D. Booth *et al.*, «Web services architecture», 2004. [Em linha]. Disponível em: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/. [Acedido: 12-Nov-2015].

[4] C. Pautasso, O. Zimmermann, e F. Leymann, «RESTful web services vs. "big" web services», em *Atas da 17th international conference on World Wide Web - WWW '08*, 2008, pp. 805–814.

[5] T. Erl, *SOA Design Patterns*. Prentice Hall, 2009.

[6] Arcitura Education Inc., «ServiceOrientation.com». [Em linha]. Disponível em: http://serviceorientation.com/whatissoa/service_oriented_architecture. [Acedido: 25-Nov-2015].

[7] JSON.org, «Introducing JSON». [Em linha]. Disponível em: http://www-.json.org/. [Acedido: 30-Nov-2015].

[8] H. Butler, M. Daly, A. Doyle, S. Gillies, T. Schaub, e C. Schmidt, «The GeoJSON Format Specification». [Em linha]. Disponível em: http://geojson-.org/geojson-spec.html. [Acedido: 30-Nov-2015].

[9] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, e F. Yergeau, «Extensible Markup Language (XML) 1.0». [Em linha]. Disponível em: http://www-.w3.org/TR/REC-xml/. [Acedido: 30-Nov-2015].

[10] Google Developers, «What is KML?» [Em linha]. Disponível em: https://developers.google.com/kml/. [Acedido: 30-Nov-2015].

[11] Open Geospatial Consortium, «Geography Markup Language». [Em linha].

Disponível em: http://www.opengeospatial.org/standards/gml. [Acedido: 30-Nov-2015].

[12] Open Geospatial Consortium, «Well-known text representation of coordinate reference systems». [Em linha]. Disponível em: http://www.opengeospatial.-org/standards/wkt-crs. [Acedido: 30-Nov-2015].

[13] «GPX: the GPS Exchange Format». [Em linha]. Disponível em: http://www.topografix.com/gpx.asp. [Acedido: 13-Set-2016].

[14] R. Ramakrishnan e J. Gehrke, *Database management systems*, 3.ª ed. McGraw Hill, 2003.

[15] International Organization for Standardization e International Electrotechnical Commission, «SQL/MM Part 3: Spatial». [Em linha]. Disponível em: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=53698. [Acedido: 26-Nov-2015].

[16] R. O. Obe e L. S. Hsu, *PostGIS in action*. Manning, 2011.

[17] Y. Fang, M. Friedman, G. Nair, M. Rys, e A.-E. Schmid, «Spatial indexing in Microsoft SQL Server 2008», em *Atas de 2008 ACM SIGMOD International Conference on Management of Data*, 2008, pp. 1207–1216.

[18] A. Furieri, «SpatiaLite». [Em linha]. Disponível em: https://www.gaia-gis.it/fossil/libspatialite/index. [Acedido: 30-Nov-2015].

[19] M. Akbari e S. R. H. Peikar, «Evaluation of free/open source software using OSMM model case study: WebGIS and spatial database», *Adv. Comput. Sci. an Int. J.*, vol. 3, n. 5, pp. 34–43, 2014.

[20] Stan Aronoff, *Geographic Information Systems: A Management Perspective*. WDL Publications, 1991.

[21] P. Selonen, P. Belimpasakis, Y. You, T. Pylvänäinen, e S. Uusitalo, «Mixed reality web service platform», *Multimed. Syst.*, vol. 18, n. 3, pp. 215–230, 2012.

[22] R. Li, S. H. L. Liang, D. Lee, e Y.-J. Byon, «TrafficPulse: A mobile GISystem for transportation», em *Atas da First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, 2012, pp. 9–16.

[23] J. M. da G. N. Pedrosa, «Visualização de dados em realidade aumentada», Unversidade de Lisboa, 2013.

[24] S. Silva, «Visualização de dados sobre radiação solar em realidade aumentada», Universidade de Lisboa, 2015.

[25] D. Jacobson, G. Brail, e D. Woods, *APIs: A strategy guide*. O'Reilly Media, 2011.

[26] The PostgreSQL Global Development Group, «PostgreSQL 9.4 Documentation». [Em linha]. Disponível em: http://www.postgresql.org/docs/9.4/interactive-/index.html. [Acedido: 30-Nov-2015].

[27] The Apache Software Foundation, «Apache HTTP server version 2.4 documentation». [Em linha]. Disponível em: https://httpd.apache.org/docs/2.4/. [Acedido: 30-Nov-2015].

[28] The PHP Group, «PHP Web Page». [Em linha]. Disponível em: https://secure.php.net/. [Acedido: 30-Nov-2015].

[29] J. Lockhart, A. Smith, R. Allen, e Slim Framework Team, «Slim framework documentation». [Em linha]. Disponível em: http://www.slimframework.com-/docs/. [Acedido: 30-Nov-2015].

[30] Oracle, «Java 8 platform standart edition API». [Em linha]. Disponível em: http://docs.oracle.com/javase/8/docs/api/. [Acedido: 30-Nov-2015].

[31] QGIS Development Team, «QGIS User Guide». [Em linha]. Disponível em: http://docs.qgis.org/2.8/en/docs/user_manual/. [Acedido: 30-Nov-2015].

[32] M. Akagi, «Qgis2threejs - About». [Em linha]. Disponível em: https://-plugins.qgis.org/plugins/Qgis2threejs/. [Acedido: 17-Set-2016].

[33] The R Foundation, «What is R?» [Em linha]. Disponível em: https://www.r-project.org/about.html. [Acedido: 30-Nov-2015].

[34] Android Open Source Project - Issue Tracker, «Issue 92329: Only one instance of FileObserver is notified». [Em linha]. Disponível em: https://code.google.com-/p/android/issues/detail?id=92329. [Acedido: 08-Ago-2016].

[35] Stack Overflow, «Multiple FileObserver on same file failed». [Em linha]. Disponível em: https://stackoverflow.com/questions/29227425/multiple-file-observer-on-same-file-failed). [Acedido: 08-Ago-2016].

[36] Android Developers, «android.location». [Em linha]. Disponível em: https://developer.android.com/reference/android/location/package-summary.html. [Acedido: 29-Set-2016].

[37] J. Simões, R. Gimènez, e M. Planagumà, «Big data y bases de datos espaciales: Un análisis comparativo», em *Atas das 9as Jornadas de SIG Libre*, 2015.

[38] «GeoHash». [Em linha]. Disponível em: http://geohash.org/. [Acedido: 25-Nov-

2015].

[39] Nvidia, «Nvidia Shield Tablet K1». [Em linha]. Disponível em: https://shield.nvidia.com/tablet/k1. [Acedido: 17-Set-2016].

[40] Apple, «iPhone 6s - especificações técnicas». [Em linha]. Disponível em: http://www.apple.com/pt/iphone-6s/specs/. [Acedido: 17-Set-2016].