UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Aplicação Descubra Acessível: Integração de Componentes de Acessibilidade para Pessoas Cegas

Dissertação de Mestrado em Engenharia Informática

João Paulo Martins Vicente

Orientador:

Professor Doutor João Manuel Pereira Barroso

Coorientador:

Doutor Hugo Ricardo Morais Fernandes

Vila Real, 2018

UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Aplicação Descubra Acessível: Integração de Componentes de Acessibilidade para Pessoas Cegas

Dissertação de Mestrado em Engenharia Informática

João Paulo Martins Vicente

Orientador:

Professor Doutor João Manuel Pereira Barroso

Coorientador:

Doutor Hugo Ricardo Morais Fernandes

Vila Real, 2018

Resumo

Ao longo dos anos o aumento do número e da utilização de aplicações móveis tem sido considerável, mas, apesar desta massificação, nem todas são acessíveis ou garantem a utilização por todos os tipos de audiência, independentemente das suas limitações. As aplicações de orientação e navegação geográfica são muito utilizadas no dia a dia, quer para deslocações programadas, quer para conhecer melhor um determinado espaço, em turismo ou na prática de atividades georreferenciadas.

No âmbito do projeto CE4Blind nasceu uma aplicação móvel desenhada com a intenção de ajudar na orientação e navegação geográfica de pessoas cegas, garantindo a acessibilidade da sua utilização através de formas de interação pessoa-computador que assentam em atuadores hápticos, tecnologia de texto-para-voz e numa bengala eletrónica que permite ao utilizador interagir com a aplicação e com sensores colocados na infraestrutura. A aplicação móvel Descubra, por outro lado, é uma aplicação de turismo, de âmbito mais generalista, que funciona como um guia turístico que pretende entregar informação cultural, histórica e georreferenciada.

Nesta dissertação é apresentada uma proposta de integração de funcionalidades do sistema CE4Blind na aplicação móvel Descubra permitindo, assim, transformar a Descubra numa ferramenta acessível e que dê a uma pessoa cega a capacidade de se deslocar de uma forma mais independente, com mais capacidade de se orientar num local que não conheça.

Ao longo desta dissertação são apresentadas algumas das várias formas utilizadas recentemente para integrar aplicações, com especial enfoque na integração de aplicações móveis. Também é apresentado o trabalho desenvolvido para incluir componentes de acessibilidade na aplicação Descubra, nomeadamente na inclusão do uso da bengala eletrónica do sistema CE4Blind para o utilizador interagir com a aplicação. São também descritas as diferentes formas como as tecnologias foram adaptadas para permitir a orientação e navegação de uma pessoa cega num local que não conheça, promovendo assim a sua independência.

Abstract

Over the years the number and the use of mobile applications has had a considerable increase, but despite this massification, not all of them are accessible or guarantee the use by all types of audience, regardless of their limitations.

Under the project CE4Blind, an application was developed with the intention of helping blind people in the orientation and geographical navigation, guaranteeing its accessibility through human-computer interaction based on haptic actuators, text-to-speech technology and an electronic cane that allows the user to interact with the application and the sensors placed in the infrastructure. On the other hand, Descubra, is a more generalist tourism application that works as a tourist guide delivering cultural, historical and georeferenced information.

This dissertation presents a proposal for the integration of some of the functionalities developed in the CE4Blind project into the mobile application Descubra, allowing to turn Descubra into an accessible application that empowers a blind person with the ability to move more independently, with increased ability to orient himself in unknow places.

This dissertation overviews some of the different ways used, nowadays, to integrate applications, with special focus on the integration of mobile applications. Then, the work developed to include accessibility components into the Descubra application is presented, namely the use of CE4Blind's electronic white cane to interact with the application, as well as how the different technologies have been adapted to allow the orientation and navigation of the blind in unfamiliar places, thus promoting their independence.

AGRADECIMENTOS

Durante a vida existem milhares de obstáculos que temos de enfrentar sendo que os mais difíceis por mais força que tenhamos não os conseguiremos superar sozinhos. Para tal nesta fase final mais um obstáculo foi superado e claro que não poderia ter sido superado isto sozinho, como tal quero agradecer a todos os que tiveram sempre do meu lado a apoiar, a todos com quem eu aprendi e cresci para eles um muito obrigado.

Primeiramente quero agradecer aos meus pais do fundo do coração por todo o apoio demonstrado desde o início ao fim, tudo aquilo que fizeram e fazem por mim, sem eles nada disto era possível, gabo-lhes a paciência que tiveram para me aturar, todas as chatices que lhes dei, todos os sermões, que me fizeram crescer e aprender, e assim conseguir o que tenho hoje. Mais uma vez obrigado a VÓS.

Gostaria de agradecer a toda a minha família, mas especialmente a minha avó que esta faz tudo pela família toda e sem ela nada tinha sido feito. Obrigado AVÓ.

Um especial agradecimento a minha namorada que sem o apoio dela isto não era possível, sem ela não teria acabado tão cedo, pois a fase inicial de escrita foi muito complicada. Um apoio de inicio ao fim, aturando todos os meus stresses e fazendo que me levantasse todas as vezes que ficava em baixo. Um grande obrigado a TI.

Um grande obrigado também a minha segunda família os escuteiros, sem estes não teria crescido da forma como cresci e o apoio deste foi sempre fundamental para tudo, destes todos um agradecimento mais especial as minhas meninas, Lara Ribeiro, Joana Rego, Bárbara Ribeiro e Beatriz Pinto e ao meu menino Stefano Guerra, obrigado a VÓS por tudo aquilo que foram e que são para mim.

Agradeço também a todos os meus amigos, todos aqueles que estiveram do meu lado a apoiar me para assim conseguir passar por isto, todos os desabafos e aventuras que tive, deste um especial agradecimento ao Fernando Freitas, Diogo Neves, José Cardoso, Jorge Silva e Rui Silva, que me ajudaram neste obstáculo e em todos os outros tanto na licenciatura como no mestrado, obrigado a VÓS, aos outros que apesar de não terem aqui o nome peço desculpa mas seria uma lista interminável obrigado a TODOS VÓS.

Gostaria de agradecer também ao professor João Barroso por todo o apoio e orientação neste projeto para assim conseguir concluir esta fase, tudo aquilo que me ensinou obrigado a SI.

Um especial grande agradecimento ao meu coorientador Hugo Fernandes que mais que orientador foi um amigo, amigo este que me ensinou, ajudou e apoio em todos os problemas ao longo deste obstáculo e que assim fosse possível terminar este árduo trabalho, sem ele nada seria possível um grande obrigado a TI por tudo, isto foi também graças a ti.

Agradeço também HCILab por toda ajuda demonstrada durante este obstáculo e permitiram que isto fosse possível, em especial ao André Sousa por tudo aquilo que aprendi com ele e que sem ele não seria possível terminar isto. Obrigado a TI.

Por fim gostaria de agradecer a WorldIT por todo a contribuição que deram a este projeto, para este fosse possível concluir, todos estes foram importantes, obrigado WorldIT.

Aqueles que me esqueci de mencionar obrigado, não é por não mencionar que deixam de ser importante pois todos foram importantes para isto. Obrigado a VÓS.

Por tudo, um MUITO OBRIGADO.

João Vicente

ÍNDICE

LISTA DE FIGURAS

Lista de Tabelas

Lista de Equações

Acrónimos

API - Application Programming Interface

AP - Access point

BLE - Bluetooth Low Energy

EAI - Enterprise Application Integration

GIS - Geographic information system

GPS - Global Positioning System

HTTP - Hypertext Transfer Protocol

JSON - JavaScript Object Notation´

REST - Representational State Transfer

RFID - Radio-frequency identification

RSSI - Received signal strength indication

SOA - Service-oriented Architecture

SOAP - Simple Object Access Protocol

UDDI - Universal Description, Discovery, and Integration

URI - Uniforme Resource Identifier

UTAD - Universidade de Trás-os-Montes e Alto Douro

UUID - Universally Unique Identifier

WSDL - Web Services Description Language

XML - Extensible Markup Language

1. INTRODUÇÃO

1.1. Enquadramento

No século XXI, tem-se assistido a um grande aumento da utilização de telefones móveis, principalmente com a evolução do telemóvel tradicional para o *smartphone*, tendo havido um crescimento bastante acentuado na última década. O *smartphone* deixou de ser apenas um telemóvel com características avançadas de assistente profissional, para se tornar num dispositivo que combina entretenimento, ferramentas profissionais e de lazer, incluído a georreferenciação. Além disso, tem-se assistido mais recentemente à inclusão de vários tipos de sensores, nomeadamente biométricos, barométricos, de visão e de movimento. Um estudo publicado pela Smart Insights (Sheppard, 2018) revela que a utilização de *smartphones* em 2010 era de 30%, tendo aumentado para 72% em 2016 na população com idade superior a 16 anos. Esse estudo revela também que em 2010 o dispositivo mais utilizado para aceder à internet era o computador, sendo superado em 2016 pelo *smartphone*, que em 2010 era utilizado em apenas 31% dos acessos. Podemos ver, assim, que os *smartphones* estão cada vez mais presentes nas tarefas do nosso dia a dia, sendo que a maior parte da população recorre quase exclusivamente às suas ferramentas. Este aumento no uso de *smartphones* levou a que as empresas tivessem de se adaptar, tornando os seus sites responsivos para as diferentes plataformas e até mesmo à criação de aplicações móveis que entreguem a informação de uma forma mais personalizada. Um estudo feito pela companhia Statista (Statista, 2018) revela que em dezembro de 2009 existiam cerca de 16 000 aplicações publicadas na Google Play Store, tendo aumentado até dezembro de 2017 para cerca de 3 500 000 aplicações. Com isto torna-se cada vez mais importante a integração de aplicações para agilizar processos de negócio.

Dada esta variedade de usos e aplicações disponíveis, a capacidade de integração entre aplicações é cada vez mais importante e, para isso, existem diferentes abordagens, sendo que a mais comum é a abordagem *Enterprise Application Integration* (EAI). A atuação, segundo esta abordagem, abrange vários níveis, nomeadamente ao nível da interface do utilizador, dos processos de negócio, e aos níveis aplicacional e de dados (Laftsidis, 2000). Estes níveis são diferentes consoante tipo de tecnologia utilizada e o tipo de integração que pretendida. Apesar da contínua evolução dos dispositivos móveis,

estes continuam a ter limitações comparativamente a um computador, tais como limitações ao nível da memória, da capacidade de processamento e da largura de banda (Linthicum, 2003). Uma abordagem muito comum, no que diz respeito à integração de aplicações móveis, é a de fazer o processo de integração com recurso a *Application Programming Interfaces* (APIs), pois as necessidades de processamento são menores. Deve ter-se em conta que, por vezes, ao nível empresarial, a integração entre aplicações, por si, não é suficiente para satisfazer todas as necessidades, sendo por vezes necessário implementar novos métodos, semelhantes a métodos já existentes, mas seguindo abordagens personalizadas.

1.2. Motivação

O trabalho descrito nesta dissertação enquadra-se num projeto resultante de uma parceria entre a Universidade de Trás-os-Montes e Alto Douro (UTAD) e a empresa WorldIT, com o objetivo global de integrar algumas das funcionalidades do sistema CE4Blind, desenvolvido na UTAD, na aplicação móvel Descubra, desenvolvida na WorldIT.

A aplicação móvel desenvolvida no âmbito do projeto CE4Blind, é uma aplicação em contínuo desenvolvimento e que tem como objetivo global ajudar na orientação e navegação inclusiva de pessoas cegas num determinado espaço, previamente mapeado, tanto em ambiente de interior como exterior. Esta aplicação é composta por diferentes módulos para realizar as várias tarefas necessárias no processo de orientação e navegação. Na vertente de orientação, o utilizador pode deslocar-se num determinado espaço recebendo informações de contexto sobre o local onde se encontra, nomeadamente pontos de interesse ou perigos existentes. Quando este solicitar, a aplicação pode também dar uma descrição do que se encontra à sua volta, enunciando os pontos de interesse que se encontram num determinado raio (programável). Na vertente de navegação, o utilizador poderá selecionar um ponto de interesse, e a aplicação irá dar instruções de navegação de forma a que possa chegar até esse ponto, avisando-o quando tem de mudar de direção e sobre quantos metros terá de andar nessa direção.

A Descubra é uma aplicação de turismo, de âmbito generalista, que tem como objetivo geral ajudar os seus utilizadores a conhecer melhor uma determinada cidade,

fornecendo informações sobre diversos pontos de interesse e diferentes atividades que estão, ou vão decorrer nessa cidade, facilitando assim a comunicação entre os gestores da cidade e as diferentes pessoas que a decidem visitar. Esta plataforma agrega informações que são entregues a diversas aplicações móveis, destinadas a diferentes municípios. Destaca-se que esta plataforma ganhou, em 2017, um prémio para a "melhor app de turismo em Portugal" (NiT, 2017).

Assim, a motivação geral relacionada com o trabalho apresentado nesta dissertação prende-se com a necessidade de integrar algumas das características de acessibilidade desenvolvidas no âmbito do sistema CE4Blind na aplicação Descubra. pessoas com necessidades especiais necessitam de aplicações e tecnologias acessíveis, e para isso é necessária uma adaptação das aplicações já existentes para que se tornem acessíveis e, assim, permitam uma utilização por todos.

1.3. Objetivos e contribuição

O fato de que existem de milhões de aplicações móveis disponíveis no mercado, para os mais variados propósitos é relevante, mas, apesar da extensão atingida por este mercado, isto não significa que estas ferramentas sejam utilizáveis por todos.

O objetivo global do trabalho descrito nesta dissertação é o de dotar uma aplicação móvel de turismo georreferenciado de funcionalidades acessíveis a pessoas cegas, de forma a que seja possível uma utilização da aplicação sem necessidade da ajuda de terceiros, tornando-as mais independentes. Desta forma, além das vantagens evidentes no que diz respeito à acessibilidade da aplicação, pretende-se que os utilizadores cegos sintam reduzido o estigma habitualmente associado à sua limitação física, passando a ser algo que não os afeta no que diz respeito à sua mobilidade. Com o auxílio da bengala eletrónica o utilizador poderá utilizar o menu da Descubra e das diferentes tecnologias de localização para se deslocar para locais turísticos do seu interesse sabendo informações relevantes daquele local e também sabendo onde se encontra. Sendo este projeto desenvolvido em colaboração com a empresa WorldIT será feito um levantamento de requisitos para se decidir se todas estas funcionalidades serão integradas ou se será necessário fazer alguma adaptação dessas funcionalidades.

Assim, pretende-se que, como resultado final do trabalho desenvolvido, um utilizador da aplicação Descubra tenha a possibilidade de utilizar a aplicação de forma acessível com o auxílio da bengala eletrónica desenvolvida no projeto CE4Blind, para, quer em interior, quer em exterior, ser capaz de se orientar e de navegar, recebendo informações sobre o seu contexto geográfico e instruções sobre como pode chegar a um destino à sua escolha, respetivamente. Pretende-se que estas funcionalidades sejam desenvolvidas e estejam disponíveis de forma modular, tal como no desenvolvimento do sistema CE4Blind.

1.4. Estrutura da dissertação

A presente dissertação encontra-se estruturada em sete capítulos, que refletem as diversas fases do trabalho desenvolvido, nomeadamente o estudo de como pode ser feita a integração de aplicações móveis, bem como o modo como foi feito o desenvolvimento do trabalho necessário para tornar a aplicação Descubra mais acessível a todos. Assim, esta dissertação começa por apresentar, no capítulo dois, um estudo sobre os diferentes níveis de integração existentes entre aplicações, de forma mais genérica. Seguidamente, no capítulo três são apresentadas as formas mais comuns de integração, ao nível das aplicações móveis. De forma conjunta, os capítulos dois e três fazem uma exposição do estado da arte no que diz respeito à integração de aplicações. No capítulo quatro é exposta, na forma de análise, uma descrição detalhada da composição das duas aplicações cujas funcionalidades foram integradas, o CE4Blind e a Descubra, referindo de forma aprofundada os diferentes módulos que foram integrados. A proposta de integração, correspondendo ao trabalho central desta dissertação, é apresentada no capítulo cinco, contendo os requisitos definidos pela WorldIT e expondo os módulos que foram definidos para integração bem como a forma da integração. O desenvolvimento associado à integração é apresentado no capítulo seis, bem como a forma como foi feita a integração das várias funcionalidades. Finalmente, no capítulo sete são apresentadas conclusões sobre o trabalho desenvolvido e são tecidas algumas considerações sobre possibilidades de trabalho futuro.

2. INTEGRAÇÃO DE APLICAÇÕES

2.1. Enquadramento

Com a existência de milhões de aplicações cujas funcionalidades ou informação podem ser partilhadas entre si, a integração entre aplicações é importante pois é pouco produtivo e pouco rentável criar repetidamente funcionalidades de raiz, quando é possível facilitar o trabalho e aumentar a rapidez de implementação das funcionalidades através da partilha de módulos e dados entre várias aplicações. Tradicionalmente, uma aplicação era pensada com o intuito de atingir um objetivo, realizar um conjunto de tarefas, um processo de negócio ou algo de relevante para um utilizador, não existindo o pensamento de que a partilha de informação entre aplicações fosse muito importante, ou necessário. As aplicações funcionavam fechadas sobre elas próprias. Ao longo dos últimos anos, principalmente com o surgimento da internet, tem ocorrido uma grande evolução no modo como funciona o negócio e também uma evolução nos interesses e necessidades dos utilizadores. Esta evolução do modo com funciona o negócio potenciou a integração de aplicações. Dentro de uma empresa surgem frequentemente a necessidade de troca de processos e dados entre os diferentes departamentos.

Tradicionalmente os sistemas informáticos de uma empresa estavam unicamente ligados entre si dentro do próprio departamento, não havendo troca de informações de forma digital entre os diferentes departamentos, resultando na existência de muitas aplicações, cada uma com propósitos fechados e bem definidos. Neste contexto surgiu uma forma de integração designada por *Enterprise Application Integration* (EAI) e que tinha a habilidade de partilhar os processos de negócio e dados entre os diferentes sistemas e departamentos como se o sistema fosse monolítico (Laftsidis, 2000). O EAI tem como objetivo geral facilitar a integração de sistemas e aplicações. É uma forma de integração relativamente antiga, onde toda a informação era processada num *mainframe*, e todos os processos e dados se encontravam num ambiente homogéneo fazendo com que a integração fosse mais simples, pois tratava-se de um sistema que era centralizado. Com a evolução dos sistemas e das necessidades, surgiram novas tecnologias que introduziram novos paradigmas de programação, mais simples para o programador. Isto levou ao desenvolvimento de novas aplicações, em sistemas distribuídos, ao desenvolvimento de

packaged applications e vários tipos de bases de dados, complicando assim a integração de aplicações (Laftsidis, 2000).

2.2. Níveis de integração de aplicações

Quando é necessário efetuar uma integração, é importante ter uma noção bem definida do que se pretende integrar e quais os objetivos. Com as grandes diferenças que, obviamente, existem entre as várias aplicações, em termos dos processos e dos dados, o EAI pode atuar em diferentes níveis, nomeadamente:

- Nível dos dados;
- Nível da interface da aplicação;
- Nível dos métodos da aplicação;
- Nível da interface do utilizador.

Ou seja é necessário saber se pretendemos os atuar sobre os dados ou sobre os processos de negocio e, para tal, é preciso entender os processos, saber o que nos é fornecido pela aplicação para efetuar a integração e se temos acesso direto ao processos/dados, ou se teremos de aceder a estes de forma indireta (Laftsidis, 2000).

2.2.1. Integração ao nível dos dados da aplicação

Na maioria dos casos esta forma de integração pode ser a mais simples e rentável de se fazer. A integração ao nível dos dados tem como objetivo principal a partilha de dados entre sistemas, podendo ser feita entre bases de dados ou entre a base de dados e diferentes aplicações. Uma das maiores vantagens da utilização deste tipo de integração é a existência atual de um grande número de ferramentas e técnicas que facilitam a transferência e a adaptação dos dados entre os diferentes sistemas, o que torna as necessidades de alterações ao nível da lógica das aplicações e na estrutura das bases de dados mínimas ou até mesmo inexistentes.

Este tipo de integração é complexo porque o responsável pela implementação tem que conhecer a complexidade existente nos diferentes tipos de tecnologias de base de dados e a forma como a informação se movimenta nos vários sistemas. Isto é muito importante pois a adição ou alteração de dados é muito frequente e é necessário manter

controlo sobre o que está a acontecer a cada momento, que alterações se devem efetuar para suportar a utilização de diferentes tecnologias e aplicações. A forma como os diferentes sistemas de gestão de base dados ou aplicações gerem os dados é diferente entre si e uma integração de forma errada pode gerar conflitos ou destruir dados, fazendo com que o sistema deixe de funcionar. Por vezes, em certas aplicações pode ser difícil utilizar este tipo de integração porque os dados podem estar tão intrinsecamente ligados com a logica da aplicação, que o uso da integração ao nível dos dados insuficiente e deve utilizada também ao nível dos métodos da aplicação. A utilização destes dois tipos de integração implica a adição de uma camada extra sobre o sistema, aumentando a sua complexidade. Assim, ocasionalmente, pode ser utilizada apenas a integração a nível dos métodos da aplicação.

A atuação ao nível dos dados tem a grande vantagem de não implicar alterações ao nível da lógica do negócio. A movimentação dos dados é transparente para utilizador pois este não tem a perceção de que isto acontece no *backend*. Atualmente as tecnologias existentes neste âmbito permitem que a movimentação dos dados seja feita em tempo real, mas devem ser implementadas por forma a que não se gerem conflitos entre os diferentes sistemas que partilham o mesmo tipo de dados.

2.2.2. Integração ao nível da interface da aplicação

Atualmente o desenvolvimento de aplicações é cada vez mais direcionado no sentido de serem capazes de fornecer serviços ou dados a outras aplicações. Esta abordagem é cada vez mais frequente, invertendo a tendência de as aplicações se fecharem sobre elas próprias. Deste modo, aquando da criação de novas aplicações é reduzida a necessidade de replicação de código.

A integração ao nível da interface da aplicação é feita através do uso de interfaces de *packaged* ou *custom applications* que são criadas pelos programados de forma a que seja possível encapsular serviços da aplicação mantendo formas universais de acesso. Estas interfaces permitem o acesso direto a processos de negócio ou dados e podem ser desenhadas de forma a que sejam bastante limitadas ou a que possibilitem o acesso a muitas caraterísticas do processo. Os programadores encapsulam assim os serviços que pretendem e expõem estas interface de forma a não forçarem outros programadores a invocar a interface do utilizador para obter informação, ou a ter que aceder diretamente à

base de dados. Desta forma são mantidos mecanismos que permitem a partilha de informação.

Neste tipo de integração pode ser notada uma semelhança com os dois níveis de integração já mencionados, nomeadamente ao nível dos métodos da aplicação, pois existe uma partilha de métodos entre a interface do utilizador e a utilização das interfaces criadas para efetuar a integração. Esta distingue-se na forma como aborda a integração na partilha de métodos e dados. Esta forma de integração, como referido anteriormente, fornece interfaces aos utilizadores enquanto nos outros tipos de integração é necessária a criação de mecanismos que permitam a partilha das interfaces. No caso da integração ao nível da interface do utilizador, esta depende de todas as interações que o utilizador realize sobre a aplicação, sendo, somente assim, possível aceder aos diferentes métodos e dados. Ao nível dos métodos da aplicação é, então, necessário o uso de diferentes mecanismos para aceder às interfaces.

Atualmente, é muito comum a utilização de *application programming interfaces* (APIs) que utilizam mecanismos comuns de forma a facilitar o acesso e o uso das interfaces. Os programadores evitam, assim, a necessidade de ter de aprender como implementar interfaces que não sejam compatíveis com as suas aplicações, sendo assim mais fácil efetuar a integração devido à universalidade que as interfaces fornecem.

2.2.3. Integração ao nível dos métodos da aplicação

A integração ao nível dos métodos da aplicação tem como objetivo principal a partilha da lógica de negócio ou dos métodos. Esta forma de partilha é bastante antiga e sempre teve como objetivo principal a reutilização de código, diminuindo a redundância. Esta partilha de métodos evoluiu para o domínio da integração de aplicações.

A integração ao nível dos métodos da aplicação é feita definindo os métodos que serão partilhados e a infraestrutura que efetua essa partilha. Esta estrutura que gere a partilha da informação pode ser um servidor ou um elemento programático que faça a troca de objetos entre aplicações. Esta definição de métodos e a necessidade de existir uma infraestrutura para dar este suporte pode torna-se pouco rentável pois, ao contrário dos outros níveis de integração, há a necessidade de efetuar alterações na forma de partilha dos métodos/dados e também nas aplicações que pretendem aceder aos diferentes

métodos. Estas alterações são dispendiosas pois implicam uma necessidade de testes de integração bastante exaustivos. Sendo assim, consoante o objetivo da integração a realizar, é necessário avaliar se implementar este tipo de integração se torna mais rentável do que implementação dos outros tipos de integração já expostos. Se por um lado este tipo de integração é dispendioso pela necessidade de uma estrutura de suporte, por outro lado é dado acesso exatamente os métodos desejados, sabendo ao certo o que cada um faz e o que cada um fornece.

2.2.4. Integração ao nível da interface do utilizador

Este nível de integração é considerado o mais primitivo de todos mas é, também, o mais frequente e necessário, pois na maioria das aplicações existentes a interface com o utilizador é a única maneira para aceder à logica e aos dados da aplicação. Este processo de integração é conhecido como *"screen scraping"*, sendo este processo uma forma de extrair a informação dos processos e dados da aplicação através do acesso à interface do utilizador. Este nível de integração é muito diferente dos outros níveis já descritos e é, por vezes, muito mais complexo, pois é necessário saber a forma como utilizador interage com a aplicação e como os dados se movimentam pelos diferentes métodos da aplicação. Isto pode trazer bastantes problemas pois a interface do utilizador não deve ser utilizada para a extração de dados pela aplicação (é concebida apenas para a apresentação dos dados ao utilizador) e nem sempre os dados que a aplicação precisa para os seus processos internos são aqueles que são apresentados ao utilizador. Este nível de integração deve ser utilizado apenas quando não for possível implementar nenhum dos outros níveis.

2.3. Considerações Finais

Pode-se verificar que a necessidade de existirem diferentes níveis de integração liga-se diretamente à forma como as aplicações são planeadas no momento do seu desenvolvimento, isto é, se foram planeadas para uma utilização isolada, ou se foram planeadas com a intenção de serem partilhadas. Alguns destes níveis são mais complexos de implementar que outros, sendo alguns deles somente utilizados em casos específicos e dependendo do que se pretende da aplicação. Na Tabela 1 podemos ver alguns das diferenças entre os vários níveis.

TABELA 1 - TABELA RESUMO DOS NÍVEIS DE INTEGRAÇÃO DE APLICAÇÕES

Nível	Vantagem	Desvantagem	Exemplo	Infraestrutura
Dados da Aplicação	Mais simples	Existência de várias tecnologias de base de dados	*Export, Transform Import*	Não
Interface da Aplicação	Formas universais de acesso	Por vezes os métodos disponibilizados não são os que o programador pretende	APIs	Não
Métodos da Aplicação	Métodos disponibilizados são os que o programador pretende	Dispendioso	Serviços	Sim
Interface do Utilizador	–	Mais complexo	*Screen Scrapping*	Não

No âmbito deste trabalho os níveis de integração que serão utilizados, serão a integração ao nível da interface da aplicação com a utilização de APIs e ao nível dos métodos da aplicação com a utilização de Serviços, estando ao mesmo tempo a ser utilizado a nível dos dados da aplicação, pois a utilização de serviços tem como objetivo o acesso a base de dados externas à aplicação.

3. INTEGRAÇÃO DE APLICAÇÕES MÓVEIS

3.1. Enquadramento

Os *smartphones*, apesar das constantes evoluções nas suas características, ainda apresentam várias limitações comparativamente a um computador. São limitações, por exemplo, ao nível da capacidade de processamento, memória e largura de banda. Por esse motivo, as formas de integrar aplicações móveis têm sido mais focadas em alguns dos níveis já expostos no capítulo anterior, recorrendo à utilização de tecnologias mais especificas e orientadas à computação no domínio dos dispositivos móveis. As formas mais comuns de integração, neste campo, são a integração com o uso de serviços, a *Service-Oriented Application Integration* e a utilização de APIs. Este último caso é o mais comum uma vez que é a forma de integração menos 'pesada' ao nível das necessidades de processamento, memória e de banda larga.

3.2. Service-Oriented Application Integration

A utilização da *Service-Oriented Application Integration* permite a partilha de serviços e informações através de uma infraestrutura que é desenvolvida para gerir as várias trocas. Esta infraestrutura garante e gere o acesso a bases de dados, outras aplicações ou processos de negócio.

Historicamente, o recurso a este tipo de integração nasceu com a tentativa de as empresas conseguirem reutilizar código e minimizar a redundância nas aplicações do seu ecossistema empresarial. Os serviços eram fornecidos às diferentes aplicações por um servidor comum, na empresa, de modo a facilitar o desenvolvimento de aplicações que partilhassem da necessidade de operar sobre informações comuns. Esta utilização de serviços é suficiente quando a partilha é feita dentro de uma única organização. No entanto, atualmente, a partilha de dados entre organizações é muito frequente, gera grandes volumes de dados e é necessário que integração entre aplicações funcione corretamente nessas condições. Para resolver esse problema, foram criados os *Web Services*, que permitem o acesso a serviços de forma remota. Desde a sua conceção, os

web services sofreram várias evoluções e, hoje em dia, existem *web services* assentes em diferentes arquiteturas (Linthicum, 2003).

3.2.1. Web Services

Como já foi referido, um *web service* permite o acesso de forma remota a serviços. O uso de *web services* teve início em 2001, um ano após ter sido lançado o protocolo *Simple Object Access Protocol* (SOAP), em 2000. O protocolo SOAP permite a comunicação de dados com base em *Extensible Markup Language* (XML) sobre o *Hypertext Transfer Protocol* (HTTP). O HTTP é um protocolo de comunicação que atua ao nível da camada de aplicação do modelo *Open Systems Interconnection* (OSI) e é a base para a comunicação de dados na internet. Ainda em 2001, outras duas expecificações foram aceites: o *Web Services Description Language* (WSDL), baseado também em XML e que é responsável por descrever a interface do *web service*; seguido do *Universal Description, Discovery and Integration*, que fornece um mecanismo padrão para a utilização dos serviços.

Com os três componentes referidos, foram definidos os primeiros *web services*. Estes são baseados na arquitetura *Service-oriented Architecture* (SOA), que é um modelo de *design* que encapsula toda a lógica dentro de um serviço e permite a interação através de um protocolo de comunicação comum. Um *web service* pode ser considerado a representação *web* de uma implementação SOA. Na sua implementação um *web service* é considerado como um bloco independente de todos os outros, e pode atuar sozinho ou pode efetuar pedidos a outros serviços. Deste modo, um *web service* não é apenas um fornecedor de serviços também um consumidor de serviços (Erl, 2004).

3.2.1.1. Web Services Definition Language (WSDL)

Para a utilização de *web services* é necessário que exista uma forma de estes poderem ser integrados com outros serviços ou aplicações. Neste contexto, surge a *Web Services Definition Language* (WSDL) que é uma linguagem de definição de uma interface padrão, com a descrição e a definição do *web service*. Com o recurso à WSDL é possível utilizar o *web service*, com os seus diferentes métodos, e para os objetivos para os quais este ele foi definido. O WSDL é baseado em XML e contém os seguintes construtores: *interface*, *message*, *service* e *binding*.

O WSDL é constituído pela definição de dois elementos distintos: o *abstract interface definition* e o *concrete definition*. Na definição do *abstract interface definition* é feita uma descrição da interface do *web service*, sendo constituído pela *interface* e pela *message*. A *interface* contém as operações que podem ser utilizadas no *web service*. Por sua vez, as operações são referentes a um único método e estão relacionadas com as *messages*, que são os parâmetros e de entrada e de saída para as diferentes operações. O *concrete definition definition* guarda os detalhes que permitem que o *web service* faça a correta utilização dos protocolos SOAP e HTTP e é constituída pelo *service* e pelo *binding*. No *service* guardam-se as diferentes informações sobre os protocolos e a sua forma de acesso. O binding tem o papeç de interligar todos estes elementos, fornece uma forma de aceder ao *web service* e faz a ligação entre os diferentes construtores, associando os protocolos e as mensagens às diferentes operações, permitindo assim a utilização dos serviços.

3.2.1.2. Simple Object Access Protocol (SOAP)

O *Simple Object Access Protocol* (SOAP) é o protocolo utilizado para fazer o transporte de mensagens XML sobre o HTTP, aquando da utilização de *web services*. Este protocolo tem um formato XML, para as suas mensagens, e permite fazer o transporte dos diferentes pedidos através dos diferentes serviços. Os pedidos podem ser síncronos, quando um pedido dá lugar a uma resposta, ou pode ser assíncrono, para permitir trocas de dados.

3.2.1.3. Universal Description, Discovery, and Integration (UDDI)

Uma das partes fundamentais da arquitetura SOA é a existência de um mecanismo que permita uma pesquisa dos serviços existentes, através das suas descrições, para que os *web services* possam ser utilizados. Este mecanismo atua na forma de repositório, permitindo a transmissão das diferentes descrições.

Esse mecanismo tem o nome de *Universal Description, Discovery, and Integration*, e permite o registo e transmissão dos diferentes serviços, fornecendo uma *Application Programming Interface* (API) para, de forma simplificada, se conseguir usar o serviço. Este mecanismo permite registar os serviços de forma publica, ou privada.

3.2.2. Web Services de segunda geração

A utilização de *web services* tornou-se cada vez mais comum dentro de empresas e a arquitetura SOA tornou-se a sucessora das plataformas distribuídas. Os *web services* sofreram uma evolução, pois o seu uso massificado apresentava alguns problemas, nomeadamente transacionais, nos processos de negócio, no que diz respeito às políticas segurança, à confiança nas mensagens, e aos tipos de dados a transportar.

A criação de *web services* de segunda geração não implicaram a criação uma nova arquitetura, mas consistiu na adição de pequenas *frameworks* e em alterações à forma como a arquitetura SOA utiliza os *web services*. Estas alterações permitiram às empresas massificar a utilização de web services, aumentando a confiança neles para a partilha de serviços tanto dentro da empresa como para fora dela.

3.3. Application Programming Interfaces

As *Application Programming Interfaces* (APIs) são mecanismos, definidos pelos programadores, que dão a capacidade de conectar diferentes tipos de recursos, como servidores, middlewares ou bases de dados. As APIs fornecem uma forma simples de invocar serviços, obter dados de uma base de dados ou aceder a métodos essenciais para o desenvolvimento de uma aplicação. À semelhança de outras formas de integração, com a utilização de APIs também há uma grande redução na necessidade de replicação de código. Isto não significa que não seja necessário perceber o que a API faz e de que forma trabalha internamente, pois uma API pode conter muitos métodos e é necessário saber exatamente que métodos pretendemos utilizar. Muitas vezes os programadores fornecem a API com pouca documentação e o estudo da API torna-se complexo, quando esta está pouco documentada, mas é necessário à sua correta utilização (Linthicum, 2000). Atualmente existe um grande número de APIs e grande parte das empresas fornecem alguma das suas APIs de forma gratuita. Uma das maiores diferenças entre as APIs gratuitas e as pagas, é precisamente o tipo e o detalhe da documentação que estas fornecem para o seu uso.

No âmbito desta dissertação foram estudadas várias APIs relacionadas, obviamente, com o desenvolvimento da integração de funcionalidades relacionadas com a navegação de pessoas cegas da aplicação CE4Blind na aplicação Descubra.

3.3.1. Web API

Uma Web API, é uma API que é utilizada em contexto *web*. Foi definida tanto para permitir a criação de pedidos HTTP como para suportar as estruturas de resposta das mensagens. Estas Web APIs são tipicamente conhecidas como *web services*. Um *web service* pode visto como uma 'aplicação' que permite acesso a outras aplicações, através da *web*, sendo esta uma definição, obviamente, muita ampla. A definição dada pelo World Wide Web Consortium (W3C), diz que os *web services* são componentes simples que podem ser integrados com aplicações distribuídas, mais complexas (Alonso, Casati, Kuno, & Machiraju, 2004).

A RESTful API é neste momento a API mais utilizada na integração de aplicações móveis. Esta API utiliza a arquitetura REST para efetuar pedidos HTTP e, neste caso, o REST vem a substituir a arquitetura SOA (Pai, 2014) já referida anteriormente na descrição dos *web services* de primeira geração.

A arquitetura REST descreve os princípios de uma arquitetura que transmite dados sobre uma *interface* padrão, como o HTTP no caso do uso de RESTful API (Pai, 2014). Esta arquitetura é descrita utilizando a *Web Application Description Language*, que descreve o que pode ser pedido e enviado pelo serviço, bem como o seu identificador, normalmente denominado *Uniforme Resource Identifier* (URI). Nesta arquitetura os modelos de dados são operados como recursos, sendo identificados com URIs, que são acedidos através de uma *interface* padrão. Este modelo apresenta várias vantagens pois apresenta uma forma standard de o programador compreender melhor como a aplicação funciona e onde os recursos disponíveis podem ser bastantes diferentes uns dos outros. Além disso, através desta arquitetura, cada pedido feito ao serviço é autónomo, nenhuma informação do cliente é guardada no servidor e em cada caso, um pedido não depende do pedido anterior (Belqasmi, Glitho, & Fu, 2011).

De uma forma resumida, pode-se dizer que a RESTful API utiliza a arquitetura REST para efetuar pedidos através dos métodos HTTP, enviando o pedido através do URL dos recursos do servidor e utilizando as operações mais comuns do HTTP: GET, POST, PUT e DELETE (Pai, 2014). Dependendo do pedido feito ao servidor, cada

resposta recebida pelo cliente pode ser diferente e poderá apresentar formatos diferentes, pois a RESTful API suporta diferentes formatos. Os formatos mais comuns são o texto simples, HTML, XML e JSON (Belqasmi et al., 2011). Através desta API torna-se mais fácil a criação e o acesso a um serviço, facilitando a integração de diferentes aplicações.

3.3.2. GraphHopper API

O GraphHopper é uma API gratuita que tem o objetivo de disponibilizar formas de efetuar o cálculo de rotas de navegação entre coordenadas geográficas. Para isso, esta API utiliza os dados fornecidos pelo OpenStreetMaps e utiliza algoritmos como o Dijkstra e A* para efetuar o cálculo do caminho mais curto entre um ponto A e um ponto B, georreferenciados. Esta API pode ser utilizada *online*, recorrendo aos servidores do GraphHopper para efetuar o cálculo das rotas, ou então de forma *offline*, em que os dados do OpenStreetMaps necessitam de ser previamente transferidos para uma forma de armazenamento local, de forma externa à API. Esta API permite o cálculo de rotas rodoviárias, a pé ou de bicicleta.

Para a utilização da API numa aplicação móvel, os programadores do GraphHopper fornecem integração com a Mapsforge, que é uma ferramenta que fornece diferentes tipos de mapas vetoriais e georreferenciados de forma a que seja possível a sua utilização na interface do utilizador, sendo também esses mapas subdivididos de forma a permitir a utilização de uma menor quantidade de dados, não sendo assim necessário um grande nível de processamento (Mapsforge, 2013).

O algoritmo de Dijkstra é usado para calcular o caminho mais curto entre nós de um grafo. A sua complexidade é de O(n2), e para efetuar o cálculo são efetuados os seguintes passos:

1. Criação de dois *sets*: S e U. S será um *set* vazio e U é o *set* que vai conter os nós do grafo com a distância de infinito;
2. Definição do nó inicial a colocar no set S, com a distância de 0, e remoção desse nó do set U;
3. O nó inicial será o nó de procura;

4. A partir do nó do de procura, fazer uma comparação com os nós vizinhos que estão no set U. Se a distância entre o nó de procura e os nós vizinhos for menor que distância entre estes no set U, essa será a sua nova distância;
5. Colocar o nó com a menor distância no set S e remover esse nó do set U;
6. O nó inserido no set S será o novo nó de procura;
7. Repetir os passos 3 até 6, até que o set U fique vazio.

Utilizando este algoritmo é encontrado o caminho mais curto entre dois nós (Yin & Wang, 2010).

O algoritmo A* é um algoritmo bastante semelhante ao algoritmo de Dijkstra, e os passos que este utiliza para fazer o cálculo da melhor rota, são muito idênticos. A maior diferença está na forma como é feita a comparação entre os nós. Na avaliação para saber qual o nó com menor custo não é analisado somente o nó seguinte, mas também o custo estimado do nó seguinte até ao nó final (Zhang & Zhao, 2014).

A GraphHopper API facilita a implementação de um sistema de navegação com a capacidade de efetuar o cálculo de rotas *offline* em dispositivos com baixa capacidade de processamento. Esta API apenas retorna a rota a ser seguida, dividida nos vários segmentos que a constituem. Para fornecer a um utilizador as direções que terá de efetuar pelo caminho sugerido, é necessário efetuar os variados cálculos à parte. Este processo é explicado mais adiante, nesta dissertação, aquando da descrição do desenvolvimento do sistema proposto (GraphHopper, 2018).

3.3.3. OSMdroid API

O OSMdroid é uma API *open source* e, tal como a GraphHopper, tem o objetivo de fazer o cálculo de rotas de navegação entre (e através) de coordenadas geográficas utilizando, para isso, os dados do OpenStreetMaps. A utilização desta API requer a existência de uma ligação à internet. O cálculo das rotas é feito exclusivamente *online*, e esta dependencia do servidor pode inviabilizar a utilização em determinados contextos. Esta dependência de uma ligação constante a um servidor pode levar a que, na falha de ligação, ocorram falhas do lado do sistema que esteja a utilizar a API (Osmdroid, 2018).

3.3.4. Android Beacon Library

A Android Beacon Library é uma biblioteca que disponibiliza APIs que permitem a deteção e utilização de Beacons, que são pequenos dispositivos que fornecem informações a dispositivos moveis através de ondas *Bluetooth Low Energy* (BLE) (Kajioka, Mori, Uchiya, Takumi, & Matsuo, 2014).

O recurso a estas APIs permite que uma aplicação instalada ambiente móvel possa fazer uma pesquisa por *beacons* existentes na proximidade do dispositivo, bem com obter informações sobre o seu identificador e a força do sinal recebido (RadiusNetworks, 2018).

4. ESTRUTURA DO SISTEMA

4.1. Enquadramento

Para integrar funcionalidades do sistema CE4Blind na aplicação Descubra deve ser feito, previamente, um estudo das funcionalidades oferecidas por cada uma destas plataformas, de forma a que se possa discriminar o que deve ser integrado e o que não é aplicável na Descubra. Estas funcionalidades são muitos diversificadas como podemos ver pela Figura 1.

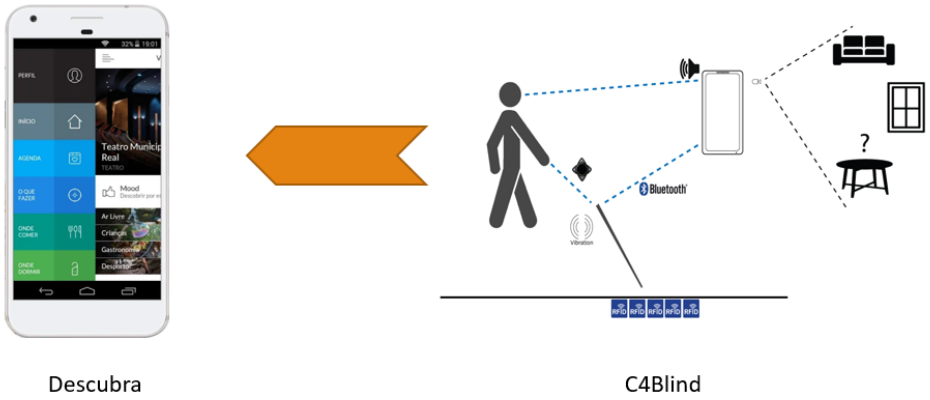

FIGURA 1 - CARATERÍSTICAS DO DESCUBRA E DO CE4BLIND

O projeto CE4Blind oferece uma aplicação móvel que assenta num desenvolvimento modular, sendo o módulo de localização, e os seus submódulos, o mais importante. No caso da aplicação Descubra, devem-se conhecer as informações que esta pretende fornecer e perceber os processos utilizados na formação dos menus e na forma como a informação é apresentada ao utilizador. No que diz respeito à integração das funcionalidades do sistema CE4Blind na aplicação Descubra, é necessário dar foco ao estudo dos módulos mais pertinentes à integração e à forma como pode ser feita a integração na aplicação de destino.

Assim, neste contexto, neste capítulo é feita uma descrição do sistema CE4Blind e da aplicação Descubra. Por fim, é sugerida uma forma de fazer a integração das duas aplicações, e serão tecidas algumas considerações sobre os motivos que levaram à não utilização de alguns dos módulos do CE4Blind, bem como algumas propostas sobre a forma de os substituir.

4.2. CE4Blind

O projeto CE4Blind é um projeto desenvolvido por um grupo de investigadores da Universidade de Trás-os-Montes e Alto Douro com o objetivo geral de ajudar pessoas cegas a navegar e se orientar num determinado local, previamente mapeado. O sistema proposto neste projeto é constituído por um modulo de *interface* com uma bengala eletrónica, um sistema de informação, um módulo de navegação responsável por determinar o caminho ideal para chegar a um destino, e um modulo de localização que contém vários submódulos que recorrem a diferentes tecnologias para efetuar o cálculo de uma estimativa da posição geográfica do utilizador. Além disso, existe também um módulo opcional que recorre a visão por computador para reconhecer determinados elementos visuais, naturais da cena, que ajudam por um lado a enriquecer a informação contextual a ser entregue ao utilizador e, por outro lado, utilizam essa informação para ajudar a localizar melhor o utilizador no espaço.

4.2.1. Módulo de interação com a Bengala Eletrónica

Pode-se dizer, de forma muito genérica, que este modulo é o modulo responsável pela interface com o utilizador, uma vez que, sendo o CE4Blind uma aplicação para cegos, é necessária a existência de uma forma de interface não visual. Atualmente, existem tecnologias capazes de fazer a leitura dos elementos presentes no ecrã de uma aplicação, facilitando a utilização das ferramentas digitais e o acesso à informação por parte das pessoas cegas. Apesar de isto ser uma solução que se adequa à maioria dos casos, no âmbito do projeto CE4Blind foi desenvolvida uma bengala eletrónica com caraterísticas semelhantes às bengalas brancas tradicionalmente utilizadas pelos cegos, mas adicionando a capacidade de interagir com um smartphone. A conexão entre estes dois dispositivos é feita através de Bluetooth e esta solução torna-se bastante apropriada pois, mantendo caraterísticas semelhantes às bengalas brancas utilizadas pelos cegos, é mais fácil a sua adaptação à utilização.

4.2.2. Bengala Eletrónica

Como referido anteriormente, a bengala eletrónica é uma bengala com caraterísticas semelhantes à bengala branca utilizada tradicionalmente pelos cegos, mas com capacidade de comunicar com um smartphone através de Bluetooth. Esta bengala contém, numa das suas extremidades, um punho instrumentado eletronicamente com um *joystick* e, na outra extremidade, uma antena leitora de etiquetas (ou *tags*, em inglês) RFID. É através deste joystick que o cego tem a possibilidade de interagir com a aplicação CE4Blind e cada tecla do *joystick* envia uma mensagem diferente para o smartphone, para que depois sejam ativadas as rotinas necessárias, consoante o que o utilizador pretender fazer. O leitor de etiquetas RFID, presenta na ponta da bengala, consegue ler o indentificador da etiqueta lida, colada ao pavimento, e enviar para o smartphone esse identificador por via Bluetooth. Este identificador é processado pela aplicação e as *tags* são assim utilizadas como fontes de localização. Mais adiante, neste documento, este processo é explicado com mais detalhe.

4.2.2.1. Modo de interação

A cada tecla do *joystick* é associada uma mensagem (Tabela 2), e o processamento de intenção associada a cada mensagem é feito no smartphone. No caso da utilização prevista no sistema no CE4Blind, estes movimentos fazem o deslocamento sobre o menu da aplicação: clicando para cima ou baixo são percorridos os itens do menu; clicando para a direita é validada a opção selecionada, entrando num subnível do menu e, clicando para a esquerda o utilizador sai do subnível selecionado, voltando ao menu anterior; clicando no botão do centro do *joystick*, o utilizador ouve uma descrição dos elementos mapeados do espaço geográfico à sua volta após uma verificação da existência de quais os pontos de interesse (POI) num raio de *x* quilómetros da localização atual do utilizador.

TABELA 2- AÇÕES DA BENGALA ELETRÓNICA.

AÇÃO	MENSAGEM RECEBIDA
Tag lida	*TAGxxxxxxxxxxx#
Joytick para cima	*BC#
Joytick para baixo	*BB#
Joytick para esquerda	*BE#
Joytick para direita	*BD#
Joytick premido no centro	*BM#

Todas as opções do menu são transmitidas ao utilizador através de tecnologia de texto-para-voz. Esta forma de interação é semelhante à da tecnologia para a leitura de ecrã, com a diferença de que neste caso, o joystick é mais percetível ao toque que o ecrã de um smartphone, ou seja, para um cego é mais fácil 'ver'/sentir o que esta a fazer, visto que com a perda da visão a única forma de um cego conseguir 'ver' é através da utilização dos outros sentidos, nomeadamente o tato e a audição.

4.2.3. Módulo de Dados

A necessidade de mapeamento dos locais é bastante importante para o funcionamento desta solução e a necessidade de mapear cada caminho e cada ponto de interesse é algo de extrema importância pois é com base nesta informação que se poderá avisar o cego acerca do caminho certo a seguir e os obstáculos existentes em cada local. É com base na qualidade do mapeamento que a aplicação se torna útil, ou não. Num cenário extremo, é a qualidade da informação que define a adesão ao sistema, e não a qualidade da ferramenta em si. Existem soluções que fornecem algum mapeamento de muitos locais de forma externa à aplicação, como por exemplo o OpenStreetMaps ou o Google Maps, mas neste caso o mapeamento é feito para pessoas com capacidades normais de visão e sem restrições à mobilidade. A necessidade de adicionar pontos de interesse relevantes à navegação de cegos, tais como como passadeiras ou obstáculos, levantou a necessidade da criação de um sistema de informação geográfico (GIS)

proprietário. O sistema CE4Blind tem o seu próprio GIS, facilitado o mapeamento de locais em ambiente de exterior e interior, adicionando pontos de interesse apropriados aos cegos, tais como os diferentes obstáculos existentes num ambiente e o caminho que interliga os vários POIs. Este GIS guarda todos os dados numa base de dados armazenada num servidor externo à aplicação CE4Blind. A aplicação faz download dos mapas do servidor recorrendo a Web Services, sincronizando os dados recebidos do servidor com a base de dados interna da aplicação no dispositivo móvel.

É através dos dados sincronizados com o GIS que a aplicação consegue saber os pontos de interesse em redor do utilizador e calcular o caminho que o utilizador tem de seguir no caso de pretender deslocar-se para certo ponto de interesse. Esse cálculo da rota a seguir é feito com recurso ao algoritmo A*, para encontrar o caminho mais curto.

4.2.4. Módulo de Localização

O sistema CE4Blind utiliza o seu módulo de localização para efetuar o cálculo da estimativa da localização do utilizador. Isto é feito com recurso a quatro tecnologias diferentes: o sistema de posicionamento global GPS, triangulação Wi-Fi, visão por computador e RFID, combinando as quatro tecnologias, associadas respetivamente a quatro submódulos, para obter uma localização mais precisa. Esta necessidade de precisão é essencial pois a localização é utilizada para a orientação de cegos, havendo a necessidade de os avisar acerca do aparecimento de um obstáculo com a maior precisão possível e antecipação possível. O utilizador não tem a possibilidade de ver naturalmente estes elementos e o sistema pode, inadvertidamente, provocar situações de perigo pela falta de precisão. Nos próximos subcapítulos irão ser explicados os diferentes módulos com mais detalhe, bem como a sua forma de funcionamento.

4.2.4.1. GPS

Um dos módulos de localização utilizado pelo CE4Blind recorre ao “Global Positioning System” (GPS), e a implementação deste módulo é feita com a implementação de baixo nível do sistema operativo Android. Este sistema operativo fornece uma localização do utilizador ao sistema sempre que se verifique uma alteração na localização do dispositivo móvel ou sempre que pedida pela aplicação (Fernandes, 2017).

O GPS recorre ao sinal enviado por uma constelação de satélites para efetuar o cálculo da localização de um utilizador com recurso aos princípios matemáticos da trilateração, determinando a posição do utilizador através da distância a que este se encontra de cada um dos satélites. Para cada cálculo, este sistema prevê a utilização de pelo menos quatro satélites: três para estimar a localização do dispositivo e um quarto satélite para validação. Apesar de ao longo do tempo a utilização deste sistema ser cada vez mais comum nos dispositivos móveis de utilização comercial, este sistema ainda apresenta erros de precisão na ordem dos 3 a 5 metros no que diz respeito à localização exata do utilizador, e este erro varia, pois depende das condições climatéricas e da infraestrutura física onde o dispositivo móvel se encontra. Outra fonte de ruido neste sistema são as interferências eletromagnéticas. Apesar de haver ferramentas matemáticas que ajudam a atenuar a amplitude destes erros, o maior problema deste sistema é a impossibilidade da sua utilização dentro de um edifício, uma vez que dentro dos edifícios o sinal de GPS degrada-se fortemente devido a densidade das paredes (Hofmann-Wellenhof, Lichtenegger, & Collins, 2012). Esta falta de precisão pode ser pouco significativa para uma pessoa normovisual, mas para uma pessoa cega é bastante importante pois pode impossibilitar a distinção entre situações em que o utilizador está de um lado da estrada ou do outro, ou até mesmo no meio da estrada. A falta de precisão no sistema pode implicar que não se avise atempadamente acerca de um obstáculo, colocando a pessoa em perigo. Dada a impossibilidade de utilizar este sistema para calcular a localização no interior dos edifícios o CE4Blind utiliza-a apenas para o cálculo da localização do utilizador no exterior (Fernandes, 2017).

4.2.4.2. Triangulação Wi-Fi

Outro módulo de localização do CE4Blind utiliza a triangulação Wi-Fi, e neste caso é necessário ter armazenada na base de dados a localização geográfica dos diferentes pontos de acesso (*Access Points*). Com esta informação é possível utilizador um algoritmo de triangulação para, de forma semelhante ao uso de GPS, determinar uma estimativa da localização do utilizador.

A triangulação de sinal Wi-Fi assenta na utilização de três, ou mais, *Access Points* para fazer o cálculo estimado a posição do utilizador. Para isso utiliza as medidas RSSI dos pontos de acesso, que representam as medidas intensidade do sinal que o dispositivo recebe dos *Access Points*. e com estas medidas estima a distância entre o dispositivo e os

vários *Access Points*. Essa distância representa o raio de um círculo que irá ser feito com centro nos diferentes *Access Points*. A intersecção desses círculos determina a posição do utilizador. O uso de RSSI é problemático e não é muito preciso pois as ondas de rádio são influenciadas pelos diferentes obstáculos existentes, e o cálculo da distância através do RSSI não é exata. Por vezes um *Access Point* pode estar mais perto que outro mas devido à existência de uma maior quantidade de obstáculos o RSSI recebido pode ser menor, fazendo com que a sua distância ao dispositivo aparente ser maior do que a distancia real. Isto provoca erros na precisão da localização do utilizador, originando por vezes erros superiores a 10 metros (Kim, Bong, & Kim, 2011).

4.2.4.3. Visão por Computador

Outro módulo de localização implementado pelo sistema CE4Blind é um módulo que recorre a visão computador. Este módulo foi implementado pelos investigadores do projeto CE4Blind, na UTAD, aquando de um projeto anterior, o projeto Blavigator.

O Blavigator previa o mapeamento de um local recorrendo à utilização de visão por computador, e armazenando numa base de dados informações de certo objetos. Após o mapeamento do local, recorrendo a algoritmos de visão por computador é feita a deteção destes objetos vistos pela câmara e guardados na base de dados, ajudando a localizar geograficamente o utilizador (H. Fernandes, Costa, Paredes, Filipe, & Barroso, 2014). O CE4Blind faz a utilização desta abordagem com ajuda da tecnologia Tango para mapear o espaço. Tendo um mapa visual do espaço e uma indicação do ponto geográfico inicial do mapeamento, é possível georreferenciar pessoas e objetos. Este modulo, quando implementado num ambiente bem estruturado é bastante preciso, podendo apresentar erros de apenas de centímetros no cálculo da localização do utilizador, em condições ótimas (Fernandes, 2017).

Este módulo é um dos módulos mais importantes no CE4Blind visto que este diminui os erros ao centímetro, sendo o ideal para os cegos terem um sistema em que confiem para não corram perigos desnecessários. Com este módulo é possível obter uma localização precisa, mas tem a desvantagem de que, quando a câmara é obstruída, ou existem grandes alterações no local, o sistema não se conseguir localizar.

4.2.4.4. RFID

Além de todas as tecnologias descritas anteriormente, o sistema CE4Blind também utiliza tags RFID para localizar um utilizador. Estas tags são pequenos microchips capazes de transmitir dados que podem ser lidos por uma antena RFID (Juels, 2006). Estes microchips contêm um identificador único e é através deste indentificador que o CE4Blind consegue determinar a localização do utilizador. Para isso, os dados dos identificadores presentes nas etiquetas RFID encontram-se guardados no sistema de informação do CE4Blind, armazenando uma relação entre esses dados e a localização das etiquetas a que correspondem. Este módulo é o mais preciso de todos pois a leitura de tags RFID é feita quando o leitor se encontra fisicamente sobre a tag. Como já foi referido, o leitor de tags RFID está colocado numa das extremidades da bengala e o utilizador irá estar fisicamente na localização da tag lida, quando esta for detetada (Fernandes, 2017).

Apesar de precisão deste modulo, e do preço reduzido de cada etiqueta, é impensável a cobertura de tags RFID pelo espaço devido, entre outros motivos, aos custos de instalação e de manutenção. Assim, esta tecnologia é somente utilizada em sítios críticos, como por exemplo escadas ou passadeiras, para assim ser possível alertar acerca de um perigo existente com a maior precisão.

4.2.4.5. Considerações Finais

Com a utilização dos diferentes módulos de localização disponíveis, o CE4Blind consegue fazer um cálculo da estimativa da localização do utilizador de uma forma precisa, conseguindo, por sua vez, fornecer ao utilizador um sistema confiável para que corra o mínimo de perigo possível nas suas deslocações Estes módulos geridos e interligados são ligados uns com os outros através de um filtro. A utilização de um filtro para este propósito permite fazer a junção de todos os dados de todas as localizações fornecidas pelas diferentes tecnologias, e colocar medidas de confiança nelas (Julier & Uhlmann, 1997). Note-se que quando um destes módulos falha é possível a utilização de outros, conseguindo que o sistema seja mais robusto e disponível.

4.3. Descubra

A Descubra é uma aplicação de turismo desenvolvida pela a empresa WorldIT, que fornece a um utilizador informações variadas relativamente às atividades, valências, infraestruturas e notas históricas de um certo município de uma forma em que um utilizador consiga, de forma independente, fazer visitas à cidade sem necessidade de um guia turístico humano.

Esta aplicação fornece a agenda do município com os próximos eventos a decorrer, diferentes pontos de interesse, tais como restaurantes, onde dormir, entre outros, e a localização geográfica dos locais juntamente com a possibilidade de pedir direções para os destinos.

Em 2017 a Descubra ganhou um prémio de melhor aplicação de turismo a nível nacional, em Portugal.

4.4. Considerações Finais

Após esta exposição das funcionalidades estudadas, relativamente ao CE4Blind e à Descubra segue-se uma definição da proposta de estratégia de integração, bem como de quais as funcionalidades que deverão ser integradas.

Numa fase inicial deve ser integrado o módulo de interação com a bengala. Este módulo funciona da mesma forma que foi implementado pelo sistema CE4Blind e a diferença estará na forma de percorrer o menu. Ainda assim, a bengala irá movimentar-se pelo menu da Descubra, com todas a informações que este fornece, de forma a que seja possível uma pessoa cega utilizar a aplicação com recurso à bengala eletrónica sem se preocupar com o conteúdo presente no ecrã do smartphone.

Numa fase seguinte deverá ser integrado o módulo de localização. Este módulo deverá conter implementações de alguns dos diferentes módulos de localização do sistema CE4Blind. Os módulos utilizados serão o modulo de GPS e o modulo de RFID. O módulo de visão por computador, apesar da sua precisão e de ser um dos módulos mais importantes, não será integrado devido a ser bastante intrusivo para o utilizador (requer a

colocação de um dispositivo com a capacidade de utilização do Tango sobre o peito), e o modulo de triangulação Wi-Fi também não será utilizado devido ao grande erro associado. No entanto, este último será substituído por outro submódulo de localização, descrito mais adiante neste documento.

Em relação ao modelo de informação geográfica, os pontos de interesse a ser utilizados pela Descubra serão os já existentes na aplicação. Como foi referido anteriormente, o CE4Blind utiliza um GIS proprietário para efetuar o cálculo do caminho que o utilizador deve seguir. Nesta integração, de forma similar, será utilizado o GIS do OpenStreetMaps, com a introdução de algumas alterações para tornar o caminho mais seguro para os cegos.

5. PROPOSTA DO SISTEMA

5.1. Enquadramento

Após a análise da arquitetura e modo de funcionamento de cada componente de cada um dos sistemas, surge a necessidade de fazer um levantamento dos requisitos funcionais do produto que resultará da integração que se pretende fazer. Esta análise de requisitos resulta numa definição do que será integrado ou implementado pois, sendo a Descubra uma aplicação já existente no mercado, é necessário que se verifique e planeie com especial atenção se todos os módulos envolvidos poderão ser integrados, e se a integração mantém o propósito da aplicação desenvolvida pela WorldIT.

5.2. Levantamento de requisitos

No âmbito do levantamento de requisitos, houve intercâmbio de opiniões com a empresa que desenvolve a aplicação Descubra no sentido de validar quais funcionalidades a integrar, com base nas necessidades discutidas. A Tabela 3 enumera os diferentes módulos propostos que foram aceites para integração, bem como a razão pelo qual os restantes foram rejeitados.

TABELA 3 - MÓDULOS PROPOSTOS PARA INTEGRAÇÃO.

MÓDULO	ACEITE/REJEITADO	MOTIVO
Bengala Eletrónica	Aceite	-
Navegação GPS	Aceite	-
Triangulação Wi-Fi	Rejeitado	Pouca precisão
Visão por Computador	Rejeitado	Intrusivo
RFID	Aceite	-

SIG	Aceite	-

Dos vários módulos apresentados e representados na Figura 2, foram rejeitados vários módulos responsáveis pelo cálculo da localização do utilizador em ambiente interior, sendo que, o único módulo aceite para isso, foi o RFID. No entanto, prevê-se que, tal como no caso do projeto CE4Blind, possam ser desenvolvidos ou integrados outros módulos que ajudem a melhorar a precisão da localização do utilizador.

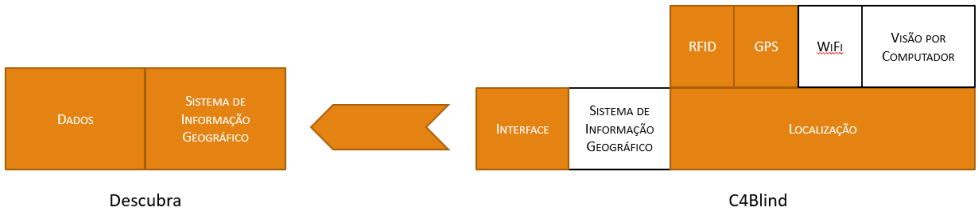

FIGURA 2 - MÓDULOS DO DESCUBRA E DO CE4BLIND

Em relação ao modelo de dados, apesar de este ter sido aceite a sua inclusão, foi decidido, em conjunto com a WorldIT, que o GIS a utilizar seria baseado na plataforma OpenStreetMaps, que fornece toda a informação necessária ao mapeamento dos locais a georreferenciar no âmbito da aplicação Descubra.

5.3. Proposta de desenvolvimento dos protótipos

Após a definição dos módulos a integrar, verificou-se que o sistema CE4Blind, muito pelo facto de ser ainda um protótipo, baseia-se fortemente em software proprietário, e não disponibiliza de uma forma simples, através APIs ou *frameworks*, formas de fazer uma integração dos seus módulos, nomeadamente com a bengala e com o módulo de navegação. Foi, então, decidido que seria utilizado o conhecimento produzido no âmbito desse projeto para reproduzir a implementação destes módulos, na Descubra. Foi proposto, assim, que numa fase inicial se iriam desenvolver protótipos para os diferentes

módulos e, somente depois de estarem funcionais, serão integrados com a aplicação Descubra.

Assim, foi proposto efetuar o desenvolvimento e integração de tecnologia/conhecimento relacionados com a inclusão: da bengala eletrónica, de formas de localização em ambientes de interior e exterior, e de um módulo de navegação. Sendo, a aplicação Descubra, disponibilizada tanto para o sistema operativo iOS como para o Android, numa fase inicial foi tomada a opção de fazer o desenvolvimento da integração do CE4Blind somente para Android, uma vez que o sistema CE4Blind seguiu a mesma abordagem inicial, existindo alguma documentação de suporte.

5.3.1. Protótipo da interação com a Bengala Eletrónica

A proposta para o protótipo de interação com a bengala eletrónica, de acordo com a Figura 3, define a utilização do Bluetooth para fazer a ligação entre a bengala e o dispositivo móvel. Para isso, será utilizada uma biblioteca, disponibilizada pelo Android, com a capacidade de encontrar, emparelhar e conectar um dispositivo Bluetooth a um smartphone.

FIGURA 3 - FLUXOGRAMA DA INTERAÇÃO COM A BENGALA ELETRÓNICA.

Após a conclusão dessa fase, será implementado o recetor das mensagens Bluetooth, como a interpretação delas, consoante os vários comandos constantes da especificação técnica da bengala. Estes comandos estarão na base da interação com o menu que irá ser utilizado pelo portador da bengala, e serão interpretados de forma a que a aplicação entregue ao utilizador as várias informações disponibilizadas pela Descubra.

5.3.2. Protótipo de localização em ambientes de interior

Sendo que dos módulos do CE4Blind responsáveis pela obtenção da localização em ambiente apenas se definiu a inclusão do modulo de localização através do RFID, é necessária a utilização de outras tecnologias para complementar a disponibilidade apenas ocasional do RFID. Assim, para aumentar a robustez deste protótipo, decidiu-se a integração da utilização de *Beacons* para a complementação do RFID.

Um *beacon* é um dispositivo de pequenas dimensões com a capacidade de emitir um sinal Bluetooth, semelhante ao sinal emitido pelos *Access Point*s de Wi-Fi. Comparativamente aos *Access Points*, um *beacon* consome muito menos energia e é alimentado apenas por uma pequena bateria com a capacidade de o alimentar durante alguns meses. Os *beacons* têm um identificador que pode ser alterado pelo instalador e têm, também, a potencialidade de armazenar e emitir alguns dados, como por exemplo URLs. Espera-se que, num futuro muito próximo, estejam presentes em muitos edifícios.

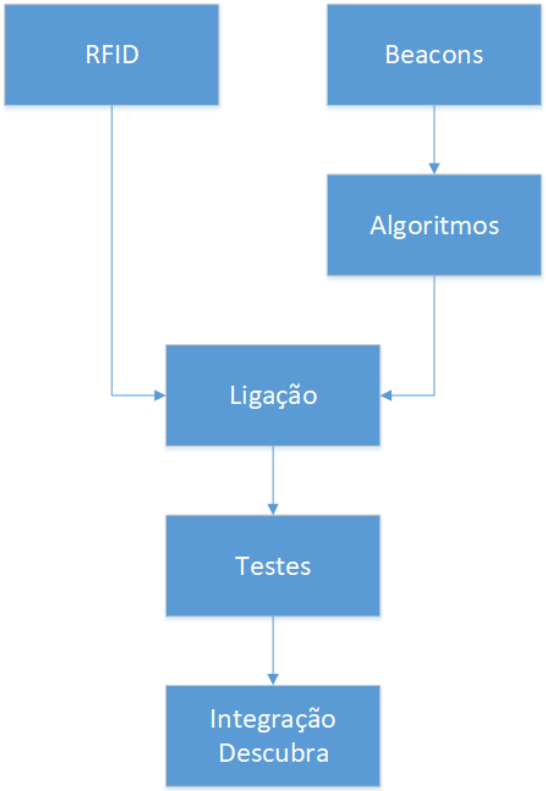

FIGURA 4 - PROPOSTA PARA O PROTÓTIPO DE NAVEGAÇÃO EM AMBIENTES DE INTERIOR.

A proposta definida para o desenvolvimento do protótipo de navegação em ambientes de interior, de acordo com a Figura 4, tem por base, numa fase inicial, a utilização simultânea de informação armazenada nos *beacons* e da tecnologia RFID presente na bengala para obter a localização do utilizador.

Numa fase seguinte, prevê-se o desenvolvimento de algoritmos que permitam calcular a localização do utilizador através dos dados fornecidos pelos Beacons, nomeadamente da potência do seu sinal. Deve-se também ter atenção, neste contexto, ao facto de que ao serem obtidos dados de localização com base em duas fontes distintas, deve-se ter o cuidado de filtrar (ou fundir) estes dados, consoante a precisão associada a cada uma.

No final desta fase, é necessária a realização de testes de forma a verificar se este protótipo de localização apresenta imprecisões no cálculo da localização do utilizador de forma a que se possa proceder a melhoramentos e à resolução de problemas. A realização deste tipo de testes relacionados com a fidelidade da localização estimada é muito importante antes de se proceder à integração com a aplicação Descubra.

5.3.3. Protótipo de localização em ambientes de exterior

A proposta para o protótipo navegação em ambientes de exterior, de acordo com a Figura 5, assenta na utilização do sinal de GPS para o cálculo da localização. Para isso, será utilizada a biblioteca disponibilizada pelo Android.

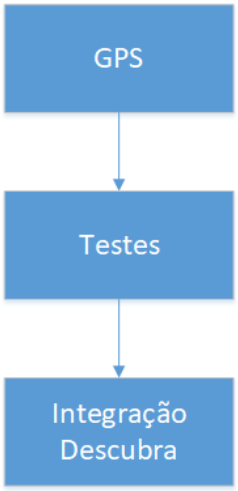

FIGURA 5 - PROPOSTA PARA O PROTÓTIPO DE NAVEGAÇÃO EM AMBIENTES DE EXTERIOR.

Tal como no módulo de localização em ambiente interior, deve ser feita a realização de testes para verificar a necessidade de reduzir o erro associado à utilização de GPS, de forma a garantir a melhor precisão possível no cálculo da localização do utilizador.

5.3.4. Protótipo do módulo de navegação

Como já foi referido, no sistema CE4Blind a definição da melhor rota a seguir para um determinado destino baseia-se na escolha do caminho mais curto. No decorrer do processo de navegação, o utilizador recebe informação armazenada no sistema de informação geográfico da aplicação. No caso do desenvolvimento deste módulo, e no âmbito da aplicação Descubra, foi decidida, conjuntamente com a empresa WorldIT, a utilização de dados armazenados na plataforma OpenStreetMaps para efetuar o cálculo da melhor rota para a navegação. Estes dados terão de ser verificados antes de ser utilizados, pois poderão existir zonas onde pode ser perigosa a passagem de pessoas cegas. Assim, sobre os dados já presentes no OpenStreetMaps, será criada uma camada de dados destinada à navegação de pessoas cegas, onde serão identificados locais

perigosos de forma a que o cálculo do caminho, além de mais curto, seja também o mais seguro possível. Assume-se que a fiabilidade e a confiança gerada num utilizador definem a massificação do uso da tecnologia.

Este protótipo será implementado durante a implementação do protótipo do módulo de localização do utilizador para possibilitar a realização de testes, e de forma a ser possível a verificação da garantia de segurança para o utilizador, que o sistema irá apresentar.

5.4. Considerações finais

Após a implementação de todos os protótipos dos módulos definidos neste capítulo, será necessário fazer a sua integração de forma a que seja possível combinar a utilização da bengala na orientação e navegação em ambiente interior e exterior. Por fim, deverá ser feita a integração dos dados geográficos, e, por sua vez, semânticos, fornecidos pela Descubra de forma a que seja possível ao utilizador utilizar a aplicação de turismo de forma independente e acessível, permitindo que este chegue ao destino desejado da forma mais segura possível.

6. DESENVOLVIMENTO DO SISTEMA

6.1. Enquadramento

De acordo com as especificações definidas na proposta de desenvolvimento do sistema, numa fase inicial da integração das novas funcionalidades na aplicação Descubra, foi previsto o desenvolvimento de protótipos para os diferentes módulos de acordo com as especificações do sistema CE4Blind. Neste capítulo serão descritos os processos de desenvolvimento dos diferentes protótipos, bem como as tecnologias utilizadas, os problemas encontrados e a sua resolução.

6.2. Interação com a Bengala Eletrónica

O protótipo de interação com a bengala eletrónica foi dividido em três componentes ou partes fundamentais: a ligação Bluetooth, o desenvolvimento de um menu capaz de ser adaptável a uma fonte de dados apropriada, ou seja, um 'menu dinâmico' e, por fim, uma componente de interface com os restantes elementos da aplicação Descubra.

6.2.1. Bluetooth

Antes de ser feito qualquer pedido à aplicação através da bengala, é necessário que esta esteja conectada com o smartphone via Bluetooth. Para isso foi utilizada a biblioteca disponibilizada pelo Android e que nos permite efetuar essa ligação. Antes de ser efetuada uma conexão de dados através da bengala é necessário que a fonte de dados (neste caso, a bengala) esteja emparelhada com o smartphone. No que diz respeito ao desenvolvimento de software para Android, é utilizada uma *Activity* [1]que nos permite procurar os dispositivos Bluetooth mais próximos, podendo ser selecionado manualmente o dispositivo ao qual nos queremos ligar. Em seguida, através de código disponibilizado

[1] Activity - https://developer.android.com/reference/android/app/Activity (visitado em Outubro 2018)

pela biblioteca, é verificada a existência ou não de emparelhamento com a bengala com vista à sua posterior utilização na aplicação.

Quando é feito um emparelhamento de um periférico via Bluetooth, são guardadas no *smartphone* informações sobre o dispositivo emparelhado, facilitando posteriores conexões. Um emparelhamento necessita de um código de emparelhamento que, por defeito, costuma ser '0000' ou '1234'. Este código é fornecido pelo fabricante do periférico e pode ser diferente para cada dispositivo, apesar de isso não ser uma prática corrente. Quando emparelhada, é possível estabelecer uma conexão de dados com a bengala. A implementação dessa conexão é feita programaticamente através de um *Service* que corre em background comunicando com a aplicação através de um *handler*. Um *handler* é uma classe que permite a troca de mensagens, neste caso entre um Service e uma Activity. Numa conexão é preciso identificar o dispositivo emparelhado e isto é feito através de um Universally Unique Identifier (UUID), que é um valor de 128 bits que identifica o tipo de dispositivo que se pretende conectar (Android, 2018b). No caso específico da bengala eletrónica, o seu interface Bluetooth tem por defeito um UUID com o valor de 00001101-0000-1000-8000-00805F9B34FB.

Após ser estabelecida a conexão da bengala com o smartphone, a biblioteca disponibilizada pelo Android permite que os dispositivos comuniquem entre si através de um *socket*. No desenvolvimento do protótipo foi implementada uma comunicação síncrona entre a bengala e o smartphone. Foi, depois, implementado um método de receção de mensagens que contém os comandos da bengala. Estas mensagens são recebidas em bytes e colocadas num *buffer*, sendo, por sua vez, entregues ao *handler*, como referido anteriormente. Foram também implementados alguns métodos para gerir situações como, por exemplo, a da impossibilidade de a bengala se conectar ao smartphone e também de os dois dispositivos perderem a conexão entre si. Os vários métodos desenvolvidos são enumerados na Tabela 4.

TABELA 4 - ESTADOS DA CONEXÃO DA BENGALA ELETRÓNICA.

SITUAÇÃO	**MENSAGEM GERADA**
Impossível conectar	Falha ao ligar a bengala
Conexão perdida	Conexão perdida

Após a implementação do método para gerir a receção das mensagens enviadas pela bengala foram efetuados alguns testes para verificar se as mensagens recebidas pelo smartphone eram corretas ou não, de acordo com cada situação prevista. Durante os vários testes foi verificado que eram recebidas várias mensagens ao mesmo tempo, e algumas delas não eram as mensagens que se esperava receber. O motivo dessa receção incorreta deve-se ao facto de que o Bluetooth recebe as mensagens por partes, sendo que, correspondentemente, estas estavam a ser enviadas para o *handler* também por partes. Para resolver este problema foi-se adicionando ao buffer todos os bytes recebidos até à receção do conjunto de bytes associado ao caracter "#". Uma vez que todas as mensagens enviadas pela bengala terminam "#", esta implementação garante que cada mensagem proveniente do buffer contém um comando completo da bengala.

Com a resolução dos problemas encontrados, após a conexão com a bengala é possível receber as mensagens de uma forma correta e fiável e que permite utilizar os comandos provenientes da bengala para a percorrer um menu. Numa fase seguinte, cada comando recebido terá de ser processado para a navegação num menu dinâmico.

6.2.2. Menu dinâmico

Numa aplicação, a um menu cabe a tarefa de apresentar todas as opções disponíveis e relacionadas com um determinado contexto, de forma a facilitar ao utilizador o acesso às diferentes informações, operações ou intenções que se relacionem com esse contexto. Um menu pode conter opções que levem o utilizador diretamente à informação desejada, ou pode, por sua vez, dar acesso a submenus que contêm opções mais refinadas, dentro de um chapéu maior. Em termos visuais, essa representação é simples pois um utilizador sem qualquer deficiência visual consegue ver todas as opções e selecionar facilmente a que desejar. No caso das pessoas cegas, apesar de ser poder utilizar o mesmo princípio, no sentido de lhe serem disponibilizadas as diferentes opções de um menu, é necessário providenciar formas alternativas de este lhe ser apresentado.

O princípio a ser utilizado para a criação do menu da aplicação é mesmo princípio utilizado para o Talkback do Android. O Talkback é uma ferramenta que o sistema

operativo Android fornece e que permite, através de tecnologia de conversão de texto para voz, receber *feedback* áudio dos conteúdos apresentados em todos os ecrãs do dispositivo, fornecendo assim formas de utilizar o dispositivo móvel sem a necessidade de olhar para o ecrã (Android, 2018a). Analogamente, o menu percorrido pela bangala fornece ao utilizador uma forma de este se deslocar pelas as diferentes opções, podendo selecionar a(s) que pretender sem a necessidade de olhar para o dispositivo, e também sem a necessidade de ter que o segurar fisicamente, pois este é acedido através da bengala. Para este propósito, foi criada uma interface de software para o menu que será implementada através do recurso a duas classes diferentes: um *menu* e um *menu entry* (Figura 6).

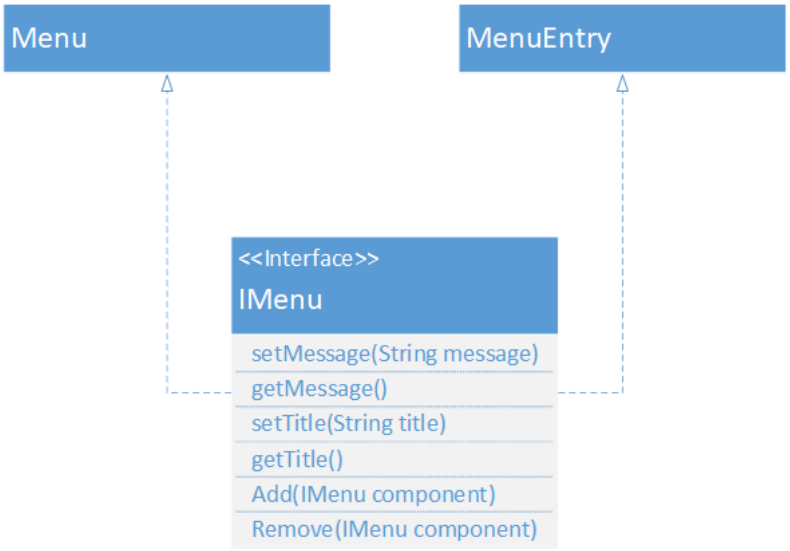

FIGURA 6 - INTERFACE COM O MENU.

A diferença entre as duas classes está no tipo de dados que cada uma representa. Um *menu* é composto por várias opções que podem ser selecionadas. Por sua vez, um *menu entry* é a opção final selecionada, independentemente do seu nível dentro do menu, selecionando as informações ou operações a que o utilizador pretende aceder. A Figura 7 representa um exemplo de um menu dinâmico.

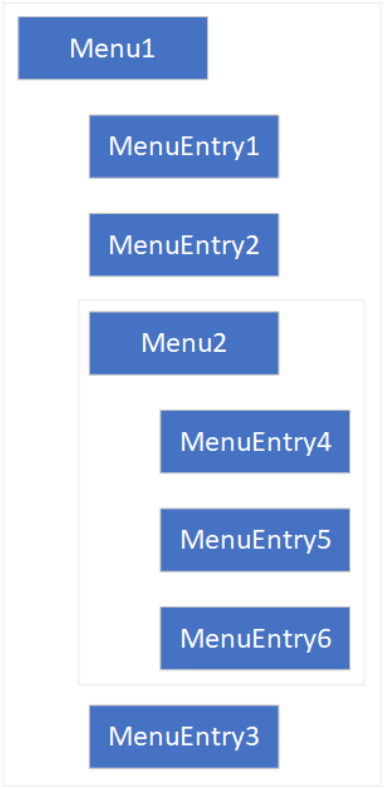

FIGURA 7- EXEMPLO DA ESTRUTURA UTILIZADA PARA O MENU.

A denominação atribuída a este tipo de menu (‘menu dinâmico’) deve-se ao facto de este apresentar a capacidade de aumentar ou diminuir o seu tamanho consoante a quantidade informação, ou opções, que nele pretendemos incluir, podendo, assim, ser configurado em cada instante com base em informação fornecida pela Descubra.

Após a implementação deste tipo de do menu, é feito o processamento das mensagens recebidas pela bengala para assim ser possível o movimento pelo menu e, através dos comandos recebidos, selecionar intencionalmente cada ação presente no menu. De forma resumida, para se movimentar no menu são utilizadas: as teclas do joystick de direção cima e baixo para percorrer verticalmente as opções do menu; as teclas direita para ativar a opção selecionada; a tecla esquerda para voltar para opção anterior; e a tecla central para ‘ver’ informações sobre das diferentes opções. Durante a movimentação pelo menu, todas as opções e informações são transmitidas ao utilizador através de *text-to-speech*, tal como no Talkback.

6.2.3. Integração com a aplicação Descubra

Como foi referido, a utilização de um menu dinâmico permite adicionar ou remover informações consoante o que se pretende apresentar ao utilizador. O menu existente na aplicação Descubra, e todas as suas informações associadas, é recebido através de Web Services que acedem à base de dados da plataforma. Com base nessa estrutura de integração, torna-se simples criar e adaptar um menu áudio com a mesma informação.

No caso dos dados já disponibilizados pela plataforma Descubra através de um Web Service, o título das diferentes opções da bengala faz correspondência direta ao título apresentado na aplicação Descubra, no seu menu e nos seu submenus, e a mensagem associada a cada item, depende do tipo de menu. No caso de um objeto *menu*, a mensagem que descreve o menu será o título do menu, tal como é apresentado na aplicação Descubra. No caso de um *menu entry*, o título será o título da opção apresentada, e a mensagem contém a informação que essa opção fornece. A Figura 8 mostra a forma como o menu dinâmico é criado através da informação disponibilizada na aplicação Descubra.

FIGURA 8 - EXEMPLO DE UM MENU CRIADO ATRAVÉS DA INFORMAÇÃO DISPONIBILIZADA PELA DESCUBRA.

6.3. Localização em ambientes de interior

Como referido antes, neste documento, o módulo do sistema CE4Blind a ser implementado para estimar a localização do utilizador em ambientes de interior foi somente o módulo de localização através de tecnologia RFID. Foi, também, já referido que este, sozinho, não é suficiente para garantir uma boa cobertura geográfica e a tecnologia escolhida para complementar o RFID foi a tecnologia de localização baseada em *Beacons*.

Neste subcapítulo é explicado o desenvolvimento do protótipo de localização em ambientes de interior, começando pela localização através do recurso a RFID e, posteriormente, a localização através dos Beacons. Por fim é exposta a forma de integração com a Descubra.

6.3.1. Localização através da tecnologia RFID

Colocar tags RFID numa tentativa de cobrir grandes áreas geográficas é uma tarefa árdua, com elevados custos de manutenção e com dificuldades inerentes à necessidade ocasional de detetar e substituir etiquetas avariadas. Tendo estes fatores em consideração, as tags RFID irão ser colocadas da mesma forma que estas são colocadas na especificação do CE4Blind, isto é, apenas em sítios de perigo iminente para o utilizador, como por exemplo escadas, elevadores, etc. São também colocadas em certos pontos de interesse turístico, como por exemplo monumentos ou quadros em museus, sendo a localização obtida utilizada para dar ao utilizador a informação adequada.

Para a utilização desta tecnologia são utilizadas as funcionalidades da bengala eletrónica. A bengala contém, na sua ponta, uma antena RFID capaz de detetar as *tags* quando é colocada sobre estas, enviando, através de Bluetooth, a informação contida na *tag*. No caso das etiquetas RFID do tipo passivo, utilizadas pelo sistema CE4Blind, as *tags* RFID fornecem somente um ID, definido pelo fabricante. A bengala envia para o smartphone uma mensagem com o formato, *TAGxxxxxxxxxxx#. Nesta forma de implementação, o ID de cada *tag* deve ser associado a uma localização especifica, sendo esta posteriormente utilizada pela aplicação para determinar se essa localização tem alguma informação associada.

Com o grande, e crescente, número de tags RFID necessárias e dos pontos de interesse associados, foram incluídas formas de alterar os dados na base de dados presente no servidor de forma a que não haja a necessidade de alterar o código da aplicação. Para isso foi criada uma base de dados em MySql e um Web Service em PHP com recurso a RESTFull, de forma a que a aplicação tenha a possibilidade de aceder a todos esses dados armazenados no servidor. Para estimar a localização através de RFID foram criadas duas tabelas na base de dados: uma referente às tags RFID (tabela *Rfid_Tag*) e outra referente aos pontos de interesse (tabela *POI*). Os vários POIs são classificados através de várias categorias para se poder distinguir entre os casos em que estes são perigos para o utilizador ou algum tipo de ponto de interesse genérico. Com este propósito, foi também criada a tabela *Categoria*. Estas tabelas estão representadas na Figura 9.

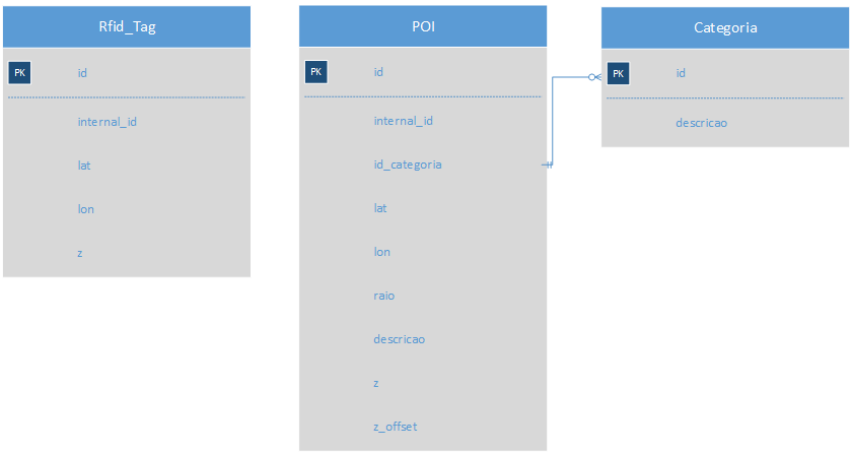

FIGURA 9 - TABELAS DA BASE DE DADOS RELACIONADAS COM A UTILIZAÇÃO DO RFID.

Uma vez que várias *tags* RFID são colocadas num ponto de interesse, não é uma boa abordagem existir apenas uma tabela com o ID da *tag* RFID e com a descrição do local que esta representa. Assim, na implementação escolhida para este trabalho, o recurso a duas tabelas diferentes cumpre o objetivo geral deste protótipo, e que é o cálculo da localização do utilizador em ambiente de interior, numa forma em que as *tags* RFID são utilizadas para obter a localização do utilizador e não a descrição do local onde este se encontra. A tabela POI é utilizada para registar os diferentes pontos de interesse, estando estes ou não assinalados com *tags* RFID.

Neste contexto, foi criado um *Web Service* de forma a aplicação possa aceder aos dados que se encontram no servidor. Para aceder a este *web service*, foram criados três métodos diferentes: getPOI(), getRfidTag() e getCategoria(), representados na Tabela 5.

TABELA 5 - MÉTODOS DO WEB SERVICE PARA ACEDER ÀS TABELAS RELACIONADAS COM O RFID.

MÉTODO	TAREFA
getPoi()	Download dos pontos de interesse.
getRfidTag()	Download das tags RFID
getCategoria()	Download das categorias.

Quando a aplicação faz o pedido ao servidor, através do Web Service, os dados são recebidos no formato JSON [2]e guardados na base de dados local da aplicação de forma a que seja possível aceder a estes dados mesmo quando o dispositivo móvel se encontrar sem acesso à internet.

Com acesso aos dados sobre as *tags* RFID e sobre os vários pontos de interesse é, então, possível fazer o processamento das mensagens recebidas através da bengala eletrónica aquando da receção de cada ID de uma *tag* RFID. Quando o ID é recebido, a aplicação procura na base de dados a existência desse ID. Caso exista, é devolvido à aplicação uma localização com dados sobre a latitude, longitude e também a altitude relativa, de forma a que seja possível fazer a distinção entre diferentes pisos em relação ao solo.

Com o conhecimento da localização do utilizador, é possível saber se este se encontra perto de algum ponto de interesse. Para essa pesquisa ser mais eficaz, do ponto de vista do utilizador, é associado a cada ponto de interesse um raio em que este ponto deve ser alertado ao utilizador. Um utilizador encontra-se num ponto de interesse, não quando se encontra 'exatamente' sobre o ponto, mas quando a distância entre a sua localização e o ponto de interesse for igual ou inferior ao raio definido. Esta medida, ou raio de deteção, pode diferir de ponto de interesse para ponto de interesse, pois existem pontos de interesse com características diferentes entre si. Exemplos disso são pontos tais

[2] JSON - https://www.json.org/ (visitado em Outubro 2018)

como pontos de perigo que devem ser avisados com maior antecedência ao utilizador para assim este evitar o perigo que esse ponto representa. Para isso, é necessário efetuar o cálculo da distância entre dois pontos geográficos (em metros), com base nas suas coordenadas. Uma vez que o planeta Terra tem uma forma aproximada a uma esfera, foi utilizado a fórmula do *Haversine* (Chopde & Nichat, 2013). Esta fórmula permite calcular a distância entre dois pontos na superfície de uma esfera. A equação utilizada encontra-se representada na Equação 1 e um exemplo da sua implementação é apresentado na Figura 10.

$$d = 2r \arcsin\left(\sqrt{sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\, sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

EQUAÇÃO 1 – HAVERSINE.

EXEMPLO DE IMPLEMENTAÇÃO EM JAVA

```
private double calculaDistancia(double lat1, double lon1, double lat2, double lon2) {
    double raioTerra = 6371;
    double dLat = Math.toRadians(lat2 - lat1);
    double dLon = Math.toRadians(lon2 - lon1);
    double sindLat = Math.sin(dLat / 2);
    double sindLon = Math.sin(dLon / 2);
    double a = Math.pow(sindLat, 2) + Math.pow(sindLon, 2) *
Math.cos(Math.toRadians(lat1))* Math.cos(Math.toRadians(lat2));
    double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
    double dist = raioTerra * c;
    return dist * 1000;
}
```

FIGURA 10 - EXEMPLO DA IMPLEMENTAÇÃO PROGRAMÁTICA DO HAVERSINE EM JAVA.

Na equação 1, φ representa a latitude e λ a longitude; *r* representa o raio da Terra, em quilómetros, e *d* é o valor da distância entre os dois pontos (de índice 1 e 2) que é

apresentado em quilómetros. No exemplo da implementação feita no âmbito desta dissertação de mestrado, foi utilizado o operador *atan2* em vez do *arcsin*, devendo-se isto ao facto de alguns sistemas móveis não disponibilizarem formas de fazer cálculos com vírgula flutuante. Com recurso ao *atan2* é feita uma aproximação ao mesmo cálculo, mas sem necessidade de processador capaz de fazer cálculos com vírgula flutuante. Por fim, o cálculo da distância *d* é convertido em metros.

Com a utilização desta fórmula é possível fazer o cálculo da distância entre dois pontos e, assim, saber se o utilizador se encontra na proximidade de um ponto de interesse. O recurso a este método de cálculo da distância a um ponto de interesse, a localização não tem que ser necessariamente dada apenas pelas *tags* RFID, mas também por outras fontes de localização.

Com o processamento da informação de localização obtida a partir das *tags* RFID é possível saber o posicionamento geográfico do utilizador com bastante precisão, pois a *tag* só é lida quando o utilizador se encontrar precisamente sobre ela. O sistema de localização baseado em RFID é, na verdade, o mais preciso do sistema CE4Blind. Como forma de testar este módulo de localização foi montada uma estrutura no evento UTAD Summer Innovation Campus, colocando *tags* RFID nas entradas das salas de cada palestra do evento. Em cada momento, nesses locais, é entregue ao utilizador informação contextual sobre a sala em que se encontra e sobre a palestra que está a decorrer nesse momento. A Figura 11 apresenta um exemplo da estrutura montada.

FIGURA 11 – INSTALAÇÃO NO EVENTO UTAD SUMMER INNOVATION CAMPUS.

6.3.2. Localização através da tecnologia de Beacons

Como já foi referido, para efetuar o cálculo da localização em ambientes de interior existe a necessidade de utilizar diferentes tecnologias, pois apesar da precisão das *tags* RFID ser bastante elevada, é impensável mapear o mundo com *tags* RFID. Assim, com recurso a outras tecnologias de localização, o sistema CE4Blind faz uma combinação das diferentes tecnologias para efetuar o cálculo da localização do utilizador da forma mais precisa possível. Como referido anteriormente também, o CE4Blind recorre ao uso de visão por computador e de triangulação Wi-Fi para determinar a localização de um utilizador. Estas duas tecnologias foram rejeitadas durante a proposta de integração na aplicação Descubra.

Para o estimar a localização do utilizador, e de forma a compensar as limitações do RFID, foi feito o recurso ao uso de *beacons*, que são pequenos dispositivos que utilizam o protocolo Bluetooth 4.0 e que transmitem informações para um *smartphone* com um consumo de energia mínimo, aquando da aproximação geográfica entre os dois (Herrera Vargas, 2016). No trabalho descrito nesta dissertação foram utilizados *beacons* do fabricante Accent Systems (modelo IBKS 105). Este modelo tem como fonte de alimentação uma pilha interna, sendo a sua autonomia aproximada de 46 meses e apresenta um alcance de deteção de 70 metros. A Accent Systems fornece uma aplicação que permite efetuar a gestão dos diferentes *beacons* de forma que seja possível alterar as informações que estes fornecem ao smartphone juntamente com o seu ID.

Ao longo dos anos foram desenvolvidos vários trabalhos relacionados com o cálculo da localização do utilizador em ambientes de interior. Em 2012 Anja Bekkelien apresentou uma dissertação relacionada com o cálculo da localização através de beacons onde diz que os métodos mais comuns para o cálculo são a triangulação e o *fingerprinting* através do RSSI. O *fingerprinting* apresenta uma maior precisão nesse cálculo, e é técnica escolhida nesse trabalho (Bekkelien, Deriaz, & Marchand-Maillet, 2012). Em 2016, Milan Herrera Vargas apresentou um trabalho relacionado com o mesmo tema, afirmando também que os métodos mais comuns são a triangulação e o *fingerprinting* (Herrera Vargas, 2016).

6.3.2.1. Métodos de localização com base na análise de sinal de rádio

Dos vários métodos utilizados para efetuar o cálculo da localização de um utilizador através das características do sinal de rádio dos *beacons*, os mais comuns são a triangulação e o *fingerprinting*. Estes dois métodos utilizam a força do sinal recebido (RSSI) pelo smartphone de forma a calcular a distância a que o utilizador se encontra dos emissores (os *beacons*). Essa distância é calcula através da equação 2.

$$d = 10^{(\frac{mp-rssi}{10*n})}$$

EQUAÇÃO 2 – CÁLCULO DA DISTÂNCIA ATRAVÉS DO RSSI.

Nesta equação, *d* representa a distância entre o beacon e o utilizador, *mp* é a potencia que o *beacon* envia a 1 metro de distância, sendo o valor comum equivalente a -69, *rssi* representa a força de sinal que o smartphone recebe do *beacon* e *n* é uma contante que depende no local e que tipicamente varia entre 2 e 4 (Faheem, Virrankoski, & Elmusrati, 2010).

Através desta equação é possível determinar a distância aproximada a que o utilizador se encontra dos *beacons*. Por vezes essa distância pode não ser muito aproximada pois os sinais enviados pelos *beacons* podem sofrer interferências provocadas pelos diferentes obstáculos existentes no espaço (Faragher & Harle, 2015).

6.3.2.1.1. Fingerprinting

O método de triangulação com a utilização de características do sinal dos *beacons* utiliza os mesmos princípios do método de triangulação de Access Points.

O *fingerprinting*. como o nome (em inglês) sugere é a criação de uma marca, ou assinatura, de um local. No caso deste tipo de aplicação, isto é feito através da recolha da força dos diferentes sinais RSSI recebidas num determinado local. De uma perspetiva mais geral, são recolhidas as forças RSSI em diferentes localizações e guardadas numa base de dados com associação ao respetivo ID de cada dispositivo emissor. Após a recolha das forças RSSI em diferentes localizações a marca está criada e é, assim, possível fazer um cálculo da localização aproximada do utilizador. Quando o utilizador se encontra num local que foi previamente mapeado, pela leitura das forças RSSI existentes no local é feita

uma comparação com as leituras presentes na base de dados. A marca armazenada que tiver mais semelhanças indica a possível localização do utilizador (Faragher & Harle, 2015). Para fazer a comparação destes dados, podem ser utilizados vários algoritmos, e um dos mais comuns é o algoritmo *k-nearest neighbor* que será explicado mais adiante.

6.3.2.2. Implementação do protótipo

Uma vez que o *fingerprinting* consegue ter uma melhor precisão que a triangulação do sinal, o método implementado para este protótipo foi o *fingerprinting*.

Para tal, foram criados dois protótipos: um protótipo de mapeamento, para recolha de forças RSSI, e outro que utiliza os dados recolhidos para estimar a localização do utilizador. Tanto para o protótipo que recolhe as forças de RSSI, como para o protótipo que faz o cálculo da localização do utilizador, foi então utilizada a *Android Beacon Library*.

Na recolha dos dados sobre um determinado local, é necessário existir uma forma de saber a localização exata do dispositivo móvel para que, aquando da comparação das forças em tempo real, seja possível obter uma localização bastante exata. Visto que a utilização de GPS em interiores não é possível, torna-se também impossível saber a localização exata do utilizador através dessa tecnologia presente no dispositivo móvel.

Para resolver este problema, uma abordagem poderia passar, numa fase inicial, pelo desenvolvimento de uma aplicação onde, num mapa, o utilizador seleciona o local onde se encontra enquanto recolhe as forças RSSI dos *beacons*. Esta abordagem foi rejeitada pois seria difícil selecionar o local de forma precisa num ecrã de pequenas dimensões, como o de um smartphone. Além disso, e visto que as coordenadas geográficas que se pretende recolher são coordenadas no interior de um edifício, é difícil indicar o local exato do dispositivo móvel no edifício através de imagens de satélite, como seria feito no recurso à utilização de uma ferramenta como o Google Maps.

Assim, no trabalho desenvolvido nesta dissertação, foi feito o recurso à ferramenta open-source QGIS, que fornece a capacidade de poder ver, editar e analisar dados geográficos. Uma das funcionalidades que esta ferramenta fornece, em particular, permite georreferenciar a planta de um edifício através do seu mapa *raster*, ou impresso. Assim, é possível determinar um ponto geográfico de forma muito precisa dentro de um edifício,

e, de forma mais particular, no interior de salas. A Figura 12 mostra um exemplo da utilização desta ferramenta para a georreferenciação de um edifício.

FIGURA 12 – EXEMPLO DA GEORREFERENCIAÇÃO DE UM EDIFÍCIO.

Na Figura 12 pode-se ver o mapa impresso de um edifício com a sobreposição da planta desse mesmo edifício em formato digital. Pode-se com esta ferramenta recolher as coordenadas geográficas de uma forma muito precisa. Como referido anteriormente, com a utilização de *fingerprinting* quanto maior for o número de recolhas, maior é a precisão. Assim, e dado que pode existir alguma demora no tempo de recolha das forças do sinal, o tempo tende a aumentar com a necessidade de identificar geograficamente mais pontos de cada local. Numa tentativa de diminuir este tempo foi desenvolvido um algoritmo capaz de calcular os pontos geográficos de uma determinada área retangular através dos quatro vértices do retângulo que a define. Para isso, foi utilizada a equação da reta $y = mx + b$, em que x é a latitude e y a longitude, e em que

$$m = \frac{longitude\ final - longitude\ inicial}{latitude\ final - latitude\ inicial}$$

EQUAÇÃO 3 - DECLIVE DA RETA.

e, por sua vez,

$$b = y - mx$$

EQUAÇÃO 4 – CÁLCULO DA ORDENADA NA ORIGEM DO REFERENCIAL CARTESIANO.

é a ordenada na origem do referencial.

Sabendo a equação da reta e os pontos inicial e final, é possível calcular os pontos intermédios consoante as linhas e colunas que desejamos. A Figura 13 representa uma divisão de 3x3, com a utilização das equações anteriores.

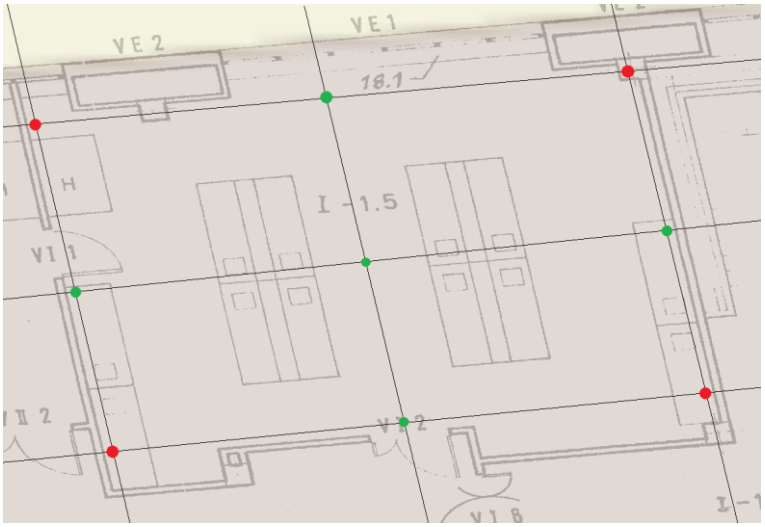

FIGURA 13 - EXEMPLO DO RESULTADO DE UM CÁLCULO DOS PONTOS INTERMÉDIOS.

Na Figura 13, os pontos a vermelho representam os quatros pontos que definem os cantos do retângulo, sendo esses os pontos também utilizados para fazer a recolha. As linhas a preto representam as diferentes retas calculadas e, por fim, os pontos verdes representam os pontos intermédios calculados através do algoritmo descrito, e que também irão ser utilizados para a recolha do sinal. Depois do cálculo dos diferentes pontos geográficos, a aplicação, durante a recolha dos dados fornece ao utilizador informações sobre quantos pontos devem ser recolhidos, a distância entre eles e quantos pontos existem por linha e por coluna. Essa informação esta representada na Figura 14.

FIGURA 14 - INFORMAÇÕES OBTIDAS APÓS A GERAÇÃO DE UM MAPA.

Após a recolha das diferentes forças RSSI nos vários locais foi necessário criar uma forma de guardar os dados de maneira que estes possam ser utilizados pela aplicação de localização. Para isso foram criadas duas tabelas na base de dados: uma para guardar as recolhas, outra para guardar informações sobre os diferentes *beacons*. Estas tabelas estão representadas na Figura 15.

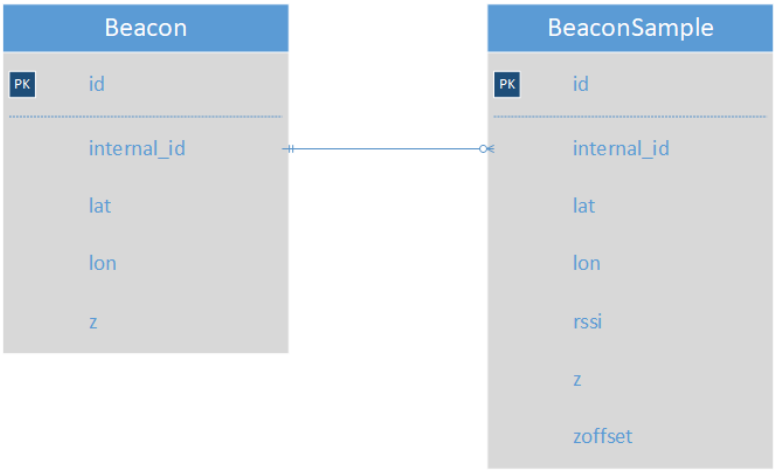

FIGURA 15- TABELAS RELACIONADAS COM O ARMAZENAMENTO DE DADOS RELACIONADOS COM OS BEACONS.

Tanto para aceder como para guardar, através de *web services*, os dados recolhidos, foram desenvolvidos os métodos getBeacon(), getBeaconSamples() e setBeaconSamples(), representados na Tabela 6.

TABELA 6 - MÉTODOS DEFINIDOS NO WEB SERVICE PARA ACEDER AOS DADOS DOS BEACONS.

MÉTODO	OBJETIVO
getBeacon()	Download dos beacons.
getBeaconSamples()	Download das forças RSSI.
setBeaconSamples ()	Upload dos RSSI recolhidos.

Para efetuar o cálculo da estimativa da localização do utilizador através do método de *fingerprinting* é feita uma comparação entre as forças recolhidas, em tempo real, e os valores armazenados na base de dados. Para fazer a comparação, foi utilizado o algoritmo de agrupamento (ou *clustering*) *k-nearest neighbor*. Este algoritmo faz uma comparação entre um vetor de entrada e um conjunto previamente treinado utilizando como critério de comparação a distância euclidiana. Este algoritmo é conhecido por apresentar bons resultados e de ser de fácil implementação (Bekkelien et al., 2012). A utilização das distâncias euclidianas como critério de comparação é uma mais valia pois, como referido anteriormente, a utilização das forças RSSI tem como objetivo o cálculo das distâncias entre o utilizador e os *beacons*.

Deve ser referido que a utilização deste algoritmo fornece uma localização do utilizador que é apenas aproximada, devido aos erros que as forças dos *beacons* podem apresentar, consoante os diferentes obstáculos presentes no meio. Devido à existência de sinais de vários *beacons*, no final da comparação são obtidas diferentes localizações, sendo necessário selecionar qual delas é a localização final do utilizador. Para isso, foram implementadas e testadas duas formas diferentes de o fazer: através da média das diferentes localizações obtidas como sendo a localização do utilizador e através da escolha da localização mais comum. Os testes a estas duas abordagens tiveram os objetivos gerais de verificar, por um lado, qual é o método mais preciso e, por outro lado, medir a diferença entre a existência de um mais ou menos elevado de amostras e de *beacons*.

6.3.2.3. Testes ao protótipo

Para a realização dos testes foram colocados *beacons* em localizações diferentes e foram utilizados números diferentes de *beacons*, na sala. Para o cenário em que foram colocados diferentes número de *beacons* foram feitos testes com base em:

- recolhas de 3x3 localizações (total de recolhas=9);
- recolhas de 4x4 localizações (total de recolhas=16);
- recolhas de 5x5 localizações (total de recolhas=25).

Com base nos diferentes cenários de recolha, o algoritmo *k-nearest neighbor* foi utilizado 10 vezes para determinar a localização do utilizador pelo método da localização mais frequente e 10 vezes através do cálculo da média das localizações estimadas. A localização real do utilizador, quando efetuado o pedido de localização, foi o centro da sala, tal como representado na Figura 16.

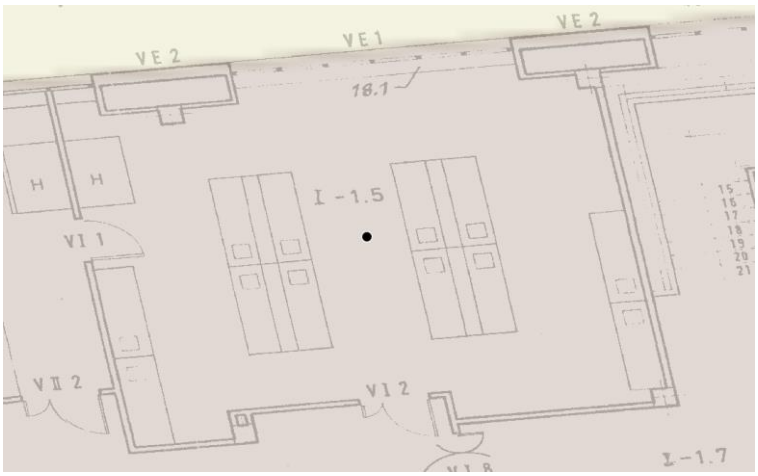

FIGURA 16 - LOCALIZAÇÃO DO UTILIZADOR.

A Figura 17 representa os testes em foram colocados 3 *beacons* a emitir sinal, e onde a localização de cada *beacon* está indicada com um círculo verde.

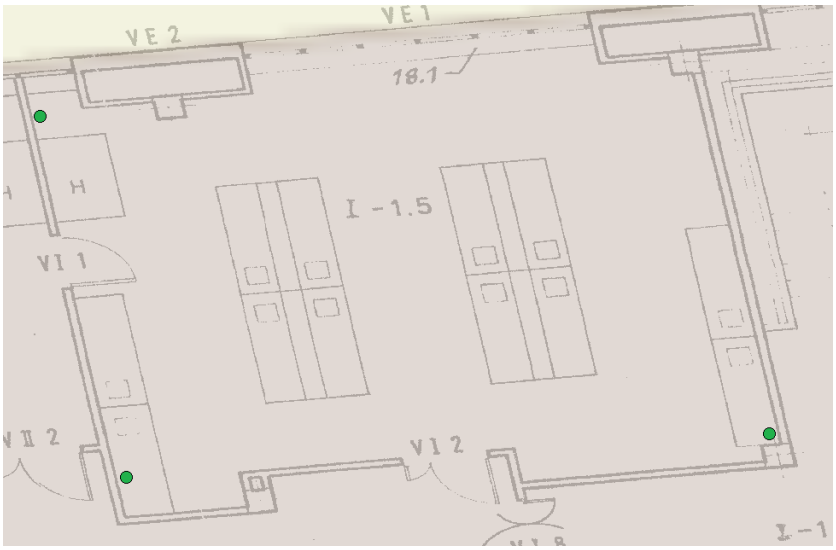

FIGURA 17 - LOCALIZAÇÃO DOS 3 BEACONS.

Após as várias recolhas e os vários pedidos de localização, os resultados obtidos, e apresentados na Tabela 7 e na Tabela 8, apresentam algumas variações nas localizações estimadas como sendo a verdadeira localização do utilizador.

TABELA 7 - VARIAÇÕES PELO MÉTODO DA LOCALIZAÇÃO MÉDIA BASEADA EM 3 BEACONS.

LOCALIZAÇÃO MÉDIA		
3x3	4x4	5x5
1,91-3,93m	1,07-4,71m	1,58-4,76m

TABELA 8 - VARIAÇÕES PELO MÉTODO DA LOCALIZAÇÃO MAIS FREQUENTE BASEADA EM 3 BEACONS.

LOCALIZAÇÃO MAIS FREQUENTE		
3x3	4x4	5x5
0,13-6,12m	0,13-5,86m	0,13-6,12m

A Figura 18 representa os testes em foram colocados 4 *beacons* a emitir sinal, e onde a localização de cada *beacon* está indicada com um círculo verde.

FIGURA 18 - LOCALIZAÇÃO DOS 4 BEACONS.

Após as várias recolhas e os vários pedidos de localização, os resultados obtidos, e apresentados na Tabela 9 e 10, apresentam algumas variações nas localizações estimadas como sendo a verdadeira localização do utilizador.

TABELA 9 - VARIAÇÕES PELO MÉTODO DA LOCALIZAÇÃO MÉDIA BASEADA EM 4 BEACONS.

LOCALIZAÇÃO MÉDIA		
3x3	4x4	5x5
0,69-4,63m	0,65-5,86m	0,27-6,12m

TABELA 10 - VARIAÇÕES PELO MÉTODO DA LOCALIZAÇÃO MAIS FREQUENTE BASEADA EM 4 BEACONS.

LOCALIZAÇÃO MAIS FREQUENTE		
3x3	4x4	5x5
1,86-4,96m	1,88-6,12m	1,86-6,12m

Por fim, a Figura 19 representa os mesmos testes num cenário em que foram colocados 5 *beacons* no mesmo espaço.

FIGURA 19 - LOCALIZAÇÃO DOS 5 BEACONS.

Após as várias recolhas e os vários pedidos de localização, os resultados obtidos, e apresentados na Tabela 11 e 12, apresentam algumas variações nas localizações estimadas como sendo a verdadeira localização do utilizador.

TABELA 11 - VARIAÇÕES PELO MÉTODO DA LOCALIZAÇÃO MÉDIA BASEADA EM 5 BEACONS.

LOCALIZAÇÃO MÉDIA		
3x3	4x4	5x5
0,45-5,60m	0,62-4,88m	0,45-2,81m

TABELA 12 - VARIAÇÕES PELO MÉTODO DA LOCALIZAÇÃO MAIS FREQUENTE BASEADA EM 5 BEACONS.

LOCALIZAÇÃO MAIS FREQUENTE		
3x3	4x4	5x5
1,64-4,94m	1,84-5,60m	1,84-6,12m

No final dos vários testes, foi possível verificar que o valor recebido pelos *beacons* varia de pedido para pedido, mesmo que esse pedido seja feito no mesmo local e no mesmo cenário. Isto mostra que existem interferências nas forças RSSI que não podem ser desconsideradas. Podemos também ver que o uso de um maior número de *beacons*

diminui o erro associado ao cálculo da localização do utilizador, assim como um quando há um maior número de amostras.

É possível concluir, também, que a utilização da média para o cálculo da localização do utilizador atenua o erro, ou seja, poderá ser utilizado como um filtro.

6.3.2.4. Considerações finais

Após a implementação das técnicas descritas e após a realização dos testes foi possível demonstrar que o uso de *beacons* é uma opção válida para obter a localização do utilizador, apesar de, obviamente, apresentar alguma falha na precisão do cálculo da localização.

Nos teste realizados, no melhor dos casos o erro situa-se entre os 0,46 metros e os 2,81 metros. Não é um valor muito elevado, mas, apesar de poder parecer insignificante para uma pessoa normal, no caso de uma pessoa cega pode representar a diferença entre estar perto ou longe de um local de perigo.

6.3.3. Integração com a aplicação Descubra

Após a implementação dos dois módulos para o cálculo da localização do utilizador em ambientes de interior realização de todos os testes realizados, verificou-se que a utilização de *beacons* apresenta uma imprecisões que não se podem desconsiderar. Assim, em ambiente de interior, numa fase inicial a integração com a aplicação Descubra será somente baseada na utilização da tecnologia RFID.

Como cenário de teste em ambiente real, tanto a tecnologia RFID como a interação com a bengala eletrónica foram integradas numa aplicação especificamente criada pela empresa WorldIT para o evento Portugal Economia Social, promovido pela Presidência da República de Portugal. Nesse evento foi testada e demonstrada a utilização da bengala e a utilização da tecnologia RFID. Vários conjuntos de etiquetas RFIS foram instaladas em sítios específicos do local do evento para que, quando um utilizador se aproximasse do local, ouvisse informação relacionada e relevante. Esta integração foi feita com sucesso tendo um *feedback* bastante positivo por parte dos utilizadores da aplicação, incluindo utilizadores cegos.

6.4. Localização em ambientes de exterior

O sistema CE4Blind utiliza o GPS para uma estimativa da localização do utilizador em ambientes de exterior. Da mesma forma, o protótipo desenvolvido no âmbito do trabalho desenvolvido nesta dissertação, utiliza a biblioteca fornecida pelo Android para esse propósito.

De forma a obter atualizações frequentes da localização do utilizador com base em sinal de GPS, e tal como no caso da gestão da ligação a periféricos via Bluetooth, foi implementado um serviço, capaz de correr permanentemente em *background*, que verifica se existe ou não sinal GPS e, em caso afirmativo, devolve através de *broadcast* a localização atual do utilizador. Durante o desenvolvimento verificou-se que a localização fornecida era, por vezes, complementarmente errada. Consoante o hardware, muitas vezes a localização alterava vários quilómetros entre amostras. Para resolver este problema foi aplicado um filtro que verifica se a localização anteriormente dada pelo GPS é muito diferente da nova amostra.

A parte mais importante do protótipo de integração de que é alvo todo o trabalho desenvolvido nesta dissertação de mestrado, é a forma como é efetuada a navegação entre dois pontos geográficos diferentes. Para este propósito, foi utilizada usada a API GraphHopper, que é uma API que utiliza os dados fornecidos pelo OpenStreetMaps para calcular a rota mais curta entre dois pontos e para, a partir daí, facilitar a geração de instruções de navegação.

6.4.1. Navegação

Para efetuar o cálculo da forma mais curta de navegar de um ponto para o outro, a API GraphHopper necessita dos dados fornecidos pelo OpenStreetMaps. Assim, o protótipo deve ter a capacidade de efetuar o download desses dados de forma a que possam ser utilizados localmente, no dispositivo. Para facilitar a implementação, foram criados, a partir do OpenStreetMaps, *datasets* que representam diferentes cidades que foram, posteriormente, colocados no servidor. Esta separação por cidades deve-se ao facto de que a Descubra se encontra também divido por municípios, não havendo a

necessidade de a aplicação armazenar localmente dados de todos os municípios, mas somente dados do município ao qual esta está associada.

Com acesso aos dados provenientes do OpenStreetMaps é possível utilizar a API GraphHopper para calcular uma rota que pode ser utilizada para gerar instruções de navegação. O cálculo da rota tem por base a indicação de dois pontos geográficos distintos: um ponto de início e um ponto de destino. O ponto de destino é, obviamente, um ponto de interesse da base de dados. No caso da integração com a aplicação Descubra, estes pontos de interesse são os pontos associados à Descubra. Durante o desenvolvimento do protótipo foi utilizada a tabela de POIs já existente na base dados anteriormente criada na para a utilização da tecnologia RFID, utilizando os métodos também já criados no Web Service para aceder a esta, e que posteriormente será gravada na base dados local do smartphone. O ponto de início da navegação é, logicamente, a localização atual do utilizador. Tendo conhecimento destes dois pontos, e com recurso à utilização da API GraphHopper, é obtido o caminho que o utilizador deverá seguir e que, segundo a especificação da API, é uma lista de nós ordenados que representa o caminho mais curto.

Esta lista de pontos consecutivos é a única informação que a API fornece, e foi, por isso, necessário criar um algoritmo capaz de saber se o utilizador se encontra no caminho correto. Assim, com base na lista de pontos gerada pela API, é criada uma lista em que se associam diferentes informações aos diferentes nós, de forma a facilitar a geração das indicações que o protótipo dará ao utilizador. Exemplos deste tipo de indicações, são por exemplo, situações em que o utilizador deve ser informado que tem de percorrer uma certa distância entre nós (em passos), ou qual a rotação que este deve efetuar para se deslocar para o nó seguinte (em horas).

A classe *Ponto* foi criada com a intenção de facilitar este processo. Esta classe é composta guarda dados sobre cada nó gerado pelo GraphHopper, bem como as diferentes informações associadas, como se representa na Figura 20.

FIGURA 20 - CLASSE PONTO.

Esta classe permite, então, facilitar a geração das informações de ajuda ao seguimento da rota, de forma a que seja possível orientar o utilizador de forma segura. A distância entre os pontos é calcula utilizando a fórmula de *haversine* anteriormente referida, e geração de cada instrução é efetuada ao longo da lista de pontos gerada pela API GraphHopper. Cada tem uma indicação relativa à distância necessária para chegar ao ponto seguinte. A Figura 21 apresenta um exemplo de como é feita a associação da distância a cada um dos nós seguintes.

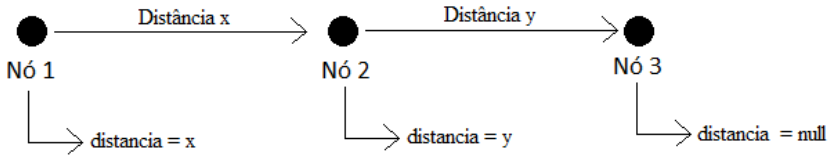

FIGURA 21 - EXEMPLO DAS DISTÂNCIAS RELATIVAS ENTRE PONTOS.

Visto que a fórmula *haversine* (Chopde & Nichat, 2013) fornece a distância em metros, é necessário fazer uma conversão para passos antes de cada indicação ser armazenada na lista de pontos. Isto é feito com recurso uma regra de proporcionalidade direta em que 1 passo equivale em média a 0,7m. Pode-se, assim, calcular de forma aproximada quantos passos o utilizador terá de dar. No caso de ser necessário fazer arredondamento, este é sempre feito de uma forma que majore a distância (por exemplo, num estimativa de 7,8 passos é armazenado o valor de 8 passos).

Para calcular a direção (rotação em cada nó) que utilizador deverá tomar, foi utilizada a seguinte fórmula.

$$angulo180 = \frac{atan2(longitude2 - longitude1, latitude2 - latitude1) * 180}{\pi}$$

EQUAÇÃO 5 – CÁLCULO DA VARIAÇÃO DE ÂNGULO ENTRE DOIS NÓS CONSECUTIVOS.

Esta fórmula devolve o ângulo correspondente ao declive entre os dois pontos que compõem cada segmento de reta, e o resultado está compreendido no intervalo [-180,180] graus. Para. posteriormente, facilitar a representação dessa informação o valor é convertido para o intervalo [0,360].

Para determinar a direção que o utilizador deve tomar é necessário saber qual a diferença de orientação do utilizador ao ponto seguinte. Assim, é utilizado o mesmo método para calcular o ângulo da reta que une os dois pontos da trajetória ideal e, por fim, sabendo as duas orientações, o utilizador terá de rodar o valor indicado pela subtração do segundo ângulo calculado com o primeiro. No caso de o primeiro ângulo ser inferior ao segundo ângulo, é necessário somar 360 graus para se obter o ângulo correto. A Figura 22 mostra um exemplo da aplicação do algoritmo e da relação entre os dois ângulos.

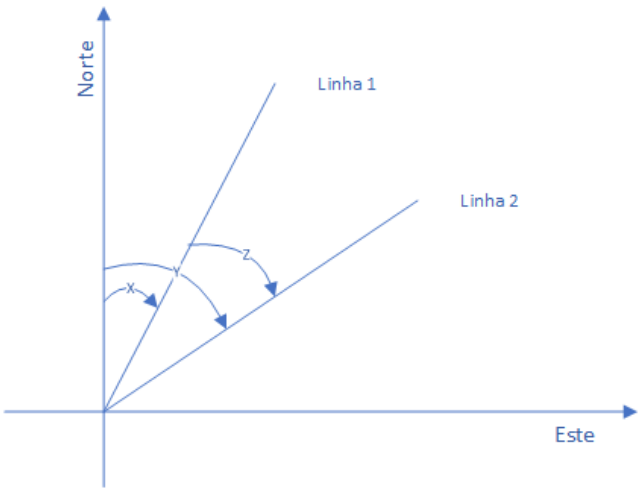

FIGURA 22 – ILUSTRAÇÃO DA ROTAÇÃO A SER CALCULADA.

Depois de calculado o ângulo que o utilizador deve rodar para chegar ao próximo nó, é necessário fazer uma conversão desse valor para um formato que o utilizador facilmente compreenda. No caso da implementação feita, esse valor de graus-norte é

transformado em horas, tal como num relógio. Tal como num círculo, num relógio cada hora incrementa com intervalos de 30º. A Figura 23 mostra um exemplo dessa divisão.

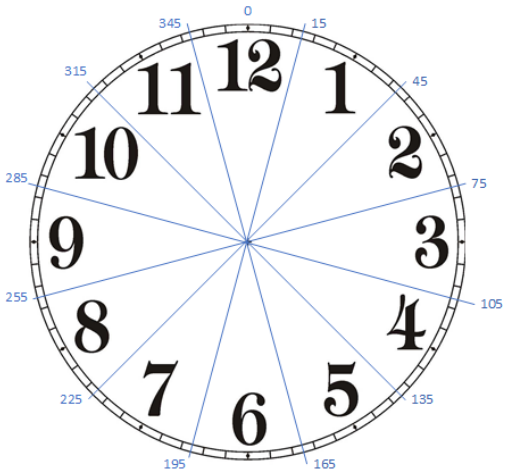

FIGURA 23 – SISTEMA ANGULAR VS. SISTEMA HORÁRIO.

Após a conversão para o referencial horário, é possível indicar ao utilizador ele deverá virar, ou rodar, em horas, para seguir para o próximo nó. É utilizada a medida 'horas' para dar essa informação em vez de 'graus' pois é de muito mais fácil (e imediata) compreensão ouvir "Virar para 2 horas." do que ouvir "Rode 63 graus".

Visto que a aplicação é destinada a pessoas cegas, e de forma a que esta seja segura, o utilizador terá que seguir sempre o caminho de nó em nó não podendo percorrer o caminho através de atalhos. Saindo da rota, este poderá encontrar perigos não georreferenciados. Assim, a cada nova coordenada do utilizador, o algoritmo de navegação verifica se utilizador se esta a aproximar do nó, e se a distância entre este e o nó diminui ao longo do tempo. Nesse caso, o utilizador encontra-se no caminho correto e pode seguir normalmente. Caso contrário, a aplicação deverá avisar o utilizador que não se está a deslocar pelo caminho correto, recalculando o caminho. Quando o utilizador chega ao novo nó, este é informado sobre a direção em horas que terá de virar e a distância em passos que terá de percorrer para chegar ao próximo no da rota. Com este algoritmo é, então, possível ajudar numa navegação segura do utilizador.

Após esta fase do desenvolvimento deste módulo, verificou-se, após alguns testes, que devido ao erro existente no GPS, já referido anteriormente, é difícil garantir que o utilizador se encontra 'exatamente' sobre o nó real, originando alguns problemas na

progressão pelo caminho desejado. Para resolver este problema foi acrescentado um atributo à classe *Ponto*: uma lista de pontos geográficos, de forma a formar um retângulo à volta dele. Este retângulo é utilizado para verificar se utilizador se encontra sobre o nó. O formato de retângulo permite abranger os dois passeios de uma via, sem aumentar o raio do ponto de interesse. Adicionalmente, foi necessário criar uma forma de 'desenhar' um retângulo no nó de uma forma em que fique sempre alinhado com a direção do caminho a seguir. Para efetuar esse cálculo é necessário saber o declive da reta que leva o utilizador ao nó. Este angulo já foi previamente calculado utilizando a equação para calcular a direção em que o utilizador se terá de deslocar, bem como as distâncias entre os vários pontos. Para efetuar o cálculo dos pontos do retângulo desejado, é necessário utilizar duas equações diferentes: uma para a latitude e outra para a longitude, como descritas na Equação X e Equação Y.

$$latitude2 = latitude1 + distância * \cos(\alpha)$$

EQUAÇÃO 6 - LATITUDE ATRAVÉS DE PONTO INICIAL DISTÂNCIA E ÂNGULO.

$$longitude2 = longitude1 + distância * \sin(\alpha)$$

EQUAÇÃO 7- - LONGITUDE ATRAVÉS DE PONTO INICIAL DISTÂNCIA E ÂNGULO.

Para o determinar as coordenadas dos vértices estas equações serão utilizadas seis vezes: duas para cálculo de dois pontos intermédios e quatro vezes para calcular os quatro vértices do retângulo, tal como se pretende e se encontra representado na Figura 24.

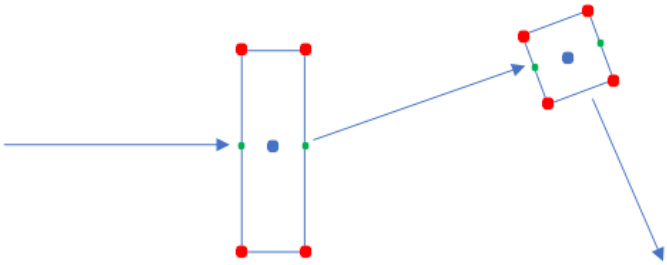

FIGURA 24 - EXEMPLO DA DEFINIÇÃO DA ÁREA GEOGRÁFICA DE COBERTURA DE UM NÓ.

Na Figura 24, os pontos vermelhos representam os vértices do retângulo e os pontos verdes representam os pontos intermédios. Conhecendo os vértices do retângulo centrado em cada nó, é necessário utilizar um algoritmo que nos permita saber quando o utilizador se encontra dentro do nó. O método de verificação utilizado recorreu à soma dos ângulos internos entre o utilizador e cada um dos vértices. Isto é válido no caso das figuras geométricas convexas, de que o retângulo é um exemplo. Quando um ponto está localizado dentro de um retângulo a soma dos vários ângulos que este faz com os diferentes vértices é sempre igual a 360º. É, assim, possível, comparativamente à utilização de um círculo, saber de uma forma mais precisa quando o utilizador se encontra num nó, de forma a que seja possível fornecer-lhe as diferentes informações.

Com esta maneira precisa de fornecer ao utilizador as informações necessárias para este ser orientado por um caminho seguro, a fase seguinte de desenvolvimento do protótipo passa por conseguir fornecer informações adicionais ao utilizador, tais como como quando este terá de passar numa passadeira ou de se deslocar sobre uma escada. Para abordar este desafio, foi necessário colocar uma camada de acessibilidade sobre os dados fornecidos pelo GraphHopper. Uma vez que este utiliza dados provenientes do OpenStreetMaps, é necessário ter a capacidade de modificar esses dados.

Durante o desenvolvimento verificou-se que o OpenStreetMaps fornece um editor que permite fazer operações sobre o seu GIS, e que nos permite alterar todos os dados existentes na sua base de dados. Estes dados podem ser alterados por qualquer pessoa, o que, por um lado, pode ser uma mais valia, uma vez que as atualizações podem ser feitas

por qualquer pessoa de uma vasta comunidade, garantindo um mapa sempre atualizado. Por outro lado, qualquer pessoa pode, também, apagar dados e isso pode provocar danos em todas as aplicações que utilizem os dados do OpenStreetMaps. O OpenStreetMaps fornece um repositório do seu editor de forma a que seja possível fazer uma instalação num servidor local e, desta forma, garantir que os dados apenas podem ser editados pelos responsáveis do desenvolvimento da aplicação. No âmbito dos trabalhos descritos nesta dissertação, foi feito um *deploy* do editor do OpenStreetMaps numa máquina local. Seguidamente, os dados foram alterados de forma a tornar o caminho mais seguro para o utilizador, removendo algumas estradas sem passeios, e identificando passadeiras e escadas. A Figura 25 mostra o mapa original antes de serem efetuadas alterações.

FIGURA 25 - MAPA ORIGINAL.

Após a identificação de locais perigosos, e passadeiras e escadas foram feitas as alterações representadas na Figura 26.

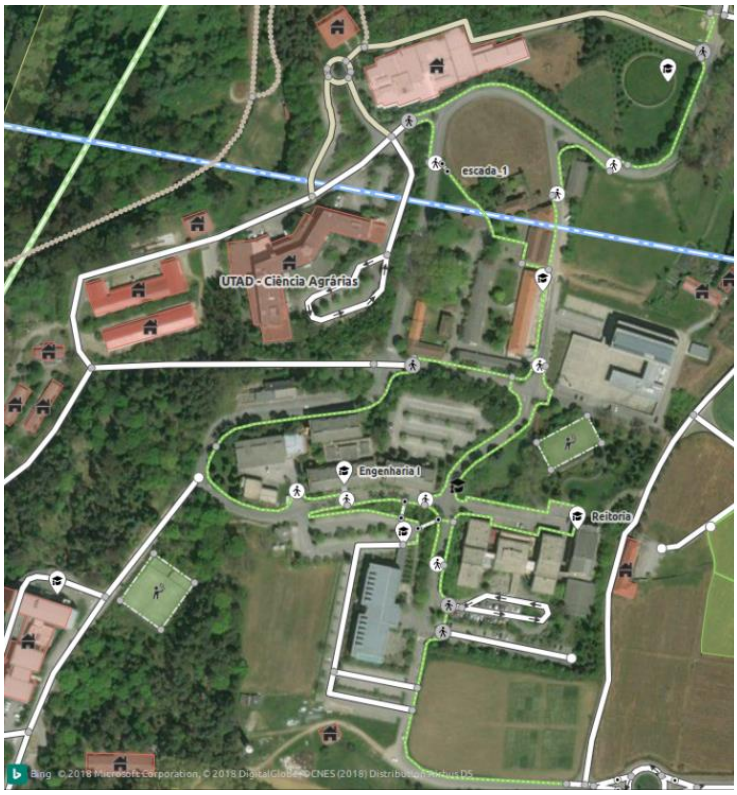

FIGURA 26 - MAPA ALTERADO.

De forma a poder identificar uma escada corretamente e para saber se o utilizador terá de a subir ou descer, os nós foram identificados de forma diferente: na zona mais baixa o nó terá o nome de 'escada_0'; por sua vez, na zona mais alta o nó terá o nome de 'escada_1'. Visto que o GraphHopper apenas fornece os nós existentes no caminho, as alterações feitas no mapa somente vão afetar o caminho. Assim, para identificar este tipo de nós tal foram criadas no servidor as tabelas *Stairs* e *CrossRoads*, tal como especificadas na Figura 27.

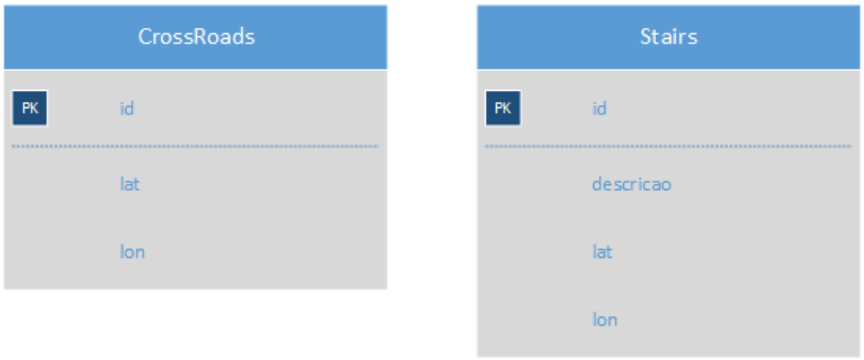

FIGURA 27 - TABELAS DA BASE DE DADOS RELACIONADAS COM AS PASSADEIRAS E AS ESCADAS.

Estas tabelas armazenam dados do OpenStreetMaps relativos às passadeiras e às escadas. Paralelamente, foram criados métodos no Web Service capazes de fazer download desses dados para a máquina local e de colocar essa informação no servidor. Foram ainda desenvolvidos métodos que permitem à aplicação fazer download desses dados, localmente. Esses métodos estão representados na Tabela 13.

TABELA 13 - METODOS DEFINIDOS NO WEB SERVICE PARA ACEDER ÀS PASSADEIRAS E ÀS ESCADAS.

MÉTODO	OBJETIVO
setStairs()	Upload dos dados relativos as escadas do OpenStreetMaps no servidor.
setCrossRoads()	Upload dos dados relativos as passadeiras do OpenStreetMaps no servidor.
getStairs()	Download das escadas.
getCrossRoads()	Download das passadeiras.

Para fazer ligação dos diferentes nós gerados pelo GraphHopper foi criado um novo método e adicionado um novo atributo à classe Ponto, que armazena uma descrição que pode ser utilizada de forma a identificar se este é uma passadeira ou uma escada. Este método verifica se na lista de nós fornecida pelo GraphHopper existe algum nó que contenha a mesma localização que os dados das passadeiras ou os dados das escadas. Se existir um nó com a mesma localização, ou seja, no caso de existir semelhança entre a localização do nó e algum dos dados, o método altera o atributo da 'descrição' para *passadeira* no caso de este ser uma passadeira ou coloca a descrição do *ponto* igual à descrição da escada que tenha a localização igual a ele. No caso de este ser uma escada, como referido anteriormente, a descrição será 'escada_0' ou 'escada_1'.

O algoritmo de navegação foi, obviamente, alterado para abranger estas características. Assim, é possível avisar o utilizador nos casos em que tenha de passar uma passadeira ou subir/descer uma escada naquele nó. Ou seja, a cada nova coordenada do utilizador, além de se verificar se utilizador se encontra dentro do retângulo, também se verifica se o nó onde o utilizador se encontra contém uma descrição. Se a descrição for

‘*escada_0*’, em vez de se indicar a rotação e a distância a seguir (tal como é feito por defeito) é indicado ao utilizador que este deve subir uma escada. O inverso é dito se a descrição do nó for ‘*escada_1*’. Caso o nó seguinte tenha a descrição ‘*passadeira’* são ditas ao utilizador as informações normais, acrescentando-se à mensagem que este terá de passar uma passadeira. No caso de não haver descrição, são apenas entregues as informações mais simples de rotação e distância.

6.5. Integração com a aplicação Descubra

Na fase atual o protótipo ainda não foi completamente integrado com a Descubra na sua versão comercial. Antes da integração final é necessária a realização de testes para verificar se funciona da forma pretendida, uma vez que na rua existem perigos que não podem ser controlados, tais como os carros ou a existência de obstáculos nos passeios que não podem ser previstos, tais como buracos. Apenas após a realização desses testes poderá ser determinado se ainda é necessário algum melhoramento de forma a apresentar uma aplicação que garanta a maior segurança possível.

Quando ocorrer essa integração final com o Descubra os pontos de interesse da aplicação serão os mesmos existentes na aplicação Descubra. Serão pontos de relevo turístico, e é, portanto, sempre necessário fazer um mapeamento do local onde esses pontos de interesse se encontram, de forma a que seja possível evitar estradas impróprias a peões e, principalmente, peões cegos, bem como fazer a identificação das passadeiras e escadas. Deve-se garantir que a navegação é sempre segura para o utilizador.

7. CONSIDERAÇÕES FINAIS

O objetivo principal que foi proposto para o desenvolvimento do trabalho realizado no âmbito desta dissertação foi o de integrar funcionalidades da aplicação CE4Blind na aplicação Descubra. Durante o seu desenvolvimento determinou-se que o CE4Blind não apresentava formas simples de efetuar integração direta das suas funcionalidades, uma vez que ainda é um protótipo demostrador. Assim, foi necessário replicar as funcionalidades que foram determinados, em conjunto com a WorldIT, para integração com a Descubra.

Após algum trabalho de especificação das funcionalidades a integrar, foram propostos e desenvolvidos os diferentes protótipos dos vários módulos a integrar na aplicação Descubra. Alguns destes módulos já se encontram, neste momento, integrados e ainda há outros por integrar. Os módulos desenvolvidos que já estão e integrados são os módulos de interação com a bengala eletrónica e de navegação em ambientes de exterior, além do sistema de informação geográfico, assente numa instância do OpenStreetMaps, que dá suporte ao armazenamento e disponibilização dos dados necessários ao cálculo de rotas e apresentação de informação contextual. Apesar de já estarem integrados com a aplicação, ainda não se encontram disponíveis na versão publicada na *playstore* pois ainda não se encontra nenhuma bengala disponível para utilização, estando este *hardware* ainda em versão de protótipo. Além disso, por decisão do consórcio da parceria, a aplicação deve ainda passar por mais fases de teste que são necessários para validar a introdução no mercado.

Relativamente ao módulo de localização através de *beacons* Bluetooth, foram desenvolvidos e testados vários tipos de disposição dos *beacons* e vários métodos de estimar a localização do utilizador. Verificou-se que esta tecnologia se adequa e pode ser utilizada pela plataforma Descubra. No entanto, a implementação deste método requer a instalação de uma infraestrutura física, e isto pode ser um entrave à sua integração em determinados locais. Além disso, esta infraestrutura requer manutenção, ainda que baixa, uma vez que os *beacons* são alimentados por bateria.

Como conclusão geral do trabalho desenvolvido e exposto nesta dissertação, é possível afirmar que um utilizador cego pode utilizar a aplicação de forma acessível com o auxílio da bengala eletrónica desenvolvida no projeto CE4Blind, para, quer em interior,

quer em exterior, ser capaz de se orientar e de navegar, recebendo informações sobre o seu contexto geográfico e instruções sobre como pode chegar a um destino à sua escolha, respetivamente.

7.1. Trabalho Futuro

A necessidade ter e manter uma aplicação acessível para pessoas cegas faz com que este trabalho tenha de ser continuado melhorado. Assim, como indicação da direção futura que este trabalho pode tomar, e como já foi referido, será necessário efetuar mais testes em ambiente real, de interior e exterior, com pessoas cegas, de forma a verificar a segurança, precisão e, por conseguinte, a fiabilidade e robustez da aplicação. Também será necessário a implementação de uma versão para os sistema operativo móvel iOS visto que a aplicação Descubra existe para os dois sistemas operativos. Essa versão também necessitará de testes pois os sistemas operativos funcionam de forma diferente, podendo algumas das funcionalidades desenvolvidas para o sistema operativo móvel Android não serem compatíveis com iOS, sendo assim acrescida a necessidade de procurar soluções semelhantes.

Nesta fase da implementação, apesar de já estarem desenvolvidos todos os módulos, devem ser feitos testes complementares aos módulos que falta integrar na versão comercial da aplicação Descubra, com mais utilizadores reais, cegos, em ambiente real, para poder ser feita a integração completa. Após a integração na versão comercial da aplicação, deve-se continuar na busca por tecnologias e métodos de estimar a localização do utilizador com a maior precisão e ubiquidade possível, em todo o tipo de ambiente, quer na rua, quer no interior de edifícios.

8. BIBLIOGRAFIA

Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). Web services *Web Services* (pp. 123-149): Springer.

Android. (2018a, 2018). TalkBack. *Android Developers.* Retrieved from https://support.google.com/accessibility/android/answer/6283677?hl=en

Android. (2018b, 2018). UUID. *Android Developers.* Retrieved from https://developer.android.com/reference/java/util/UUID

Bekkelien, A., Deriaz, M., & Marchand-Maillet, S. (2012). Bluetooth indoor positioning. *Master's thesis, University of Geneva*.

Belqasmi, F., Glitho, R., & Fu, C. (2011). RESTful web services for service provisioning in next-generation networks: a survey. *IEEE Communications Magazine, 49*(12).

Chopde, N. R., & Nichat, M. (2013). Landmark based shortest path detection by using A* and Haversine formula. *International Journal of Innovative Research in Computer and Communication Engineering, 1*(2), 298-302.

Erl, T. (2004). Service-oriented architecture: a field guide to integrating XML and Web services/Thomas Erl: Prentice Hall PTR.—2004.—536 p.

Faheem, A., Virrankoski, R., & Elmusrati, M. (2010, 2010). Improving RSSI based distance estimation for 802.15. 4 wireless sensor networks.

Faragher, R., & Harle, R. (2015). Location fingerprinting with bluetooth low energy beacons. *IEEE journal on Selected Areas in Communications, 33*(11), 2418-2428.

Fernandes, L. M. A. (2017). Wearable devices for blind orientation and navigation.

GraphHopper. (2018, 2018). GraphHopper. *GraphHopper Directions API.* Retrieved from https://www.GraphHopper.com/

Herrera Vargas, M. (2016). Indoor navigation using bluetooth low energy (BLE) beacons.

Hofmann-Wellenhof, B., Lichtenegger, H., & Collins, J. (2012). *Global positioning system: theory and practice*: Springer Science & Business Media.

Juels, A. (2006). RFID security and privacy: A research survey. *IEEE journal on Selected Areas in Communications, 24*(2), 381-394.

Julier, S. J., & Uhlmann, J. K. (1997, 1997). New extension of the Kalman filter to nonlinear systems.

Kajioka, S., Mori, T., Uchiya, T., Takumi, I., & Matsuo, H. (2014). Experiment of indoor position presumption based on RSSI of Bluetooth LE beacon - IEEE Conference Publication.

Kim, B., Bong, W., & Kim, Y. C. (2011, 2011). Indoor localization for Wi-Fi devices by cross-monitoring AP and weighted triangulation.

Laftsidis, A. (2000). Enterprise Application Integration. *IBM Zweden*.

Linthicum, D. S. (2003). Next generation application integration: from simple information to Web services: Addison-Wesley Longman Publishing Co., Inc.

Mapsforge. (2013, 2013). Mapsforge. Retrieved from https://github.com/mapsforge/mapsforge

NiT. (2017, 2017). Aplicação Descubra distinguida pelos Prémios Navegantes XXI. *NiT*. Retrieved from https://nit.pt/out-of-town/back-in-town/aplicacao-descubra-distinguida-premios-navegantes-xxi

Osmdroid. (2018, 2018/08/02/T12:48:30Z). OSMDroid. Retrieved from https://github.com/osmdroid/osmdroid

Pai, M. V. (2014). RESTful Web Services.

RadiusNetworks. (2018, 2018). Android Beacon Library. Retrieved from https://altbeacon.github.io/android-beacon-library/

Sheppard, P. (2018). Adults' Media Use and Attitudes Report. 219.

Statista. (2018, 2018). Number of Google Play Store apps 2018 | Statistic. *Statista*. Retrieved from https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

Yin, C., & Wang, H. (2010, 2010). Developed Dijkstra shortest path search algorithm and simulation.

Zhang, Z., & Zhao, Z. (2014). A multiple mobile robots path planning algorithm based on A-star and Dijkstra algorithm. *International Journal of Smart Home*, *8*(3), 75-86.