Universidade do Minho
Escola de Engenharia

Ivo Miguel Menezes Silva

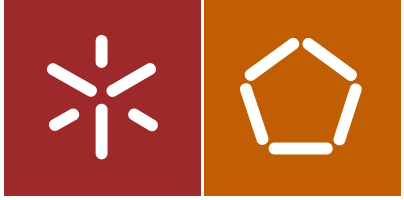
Construção de mapas de rádio para sistemas de posicionamento WiFi

julho de 2016

Construção de mapas de rádio para
sistemas de posicionamento WiFi

Ivo Miguel Menezes Silva

UMinho | 2016



Universidade do Minho
Escola de Engenharia

Ivo Miguel Menezes Silva

Construção de mapas de rádio para sistemas de posicionamento WiFi

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de
Mestre em Engenharia de Telecomunicações e Informática

Trabalho efetuado sob a orientação de
Professor Doutor Adriano Jorge Cardoso Moreira
Professor Doutor Filipe Miguel Lopes Meneses

julho de 2016

AGRADECIMENTOS

Dedico este espaço a todos que me ajudaram, direta ou indiretamente a cumprir os meus objetivos e a terminar esta etapa da minha formação académica. Gostaria de destacar o valor enorme que esta experiência teve para mim, tanto a nível pessoal como a nível profissional.

Gostaria de começar por agradecer aos meus orientadores, o Professor Doutor Adriano Moreira e o Professor Doutor Filipe Meneses, pela sua dedicação, orientação e disponibilidade. Fico grato pelo acompanhamento e ajuda que me deram no decorrer deste trabalho assim como pelas opiniões e críticas que ajudaram a valorizar e tornar este trabalho mais rico. Fico agradecido por ter trabalhado sob orientação de ambos porque me tornei melhor pessoa e profissional graças a eles.

Agradeço aos amigos e colegas que me acompanharam e motivaram ao longo do curso, em particular ao Ricardo Peixoto e ao Ricardo Mesquita pelo companheirismo, amizade e apoio que me deram ao longo desta etapa.

Agradeço sobretudo à Vera pela amizade, amor e suporte. Fico especialmente grato pela sua companhia nos bons e maus momentos e pelas palavras de incentivo ao longo do meu percurso universitário.

Para terminar gostaria de agradecer aos meus pais, Valdemar e Violeta por toda a sua dedicação, motivação, paciência e pela educação e valores que me transmitiram. Ao meu irmão, André, agradeço a sua amizade e ajuda na superação de obstáculos que foram surgindo ao longo do meu percurso académico.

RESUMO

A utilização de dispositivos móveis em ambientes empresariais e académicos proliferou pelo mundo inteiro contribuindo para a criação de aplicações nas mais diversas áreas. Há situações em que o conhecimento sobre o local onde o utilizador se encontra traz várias vantagens no domínio aplicacional. Há muitos anos que a localização é explorada em aplicações de navegação ou na localização de objetos e pessoas. A localização de dispositivos móveis em ambientes interiores poderá ter relevância para a monitorização e gestão de recursos humanos ou mesmo para o melhoramento do mercado para o consumidor. Normalmente, a tecnologia utilizada na obtenção da localização é o GPS, mas em ambientes *indoor* esta tecnologia não apresenta bons resultados.

Nos últimos anos observaram-se bastantes desenvolvimentos nas tecnologias de posicionamento que exploram as infraestruturas já existentes nos edifícios. Uma das tecnologias de posicionamento em ambientes interiores mais utilizada é o Wi-Fi *fingerprinting*, que usufrui das redes Wi-Fi para obter informações que permitem associar um local aos níveis de sinal rádio dos pontos de acesso existentes nesse local. Como estes sistemas oferecem soluções de baixo custo que permitem a localização de objetos e/ou pessoas, esta é uma área que tem obtido muito interesse por parte de empresas e investigadores. Por outro lado, surgem outros desafios que se devem à escalabilidade, por exemplo quando se trata de um edifício de grande dimensão.

O objetivo principal desta dissertação é desenvolver uma solução, baseada numa aplicação para *tablets*, que facilite a construção de mapas de rádio para sistemas de posicionamento de interiores recorrendo a técnicas de Wi-Fi *fingerprinting* com o intuito de facilitar o mapeamento de um edifício. Além disso espera-se que a aplicação forneça informação relativa à qualidade do mapa de rádio à medida que este vai sendo construído. Nesta dissertação é feito o estudo do estado da arte e a análise do problema. Posteriormente, é apresentada uma solução, a descrição da mesma assim como a discussão dos resultados obtidos a partir dos testes realizados.

Também se pretende desenvolver uma solução para a visualização da posição de utilizadores de um sistema de posicionamento de interiores. Espera-se que esta solução apresente a localização dos utilizadores ao nível dos espaços interiores de um edifício. A arquitetura e o desenvolvimento desta solução são descritos nesta dissertação assim como os testes realizados.

ABSTRACT

The use of mobile devices in business and academic environments has proliferated across the world contributing for the development of mobile applications in many areas. In the application domain, the knowledge of the user's location can bring several advantages. The localization has been explored for many years in navigation apps or in locating objects and people. The indoor localization in indoor environments is relevant for human resource monitoring and management or even to improve consumer's market. Usually, GPS is the most used technology to obtain the location, but in indoor environments this technology is not reliable.

Over the recent years, there has been many developments in positioning technologies that explore the already existing building infrastructures. Wi-Fi fingerprinting is one of the most used indoor positioning technologies which uses Wi-Fi networks to obtain information that allows to link the existing radio signal levels from access points to that place. Since these systems offer low cost solutions allowing to locate people and/or objects, this is an area that has got a lot of interest from companies and researchers. On the other hand, there are challenges due to scalability, for example when the building is large.

The main objective is to develop a solution based on a *tablet* application that facilitates radio map construction for indoor positioning systems using Wi-Fi fingerprinting techniques in order to simplify the building's mapping. Furthermore, it is expected that the application provides feedback regarding the quality of the radio map as it's being built. The state of the art study and the problem analysis is made in this thesis. Subsequently, a solution and its description is presented, as well as the discussion of the obtained results from the tests.

Also, it is intended to develop a solution for viewing the position of users of an indoor positioning system. It is expected that this solution presents the users' location at the level of the building's indoor spaces. Both the architecture and the development of this solution are described in this thesis as well as the carried tests.

ÍNDICE DE CONTEÚDOS

LISTA DE SIGLAS E ACRÓNIMOS

AP	*Access Point*
API	*Application Programming Interface*
BD	Base de Dados
BSSID	*Basic Service Set Identification*
CoO	*Cell of Origin*
CRUD	*Create Read Update Delete*
DES	*Data Encryption Standard*
GPS	*Global Positioning System*
HTTP	*Hypertext Transfer Protocol*
IPS	*Indoor Positioning System*
JSON	*JavaScript Object Notation*
LIFO	*Last In First Out*
MAC	*Media Access Control*
OSM	*OpenStreetMap*
PDO	*PHP Data Objects*
PHP	*Personal Home Page*
PoA	*Phase Of Arrival*
REST	*Representational State Transfer*
RFID	*Radio Frequency Identification*
RSS	*Received Signal Strength*
RSSI	*Received Signal Strength Indicator*
RToF	*Round-Trip Time of Flight*
RTT	*Round-Trip Time*
SLAM	*Simultaneous Localization and Mapping*
SMTP	*Simple Mail Transfer Protocol*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
SSID	*Service Set Identifier*
SSL	*Secure Sockets Layer*
TCP	*Transmission Control Protocol*

TDoA	*Time Difference of Arrival*
ToA	*Time of Arrival*
UDP	*User Datagram Protocol*
UI	*User Interface*
UM	Universidade do Minho
URI	*Uniform Resource Identifier*
UWB	*Ultra-Wide Band*
Wi-Fi	*Wireless Fidelity*
WLAN	*Wireless Local Area Network*
XML	*Extensible Markup Language*

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1 – Introdução

1.1. Enquadramento e motivação

A utilização de dispositivos móveis tem crescido bastante nos últimos anos e hoje pode-se verificar a presença dos *smartphones* e *tablets* no nosso quotidiano. As aplicações para estes aparelhos fornecem diversas funcionalidades e facilidades. A grande maioria dos dispositivos móveis são providos de módulos GPS (*Global Positioning System*) que obtêm a localização com recurso a uma constelação de satélites resultando nas coordenadas geográficas. Há várias aplicações que utilizam o GPS para fazer a localização num espaço interior, mas este sistema não apresenta bons resultados uma vez que o sinal GPS falha em ambientes interiores. É aqui que as técnicas para posicionamento em ambientes interiores têm relevância porque normalmente utilizam infraestruturas já existentes nos edifícios [1].

Há uma quantidade elevada de redes Wi-Fi instaladas que fornecem cobertura especialmente em espaços urbanos. As redes Wi-Fi podem ser encontradas nos mais diversos locais, em instituições públicas, hospitais, escolas, aeroportos e até mesmo em residências. As redes Wi-Fi fornecem várias funcionalidades no âmbito de rede local, mas também de acesso ao exterior, à Internet. Com a vulgarização destas redes existe uma elevada quantidade de utilizadores que fazem uso dos serviços fornecidos pelas mesmas. Os pontos de acesso criam assinaturas rádio que são distintas em cada local de um edifício, permitindo associar a um local uma assinatura rádio específica [1].

Há vários sistemas que permitem obter a localização *indoor* apoiados nas mais diversas tecnologias [1][2][3][4]. Há sistemas baseados em *beacons* que utilizam sinais de pontos de acesso 802.11 [5], mas também se podem usar dispositivos Bluetooth, torres celulares GSM, transmissores rádio FM e combinações entre estes [2][6]. Os sistemas de *fingerprinting* como o RADAR [3] utilizam a radiofrequência para obter a localização e registo de movimentação em ambientes interiores; isto é feito a partir do mapa que é constituído pela coleção de *fingerprints* obtidas em cada local em que se fez a medição dos níveis de sinal dos APs existentes.

A técnica Wi-Fi *fingerprinting* baseia-se em estruturas Wi-Fi já existentes, no entanto, implica a construção de mapas de rádio através da análise dos níveis de sinal em cada local do espaço interior. Como a análise é feita aos sinais rádio em cada local do espaço interior, é possível obter uma *fingerprint* que relaciona os níveis de sinal ao local em questão [5].

Nos sistemas baseados em *fingerprinting*, como é o caso do RADAR, o processo de localização é composto por duas fases distintas: a fase de calibração e a fase *real-time* da qual

resulta a localização estimada. A fase de calibração (*offline*) é aquela na qual são recolhidos os dados relativos às potências recebidas em cada ponto, organizados e armazenados devidamente. A outra fase denomina-se de *real-time*, onde são recolhidos os dados relativos às potências do sinal em tempo real. É feita uma comparação entre os dados obtidos em tempo real e os dados recolhidos na fase de calibração através de um algoritmo que resulta nas coordenadas estimadas do utilizador/dispositivo.

No mercado já existem vários produtos que incluem os sistemas de posicionamento de interiores (IPS) em aplicações móveis [7]–[10]. A introdução desta tecnologia veio trazer várias facilidades, como por exemplo a navegação dentro de um edifício e a monitorização e segurança de pessoas e objetos. Estas facilidades têm aplicação em hospitais (navegação, monitorização e controlo de pacientes), estádios (navegação e segurança), centros comerciais (navegação, segurança, monitorização de crianças), universidades, aeroportos (navegação e segurança), há um leque vasto onde as soluções aplicacionais nesta área terão funcionalidade.

Um dos principais desafios nos sistemas de posicionamento de interiores baseados em *fingerprinting* prende-se com a construção dos mapas de rádio, ou seja, no processo de calibração, principalmente quando se trata de um edifício de grande escala.

O interesse nesta área comprova-se quando se verifica que as maiores empresas tecnológicas do mundo estão a explorar as tecnologias de posicionamento em ambientes interiores, nomeadamente a Google [11], Apple e a Microsoft.

1.2. Objetivos

O objetivo deste trabalho é o desenvolvimento de uma solução eficaz para a construção dos mapas de rádio, destinado a suportar um sistema de posicionamento de interiores com capacidade para realizar as duas fases de um sistema baseado em *fingerprinting*, a fase de calibração (*offline*) e a fase *real-time*. O intuito do sistema passa por facilitar a construção de mapas de rádio de um edifício através da recolha de *fingerprints* em cada espaço interior. A fase de calibração, também conhecida como fase de treino, é uma tarefa cansativa principalmente quando se trata de um edifício de proporções elevadas, a solução desenvolvida terá um impacto positivo facilitando nesta tarefa. A fase *real-time,* em combinação com um motor de posicionamento, permitirá verificar a posição do utilizador/dispositivo fazendo a comparação entre dos dados recolhidos previamente na fase de calibração com os dados obtidos em tempo real.

Uma aplicação para *tablets* é ideal para apresentar o *layout* do edifício para a construção do mapa de rádio e controlo do estado atual do mesmo. Tendo em consideração o processo de calibração, a otimização para *tablets* facilita a interação com o utilizador.

Pretende-se dar resposta aos problemas e às questões que surgirão ao longo do desenvolvimento da dissertação, complementando com o trabalho de investigação inicial que será relevante para o contexto da construção de mapas de rádio recorrendo a técnicas de Wi-Fi *fingerprinting*.

1.3. Abordagem

De modo a alcançar os objetivos propostos anteriormente é necessário definir um conjunto de tarefas que permitam desenvolver o projeto de uma forma progressiva para que no fim seja possível obter os resultados esperados. Das tarefas que serão realizadas neste trabalho distinguem-se as seguintes:

1. Estudo inicial das tecnologias e sistemas de posicionamento, através da recolha bibliográfica de forma a aprender as principais técnicas de posicionamento, os sistemas que já foram desenvolvidos, e as principais soluções disponíveis no mercado nos dias de hoje. É importante fazer uma abordagem com mais detalhe nos sistemas e aplicações móveis que realizam a calibração dos espaços interiores tendo em atenção o modo como são utilizados num contexto real.
2. Depois do estudo do Estado da Arte, será necessário perceber quais são as questões associadas ao trabalho que será desenvolvido, passando pela análise das soluções disponíveis e compreender como resolver alguns dos problemas associados aos sistemas de posicionamento de interiores. Esta tarefa aborda um estudo às soluções para mapas de interiores e são levantadas várias questões sobre o processo de calibração. Sabendo que não existe um *standard* para plantas de interiores, o estudo e análise das soluções disponíveis é determinante para o trabalho desenvolvido.
3. Determinar a solução adequada para o problema as plantas dos edifícios com capacidade para múltiplos operadores e utilizadores. Definir o desenho da aplicação de calibração com o intuito de apoiar tarefas de calibração e respetiva implementação. Esta tarefa engloba um estudo inicial das tecnologias que serão utilizadas na implementação, assim como a definição da arquitetura do sistema com suporte para múltiplos operadores e utilizadores. Espera-se a integração da solução das plantas dos edifícios na aplicação de

calibração. A importância da calibração está associada à qualidade do mapa de rádio e à informação que poderá ser disponibilizada pela aplicação para a análise do mapa de rádio. Após a definição do desenho do sistema, é iniciado o processo de implementação da arquitetura definida utilizando as tecnologias escolhidas inicialmente. No final, o sistema de calibração será alvo de vários testes para garantir que são obtidos os resultados esperados, para verificar a performance da aplicação e para detetar possíveis falhas. O desenvolvimento de testes num contexto real também será imprescindível para se fazerem alguns ajustes caso surjam pequenos detalhes que necessitem alterações. Além disso a experimentação em ambiente real permitirá fazer a validação da solução.

4. Desenho da componente de visualização tendo em conta os requisitos definidos anteriormente. Esta componente será um módulo integrado na aplicação Where@UM que utiliza a solução para as plantas dos edifícios. Inicialmente é relevante fazer um levantamento de cada elemento da componente de visualização. Depois é feita a implementação tendo em conta o desenho feito inicialmente e as tecnologias definidas. Tal como na tarefa 3, são feitos testes de desenvolvimento no módulo integrado na app Where@UM. As tarefas de implementação têm especial destaque porque é delas que depende a solução final. Depois da implementação e desenvolvimento do sistema, a avaliação do mesmo permite verificar o que foi desenvolvido ao longo do tempo e determinar como é que o sistema desenvolvido se aproxima dos objetivos propostos.
5. Redação da dissertação. É uma tarefa paralela a todas as outras, uma vez que se trata de um trabalho progressivo que é desenvolvido no domínio da dissertação. Assim acompanhando todas as tarefas é feita a documentação de forma concreta e detalhada. A dissertação é o documento que apresenta o desenvolvimento do trabalho, tarefa-a-tarefa.

1.4. Estrutura da Dissertação

Esta dissertação divide-se em sete capítulos. Partindo do primeiro capítulo onde esta secção se encontra inserida, é feita uma introdução à dissertação, passando pelo enquadramento, objetivos e abordagem.

No segundo capítulo é feita a introdução ao problema através da introdução e do estudo bibliográfico (Estado da Arte). A base do estudo posterior está dependente da análise feita no segundo capítulo, onde é feita a descrição das técnicas de posicionamento, de sistemas de posicionamento de interiores e de produtos comerciais existentes no mercado.

O terceiro capítulo aborda o estudo do problema onde são analisados os principais problemas relativos ao processo de construção dos mapas de rádio, aos quais se pretende dar resposta com o trabalho desenvolvido nesta dissertação.

Iniciando a fase de desenvolvimento, no quarto capítulo é feita uma descrição detalhada relativamente à solução para as plantas dos edifícios que será implementada. Neste capítulo surge a análise do desenho e implementação da solução para as plantas dos edifícios.

O quinto capítulo engloba todo o sistema de calibração, desde a sua conceção com o desenho do sistema, até à sua implementação. Este capítulo descreve detalhadamente o funcionamento do sistema de calibração, cada componente e o modelo de comunicação. O capítulo termina com uma análise à construção de um mapa de rádio através de uma experiência no mundo real.

O penúltimo capítulo aborda a componente de visualização integrada no sistema Where@UM. Além da nova arquitetura deste sistema, são apresentadas alterações em cada constituinte do sistema passando pelos modelos de dados e os serviços que permitirem a comunicação entre componentes. Por fim, são apresentados os testes realizados e a análise ao funcionamento da aplicação onde são testadas todas as funcionalidades.

As conclusões e trabalhos futuros são expostos no sétimo capítulo, passando por uma análise ao trabalho desenvolvido e aos aspetos que poderão ser melhorados no futuro.

CAPÍTULO 2 – Estado da Arte

Neste capítulo são revistos os sistemas de posicionamento dando maior destaque aos sistemas que se baseiam em tecnologias Wi-Fi. As técnicas de localização *indoor* também serão abordadas e descritas de uma forma breve, sendo que a técnica de *fingerprinting* terá mais atenção. Alguns dos produtos comerciais presentes no mercado também serão apresentados e descritos tendo em consideração as tecnologias que utilizam. Por fim os produtos que apresentam soluções parecidas com a solução deste projeto serão discutidos e analisados.

2.1. Introdução

Como foi possível verificar anteriormente, há vários sistemas de posicionamento de interiores baseados em diferentes tecnologias. O sistema de posicionamento mais comum é o GPS, que apesar de ter inúmeras aplicações, em ambientes *indoor* este não é considerado útil, e por isso recorre-se a outras tecnologias, tais como a tecnologia Bluetooth [12], sistemas baseados em RFID [13][14], infravermelhos (IR) [15], redes sem fios Wi-Fi [1][3][4][5][16], redes de sensores [17], sistemas UltraWideBand (UWB) com largura de banda elevada [18], ultrassons, sistemas baseados em visão e ainda campos magnéticos [19]. O elevado número de sensores permite a existência de vários esquemas de localização entre os quais se podem destacar a triangulação, *trilateration*, correspondência de dados, entre outras [20].

Por outro lado existem outros sistemas de posicionamento de interiores que combinam várias tecnologias para obterem melhores resultados como se pode verificar em [15][19][20][21], onde se podem verificar várias comparações entre as diferentes técnicas e são analisadas as técnicas com melhores resultados.

Na Figura 2.1 estão representados os sistemas de posicionamento de acordo com a sua resolução e a escala.

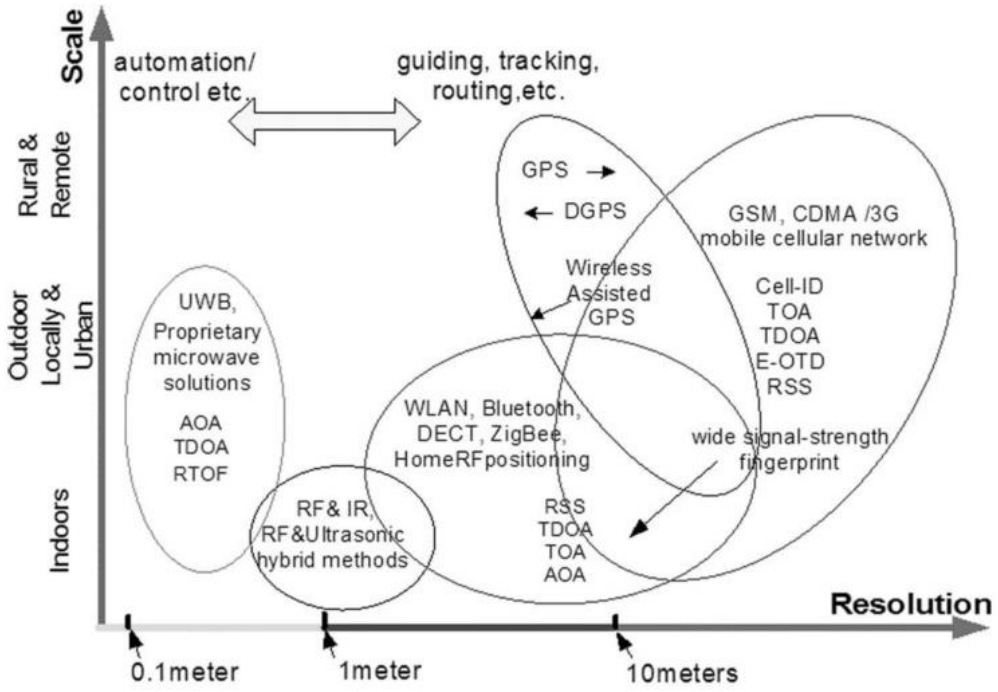


Figura 2.1: Comparação entre vários sistemas de posicionamento baseados em tecnologias sem fios [15]

Neste trabalho a tecnologia mais relevante é a de redes sem fios Wi-Fi, que será a tecnologia utilizada no sistema de posicionamento alvo deste trabalho.

Um mapa de rádio Wi-Fi é uma base de dados que contém as *fingerprints* (muitas vezes recolhidas manualmente, como será o caso neste projeto) nas diversas localizações. Os sistemas de posicionamento, também conhecidos como motores de posicionamento, comparam a *fingerprint* Wi-Fi recolhida pelo dispositivo móvel com o conjunto de *fingerprints* previamente recolhidas no processo manual do qual resultou o mapa de rádio. A localização do dispositivo móvel é estimada por similaridade ou pela função de melhor correspondência, e como cada registo está associado ao nome da localização ou às coordenadas, o sistema de posicionamento pode fornecer a localização do utilizador [22].

Neste trabalho o foco será na construção inicial dos mapas de rádio com recurso a um dispositivo móvel, facilitando bastante o trabalho manual que é feito inicialmente.

2.2. Técnicas de posicionamento

Já foram referidas as tecnologias que são utilizadas nos sistemas de posicionamento *indoor*, para obter a localização estes sistemas recorrem a técnicas de posicionamento para

determinar a localização do utilizador/dispositivo. Cada técnica utiliza métricas diferentes para obter a estimação da localização.

2.2.1. *Fingerprinting* ou *Scene Analysis*

Scene analysis refere-se ao tipo de algoritmo que primeiramente recolhe *fingerprints* (impressões digitais) de um espaço e depois estima a localização de um objeto fazendo a correspondência dos dados com as *fingerprints* obtidas *a priori*. A técnica de *scene analysis* [15] fundamenta-se na ideia de cada espaço ter associado um conjunto de sinais rádio únicos que o permitem distinguir de outros locais. A *fingerprint* é conhecida como uma associação de uma medição única a um local. Podem ser associadas várias *fingerprints* ao mesmo local. Esta técnica consiste em duas fases: "treino", é a fase *offline* também conhecida como fase de calibração, e a fase de "posicionamento" ou fase *real-time* ou *online*.

A técnica de localização *Received Signal Strength* (RSS) que usa como característica a potência do sinal recebido é a técnica mais utilizada em sistemas que utilizam o *scene analysis*.

Na primeira fase é feita a construção de uma base de dados de *fingerprints*, também conhecida como mapa de rádio. Para construir a base de dados é preciso selecionar os diversos pontos de referência onde serão medidos os níveis de sinal dos pontos de acesso. Em cada sala/espaço convém recolher várias *fingerprints* em vários pontos do local de forma a tornar o sistema mais robusto, uma vez que a estrutura da própria sala afeta os sinais rádio. A fase de calibração é uma tarefa penosa por si só e torna-se bastante complicada de realizar quando se pretende fazer a construção do mapa de rádio de um edifício de elevadas dimensões através da recolha de *fingerprints* em todos os locais.

A principal dificuldade nas técnicas de localização baseadas em *fingerprinting* é o nível/força do sinal recebido que pode ser afetado por difração, reflexão e *scattering* na propagação em ambientes interiores. Nos edifícios, podem ser adicionados móveis, as suas posições podem ser alteradas assim como podem ser adicionados ou removidos pontos de acesso. Os sinais dos pontos de acesso são variáveis, por isso é necessário manter a base de dados atualizada para que os resultados sejam melhores. A construção e atualização dos mapas de rádio pode ser feita através de um processo colaborativo [23] ou automático [5], os quais facilitam bastante o processo da construção e atualização de mapas de rádio. O principal intuito desta dissertação é desenvolver uma solução que facilite o processo da recolha de *fingerprints*.

O nível de detalhe da fase de treino está associado à precisão e acuidade dos sistemas que utilizam *fingerprinting*, quanto mais detalhada for a fase de calibração, melhores serão os resultados obtidos.

Na fase *real-time*, o *software* de cálculo recebe periodicamente os valores obtidos por um ou mais dispositivos móveis. Esta informação é comparada com os valores da base de dados armazenados na fase *offline* permitindo que a localização seja determinada.

Podem ser listados cinco algoritmos de posicionamento para as técnicas baseadas em *fingerprinting* [15]: métodos probabilísticos, redes neurais, *k-nearest-neighbor* (*k*NN), *support vector machine* (SVM), e ainda *smallest* M-*vertex polygon.*

2.2.2. Triangulação – lateração e angulação

Utilizada em várias tecnologias, a triangulação utiliza as propriedades geométricas dos triângulos para obter a localização dispositivo alvo. Conhecem-se duas variantes: a lateração e a angulação. A lateração estima a posição de um objeto medindo a distância do alvo a vários pontos de referência. Por sua vez, a angulação localiza um objeto fazendo a computação dos ângulos relativos aos diversos pontos de referência.

As formas de medir a distância ao ponto de referência são:

- *Time of Arrival* (TOA), definido pela distância que vai desde o terminal móvel até à unidade de medida diretamente proporcional ao tempo de propagação, ou seja, o TOA mede o tempo de propagação de um sinal entre o emissor e o recetor;
- *Time Difference of Arrival* (TDOA), a ideia principal é examinar a diferença temporal na chegada de dois ou mais sinais a um dispositivo para estimar a posição relativa do mesmo;
- RSS-Based ou *Signal Attenuation-Based*, utiliza a atenuação que afeta o sinal emitido como medida para estimar a distância a cada ponto de referência e depois aplicar o algoritmo do qual resulta a localização;
- *Roundtrip Time of Flight* (RTOF) é um método que mede o tempo de ida e volta do sinal a deslocar-se desde o emissor até ao dispositivo recetor, sabendo exatamente o tempo de ida e volta é possível estimar a distância;
- *Phase of Arrival* (POA) ou método de fase do sinal recebido, determina a diferença de fase partindo do principio que todas as fontes emitem sinais sinusoidais com a mesma frequência *f* e uma fase igual a zero, fazendo a diferença entre a onda emitida e a onda

recebida num dispositivo recetor, é possível calcular qual será a distância a que está o dispositivo.

2.2.3. Map Matching (MM)

A técnica de *map matching* associa a posição do dispositivo a um mapa digital. O principal objetivo deste algoritmo é identificar o segmento onde o objeto se movimenta e determinar a localização do objeto nesse segmento. Os algoritmos de *map matching* baseiam-se na teoria do reconhecimento de padrões. Essencialmente é feita uma comparação da localização do dispositivo com o mapa digital. Os sistemas de GPS são os que mais utilizam esta técnica principalmente nos veículos acompanhando o seu movimento relativamente aos mapas digitais.

Normalmente, a técnica de *map matching* é utilizada em conjunto com outra técnica de posicionamento com o intuito de melhorar o desempenho desta última.

2.2.4. Dead Reckoning (DR)

O principal objetivo da *dead reckoning* é determinar a localização partindo de um ponto fixo (posição determinada previamente) e estimar a localização recorrendo à da velocidade (real ou estimada), tempo e direção do movimento. Como o ponto de partida é fixo, os erros associados à estimação da velocidade e direção, que são cumulativos ao longo do tempo, fazem com que o erro na estimativa da posição vá aumentando também ao longo do tempo. Esta técnica é particularmente útil em sistemas com GPS que por vezes perdem sinal e a técnica DR permite manter a localização atualizada enquanto o sinal GPS não é recuperado.

Utiliza-se muito *dead reckoning* em conjunto com a técnica MM uma vez que a combinação de ambos oferece maior acuidade na localização.

2.2.5. Cell of Origin (CoO)

CoO tal como o nome sugere, identifica a célula de origem à qual o terminal móvel está ligado quando tem acesso à rede móvel GSM. Esta técnica não é muito precisa uma vez que depende do número de estações base de uma área, o que representa uma acuidade que pode variar de 100 m em zonas urbanas até 30 km em zonas pouco povoadas. Por outro lado, tem várias vantagens, uma vez que identifica rapidamente a localização e não necessita de quaisquer *upgrades* de rede.

Quando a precisão é moderadamente importante, utiliza-se CoO combinado com outras tecnologias como o GPS por exemplo.

2.3. Sistemas de posicionamento

Depois de se abordarem os conceitos essenciais relativos às técnicas de posicionamento, serão analisados alguns sistemas de posicionamento de interiores, tendo em especial atenção os sistemas que se baseiam em Wi-Fi *fingerprinting* uma vez que é esta a técnica que será abordada neste trabalho.

2.3.1. Radar

Tal como foi referido anteriormente, o sistema RADAR é um sistema de posicionamento *indoor* desenvolvido por Paramvir Bahl e Venkata N. Padmanabhan [3], que faz a localização e registo de movimentação de utilizadores dentro de edifícios.

O RADAR é um sistema baseado em *fingerprints* de sinais rádio obtidas em diversos locais que depois são utilizadas para determinar as coordenadas do utilizador.

Tal como foi descrito previamente este sistema inclui duas fases distintas, a fase de calibração e a fase *real-time.* São recolhidos dados nas duas fases, por parte da estação base na forma (*t*, *bs*, *ss*) em que *t* é o *timestamp* da recolha, *bs* é a estação base que faz o registo da força do sinal (*ss*). Por defeito são feitas 4 recolhas por segundo. As estações base recebem os sinais por parte do dispositivo do utilizador e registam a força do sinal.

Na fase de calibração, o utilizador indica a sua posição clicando no mapa do piso em que está. As coordenadas (t, x, y, d) são registadas, onde *t* é o *timestamp* do momento em que se fez o registo, por sua vez (x, y) são as coordenadas cartesianas da posição no mapa, enquanto *d* é a direção (que pode tomar os seguintes valores: norte, sul, este, oeste). Para cada registo na fase de calibração foram recolhidas pelo menos 20 amostras da força do sinal. Cada uma das amostras é chamada de *fingerprint.* Desta forma o mapa de rádio do piso é criado, sendo constituído pelo conjunto de todas as amostras recolhidas.

Na Figura 2.2, cada estrela representa as localizações das estações base, os pontos pretos são representações dos locais onde foram recolhidas informações relativamente à potência do sinal.

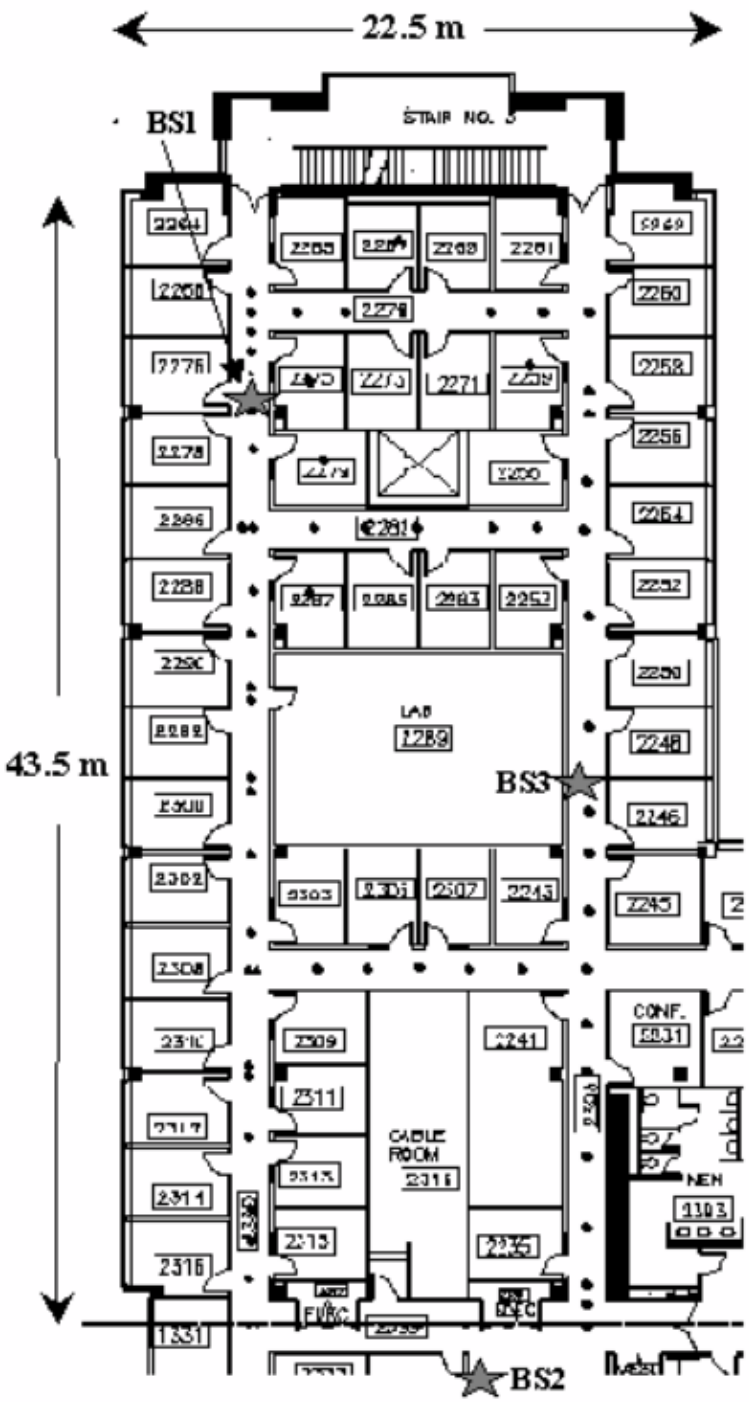


Figura 2.2: Mapa do piso onde se realizaram as experiências [3]

A técnica utilizada para a estimação da localização é baseada em *fingerprinting*. Dado um conjunto de medidas da força do sinal de cada um dos pontos de acesso, determina-se a localização que corresponde da melhor forma aos dados observados da força do sinal.

É aplicado o algoritmo NNSS (*nearest neighbor(s) in signal space*) com o intuito de determinar a distância (em espaço de sinal) entre o conjunto de medidas SS (força do sinal) (ss_i,

ss_2, ss_3) que são os valores da força do sinal de cada ponto de acesso, e o conjunto (ss'_1, ss'_2, ss'_3) que são os valores registados previamente na fase *offline*. Foi utilizada a distância euclidiana dada pela expressão $\sqrt{(ss_1 - ss'_1)^2 + (ss_2 - ss'_2)^2 + (ss_3 - ss'_3)^2}$ para determinar a estimação da localização do utilizador. Foram analisados outros critérios para se fazer a melhor correspondência, sendo que este foi o que apresentou melhores resultados.

O sistema RADAR tem capacidade para determinar a posição do utilizador com um erro médio de 2 a 3 metros, que permite estimar o posicionamento com a acuidade necessárias para várias aplicações em espaço interior.

Uma das desvantagens deste sistema é a necessidade de recalibração sempre que houver alterações no espaço interior.

2.3.2. Herecast

O sistema Herecast [24] utiliza a infraestrutura de pontos de acesso Wi-Fi já existentes num edifício. A sua arquitetura aborda três áreas distintas: modelação e gestão de dados, *software* do dispositivo cliente e possíveis tipos de serviços.

Este sistema tem uma abordagem colaborativa em que os utilizadores introduzem informações relativas a pontos de acesso de modo a associa-los a localizações específicas. Associam-se várias informações a cada ponto de acesso, nomeadamente: a informação sobre a localização do AP, a rede em que o ponto de acesso está ligado e serviços associados ao ponto de acesso. A informação relativa à localização na Figura 2.3 tem a forma: país, distrito/província, cidade, área(s), edifício (com um nome e morada da rua associados ao mesmo). O ponto de acesso é associado ao edifício sendo o piso e a sala atributos opcionais que têm bastante relevância em edifícios maiores com mais do que um ponto de acesso. Os pontos de acesso são adicionados e removidos constantemente, e a informação é atualizada na base de dados por um administrador ou um pequeno grupo de administradores. Por sua vez, os utilizadores podem mesmo atualizar informações sobre um ponto de acesso que não tenha sido descoberto previamente.

Mantendo os dados dos pontos de acesso é possível estimar a localização dos utilizadores fazendo a correspondência com a localização dos pontos de acesso registados no sistema.

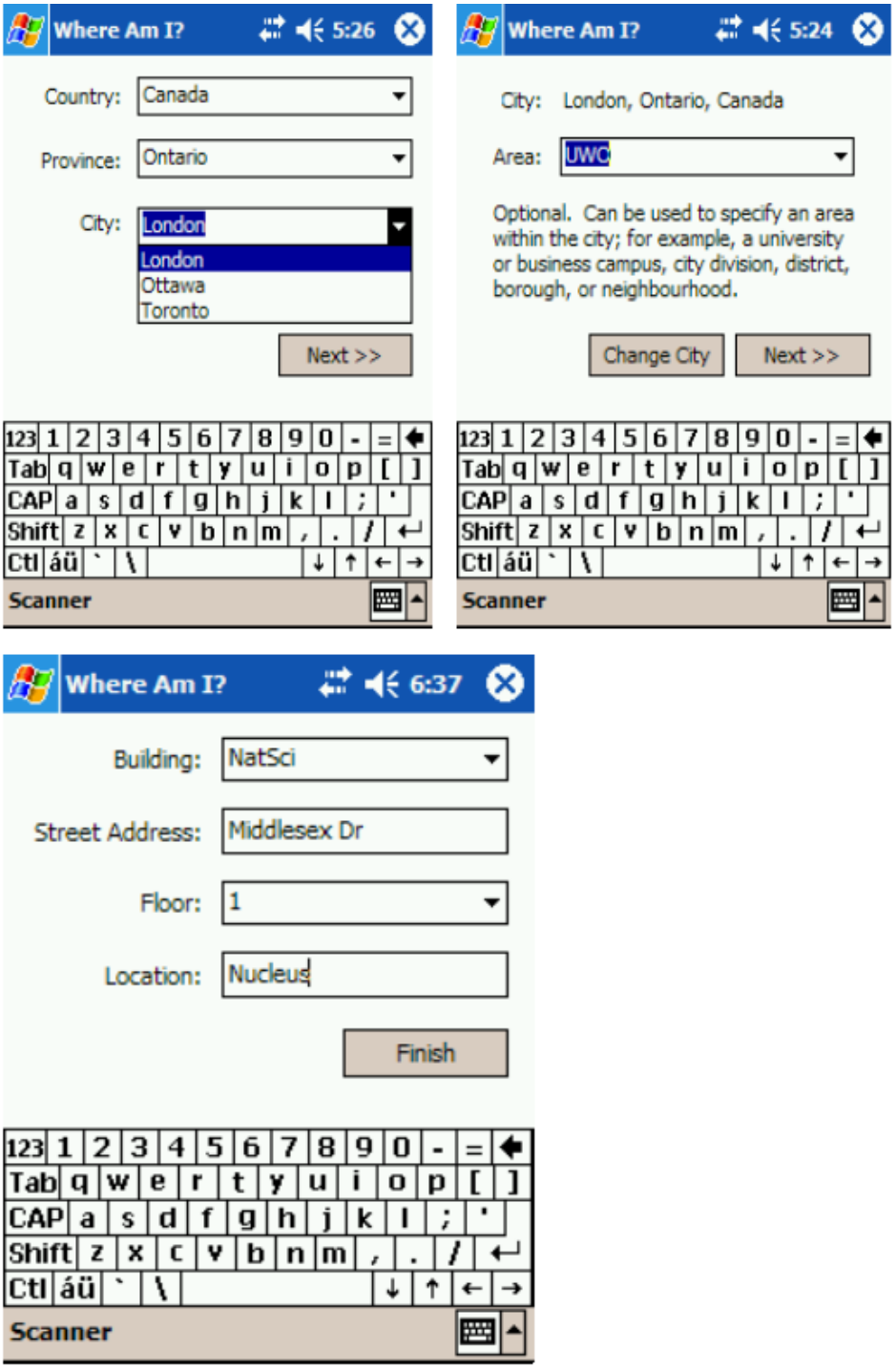


Figura 2.3: Formulários de introdução de um ponto de acesso [24]

Os formulários permitem que o utilizador contribua para o processo de armazenamento das informações relativas a pontos de acesso que não foram descobertos previamente, assim o utilizador participa na construção dos mapas de rádio daquele piso.

Foi desenvolvido *software* para um dispositivo cliente (um PDA). Há uma componente que recolhe o endereço MAC de um AP e o seu *Received Signal Strength Indicator*(RSSI) fornecidos

pela placa de rede 802.11 do dispositivo. A aplicação tem uma componente que armazena todas as informações sobre os pontos de acesso, incluindo uma lista de pontos de acesso. Esta componente contém um método que determina o ponto de acesso com maior RSSI. A comunicação com o servidor é feita via conexão HTTP com pedidos do tipo GET e do tipo POST. Por fim, há uma componente relativa aos dados em cache que tem o intuito de manter a lista de pontos de acesso armazenada no dispositivo. Quando é feito um pedido sobre a informação de localização de um ponto de acesso, o servidor devolve essa informação juntamente com informação acerca de outros pontos de acesso no edifício.

Foram desenvolvidas várias aplicações para o Herecast com finalidades diferentes:

- *Area Maps* – Normalmente os serviços de navegação e de mapas apenas têm a finalidade da navegação em ruas e de carro. Por outro lado, seria relevante ter acesso a um mapa com mais detalhes sobre a zona em questão. Por exemplo, num centro comercial seria relevante ter um mapa com os nomes das lojas. O serviço *Area Maps* permite a produção de mapas detalhados para uma área específica. Os mapas são otimizados para serem observados num ecrã de um Pocket PC e a navegação nos mapas é feita facilmente clicando na imagem.
- *Friend Finder* – é uma aplicação que apresenta a localização atualizada do utilizador num *website.* Quando o utilizador subscreve ao serviço, o dispositivo publica automaticamente a localização do utilizador num *website.* É um serviço de demonstração que teve poucas preocupações na privacidade do utilizador.
- *Heresay* – é uma versão inicial de um serviço que pode ter algumas funcionalidades. Os utilizadores deixam uma mensagem na sua localização atual e veem as mensagens que foram enviadas por outros utilizadores no mesmo edifício.
- *Bandwidth Advisor* – é uma aplicação que orienta o utilizador para os pontos de acesso com mais largura de banda disponível. Se o dispositivo verificar que a rede está congestionada e lenta, ele ira aconselhar proactivamente o utilizador para se deslocar para uma localização com menor utilização. Esta aplicação além de ajudar o utilizador também contribui para o balanceamento do tráfego da rede. A implementação deste serviço requer a recolha das estatísticas de utilização dos pontos de acesso utilizando SNMP.

2.3.3. Place Lab

A premissa deste sistema foca-se na possibilidade de haver uma cobertura global através de um serviço comparável ao GPS, onde as pessoas que possuem dispositivos providos de interface Wi-Fi poderão usufruir de serviços que utilizam a localização, só que neste caso a localização será determinada em ambientes exteriores e interiores.

A iniciativa Place Lab apresenta algumas soluções para os principais obstáculos: a utilização de soluções de baixo custo em tecnologias de posicionamento, deixar os utilizadores confortáveis tendo em atenção a privacidade da sua localização e ter conteúdos *web* facilmente personalizáveis para localizações geográficas [25].

Segundo Anthony LaMarca, *et al.* [2] os utilizadores com dispositivos tais como PDAs, computadores portáteis, telemóveis são localizados com recurso a *beacons* de rádio tais como APs 802.11, torres celulares GSM e dispositivos Bluetooth fixos que já existem no meio. Estas *beacons* apresentam identificadores (IDs) únicos ou semi-únicos, por exemplo, o endereço MAC. O dispositivo está à escuta dos IDs e quando é detetado um ou mais IDs, o cliente computa a sua própria localização verificando os IDs das *beacons* associados aos locais que se encontram num mapa que está armazenado em memória cache.

A arquitetura deste sistema é constituída por *beacons* de rádio, bases de dados que contêm informações relativas à localização dos *beacons* e os clientes Place Lab que estimam a sua localização com recurso à informação disponibilizada pelas bases de dados.

Tal como referido anteriormente, o Place Lab funciona através da escuta das transmissões de fontes de rede tais como APs 802.11, dispositivos Bluetooth fixos, e torres celulares GSM, também chamados de *beacons.* O processo de estimação da localização do cliente é simplificado quando o ID de uma *beacon* é escutado. Os clientes Place Lab não precisam de transmitir dados para determinar a localização, nem precisam de escutar as transmissões de dados de outros utilizadores. Os processos de escuta para cada tipo de *beacon* variam, por exemplo, no caso dos dispositivos Bluetooth é necessário os clientes fazerem um *scan* de modo a encontrar as *beacons* na área de alcance.

As bases de dados têm o papel de fornecer a informação da localização das *beacons* para os dispositivos cliente. Pode haver várias bases de dados, não é especificado se as bases de dados são públicas ou privadas, nem como é que a autenticação entre o cliente e a base de dados é feita e por último não é definido o número de bases de dados das quais o cliente deve carregar os

dados. As bases de dados provêm de diversas instituições tais como universidades, empresas, departamentos que têm conhecimento das localizações de pontos de acesso 802.11, etc.

Por fim, os clientes Place Lab usam observações rádio em tempo real e obtêm a estimação da localização com recurso às localizações das *beacons* em cache. Os clientes Place Lab apresentam três elementos fundamentais: *spotters, mappers,* e *trackers.* Os *spotters*, tal como o nome sugere, são elementos que procuram observar um fenómeno, é criada uma instância *spotter* por cada protocolo rádio suportado pelo dispositivo. Por exemplo, um *smartphone* necessita de um *spotter* para 802.11, um para Bluetooth e outro para GSM. A tarefa de um *spotter* é monitorizar a interface rádio e partilhar os IDs das *beacons* observados com outros componentes do sistema. O *mappar* é o elemento que fornece a localização das *beacons* conhecidas, sendo que esta localização inclui a latitude e longitude, mas pode ter outras informações complementares. Por sua vez, o *tracker* utiliza *streams* de observações do *spotter* e dados associados do *mapper* para estimar a posição do utilizador.

Os resultados experimentais obtidos por este sistema revelam que a cobertura e precisão dependem do número e combinação das *beacons* no meio. Como há mais densidade de dispositivos 802.11 e torres celulares GSM em áreas urbanas, o Place Lab tem melhores resultados nestas áreas que são as que apresentam mais *beacons* contribuindo assim para maior precisão na localização que é estimada. Num ambiente deste género, com densidade suficiente de *beacons* 802.11 o erro na estimação da posição tem uma média que varia entre 15 e 20 metros. Quando apenas são utilizadas *beacons* GSM a média aumenta drasticamente para valores entre 100 e 200 metros.

2.3.4. Redpin

Tendo por base os sistemas Radar e Place Lab, o sistema Redpin [23] é um sistema baseado em *fingerprinting* desenhado e construído para funcionar em telemóveis. Com o Redpin é possível ter em consideração a força de sinais GSM, Bluetooth e pontos de acesso Wi-Fi num telemóvel. Esta abordagem permite que o sistema se adapte às alterações no ambiente, por exemplo quando se adiciona um ponto de acesso.

Aquando da conceção, este sistema tinha vários objetivos: utilizar *hardware* que já toda a gente possui, desenvolver um sistema que não precise de infraestruturas especiais no edifício, com fácil instalação e manutenção, fornecer precisão ao nível de uma sala/escritório e ter capacidade para se adaptar às alterações no meio.

Uma das características deste sistema é que a fase de treino é elaborada pelos utilizadores do sistema de uma forma colaborativa. As informações relativas às localizações são criadas e modificadas pelos utilizadores através de um interface apelativo, mostrado na Figura 2.4.

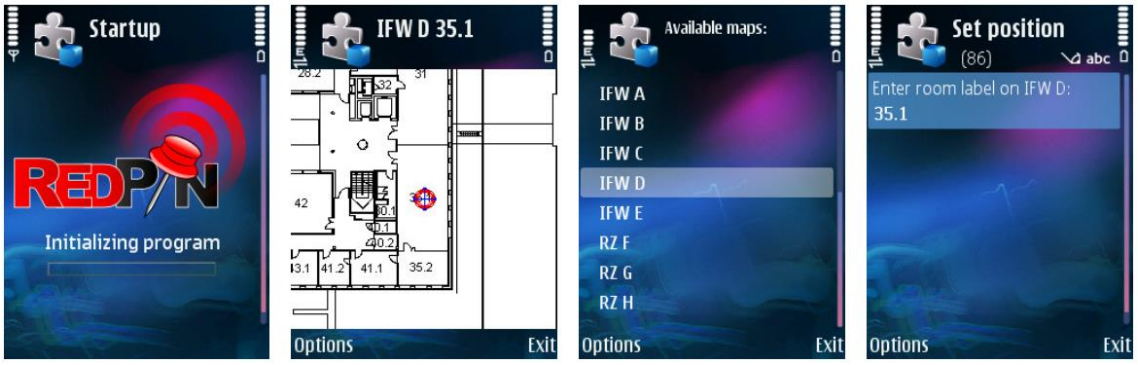


Figura 2.4: Aplicação Redpin - interface de utilizador num Nokia N95 [23]

Para chegar a atingir uma precisão ao nível de uma sala/escritório seleciona-se a localização correta tendo em consideração a força do sinal ativo da célula GSM ativa, a força dos sinais de todos os pontos de acesso Wi-Fi assim como o identificador Bluetooth de todos os dispositivos Bluetooth fixos dentro do alcance.

A arquitetura deste sistema, apresentada na Figura 2.5, tem duas componentes: um *sniffer* que reúne e recolhe informação sobre vários dispositivos sem fios dentro do alcance de forma a criar uma *fingerprint*; um localizador, que armazena as diversas *fingerprints* num repositório e contém o algoritmo para localizar o dispositivo móvel. O *sniffer* necessita de ser executado no dispositivo móvel, mas o localizador pode estar num servidor central ou em cada dispositivo móvel separadamente. Para a solução ser colaborativa entre os utilizadores o servidor central é utilizado para manter os dados fornecidos por todos os utilizadores.

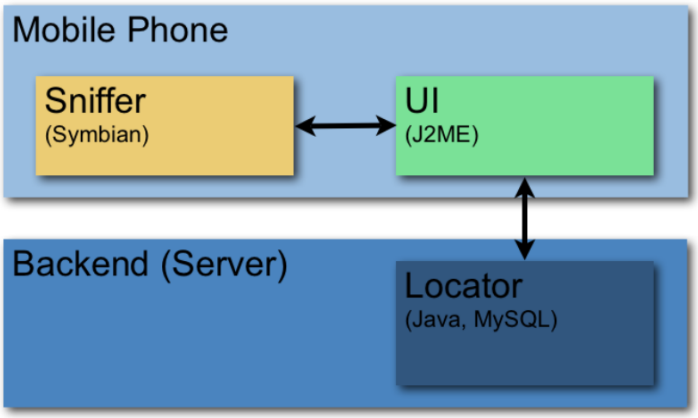


Figura 2.5: Arquitetura do sistema Redpin [23]

O servidor tem com principais funções o armazenamento de *fingerprints* e o serviço que estima a localização do dispositivo móvel através da correspondência da *fingerprint* armazenada com a medição feita no dispositivo móvel.

O *sniffer* faz medições dos sinais GSM e Wi-Fi (que sofrem flutuações), já os dispositivos Bluetooth por vezes não são detetados dentro do tempo de análise o que leva a variações nas medições. As informações dos pontos de acesso Wi-Fi permitem associar o *basic service set identification* (BSSID) ao valor RSS. Os dispositivos Bluetooth são identificados inequivocamente pelo *Bluetooth device address* (BD_ADDR), semelhante ao endereço MAC numa placa de rede.

Para testar a qualidade do sistema foram feitas algumas experiências com vários telemóveis. De modo a testar a taxa de sucesso (quando o sistema é capaz de determinar a localização do dispositivo com sucesso), foram adicionadas à base de dados 26 *fingerprints* de locais escolhidos aleatoriamente. Cada local (sala/escritório) é inferior a 5 por 3 m. Foi utilizado um telemóvel para determinar a localização atual várias vezes ao dia, tanto nas horas de trabalho como à noite, e os resultados demonstram que o sistema é capaz de determinar a localização corretamente em 9 de 10 casos. Tendo em consideração estes resultados, o tempo que demora para se obter pelo menos uma *fingerprint* em cada espaço depende apenas da atividade dos utilizadores e da sua mobilidade. Foi feito um inquérito que demonstra que quando apenas 10 (de um total de 50 pessoas que trabalham naquele piso) contribuem para o sistema, o mapa fica completo em apenas um dia.

Há muitos casos em que apenas é necessária uma *fingerprint* por cada espaço para que o sistema seja capaz de determinar a localização com sucesso, além disso, a mobilidade dos utilizadores permite que sejam obtidas várias *fingerprints* nos mesmos locais permitindo a atualização das mesmas, fazendo assim uma adaptação às alterações do meio.

Mais recentemente foram desenvolvidas aplicações para Android [26] e iOS [27] dando capacidade ao sistema de ser utilizado em diversas plataformas.

2.3.5. COMPASS

O sistema COMPASS [28] é um IPS probabilístico baseado em 802.11 e bússolas digitais. São combinadas as vantagens das infraestruturas WLAN e bússolas digitais para fornecerem um serviço de posicionamento de baixo custo e com boa precisão aos utilizadores portadores de um dispositivo com interface 802.11 e bússola digital.

A direção do utilizador é medida por uma bússola digital para reduzir a influência do bloqueio para o processo de posicionamento. Uma bússola digital é um componente de baixo custo e consumo com tamanho reduzido, uma vez que está integrada num *chip.*

As bússolas digitais permitem que a orientação do utilizador seja detetada e tendo conhecimento desta informação apenas se selecionam as *fingerprints* com a orientação semelhante à do utilizador, que foram obtidas durante a fase *offline,* para determinar a posição do utilizador. O algoritmo probabilístico apenas utiliza este subconjunto de *fingerprints* para estimar a posição do utilizador.

Tanto no sistema RADAR como no sistema COMPASS os seus criadores deram especial atenção ao impacto da orientação do utilizador no registo de movimentação de um utilizador móvel. Como um corpo humano contém mais do que 50% de água, que absorve sinais de rádio de 2.4GHz, o efeito que o corpo humano reproduz influencia a precisão das medições. O sistema COMPASS apresenta uma solução para este problema através do aumento do número de amostras de potência de sinal obtidos em cada local com orientações diferentes.

A fase de calibração é fulcral para o bom funcionamento do sistema, e é das *fingerprints* que o sistema depende quando é necessário estimar a localização. Esta fase exige que sejam feitas medições em vários pontos alinhados em forma de grelha com 1 m de espaço entre si onde são obtidas medições em 8 direções diferentes para cada ponto. Mesmo num pequeno edifício com uma área de 125 m^2, a fase de treino demoraria mais do que 4h.

O tamanho do conjunto de treino é um parâmetro bastante importante, sendo que este parâmetro determina o limite inferior no tempo necessário para se recolher os dados para a base de dados de *fingerprints.* O gráfico da Figura 2.6 apresenta o tamanho do conjunto de treino em função da distância de erro média. A precisão aumenta mais de 10 cm se o tamanho do conjunto de treino for aumentado de 3 para 20 medições. Contudo, se houver um aumento deste tamanho do conjunto de treino, a precisão melhora ligeiramente e por isso foi determinado que o tamanho do conjunto de treino de 20 medições é o mais adequado.

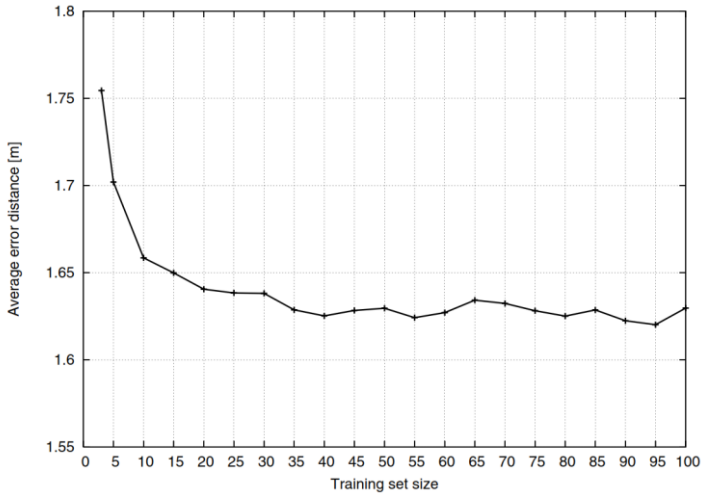


Figura 2.6: Distância de erro média em função do tamanho do conjunto de treino em função da [28]

Na fase *online* o número de medições também influencia a precisão do sistema. o gráfico da Figura 2.7 apresenta variação do erro média quando há uma variação no número de medições *online*.

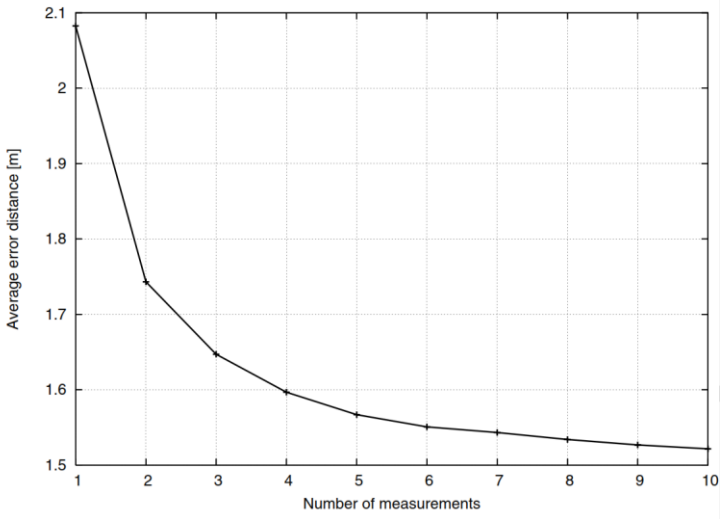


Figura 2.7: Distância de erro média em função do tamanho do conjunto de treino [28]

Foi utilizado o algoritmo *Multiple Nearest Neighbors* proposto por Bahl *et al.* [3] uma vez que este apresenta melhor performance. Comparando este sistema com o sistema RADAR, é possível verificar que o COMPASS tem melhores resultados que o RADAR, o algoritmo é mais preciso revelando uma distância de erro inferior a 2 m em 70% de todas as medições, um valor bastante superior aos 50% conseguidos com o sistema RADAR.

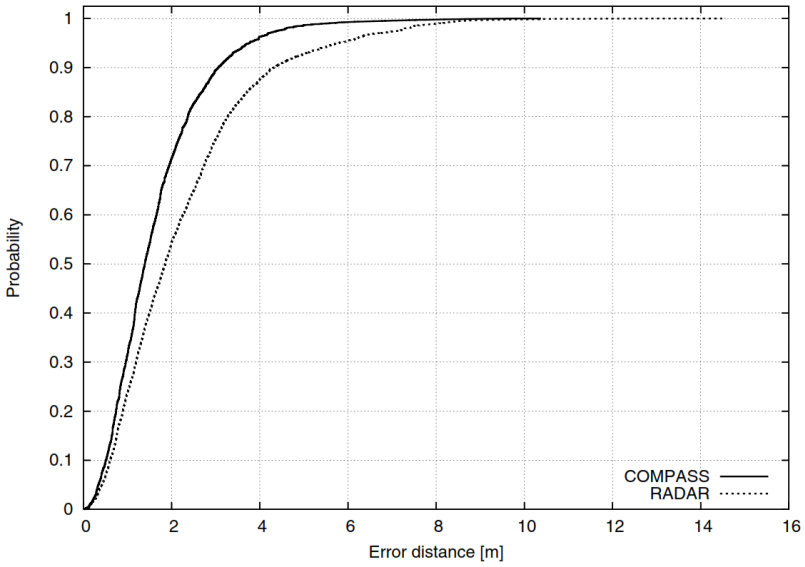


Figura 2.8: Função de distribuição cumulativa da performance dos sistemas COMPASS e RADAR [28]

A precisão deste sistema demonstra uma distância de erro médio de 1,65 m, um valor inferior aos 2,26 m obtidos com o sistema RADAR. Por outro lado, será relevante referir que no sistema COMPASS apenas se fez a localização de um utilizador e não foi discutida a possibilidade de múltiplos utilizadores. Assim, a escalabilidade do sistema COMPASS é baixa para fornecer a deteção da localização de vários utilizadores. Além disso, a fase de treino é uma tarefa que consome bastante tempo para ser feita corretamente e é uma questão que deve ser tomada em consideração comparando os resultados deste sistema com resultados obtidos por outros sistemas.

2.4. Empresas, produtos comerciais e aplicações móveis

Com a evolução dos últimos anos na área dos dispositivos móveis e a utilização em massa destes dispositivos em combinação com os desenvolvimentos na área de sistemas de posicionamento de interiores, verifica-se um aparecimento de vários produtos e aplicações móveis que envolvem os sistemas de posicionamento de interiores. Além das grandes empresas, os programadores independentes também contribuem com as suas soluções para o mercado resultando em produtos bastante competitivos com muitas funcionalidades. Também houve o aparecimento de diversas *startups* que tentam ainda afirmar-se no mercado. A WiFiSLAM [29] foi uma das primeiras *startups* a surgir com um sistema de posicionamento *indoor* baseado em Wi-Fi *fingerprinting,* e foi adquirida pela Apple em 2013. Entretanto surgiram vários produtos capazes de localizar dispositivos em ambientes fechados utilizando várias tecnologias.

Nesta secção serão abordadas várias soluções, tendo especial atenção os sistemas mais focados em tecnologias Wi-Fi.

2.4.1. IndoorAtlas

A empresa IndoorAtlas [7] desenvolveu um sistema baseado em posicionamento magnético o que significa que a solução é independente das infraestruturas externas como por exemplo os pontos de acesso Wi-Fi.

É disponibilizado um SDK para Android e iOS que facilita o desenvolvimento de aplicações para este sistema. O sistema funciona em ambientes interiores e exteriores e utiliza como pontos de referência as variações naturais do campo geomagnético. Os sensores de um *smartphone* são capazes de detetar as variações do campo geomagnético e tal como numa *fingerprint* é possível associar as variações do campo geomagnético medidas pelos sensores a um local específico. A precisão deste sistema ronda os 3 m.

O serviço *cloud* da IndoorAtlas permite que o mapa do campo magnético seja criado depois dos dados do campo magnético serem recolhidos.

Os utilizadores do serviço IndoorAtlas podem desenvolver as suas aplicações e utilizar a API desenvolvida para fazer a comunicação entre o serviço de posicionamento e a aplicação.

A empresa tem disponível a aplicação IndoorAtlas MapCreator para Android [30] e iOS [31], que é uma aplicação com muitas funcionalidades principalmente no âmbito da fase de treino, onde é possível adicionar a planta de um edifício e sobrepô-la ao edifício no Google Maps. O utilizador deve estar registado na aplicação para poder usufruir do serviço e o dispositivo tem que possuir magnetómetro (bússola), acelerómetro e giroscópio para a aplicação funcionar. A página inicial é um mapa centrado na localização do utilizador, uma lista de locais próximos (edifícios/locais adicionados por outros utilizadores na mesma área) e a lista de locais adicionados pelo utilizador.

Para o utilizador adicionar um local, tem que marcar o edifício no mapa e carregar a planta de um piso à escolha (através de uma imagem ou fotografia). Depois de escolhida a planta do edifício, é necessário alinhar a planta com o edifício no mapa, como ilustrado na Figura 2.10. A fase de treino requer que o utilizador marque uma trajetória entre dois pontos que deverá seguir durante a recolha de dados (são recolhidos os dados dos diversos sensores do *smartphone*, incluindo as informações dos pontos de acesso Wi-Fi). Por fim é necessário que os dados recolhidos sejam enviados para a *cloud* de forma que o mapa de rádio seja construído.

Para fazer o mapeamento foi necessário fazer uma trajetória definida entre 2 pontos, percorrendo um caminho de teste para fornecer melhores resultados no mapa de rádio criado. A Figura 2.11 ilustra o processo de calibração.

A funcionalidade de posicionamento estima a posição do utilizador dentro do edifício em tempo real fazendo a comparação dos dados recolhidos na fase de calibração com os dados recolhidos na fase *online.*

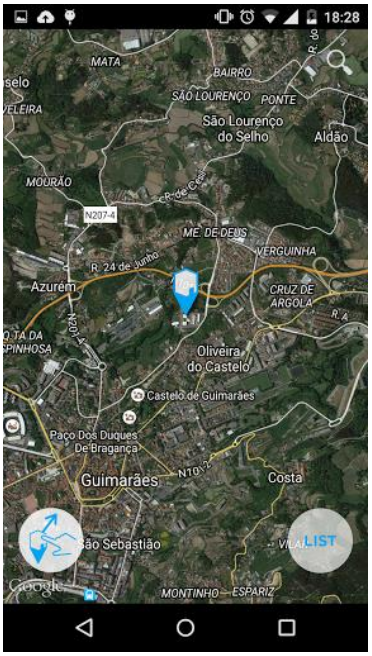


Figura 2.9: Página inicial da aplicação IndoorAtlas

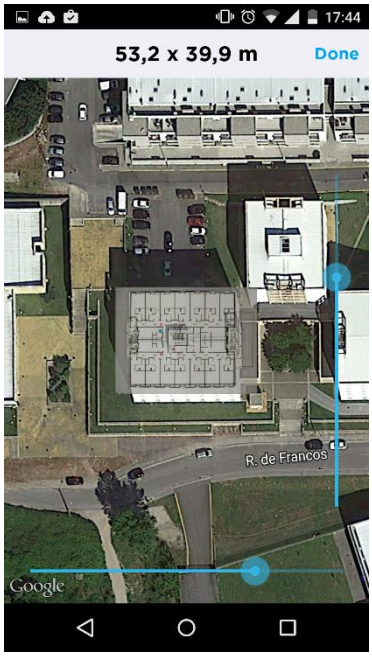


Figura 2.10: Alinhamento da planta do edifício com o mapa

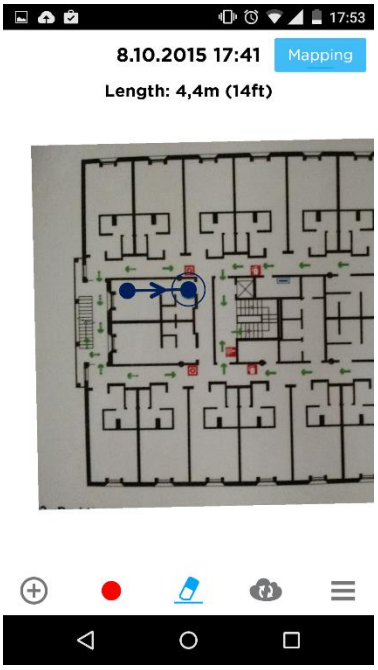


Figura 2.11: Fase de calibração

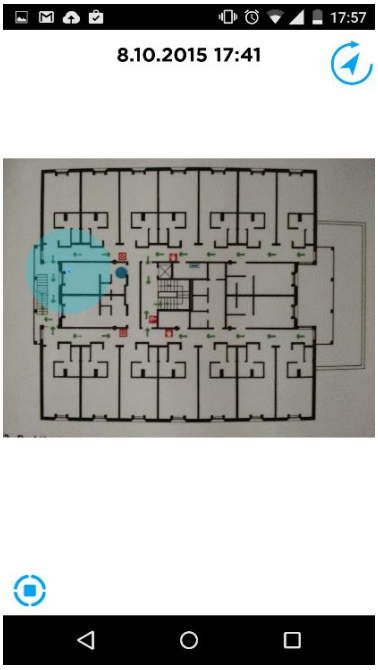


Figura 2.12: Fase real-time

2.4.2. NFER Real-Time Location Systems

A Q-TRACK [32] comercializa produtos baseados no seu sistema de localização em tempo real com tecnologia NFER(*Near Field Electromagnetic Ranging*). Esta tecnologia *near field* utiliza ondas de baixa frequência com elevado comprimento de onda que permitem obter uma precisão por volta de 1 m. Este sistema localiza através de *tags* que são espalhadas pelo espaço interior. O sistema tem capacidade de detetar até 500 *tags*.

Atualmente os produtos que utilizam a tecnologia NFER têm várias finalidades: localização de militares em edifícios para treino de grande dimensão; registo da localização de bombeiros, policias, mineiros e enfermeiros.

2.4.3. Ekahau

A empresa Ekahau [33] desenvolveu um sistema que utiliza as infraestruturas WLAN existentes para monitorizar continuamente o movimento de dispositivos e *tags* Wi-Fi.

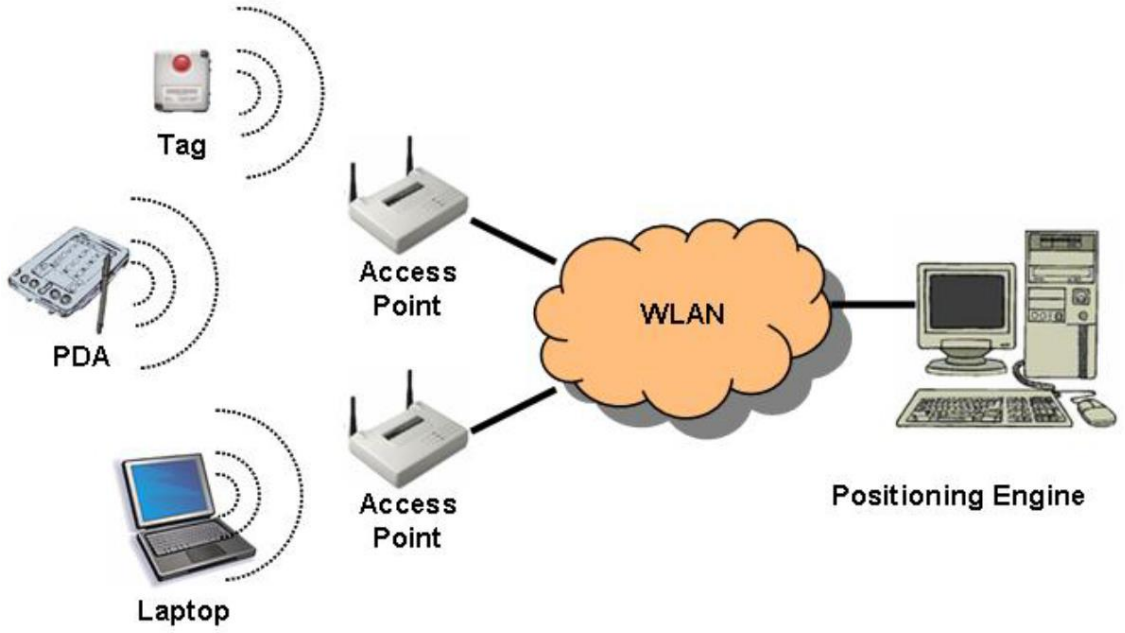


Figura 2.13: Arquitetura do sistema Ekahau [34]

A técnica de triangulação é utilizada para localizar qualquer dispositivo com Wi-Fi no sistema de posicionamento Ekahau. Os valores RSSI dos sinais de radiofrequência registados nos diferentes APs são utilizados para determinar as localizações alvo.

Esta solução é de baixo custo e flexível, com capacidade para fazer a localização *indoor* através da localização de *tags* via dispositivos de referência, que são os pontos de acesso Wi-Fi.

A arquitetura do sistema tem três elementos. O primeiro elemento é uma ferramenta de *software* que permite a calibração dos locais com recurso a algumas informações sobre a rede. O segundo elemento é a *tag* de localização Wi-Fi, que pode ser colocada em qualquer objeto que se pretende localizar. A *tag* transmite sinais de radiofrequência. Os pontos de acesso medem a força de sinal dos sinais de radiofrequência recebidos. O terceiro elemento é um motor de posicionamento que localiza qualquer dispositivo em tempo real utilizando tecnologia WLAN.

A precisão do sistema Ekahau pode chegar a 1 m e o sistema tem capacidade para localizar milhares de dispositivos simultaneamente. Por outro lado, esta robustez só é atingida se houver mais do que 3 APs para que o sistema consiga localizar o dispositivo alvo com uma precisão até 1 m.

2.4.4. WiFiSlam

Em 2013 surgiu um vídeo [35] com uma demonstração de um produto desenvolvido pela WiFiSlam [29]. A aplicação móvel permite que o utilizador adicione várias localizações (edifícios ou pisos de edifícios) através de uma fotografia da planta do piso obtida na planta de emergência. Depois a aplicação processa a imagem e o utilizador seleciona o ponto da planta em que se encontra. Para fazer a calibração, o utilizador desloca-se em linhas retas no espaço interior e marca os pontos em que está localizado depois de fazer o percurso em linha reta. Quando este processo está concluído, os dados são enviados para a *cloud* e processados por um servidor.

Na fase de tempo real a posição estimada do utilizador aparece enquanto este se desloca no espaço interior. Este produto não se encontra no mercado porque a WiFiSlam foi adquirida pela Apple em 2013, por isso será de esperar num futuro próximo uma solução nesta área desenvolvida pela Apple.

2.4.5. Navizon

A Navizon [36] comercializa produtos baseados no seu sistema de triangulação *indoor.* Este sistema funciona com recurso a *hardware* adicional que são chamados de *nodes* que permitem detetar dispositivos Wi-Fi ativos e faz o *upload* da lista de endereços MAC e níveis de sinal. O servidor deste sistema estima as posições dos dispositivos relativamente aos *nodes.* As aplicações fazem pedidos ao servidor para obterem as localizações dos dispositivos.

O sistema depende dos *nodes* que necessitam de ser instalados com disposição de grelha no espaço interior com uma distância entre 12 e 20 m entre cada *node.* Quanto maior for a distância entre os *nodes*, pior será a precisão deste sistema. O valor mais baixo da precisão vai dos 2 aos 3 m quando os *nodes* têm 12 m de distância entre si.

A aplicação Navizon Indoors para Android [37] e iOS [38] apesar de ter muitas das funcionalidades restritas aos utilizadores dos sistemas Navizon, é possível utilizar a aplicação na fase de treino e de tempo real. O utilizador não pode adicionar a planta do edifício (funcionalidade apenas disponibilizada aos subscritores do serviço Navizon), mas tem acesso ao mapa do Google Maps com a vista de satélite.

Como não existe a planta do edifício nem está associada a informação relativa a cada piso, não é possível determinar em que piso o utilizador se encontra. Mas é possível determinar a localização do utilizador em tempo real e verificar a movimentação do mesmo dentro do edifício (a partir da vista de satélite fornecida pelo Google Maps).

Antes de se realizar a fase de treino, o utilizador fixa a zona do mapa em que está, tendo em vista o edifício onde está localizado. Depois o utilizador marca com dois ou mais pontos um trajeto para percorrer ao longo da fase de treino. A fase de treino é concluída quando o utilizador terminar o percurso que definiu previamente, pode também optar por repetir o processo para recolher mais amostras e assim aumentar o nível de precisão.

Figura 2.14: Fase de treino na aplicação Navizon Indoors

2.4.6. Infsoft

A empresa Infsoft [39] faz parte do conjunto de empresas que tem no mercado vários produtos na área do posicionamento de interiores. Os produtos comercializados pela Infsoft são soluções de posicionamento, acompanhamento e navegação em edifícios de elevadas dimensões.

O produto que permite a navegação *indoor* não precisa da instalação de *hardware* adicional e para estimar a localização com uma precisão dentro de 1 m utiliza vários sensores num *smartphone* (GSM, 3G/4G, Wi-Fi, campos magnéticos, bússola, barómetro, barómetro acelerómetro, giroscópio, Bluetooth e GPS). Este produto inclui um SDK para as plataformas Android, iOS e Windows Phone o que permite que cada cliente desenvolva a sua aplicação de acordo com as suas necessidades. A calibração dos espaços interiores é feita com recurso a uma aplicação da Infsoft para Android [40] e iOS.

2.4.7. Guardly

O produto principal da empresa Guardly [9] é um sistema de posicionamento de interiores que pode ser utilizado principalmente em aeroportos para monitorização de pessoas e malas, centros comerciais e universidades para finalidades de segurança, e hospitais para emergências médias.

O sistema de posicionamento combina sinais de rede móvel (GSM) provenientes das torres celulares, GPS e redes Wi-Fi existentes para determinar a localização do dispositivo. A precisão do sistema é ao nível de um quarto/escritório. Numa demonstração do sistema em vídeo [41] é possível analisar todas as potencialidades do sistema, principalmente na área da segurança. Na demonstração, os utilizadores através de uma aplicação móvel podem pedir auxílio aos seguranças de um *campus* universitário em situações de emergência.

A fase de calibração do sistema tem que ser realizada por técnicos especializados que fazem o mapeamento do edifício e depois realizam vários testes no sistema. Esta solução tem por isso um custo aumentado dado que é necessário a deslocação, implementação e configuração prévia por parte de técnicos.

2.4.8. Where@UM

A aplicação Where@UM [42] foi desenvolvida no âmbito de uma dissertação de Mestrado [43] por um aluno da Universidade do Minho. A aplicação utiliza um sistema de posicionamento baseado em Wi-Fi *fingerprinting* que cria o mapa de rádio dos espaços através de um processo colaborativo. O objetivo é envolver os utilizadores do sistema no processo da construção do mapa de rádio. Para isso os utilizadores têm que introduzir a sua localização, com isso o sistema vai construindo o mapa de rádio de cada localização. A aplicação foi desenvolvida especialmente para alunos e professores da Universidade do Minho e tem uma abordagem semelhante a uma rede social onde os utilizadores partilham a sua localização e podem consultar a localização dos seus amigos. Os utilizadores podem adicionar novos locais dentro ou fora da Universidade do Minho, quando o sistema ainda não suporta esses locais.

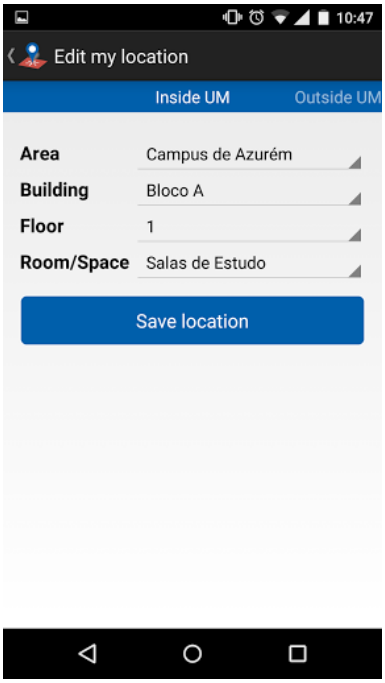


Figura 2.15: Editar localização na aplicação Where@UM

Figura 2.16: Lista de amigos na aplicação Where@UM

A arquitetura geral do sistema (Figura 2.17) é constituída por três componentes: o sistema envolve o dispositivo móvel, um servidor e um motor de posicionamento. O dispositivo móvel comunica via HTTP com o servidor através da aplicação, o servidor fornece os serviços de suporte à aplicação móvel e faz o armazenamento dos dados referentes ao mapa de rádio. O servidor comunica com o motor de posicionamento via HTTP. O motor de posicionamento permite estimar a posição dos utilizadores fazendo a correspondência da medição efetuada no dispositivo móvel com as *fingerprints* armazenadas na base de dados.

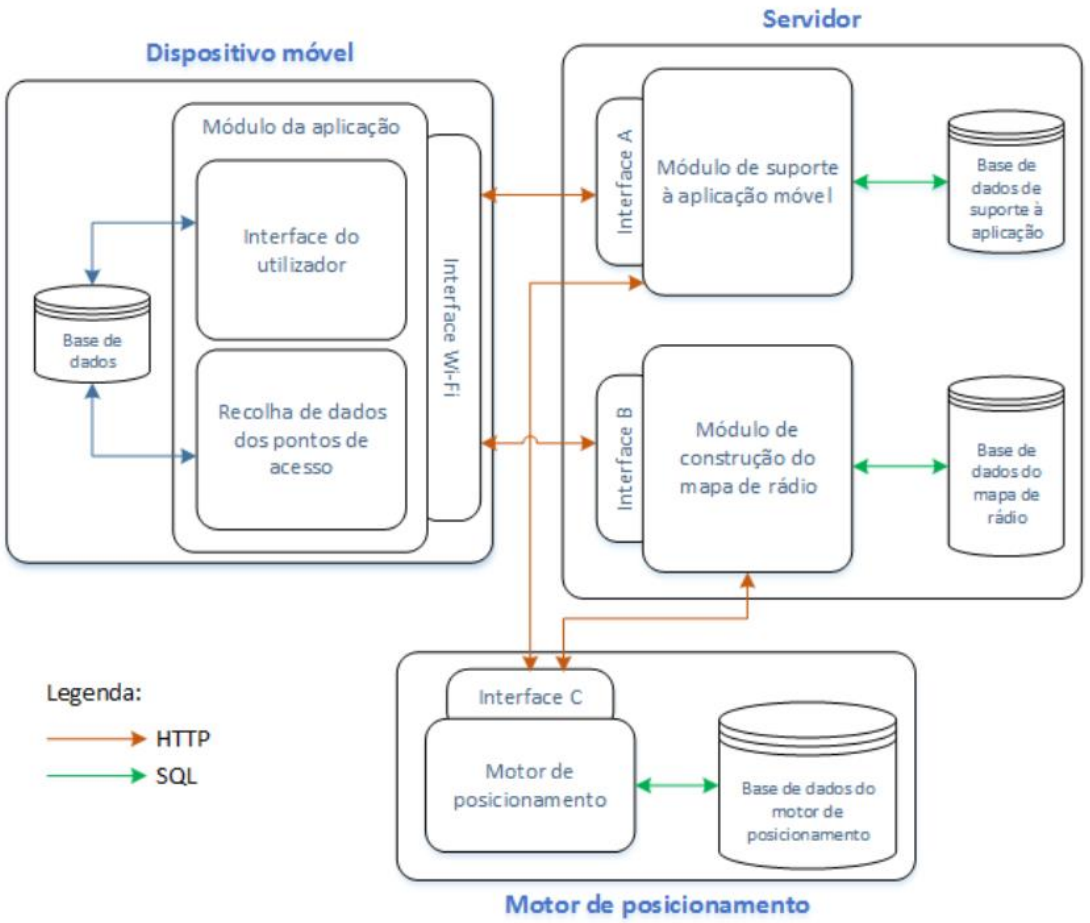


Figura 2.17: Arquitetura geral do sistema - Where@UM [43]

A aplicação é bastante útil no ambiente académico entre grupos de estudantes e mesmo entre docentes e alunos uma vez que permite que os utilizadores se encontrem com facilidade.

Este sistema é um ponto de referência para o trabalho desenvolvido nesta dissertação uma vez que aborda algumas temáticas desta dissertação.

Recentemente foi desenvolvida outra dissertação de Mestrado [44] com o intuito de introduzir novas funcionalidades na aplicação, nomeadamente: suporte para Android 6.0, troca de mensagens entre amigos, avaliação da qualidade da informação de localização, melhorias em termos de estabilidade e correção de pequenos erros.

2.4.9. Wifarer

A Wifarer [10] tem várias aplicações que utilizam o seu sistema de posicionamento de interiores. Esta empresa disponibiliza um SDK para as plataformas Android e iOS para que sejam

desenvolvidas aplicações que utilizam o sistema de posicionamento. Além disso são disponibilizadas várias ferramentas para a gestão da aplicação.

As soluções desta empresa utilizam uma ou várias tecnologias para obter a localização de um dispositivo (Wi-Fi, Bluetooth e GPS para as soluções que envolvem ambientes interiores e exteriores).

As principais funcionalidades das aplicações Wifarer são: posicionamento *indoor*, navegação, disponibilização de conteúdos em localizações específicas, rotas de acesso em cadeiras de rodas, e navegação em ambientes interiores e exteriores. Estas aplicações podem ser utilizadas em diversas áreas como por exemplo em hospitais, aeroportos, museus, estádios, lojas e universidades. As funcionalidades podem ser acedidas a partir do menu da aplicação exibido na Figura 2.18.

Figura 2.18: Menu do aplicativo Wifarer [45]

A aplicação Wifarer [45] disponível na Google Play e na App Store [46] é uma aplicação que utiliza os sensores do *smartphone* (Wi-Fi, GPS e Bluetooth) para determinar a posição do utilizador em ambientes interiores e exteriores. A aplicação permite a navegação e a disponibilização de conteúdos de acordo com a localização do utilizador. Por exemplo, num aeroporto a Wifarer ajuda na navegação em terminais complexos e também notifica o utilizador se houverem atrasos no voo ou se a *gate* foi alterada.

2.4.10. Maps Inside

A aplicação para Android Maps Inside [47] permite que o utilizador introduza o mapa do edifício a partir do sistema de ficheiros do dispositivo, introduz as informações relativas ao edifício e depois a aplicação faz a análise do ambiente rádio do piso em que está quando o utilizador pretender. A planta do edifício e as informações relativas ao processo de calibração estão apresentadas na aplicação como demonstra a Figura 2.19.

Figura 2.19: Interface da aplicação Maps Inside [47]

Os utilizadores desta aplicação depois de adicionarem o mapa podem criar o mapa de rádio com a opção de gravação, que é simplesmente andar pelo piso e consultar na aplicação as medições que estão a ser feitas. É aconselhado que sejam feitas entre 5 e 10 medições em cada espaço. A gravação pode ser feita a partir do item "Recorder" do menu da aplicação (Figura 2.20).

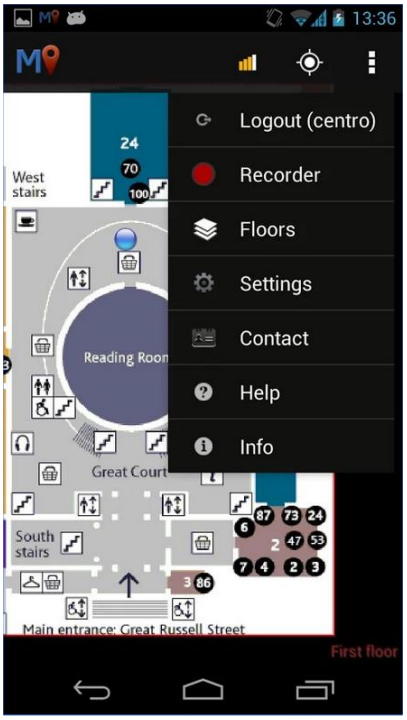


Figura 2.20: Menu da aplicação Maps Inside [47]

Outra funcionalidade disponibilizada pela aplicação é a capacidade do sistema estimar a localização do dispositivo em movimento. Por fim é possível armazenar os dados que foram adicionados (o mapa do edifício, as informações introduzidas e as medições que ocorreram) na *cloud*.

2.4.11. Indoor GPS

O Indoor GPS [48] é um aplicativo para Android que utiliza como tecnologia base o Wi-Fi e tem capacidade para construir o mapa de rádio de um edifício.

Na Figura 2.21 está a página principal da aplicação onde é possível introduzir a planta do piso através de uma fotografia ou uma imagem armazenada no sistema de ficheiros do dispositivo.

Figura 2.21: Página inicial da aplicação Indoor GPS

A fase de calibração, mostrada na Figura 2.22, é feita pelo utilizador fixando um ponto inicial no local onde se começa o movimento pelo espaço interior. De seguida, desloca-se para outro ponto do espaço e marca esse ponto como ponto de destino. A partir desse momento o processo de calibração inicial está concluído. A aplicação permite que sejam feitas várias calibrações com outras rotas para melhorar os resultados na fase *real-time.* Depois da fase de calibração a aplicação permite que a localização do dispositivo seja estimada comparando os dados recolhidos na fase de treino com os dados medidos em tempo real.

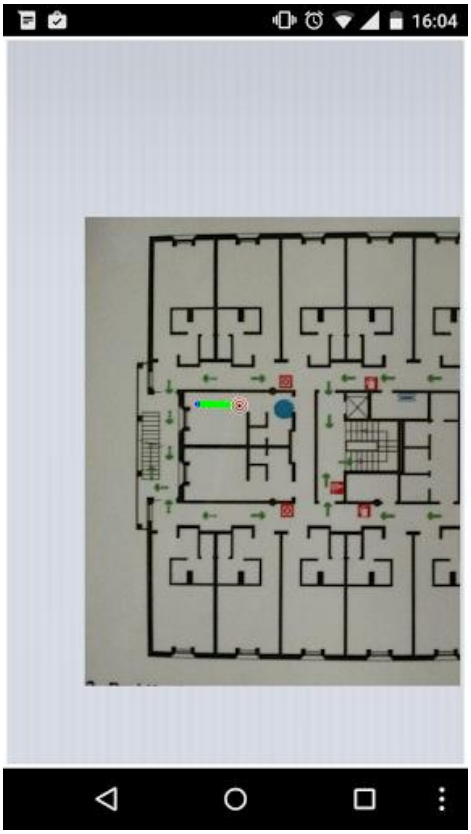


Figura 2.22: Fase de calibração da aplicação Indoor GPS

Esta é a versão gratuita da aplicação que demonstra a funcionalidade dos sistemas baseados em Wi-Fi *fingerprinting*. Há uma versão paga que permite guardar os dados do mapa de rádio obtidos na fase de calibração e assim dá a possibilidade de desenvolver uma solução com várias funcionalidades dentro deste sistema porque o SDK criado pelo desenvolvedor pode ser utilizado.

2.4.12. WiFi Indoor Localization

O aplicativo para Android WiFi Indoor Localization [49] é completamente baseada em Wi-Fi *fingerprinting* e tem uma abordagem diferente das outras aplicações analisadas. O utilizador pode carregar a planta do piso num formato de imagem a partir do sistema de ficheiros e depois de inserir a imagem define o tamanho das células que estarão sobrepostas à planta do piso (Figura 2.23). As células em forma de grelha entram na fase de treino (Figura 2.24), onde o utilizador faz a calibração selecionando a célula correspondente à sua posição e a aplicação recolhe os dados dos pontos de acesso (endereço MAC e valores BSSI) referentes à célula em que o utilizador está localizado. O utilizador repete o processo nas células em que pretende recolher *fingerprints* completando assim a fase de calibração. Os dados recolhidos podem ser exportados para um

ficheiro de texto (informação da célula GSM, o BSSID do ponto de acesso, o SSID (*Service Set Identifier*), o nível de sinal e o *timestamp*). O mapa de rádio do espaço é construído tendo em base as *fingerprints* obtidas.

A localização do utilizador é estimada através da correspondência dos dados medidos pelo dispositivo em tempo real com as *fingerprints* recolhidas anteriormente na fase de treino. A Figura 2.25 ilustra este processo que corresponde à fase de tempo real do sistema.

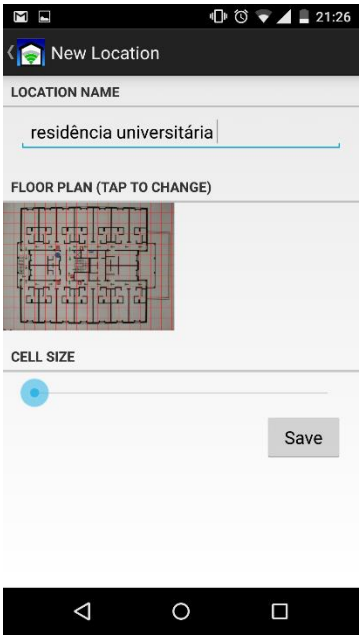


Figura 2.23: Adicionar a planta do piso

Figura 2.24: Fase de treino

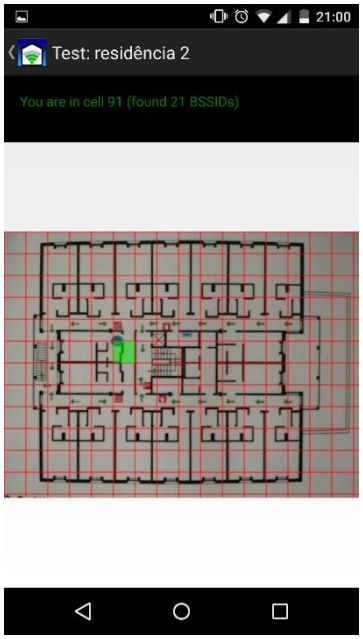


Figura 2.25: Fase de tempo real

2.5. Considerações finais

O trabalho de pesquisa e análise de vários sistemas é relevante para o trabalho de desenvolvimento que será abordado nos capítulos seguintes, principalmente com os sistemas que englobam aplicações móveis.

A análise de aplicações revelou a existência de algumas soluções semelhantes ao que é pretendido com este trabalho, das quais se podem tirar ideias pertinentes para o contexto da solução que será desenvolvida. A opção para inserir a planta do edifício através de uma fotografia (que pode ser obtida nas plantas de emergência) parece bastante interessante e prática do ponto de vista do utilizador. Funções como a marcação de dois pontos como trajetória a tomar na fase de treino são práticas em situações de recolha de *fingerprints* de um espaço.

É importante realçar que muitas das aplicações utilizam sistemas com *beacons* e combinam várias tecnologias para obter a localização do utilizador, nomeadamente através dos serviços de localização da Google, GPS, Bluetooth e GSM. Também é relevante denotar que algumas das aplicações analisadas tinham funcionalidades restritas aos clientes do serviço, uma vez que exigiam uma subscrição paga.

No Anexo A está uma tabela onde são comparadas as funcionalidades das aplicações móveis analisadas.

CAPÍTULO 3 – ESTUDO DO PROBLEMA

3.1. Introdução

Quando se pretende desenvolver uma solução para a construção dos mapas de rádio surgem várias questões: as plantas dos edifícios (quais as soluções e alternativas?), as coordenadas em ambientes interiores (qual é a melhor alternativa para identificar um espaço num piso específico?) e a atualização do mapa de rádio (como resolver o problema da desatualização dos mapas de rádio?). Com este trabalho pretende-se dar resposta a estas questões no sistema que será desenvolvido.

Num sistema de posicionamento de interiores, a fase de calibração é muito importante porque a eficácia do sistema depende da qualidade/quantidade dos dados obtidos na fase inicial. Na fase de treino é feito o reconhecimento do ambiente rádio de cada espaço de um edifício.

Os níveis de sinal de cada ponto de acesso dentro do alcance do dispositivo são recolhidos em vários pontos do espaço interior, escolhidos propositadamente para que sejam recolhidas amostras que representem o ambiente de rádio de cada espaço.

Por si só a fase de calibração é um trabalho moroso onde é necessária a recolha em todos os espaços que constituem o mapa de rádio de um edifício. Esta tarefa quando é efetuada manualmente (maioria dos casos) exige que o utilizador se desloque para cada espaço do piso, e obtenha várias *fingerprints*. Se esta tarefa se torna difícil num edifício de dimensões pequenas é fácil prever que será uma tarefa bastante demorada, por exemplo, quando se pretende construir o mapa de rádio de todos os edifícios de um campus universitário.

3.2. O problema das plantas dos edifícios

Num ambiente *indoor* um utilizador tem capacidade para se orientar com recurso à planta do piso. Se não houver um elemento gráfico que represente a estrutura interna do edifício, uma pessoa apenas sabe guiar-se pela sua intuição e pelo que observa em seu redor.

Realizar a recolha de amostras para a fase de treino sem as plantas do edifício é uma tarefa complicada uma vez que o utilizador não tem forma gráfica de associar a informação ao local onde está. Com a planta do piso, o utilizador pode indicar facilmente o local onde se encontra para realizar a fase de treino com facilidade e rapidez.

Assim surge a questão da utilização das plantas de edifícios nos sistemas de posicionamento de interiores. As empresas tecnológicas Google, Apple e Nokia já têm em prática as suas versões de mapas *indoor* [11] [50]. Neste momento, é possível adicionar as plantas de

um edifício nas plataformas da Apple e da Google com o cumprimento de condições específicas. A solução da Apple é pouco flexível uma vez que apenas está disponibilizada para edifícios com um número de visitantes anual superior a um milhão de pessoas, entre outras características. Para adicionar uma planta de um edifício na plataforma da Google basta localizar o edifício no mapa, preencher as informações relativas ao edifício (nome, andar, etc.), carregar o ficheiro da planta (formatos possíveis: JPG, PNG, PDF, BMP e GIF), de seguida é necessário alinhar a planta do edifício com o mapa e por fim fazer a submissão dos dados. Quando se trata de um edifício com vários andares, é necessário carregar a planta de cada andar resultando num processo repetitivo. Depois de concluído, é necessário aguardar validação por parte da Google e por fim será possível consultar a planta a partir da aplicação Google Maps [51], onde não existe uma previsão temporal para a disponibilização das informações dos espaços interiores do edifício. A Nokia apresenta mapas *indoor* na aplicação Here Maps [52] para Android mas não disponibiliza uma plataforma para inserção das plantas dos edifícios.

Mesmo havendo soluções disponíveis pelas maiores empresas tecnologias do mundo, não existe um *standard* para mapas de interiores nem para coordenadas em ambientes interiores, o que traz desvantagens quando se pretende desenvolver um sistema de posicionamento de interiores. É necessário ter em consideração todas estas questões principalmente para a componente que realiza a fase de calibração.

Existem algumas soluções de mapas de interiores como se pode verificar em [53] e na demonstração de mapas de interiores da OpenStreetMap em [54]. A primeira solução é um produto comercializado e a segunda solução requer a criação da planta do edifício com recurso à ferramenta JOSM [55].

OpenStreetMap [56] trata-se de um projeto colaborativo para criar um mapa do mundo livre e que pode ser editado por parte dos utilizadores. No âmbito dos ambientes interiores existe uma página dedicada a diversos temas de mapeamento em ambientes interiores [57], também podem ser consultadas várias propostas de esquemas de *tagging* para ambientes interiores em [58] e [59] onde também consta um pequeno tutorial que explica como se faz o mapeamento de dados com a ferramenta JOSM. Através desta ferramenta é possível fazer o mapeamento dos espaços interiores de um edifício com recurso a um ficheiro OSM (mapa) e a planta do edifício (ficheiro de imagem). A planta do edifício é alinhada com o edifício no mapa, e depois através da ferramenta JOSM é possível utilizar os elementos do OSM (*nodes, ways*, *relations* e *tags*) para descrever o ambiente interior do edifício (salas, paredes, escadas, elevadores, etc). O resultado do

mapeamento do ambiente interior é um ficheiro no formato OSM (XML) que é uma lista de instâncias dos elementos OSM (*nodes, relations* e *ways*).

Outra forma de utilizar os mapas de interiores é sobre a forma de uma imagem da planta do piso, fazendo a gestão de vários ficheiros associados a cada edifício. Um local pode ser identificado através da informação do piso e das coordenadas cartesianas do local no ficheiro de imagem da planta.

Dadas as necessidades deste trabalho, será necessário desenvolver uma solução própria para as plantas dos edifícios com capacidade para:

- Criação das plantas dos edifícios;
- Armazenamento das plantas dos edifícios;
- Disponibilização do conteúdo das plantas;
- Apresentação das plantas num dispositivo Android.

3.2.1. Coordenadas num sistema *indoor*

Sabendo que serão recolhidas várias *fingerprints* com os níveis de sinal em cada ponto dentro do alcance do dispositivo de leitura, será necessário associar às *fingerprints* um conjunto de coordenadas que identifica o local dentro do edifício inequivocamente, tendo em consideração o piso do edifício.

Infelizmente, não existe um *standard* que defina a melhor forma de identificar um local dentro de um edifício. Há vários modelos possíveis para identificar o local inequivocamente, nomeadamente um par de coordenadas (latitude, longitude), o nome do local/espaço, o piso, o nome do edifício, a rua, a cidade e o país. As informações referentes à morada podem ser obtidas facilmente, através de serviços que permitem identificar a morada de um edifício num mapa, além disso é necessário associar as informações relativas ao nome do edifício, ao piso e ao nome da sala/espaço.

Neste trabalho o objetivo é apresentar a localização do utilizador ao nível da sala/espaço e por isso as coordenadas e a forma de armazena-las serão discutidas posteriormente no capítulo do desenho do sistema uma vez que a arquitetura terá influência no formato de coordenadas que serão utilizadas.

3.3. Processo de calibração

A calibração em dispositivos móveis pode ser feita de duas formas: de forma automática (com recurso aos sensores disponíveis nos dispositivos móveis), ou de forma manual.

É possível obter várias informações sobre a movimentação do utilizador e a sua orientação quando a calibração é realizada com recurso aos sensores do dispositivo (acelerómetro, magnetómetro, giroscópio, entre outros). Isto permite que a calibração seja feita enquanto o utilizador se move de um ponto de origem até um ponto de destino, ou seja, enquanto o utilizador se movimenta são recolhidas *fingerprints*. Algumas das aplicações enumeradas na secção 2.4 utilizam estes processos que procuram facilitar a fase de calibração, como é o caso da aplicação IndoorAtlas.

A fase de treino pode ser feita manualmente através a interação com o utilizador. Basta selecionar um ponto ou área pequena na planta do andar e as *fingerprints* recolhidas são associadas a esse ponto específico. A aplicação WiFi Indoor Localization utiliza esse processo na fase de calibração. Este processo é mais simples e permite obter dados com mais qualidade uma vez que se parte do princípio que o utilizador se encontra imóvel quando faz a recolha de amostras.

Neste trabalho pretende-se desenvolver um sistema de calibração que utiliza a planta do edifício para facilitar o processo de calibração manual. Através de uma aplicação será possível acompanhar o processo de criação do mapa de rádio através de *feedback* dado pela aplicação (como por exemplo, a densidade de amostras e a sua antiguidade). A aplicação de calibração também poderá servir para aferir a qualidade da cobertura de redes Wi-Fi através de informações recolhidas aquando da calibração.

3.4. Calibração automática

A calibração automática em espaços fechados pode ser feita de várias formas, das quais se destaca a técnica *Simultaneous Localization and Mapping* (SLAM), utilizada na robótica e que consiste na construção de um mapa de um ambiente desconhecido sendo que ao mesmo tempo é feita a localização. O conceito de SLAM engloba a localização e o mapeamento, por isso o resultado de um processo tem influência no outro.

A aplicação desta técnica em dispositivos móveis é uma tarefa complexa uma vez que são necessárias determinadas especificações que os dispositivos móveis normalmente não dispõem. Os sensores utilizados nos sistemas que utilizam SLAM, entre outros, são: sensor sonar, medidor de distâncias laser, e câmaras que permitem o posicionamento de alta precisão. O Project Tango

[60] da Google introduziu um *tablet* que apresenta alguns dos sensores que permitem o mapeamento e localização interior. Apesar de ser uma solução possível, este dispositivo é caro, só pode ser adquirido através de uma plataforma dedicada a esse fim e a sua utilização exige conhecimentos avançados.

A principal vantagem da técnica SLAM é o facto de não ser necessário utilizar as plantas dos edifícios para realizar a calibração. Por outro lado, existem várias desvantagens, nomeadamente, a utilização de dispositivos específicos de custo elevado e de utilização complexa.

Um dos trabalhos desenvolvidos nesta área [61] é um sistema de localização e mapeamento para pedestres em ambientes fechados. São montados nos pés dos utilizadores e fazem o mapeamento e localização através da movimentação do utilizador. Esta solução apresentou resultados positivos, contudo requer a utilização dos sensores para auxílio do mapeamento e localização, por isso não parece uma opção viável nos dias de hoje quando se opta apenas pela utilização de dispositivos móveis nos sistemas de localização *indoor.*

3.5. Atualização do mapa de rádio

Como é expectável, o ambiente rádio de um espaço interior está sujeito a mudanças constantes devido às variações de sinal, alterações no espaço e adição ou remoção de pontos de acesso. Para colmatar este problema, é necessário manter o mapa de rádio atualizado para que na fase *online* o sistema tenha capacidade para identificar a *fingerprint* correspondente aos dados medidos pelo dispositivo em tempo real.

Existem algumas soluções para manter o mapa de rádio de um edifício atualizado como é o caso da aplicação Where@UM que foi abordada anteriormente. Esta aplicação recorre a um sistema colaborativo onde os utilizadores contribuem para a evolução do sistema ao adicionarem locais desconhecidos através da recolha de *fingerprints* e também recolhe *fingerprints* em locais que já foram calibrados anteriormente, de forma atualizar o mapa de rádio. Este sistema depende da colaboração dos utilizadores, mas é mais prático do que repetir a fase de calibração em todos locais dos edifícios. Por outro lado, as abordagens colaborativas estão sujeitas a erros por parte dos utilizadores quando introduzem a sua localização que podem induzir o sistema a erros. Estes erros podem ser corrigidos através da recolha de *feedback* por parte dos utilizadores, partindo do principio que os utilizadores colaboram com informações verdadeiras.

Outra possibilidade é a atualização do mapa de rádio manualmente recorrendo a ferramentas específicas. Com o auxílio de uma aplicação para calibração manual, um

utilizador/técnico tem conhecimento dos locais onde o mapa de rádio está desatualizado, e deste modo, podem recolher amostras apenas nos locais onde o mapa de rádio está desatualizado. A gestão do mapa de rádio com recurso a uma aplicação pode ser efetuada, através do fornecimento de informações relativas à qualidade do mapa de rádio que permitem aos utilizadores a deteção de espaços que necessitam de ser calibrados ou onde o mapa de rádio está desatualizado. Esta tarefa exige um trabalho mais demorado por parte do utilizador da aplicação de calibração, mas fornece dados com mais qualidade para o mapa de rádio.

A melhor solução para a atualização dos mapas de rádio é através de uma solução colaborativa em conjunto com a calibração manual. Através do envolvimento dos utilizadores de um sistema colaborativo permite a atualização dos mapas de rádio de uma forma simplificada com a sua contribuição. De outro modo, a calibração manual permite que o mapa de rádio tenha dados com mais qualidade e, por isso a recalibração manual dos espaços periodicamente permite manter o mapa de rádio atualizado.

CAPÍTULO 4 – PLANTAS DOS EDIFÍCIOS

O presente capítulo mostra a solução para os problemas enunciados no terceiro capítulo. Inicialmente são descritos os cenários de utilização das plantas dos edifícios e é feito um estudo sobre as soluções para as plantas dos edifícios, do qual resulta a solução adotada. Posteriormente é introduzida a solução desenvolvida para as plantas dos edifícios através da apresentação do desenho e implementação do sistema.

4.1. Cenários de utilização das plantas dos edifícios

Existem dois cenários para a utilização das plantas dos edifícios: um cenário para efeitos de calibração (situação em que o utilizador indica explicitamente o local em que se encontra, clicando num ponto da planta do edifício) e um cenário para a visualização da localização de outros utilizadores (e do próprio).

No cenário de calibração, o utilizador identifica a sala/espaço em que se encontra e o sistema faz a recolha de *fingerprints* nesse local. É expectável que o utilizador recolha *fingerprints* em vários pontos da sala/espaço interior marcando os locais onde recolhe *fingerprints*. Além disso é necessário implementar uma solução que permita associar cada sala/espaço da planta do edifício a um nome que identifica esse local no respetivo andar do edifício. Associado a este cenário está um conjunto de serviços capazes de lidar com a gestão da informação necessária para a calibração, desde as plantas dos edifícios às informações recolhidas nas *fingerprints.* Neste cenário, as plantas dos edifícios têm um papel fundamental uma vez que os pontos de recolha são obtidos a partir das plantas e são esses pontos que servirão de referência na fase em que o sistema estima a localização.

O cenário de visualização consiste na apresentação da localização de diversos utilizadores e do próprio utilizador como pontos num mapa de interiores. Pretende-se mostrar onde estão as pessoas ou objetos cuja posição/localização está a ser seguida. A localização de cada pessoa/objeto é apresentada ao nível da sala/espaço do piso.

4.2. Soluções de plantas de interiores para implementação

Tendo em consideração as soluções para plantas de interiores, foram selecionadas duas possibilidades de implementação que são as melhores tendo em conta as soluções analisadas: a solução do Google Maps que disponibiliza as plantas de edifícios suportados, e a solução do OpenStreetMap, onde é possível descrever o espaço interior através da ferramenta JOSM. Cada

uma das soluções influencia diretamente a arquitetura do sistema, e por isso, inicialmente é necessário definir qual a melhor solução de mapas de interiores para o sistema.

A solução da Google para as plantas dos edifícios parece ser a menos flexível, mas com maior probabilidade de ser utilizada a uma grande escala uma vez tem uma API dedicada ao Google Maps. Por outro lado surgem dificuldades como por exemplo, o utilizador ter que aceder a um computador para adicionar as plantas dos edifícios e o facto de não existir nas APIs do Google Maps [62] e do Google Places [63] (API dedicada a pontos de interesse) uma solução que identifique inequivocamente um local no interior do edifício. Apenas é possível identificar pontos de interesse (locais adicionados e configurados previamente, como lojas, bancos, instituições, empresas, etc), escritórios e outros tipos de espaços dentro de um edifício não são identificados pela plataforma e não é possível associar estes espaços a um piso específico. Relativamente aos edifícios que apresentam as plantas de cada piso interior, não é possível obter as informações que identificam cada um desses espaços (nome do espaço) através das APIs disponibilizadas pela Google, sendo que a única informação que se pode obter são as coordenadas geográficas e o piso específico. Apesar disto, a solução da Google pode ser implementada se o utilizador intervier com a introdução das informações de cada local do espaço interior (apenas é necessário o nome).

A outra solução é baseada nos mapas OpenStreetMap para apresentar as plantas dos edifícios. Os mapas OpenStreetMap são bastante completos, e estão em constante atualização por parte dos utilizadores que colaboram através da introdução e atualização das informações associadas aos mapas. Tendo por base os mapas OpenStreetMap, o utilizador tem a capacidade de desenhar as plantas dos edifícios com recurso a uma ferramenta de edição de ficheiros OSM (JOSM), onde é possível adicionar a estrutura interior de cada piso. Trata-se de um trabalho que requer algum tempo, prática e exige o conhecimento da ferramenta JOSM. Tendo os espaços interiores definidos, o ficheiro resultante contém todas as informações para se renderizar a planta do piso em dispositivos móveis, o que permite o desenvolvimento da aplicação móvel utilizando esta solução. Desta forma existe um controlo total sobre os mapas de interiores, principalmente sobre os espaços interior e os nomes atribuídos aos mesmos. Além disso existe a possibilidade de introduzir navegação *indoor* através da introdução de grafos que representem a topologia aquando da construção da planta do edifício com a ferramenta JOSM.

A melhor solução para integrar na arquitetura é a solução de mapas de interiores OpenStreetMap uma vez que existe uma ferramenta que permite desenhar as plantas e que oferece algumas funcionalidades que podem ser utilizadas facilmente. Apesar do utilizador ter a

necessidade de fazer o desenho de cada piso do edifício manualmente através da ferramenta JOSM, isto oferece mais flexibilidade a qualquer utilizador com alguma prática que seja capaz de construir as plantas de um edifício e assim não fica sujeito ao tempo de espera dos mapas de interiores da Google: após a submissão das plantas, estas têm que ser convertidas para o formato utilizado no Google Maps e o tempo de espera não pode ser estimado. Na solução OSM, os dados das plantas dos edifícios são guardados em ficheiros OSM (XML), as estruturas são simples e legíveis tanto por utilizador como por máquinas (computadores), permitindo que estes dados sejam facilmente armazenados em bases de dados, o que oferece mais flexibilidade porque existe controlo total sobre as plantas dos edifícios e os seus dados, ao contrário da solução da Google onde se depende completamente das informações disponibilizadas pelas suas APIs e serviços.

O Google Maps poderá ser útil para apresentar a localização dos utilizadores em *overview*, onde os utilizadores aparecem espalhados por uma grande área geográfica (e.g. uma cidade inteira). Quanto às plantas dos edifícios, a solução OSM é a mais indicada sendo que a sua utilização será através dos ficheiros obtidos com a ferramenta JOSM.

4.3. Requisitos

Para colmatar os problemas das plantas dos edifícios enunciados anteriormente é necessário desenvolver uma solução para as plantas que seja capaz de:

- Permitir a criação de plantas com facilidade;
- Permitir o armazenamento das plantas num servidor;
- Permitir o acesso às plantas através de *web services* (disponibilização das plantas);
- Dar suporte à calibração e visualização da posição estimada para dispositivos Android, através do *rendering* de plantas nos dispositivos Android.

4.4. Desenho da solução para as plantas dos edifícios

O desenho do sistema aborda a informação que é necessária para o bom funcionamento do mesmo, assim como os requisitos e arquitetura do sistema. Assim, esta secção apresenta o desenho do sistema para as plantas dos edifícios.

4.4.1. Arquitetura da solução para as plantas dos edifícios

A definição da arquitetura da solução para as plantas dos edifícios é um passo importante do qual resulta um conjunto de elementos interligados. A arquitetura permite compreender o funcionamento do sistema de uma forma geral.

A arquitetura divide-se em três componentes distintos que se repartem em três tipos de dispositivos: dispositivo móvel Android, onde será apresentada a planta do edifício; computador, para desenhar a planta do piso; sistema central para o processamento, respetivo armazenamento e disponibilização das plantas. A Figura 4.1 mostra a arquitetura geral da solução para as plantas dos edifícios.

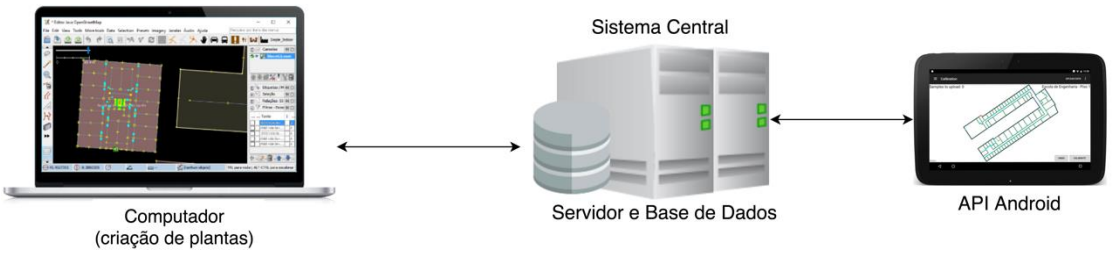


Figura 4.1: Arquitetura da solução para as plantas dos edifícios

4.4.1.1. Componentes da solução para as plantas dos edifícios

A função de cada componente do sistema é diferente, principalmente porque neste caso cada componente do sistema está associado um tipo de dispositivo. A Figura 4.2 demonstra os sub-módulos de cada componente e as interfaces de comunicação.

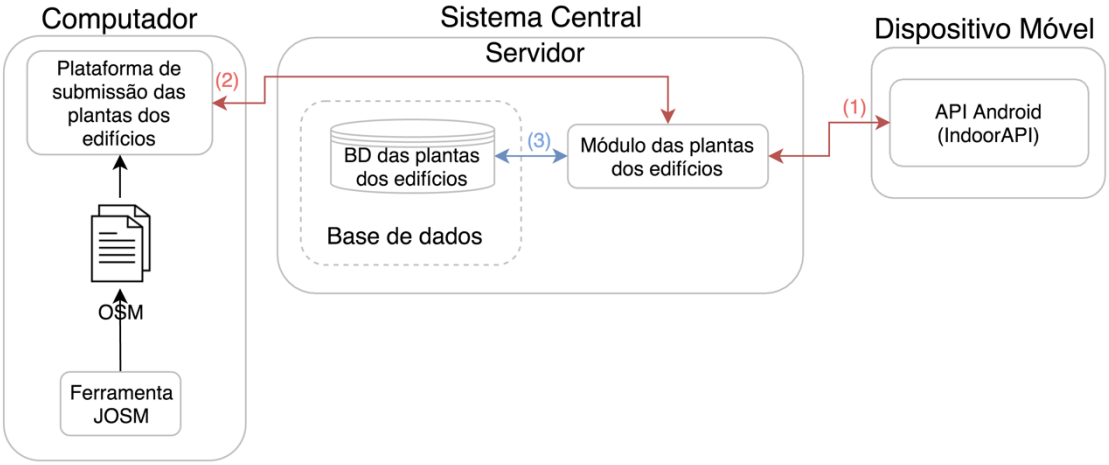


Figura 4.2: Arquitetura da solução para as plantas dos edifícios

Sistema central

O sistema central divide-se num servidor e numa base de dados. Quando é submetida a planta de um piso (2), o servidor recebe os dados através do módulo das plantas dos edifícios, processa-os e introduz as informações relativas ao edifício na base de dados. De seguida, o servidor procede ao armazenamento do ficheiro da planta para que esta possa ser disponibilizada. Quando é feito um pedido de obtenção da planta (1), o servidor obtém os dados da planta armazenada e devolve-os à aplicação cliente.

Plataforma de submissão das plantas

Permite ao utilizador fazer o envio das plantas dos edifícios para o servidor através de uma aplicação Web. Trata-se de uma plataforma dedicada à submissão de um ficheiro OSM com os dados da planta de um piso. A plataforma de submissão das plantas serve de interface com o servidor para que a planta possa ser devidamente armazenada no sistema. Os dados de cada planta são processados e armazenados. Isto permite associar os dados das plantas com os dados na base de dados. Além disso é armazenado o ficheiro da planta do piso no servidor.

É suposto o utilizador adicionar uma planta a partir de um ficheiro do seu computador e fazer a submissão da mesma. O processo de criação da planta requer a utilização da ferramenta JOSM para o desenho da planta e criação do ficheiro OSM.

API Android

O objetivo é que a planta do piso seja apresentada em qualquer dispositivo Android. Para isso, a API deverá receber e interpretar os dados da planta de forma a realizar o *rendering* da planta no dispositivo Android.

A API Android permite que a solução das plantas dos edifícios seja facilmente integrada em aplicações Android.

4.4.1.2. Comunicação entre componentes

As interfaces de comunicação entre componentes, na Figura 4.2 distinguem-se por serem ligações HTTP ou SQL.

(1) Módulo das plantas dos edifícios ↔ API Android

(1) Define a comunicação HTTP entre o módulo das plantas dos edifícios e a API Android. A comunicação entre estes dois componentes permite a obtenção da planta do piso. A planta do piso é pedida por parte da API Android e o módulo das plantas dos edifícios fornece a planta do piso, caso ela exista, à API Android.

(2) Plataforma de submissão das plantas dos edifícios ↔ Módulo das plantas dos edifícios

Depois do utilizador criar a planta de cada piso, é necessário um serviço que permita ao utilizador a submissão das plantas. A ligação entre plataforma de submissão das plantas dos edifícios e o módulo das plantas dos edifícios permite que o utilizador envie as plantas dos edifícios para o servidor de uma forma simples e intuitiva. A submissão dos ficheiros OSM com as plantas de cada piso pode ser feita através da página *web*. Como resultado da submissão das plantas dos edifícios, os dados são armazenados numa base de dados.

(3) Módulo das plantas dos edifícios ↔ BD das plantas dos edifícios

A ligação SQL entre o módulo das plantas dos edifícios e a BD das plantas dos edifícios permitirá que os dados das plantas dos edifícios sejam recebidos, processados e armazenados na base de dados. Sempre que a aplicação móvel faz um pedido para obter os dados de uma planta, esses dados são obtidos através do módulo das plantas dos edifícios que obtém os dados na base de dados das plantas dos edifícios.

Essencialmente, o módulo das plantas dos edifícios interage com a base de dados para inserir ou consultar os dados das plantas, assim como para obter os ficheiros das plantas.

4.4.1.3. Protocolos de comunicação

A implementação de *web services* revelou-se a melhor solução para os protocolos de comunicação entre as componentes do sistema, depois de analisadas as soluções disponíveis. Os *web services* são componentes de uma aplicação, normalmente disponibilizados por um *web server* para vários tipos de aplicações disponíveis na *web*. Estes serviços são capazes de comunicar e trocar dados com outras aplicações e plataformas, além disso, resolvem o problema a interoperabilidade entre aplicações oferecendo uma forma de trocar dados entre as mesmas,

independentemente da linguagem de programação na qual estão implementadas.

Essencialmente, um *web service* apresenta as seguintes características:

- É um serviço que está disponível na internet ou numa *intranet* (rede privada);
- A troca de dados é feita através de linguagens estruturadas, por exemplo, o XML;
- Independência da linguagem de programação e do sistema operativo;
- Permite troca de dados com outros sistemas utilizando o protocolo HTTP, que facilita o acesso nos sistemas com *firewalls*, uma vez que as *firewalls* normalmente permitem o protocolo HTTP.

Os *web services* apresentam duas soluções de interface através de HTTP, nomeadamente SOAP (*Simple Object Access Protocol*) e RESTful (*Representational State Transfer*). Existem vários trabalhos desenvolvidos que debatem as diferenças entre os dois [64]–[67].

O protocolo SOAP tem a capacidade de fazer a troca de informação estruturada (em XML) num ambiente distribuído e descentralizado [68]. É utilizado XML como formato para troca de mensagens e os protocolos HTTP e SMTP da camada aplicacional são utilizados para negociação e transmissão de mensagens.

Este protocolo baseado em XML fornece uma *framework* básica de comunicação para *web services,* que se divide em três elementos*:*

- Um envelope que define a estrutura da mensagem e a forma de como esta deverá ser processada;
- Um conjunto de regras de codificação para expressar os tipos de dados definidos pela aplicação;
- Uma convenção para a representação dos pedidos e respostas aos procedimentos.

As principais características do protocolo SOAP são a extensibilidade (segurança e outras extensões estão em desenvolvimento), a neutralidade (o protocolo tem a capacidade para funcionar sobre qualquer protocolo de transporte, como por exemplo o TCP, UDP, HTTP ou SMTP) e a independência (uma vez que permite qualquer modelo de programação).

Os *web services* REST assentam num estilo de arquitetura diferente que especifica restrições, tais como uma interface uniforme, que quando é aplicada num *web service* introduz algumas propriedades desejáveis, como a performance e escalabilidade. Numa arquitetura REST, os dados e as funcionalidades são considerados recursos e são acedidos através de URIs (*Uniform Resource Identifiers*), tipicamente *links* na *web.* É possível fazer atuar sobre os recursos através de um conjunto simples de operações, definidas no protocolo HTTP (GET, POST, PUT e DELETE).

A arquitetura cliente/servidor utiliza um protocolo de comunicação *stateless*, onde clientes e servidores trocam representações de recursos pela utilização de uma interface e protocolo *standard* [69].

A abordagem RESTful apresenta vários princípios que permitem dar às aplicações mais simplicidade, leveza e rapidez:

- Identificação de recursos através de URIs – permite um espaço de endereçamento global para a descoberta de recursos e serviços.
- Interface uniforme – através das operações simples do HTTP, é possível realizar operações do tipo CRUD (*create, read, update, delete*) sobre os recursos. A operação PUT cria um novo recurso, que pode ser apagado com o DELETE. Por sua vez, o GET obtém o estado atual de um recurso e o POST atualiza o estado do recurso.
- Mensagens autodescritivas – os recursos são desacoplados da sua representação para que o seu conteúdo seja acedido através de vários formatos, tais como o HTML, XML, texto simples, PDF, JPEG, JSON, entre outros. Os metadados sobre o recurso estão disponíveis e podem ser utilizados para fazer controlo de *caching,* detetar erros de transmissão, negociar o formato de representação, e para realizar autenticação ou controlo de acesso.

Para aceder a um recurso específico, por exemplo, obter os dados de um cliente chamado Manuel, a operação é do tipo GET, e o URI é o seguinte:

- http://exemplo.com/clientes/Manuel

No URI, "clientes" especifica o recurso que se pretende consultar, e o nome do cliente, "Manuel" refere-se ao cliente cujo nome tenha o valor "Manuel". Como resultado são devolvidas as informações do cliente Manuel.

Normalmente o formato de dados utilizado nos *web services* RESTful é o JSON (*JavaScript Object Notation*) caracterizado por ser um formato de dados facilmente convertido para outros formatos e é simples para os humanos lerem e escreverem. Além disso é fácil as máquinas fazerem o *parsing* e geração deste tipo de dados. [70]

Os formatos de dados associados ao SOAP e ao REST são XML e JSON respetivamente. Comparando a solução REST+JSON com SOAP+XML em [71], é possível obter a Tabela 4.1.

Tabela 4.1: Comparação entre a abordagem REST+JSON e SOAP+XML

	REST+JSON	**SOAP+XML**
Tamanho (quantidade de dados)	Menor - menos dados enviados, uma vez que o tipo de dados JSON apresenta menos	Maior - mais dados enviados porque o formato de dados XML apresenta *overhead* de *markup*,
Eficiência	Maior - o *parsing* do JSON é mais rápido comparando com o XML, o que requer menos tempo de CPU além da facilidade em extrair e converter os dados.	Menor - como o XML apresenta as TAGs é necessário processa-las e por isso o *parsing* de XML é mais lento, sendo por isso menos eficaz.
Caching	Apresenta tempos de resposta melhorados e de carregamento do servidor devido ao suporte para *caching.* Como os pedidos podem ser feitos através de GET, PUT e DELETE, é feito o *caching* dos dados, uma vez que estes pedidos são idempotentes [72].	Não suporta mecanismos de *caching*, uma vez que quando é utilizado o HTTP como mecanismo de transferência, os pedidos são enviados via pedidos HTTP POST, como a operação POST não é idempotente, não é feito o *caching* dos dados [72].
Pedidos	São transmitidos na forma de um URI: • Apresentam um limite no seu comprimento; • Pode utilizar campos de entrada.	Os pedidos são transmitidos em XML.

Tendo em conta as características dos *web services* comparadas na Tabela 4.1 e a análise de alguns trabalhos [64][67][71], a abordagem REST+JSON é mais vantajosa principalmente porque é mais adequada para aplicações móveis que necessitam de obter dados de *web services* onde não existe necessidade para a estrutura pesada do XML. Por todas essas razões e sabendo que a aplicação móvel é um dos principais elementos da arquitetura do sistema, será utilizada a abordagem REST+JSON na implementação dos *web services* para a comunicação entre os diferentes componentes da arquitetura.

4.4.1.4. Modelo de dados

Uma tarefa importante quando se faz o desenho de um sistema é a definição do modelo de dados uma vez que este apresenta toda a informação que será necessária para que o sistema funcione da melhor forma e cumpra os requisitos definidos.

Quando se define um modelo de dados é necessário eliminar a redundância ao máximo nos casos em que surge informação duplicada permitindo assim um modelo simples, apenas com a informação necessária. Disto resulta o modelo mais adequado, em que a informação está organizada de forma a suportar os dados das aplicações. O objetivo é desenvolver bases de dados capazes de lidar com os dados das aplicações implementadas de uma forma simples com a redundância minimizada, permitindo alterações e adaptação das mesmas para utilização noutras aplicações.

O modelo relacional foi escolhido para a implementação das bases de dados. Este modelo baseia-se em lógica e na teoria de conjuntos, introduzindo dois conceitos fundamentais: entidade e relação. Uma entidade é definida com vários dados que são definidos inicialmente. Quando se constrói a tabela, identificam-se os dados da entidade. Um registo na tabela é a atribuição de valores a uma entidade. Uma relação indica a forma como os registos de uma tabela estão associados a registos de outras tabelas.

Base de dados

No âmbito das plantas dos edifícios, surge um conjunto de entidades que se relacionam entre si:

- Operadores (são entidades que englobam um conjunto de áreas de edifícios, por exemplo: Universidade do Minho):
    - Identificador do operador;
    - Nome do operador.
- Áreas (um operador normalmente abrange um conjunto de áreas ou uma só área):
    - Identificador da área;
    - Identificador do operador;
    - Nome.
- Edifícios (pertencem a uma área específica):
    - Identificador do edifício;
    - Identificador da área;

  - Nome;
  - Descrição;
  - País;
  - Cidade;
  - Rua;
  - Código-postal.
- Pisos (representam cada um dos pisos de um edifício):
  - Identificador do piso;
  - Identificador do edifício;
  - Nome;
  - Diretoria da planta do piso armazenada no sistema de ficheiros do servidor.
- Espaços/Salas (cada piso apresenta um conjunto de salas/espaços):
  - Identificador do espaço;
  - Identificador do piso;
  - Nome;
  - Identificador do espaço no ficheiro da planta.

4.4.2. Informação necessária para o bom funcionamento do sistema

A informação necessária para a solução das plantas dos edifícios está associada às características de cada edifício, mas também requer informação detalhada que descreve a estrutura interior do edifício. Para o bom funcionamento do sistema são necessárias as seguintes informações:

- Utilizadores (nome, e-mail, *password* e operador);
- Plantas dos edifícios (plantas de cada piso do edifício com as respetivas salas/espaços devidamente identificados);
- Informações de cada sala/espaço interior (nome, piso, tipo de espaço interior).

4.4.3. Requisitos técnicos

Apesar de desempenhar uma função de suporte, a solução para as plantas apresenta requisitos de carácter funcional e não funcional.

Requisitos funcionais

- Receber, processar e armazenar plantas dos edifícios;

- Suporte para plantas de múltiplos operadores (instituições, universidades, etc);
- Disponibilizar plantas dos edifícios sempre que pedido através de *web services*;
- Suporte à calibração e visualização da posição estimada através da apresentação da planta do edifício num dispositivo Android com capacidade de vários modos de interação (*zoom* e *pan*).

Requisitos não funcionais

- Rapidez na obtenção da planta do servidor;
- Desempenho e fluidez na apresentação da planta nos dispositivos Android;
- Comunicação entre componentes do sistema via Internet.

4.5. Implementação da solução das plantas dos edifícios

Cada componente do sistema é analisada detalhadamente, com abordagem nas tecnologias e modos de funcionamento de cada componente. Nesta secção é apresentada a implementação da solução das plantas dos edifícios onde também é explicado com detalhe o funcionamento do sistema.

4.5.1. Criação de uma planta de interiores

Tal como referido anteriormente, a implementação de uma solução para as plantas dos edifícios depende da ferramenta JOSM para auxiliar no desenho da planta.

A criação da planta de um piso é uma tarefa importante que requer alguma dedicação e um utilizador profissional que seja capaz de criar uma planta que cumpra os requisitos para que seja devidamente processada e desenhada num dispositivo móvel Android.

O utilizador necessita de uma imagem da planta do piso (pode ser a planta de emergência) e um computador com a ferramenta JOSM instalada.

Deverá ter-se em consideração os seguintes cuidados ao fazer o mapeamento dos espaços interiores:

- Alinhar devidamente a imagem da planta com a estrutura do edifício no mapa;
- Não desenhar elementos dentro de outros elementos interiores, com exceção da estrutura exterior do edifício;
- Desenhar todos os elementos do espaço interior: salas, áreas, corredores, escadas, portas e elevadores;

- Atribuir nomes aos espaços interiores (todas as salas/escritórios deverão ter um nome, sendo que não é necessário atribuir nomes a corredores ou áreas de escadas).

O Anexo D – Tutorial Josm apresenta o processo de criação de uma planta com a ferramenta JOSM com detalhe.

4.5.2. Servidor

A Figura 4.3 demonstra a componente das plantas dos edifícios do servidor e a tecnologia que a caracteriza. O módulo das plantas dos edifícios está implementado em PHP com interface REST para comunicação com aplicações.

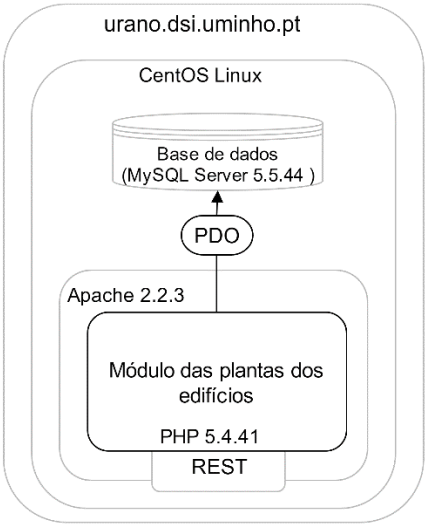


Figura 4.3: Servidor - componentes e tecnologias

4.5.2.1. Base de dados

Na Figura 4.4 está representado o diagrama de entidades e relacionamentos associado às plantas dos edifícios. O modelo tem uma estrutura hierárquica que tem como entidade principal o operador. O operador representa uma entidade que engloba conjuntos de edifícios. Cada área pertence a um operador e está associada a um conjunto de edifícios. Por sua vez, um edifício pode ter vários pisos constituídos por vários espaços interiores. Associado a cada piso existe o caminho completo incluindo o nome do ficheiro onde está armazenada a planta no servidor.

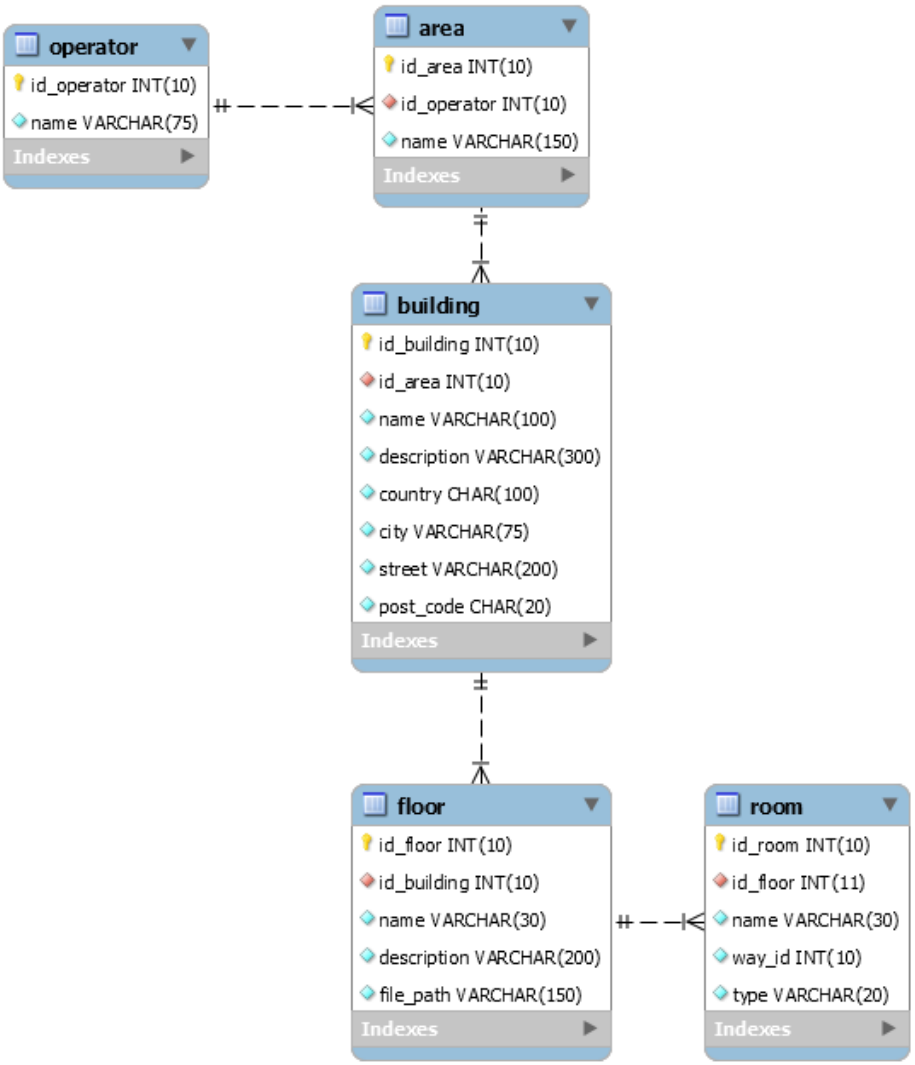


Figura 4.4: Modelo de dados das plantas dos edifícios

Tabela Operador (operator)

A tabela operador é importante porque possui todos os edifícios que se distribuem por várias áreas. Um operador é caracterizado pelo seu identificador e por um nome. No caso da universidade, a Universidade do Minho é o operador que tem domínio sobre os *campi* de Gualtar e Azurém.

Tabela área (area)

Uma área está sempre associada a um operador sendo que um operador pode ter sempre uma ou mais áreas. No caso da Universidade do Minho, as suas áreas são o Campus de Gualtar ou o Campus de Azurém, por exemplo. A área tem um identificador e um nome além do identificador do operador.

Tabela edifício (building)

Da mesma forma que uma área está associada a um operador, um edifício está associado a uma área sendo que uma área pode ter vários edifícios. Um edifício apresenta vários dados que são armazenados na tabela: identificador do edifício, identificador da área, nome, descrição, cidade, país, rua e código-postal. Estas informações do edifício são relativas à sua morada.

Tabela piso (floor)

Cada piso faz parte de um edifício, por isso existe uma relação entre ambos. As informações armazenadas sobre cada piso são: identificador do piso, identificador do edifício, nome e a diretoria da planta do piso que se encontra armazenada no sistema de ficheiros no servidor.

Tabela espaço/quarto (room)

Existe um conjunto de espaços interiores num piso, podem ser de vários tipos:

- "area";
- "corridor";
- "room" (inclui os espaços fechados tais como salas, escritórios, quartos e elevadores);
- "stairs".

Cada espaço interior associado a um piso caracteriza-se pelo seu tipo, pelos identificadores de piso e do espaço, assim como o identificador único (way_id) que identifica o elemento *indoor* no ficheiro OSM.

4.5.3. Plataforma de submissão da planta para o servidor

A submissão de plantas é feita através de uma página web com a capacidade de submissão de um ficheiro OSM. Apresentado na Figura 4.5, o formulário recebe um ficheiro que é enviado para o servidor.

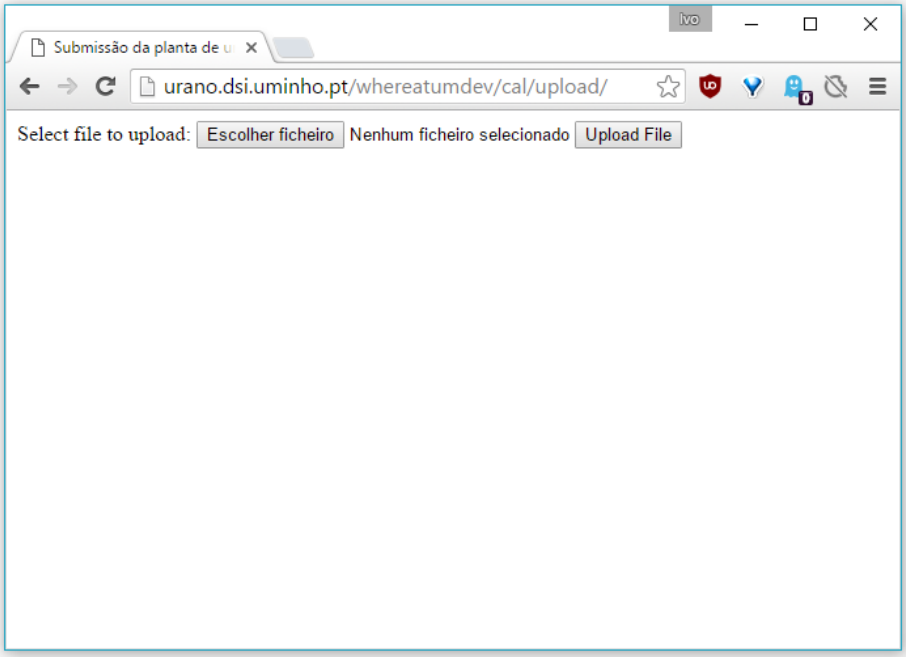


Figura 4.5: Formulário de submissão do ficheiro de uma planta

Após a receção do ficheiro, o servidor processa o seu conteúdo com o intuito de verificar se a planta submetida já existe no sistema. Esta verificação também verifica se as entidades operador, área e edifício já existem no sistema. Quando não existem, estas entidades são adicionadas à base de dados. Por fim, o ficheiro submetido é armazenado na diretoria "upload" com o nome "id_piso.osm", onde o id_piso é obtido a partir da entrada na base de dados referente à planta do piso.

Por fim, é apresentado o resultado relativo à submissão da planta com várias informações sobre a planta adicionada, como é mostrado na Figura 4.6.

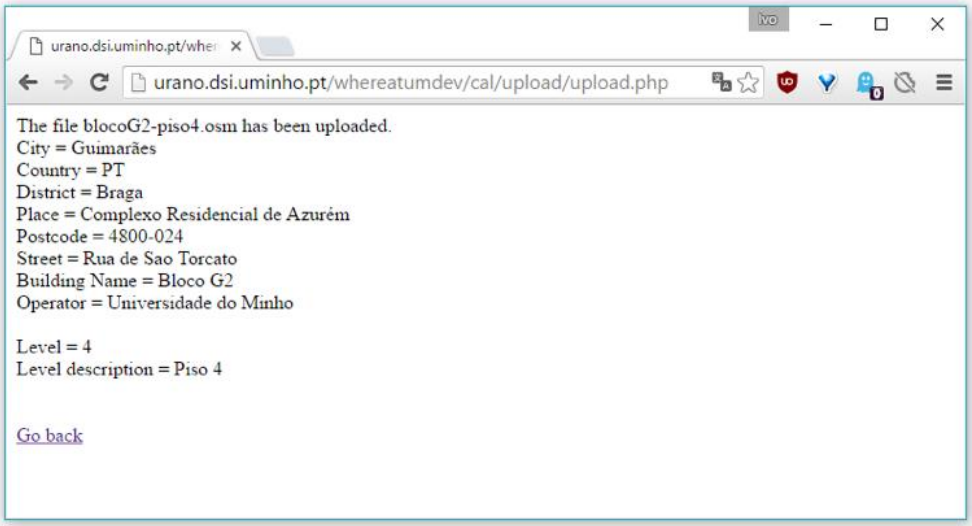


Figura 4.6: Resultado da submissão de uma planta

4.5.4. API Android

Os dados das plantas estão no formato XML e apresentam todas as informações relativas à estrutura do piso e aos espaços interiores que o constituem. Para apresentar a planta no ecrã foi necessário desenvolver uma API capaz de fazer o processamento dos dados e apresenta-los no ecrã do dispositivo. Uma planta deverá ser criada de acordo com o tutorial apresentado no Anexo D – Tutorial Josm, para que seja devidamente apresentada no ecrã do dispositivo.

A API Android, denominada de *IndoorAPI*, é constituída por várias classes Java, tendo a classe principal (BuildingView.java) que utiliza as outras classes para auxiliar na representação gráfica da planta. As classes que constituem a API são as seguintes:

- BuildingView – Responsável pelo desenho da planta no ecrã;
- ScaleListener – Lida com os gestos do utilizador para aumentar ou diminuir o zoom;
- IndoorPart – Representa um espaço interior (área, corredor, sala, etc);
- IndoorNode – Representa um nó (por exemplo uma porta);
- PointNode – Está associado a um ponto que é desenhado no ecrã.

Classe BuildingView.java

A classe principal da API Android é a classe *BuildingView* e trata-se de uma *custom view* uma vez que estende a classe *View* do Android, permitindo várias funcionalidades. A classe *BuildingView* encapsula um conjunto de funcionalidades provenientes da classe *View* oferecendo facilidade de utilização da interface e utilização eficiente de memória e CPU. Sendo uma *custom view*, a classe BuildingView deverá [73]:

- Estar em conformidade os *standards* Android;
- Fornecer atributos personalizados que funcionam com layouts XML Android;
- Enviar eventos de acessibilidade;
- Ser compatível com várias plataformas Android.

Fazendo o *override* ao método onDraw é possível desenhar o ecrã a partir do parâmetro recebido que é um objeto do tipo *Canvas*. A classe *Canvas* define métodos para desenhar texto, linhas, bitmaps e outras primitivas gráficas [74]. A planta é apresentada com recurso ao método onDraw que permite criar a interface pretendida (planta do piso), através de linhas e texto. O método onDraw é despoletado sempre que é feita uma interação por parte do utilizador (*zoom*, *pan*), ou quando se pretende adicionar elementos na representação da planta, pode chamar-se o método onDraw.

O método drawBuilding da classe BuildingView é chamado dentro do método onDraw e engloba todo o processamento dos dados obtidos na planta do piso, e o desenho dos elementos da planta (espaços interiores, nomes de cada espaço, portas, etc) no ecrã do dispositivo.

4.5.4.1. Funcionamento da API

O funcionamento desta API é o seguinte: os dados são devidamente processados, criando as estruturas de dados que descrevem cada espaço interior do piso (quartos/salas, áreas, corredores e escadas). Cada ponto é convertido do sistema de coordenadas (latitude, longitude) para o sistema de coordenadas (x, y) do ecrã Android (o canto superior esquerdo do ecrã é a origem do referencial). Por fim, cada espaço interior é desenhado sendo que elementos diferentes são desenhados com cores diferentes, tendo em conta as suas características. A estrutura do edifício é desenhada no centro do ecrã e a solução desenvolvida permite várias interações ao utilizador: *zoom* (ampliar a planta, Figura 4.8), *pan* (deslocar a planta). Estas interações permitem que o utilizador desloque e amplie a planta de acordo com o que pretende. A ampliação é vetorial, e por isso a estrutura do piso e os elementos desenhados são apresentados com detalhe.

O resultado do desenho de uma planta é mostrado na Figura 4.7 na qual está apresentada a planta do piso 0 do Bloco G2 do Complexo Residencial de Azurém.

Figura 4.7: Desenho da estrutura de um piso sem zoom

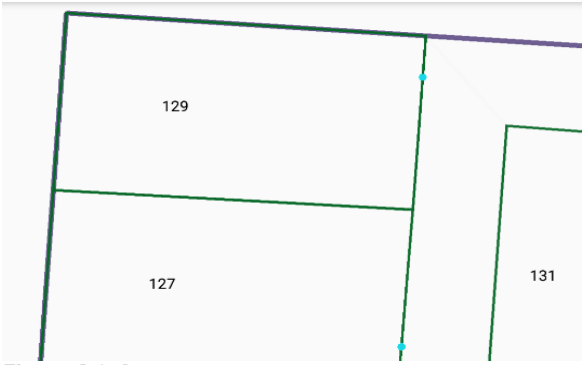


Figura 4.8: Desenho da estrutura interior de um piso com zoom

Os dados da planta são obtidos através da comunicação com o servidor. Fazendo um pedido através de *web services* para obtenção da planta é possível obter os dados da planta. De seguida a aplicação recebe-os e com o suporte da API é capaz de fazer o *rendering* da planta no ecrã do dispositivo.

As cores de cada elemento do espaço interior são:

- Roxo – estrutura externa do edifício;
- Verde – salas/quartos;
- Laranja – escadas;
- Azul – portas;
- Rosa – elevadores;
- Castanho – paredes;
- Transparente (sem cor) – corredores.

4.5.4.2. Funções implementadas

A API fornece um conjunto de funções sem as quais não seria possível apresentar a planta num dispositivo Android. De seguida está apresentada uma lista com algumas das funções implementadas na classe BuildingView.java, fundamentais para o desenho das plantas no ecrã:

- `public HashMap<String,Element> readOSMXMLfileFromString(String file)` – Tem a função de fazer o *parsing* do conteúdo XML da planta do piso e devolve os dados sob a forma de um objeto `HashMap<String,Element>` que apresenta todos os elementos do ficheiro XML.
- `public boolean onTouchEvent(MotionEvent event)` – Este método é despoletado de cada vez que o utilizador interage com o ecrã, para tocar, fazer *zoom* ou *pan*. A gestão de interações com o utilizador é feita no onTouchEvent que permite fazer toque normal, *zoom* e *pan*.
- `public int getScaleFactor()` – através das dimensões do ecrã do dispositivo, é possível determinar o fator de escala com o intuito de ajustar a planta ao ecrã, sem desperdiçar espaço no ecrã.
- `public void onDraw(Canvas canvas)` - Recebe um objeto Canvas que permite desenhar no ecrã. Este método permite desenhar as plantas no ecrã do dispositivo.
- `public void drawBuilding(Canvas canvas)` - Chamado no método onDraw, o método drawBuilding inclui o desenho do edifício, a estrutura interior, e as tags de texto que identificam cada divisão. Este método recebe os dados obtidos do ficheiro da planta a partir de objetos java que representam cada espaço interior. Neste método são desenhadas todas as estruturas a partir de linhas, e é feita a distinção entre cada espaço para que seja desenhado com a cor respetiva.

4.6. Comunicação entre componentes do sistema (*web services*)

A comunicação com o servidor deverá ser segura utilizando o protocolo HTTPS (essencialmente é uma implementação do protocolo HTTP sobre uma camada adicional de segurança que utiliza o protocolo SSL/TLS). Porém, o servidor utilizado na implementação do projeto não apresenta certificados assinados por uma CA (*Certification Authority*), apenas possui certificados auto-assinados o que não tem qualquer valor do ponto de vista criptográfico. Tendo em conta a comunicação com o servidor e os dados trocados com o mesmo, considera-se que não é crítico o uso de uma conexão totalmente segura, apesar disso este é um fator importante para a segurança de todo o sistema.

Existem várias funcionalidades que dependem da comunicação com o servidor, que permitem obter várias informações sobre os edifícios e as plantas disponíveis. Os pedidos desenvolvidos *web services* (consultar Anexo C) permitem obter várias informações:

- Obter lista dos operadores – através deste pedido é possível obter uma lista no formato de dados JSON que apresenta os operadores existentes;
- Obter a lista de áreas de um operador – dado o operador como parâmetro é devolvida a lista de áreas desse operador;
- Obter a lista de edifícios de uma área – através do identificador da área e do operador é devolvida a lista de edifícios dessa área;
- Obter a lista de pisos de um edifício – devolve a lista de pisos de um edifício em JSON tendo em conta o identificador do edifício, da área e do operador;
- Obter a planta de um piso - devolve a planta do piso em questão tendo em conta o identificador do piso, edifício, área e operador. O resultado é devolvido no formato de dados XML para ser processado e desenhado no ecrã do dispositivo móvel.

CAPÍTULO 5 –SISTEMA DE CALIBRAÇÃO

No presente capítulo é feita a abordagem inicial à aplicação de calibração partindo dos requisitos de alto nível e o desenho do sistema onde é apresentada a arquitetura.

Posteriormente é feita a descrição da implementação do sistema, onde é feita a análise a cada componente passando pelas tecnologias e funcionalidades de cada componente do sistema.

O sistema tem como objetivo permitir a calibração de uma forma simples e interativa dando *feedback* ao longo da construção do mapa de rádio de cada piso.

5.1. Requisitos

O sistema de calibração deverá ter por base uma arquitetura cliente-servidor, tendo por isso uma aplicação móvel (lado do cliente) e um sistema central (lado do servidor).

É expectável que o sistema permita o registo e autenticação de utilizadores de múltiplos operadores para que estes possam fazer a calibração de espaços associados a diversos operadores. Através de uma aplicação móvel, os utilizadores deverão ter capacidade para a recolha de *fingerprints* Wi-Fi para a construção do mapa de rádio. Ao longo do processo de calibração, o sistema deverá dar *feedback* relativamente ao estado do mapa de rádio, auxiliando na construção do mesmo. Este processo informa o utilizador dos locais que ainda não foram calibrados ou ainda não estão devidamente calibrados. Além disso, permite auferir o estado da cobertura Wi-Fi através da deteção dos locais com mais sobreposição de canais e com mais densidade de pontos de acesso.

A interface da aplicação tem bastante importância porque é o meio utilizado para interagir com o utilizador, daí a relevância da interface ser simples, funcional e apelativa para o utilizador. Esta é uma forma de manter o utilizador interessado e motivado a utilizar as aplicações. Todas as interfaces da aplicação de calibração deverão estar em inglês uma vez que é uma língua universal que abrange uma quantidade enorme de pessoas e permite que um utilizador com conhecimentos básicos de inglês em qualquer parte do mundo tenha possibilidade de usufruir da aplicação sem dificuldades relativas ao idioma.

O sistema central, composto pelos componentes do lado do servidor, tem como principais funções processar e armazenar a informação recebida, atender a pedidos por parte da aplicação móvel e criar os mapas de rádio através da recolha de *fingerprints* obtidas no dispositivo móvel. Como se trata das componentes do lado do servidor, o sistema central deve estar sempre

disponível garantindo assim a interação com os utilizadores. Os serviços de comunicação, envio e receção de dados deverão ser assegurados por serviços disponibilizados pelo servidor.

5.2. Desenho do sistema de calibração

O desenho do sistema de calibração aborda vários aspetos necessários para realizar a análise dos requisitos e a arquitetura. Da arquitetura resulta cada componente do sistema e como se relacionam entre si.

5.2.1. Arquitetura do sistema

Nesta secção é feita uma análise às soluções para implementação da arquitetura. De seguida é abordada a arquitetura do sistema de calibração, inicialmente com a descrição da arquitetura geral e posteriormente com uma descrição mais detalhada da arquitetura, dos componentes que a constituem e das comunicações entre os componentes do sistema.

5.2.1.1. Arquitetura geral do sistema

A arquitetura do sistema é constituída essencialmente por um dispositivo móvel com a respetiva aplicação e o sistema central que representa o lado do servidor no qual são disponibilizados todos os serviços necessários para o sistema de calibração: suporte à aplicação, fornecimento das plantas dos edifícios e construção dos mapas de rádio.

A arquitetura segue o modelo cliente-servidor onde o cliente faz pedidos ao servidor e a comunicação entre ambos é feita via pedidos HTTP. Na Figura 5.1 distinguem-se dois tipos de dispositivos: o dispositivo móvel (*tablet*) e o servidor.

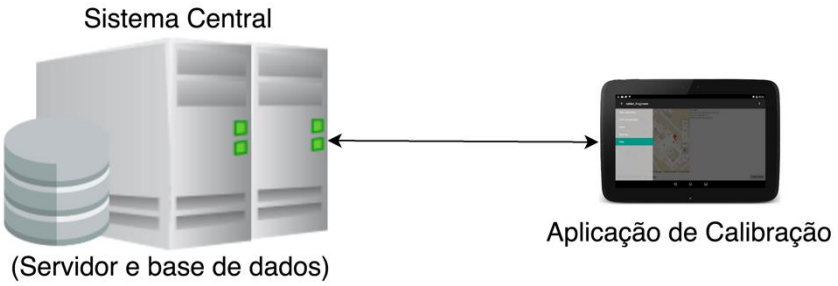


Figura 5.1: Arquitetura geral do sistema de calibração

Na Figura 5.2 está representada a arquitetura do sistema com todos os constituintes e as ligações entre os mesmos. De uma forma geral, o *client-side* é representado pela aplicação

instalada no dispositivo móvel, a aplicação de calibração. O *server-side* apresenta um servidor composto por três módulos sendo que um tem como objetivo oferecer o suporte à aplicação móvel, outro é responsável pela construção dos mapas de rádio e por fim, existe um módulo dedicado ao processamento e armazenamento das plantas dos edifícios. O módulo de suporte à aplicação utiliza uma base de dados de suporte à aplicação e o módulo de construção dos mapas de rádio utiliza uma base de dados dedicada à construção desses mapas de rádio. Por sua vez, o módulo das plantas dos edifícios tem uma base de dados distinta para armazenar os dados das plantas de interiores.

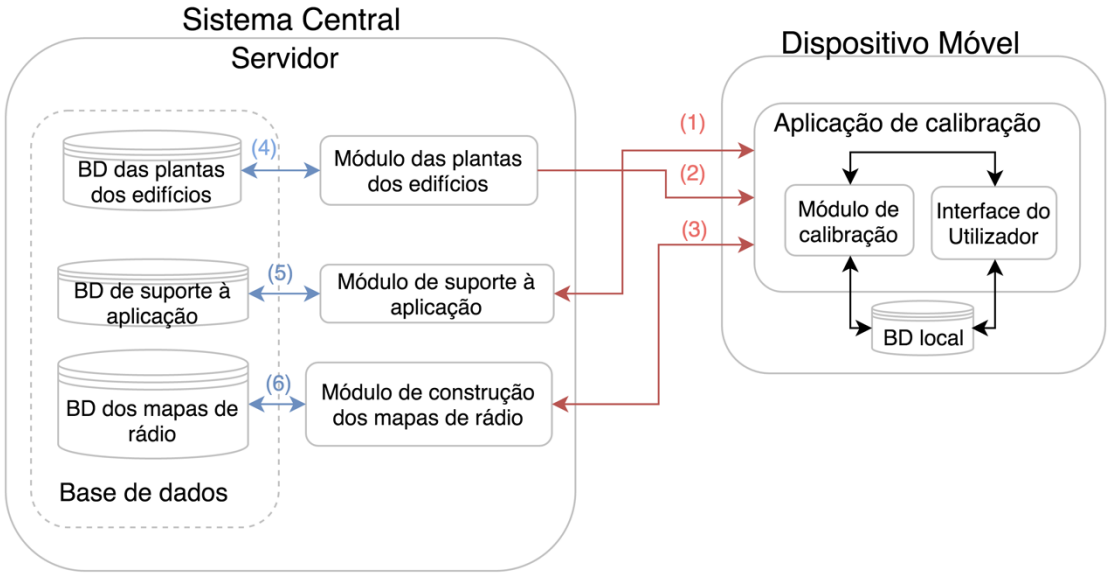


Figura 5.2: Arquitetura do sistema de calibração - componentes

Cada componente da arquitetura exerce uma função específica no sistema:

- **Aplicação de calibração**
  - Módulo de calibração: responsável pela análise do ambiente rádio através do *scan* dos pontos de acesso e respetivos níveis de sinal através da interface WiFi do dispositivo. Os valores recolhidos são armazenados na base dados local.
  - Interface do utilizador: o módulo de visualização apresenta e recolhe informações através da interface do utilizador.
  - BD local: permite armazenar temporariamente no dispositivo, os dados das *fingerprints* recolhidas aquando do processo de calibração sem conectividade à internet.

- **Sistema central**
    - Módulo das plantas dos edifícios: fornece as plantas dos edifícios à aplicação móvel através do envio dos dados de cada planta sempre que é pedido.
    - Módulo de suporte à aplicação: deve fornecer à aplicação os serviços necessários para que o bom funcionamento da aplicação seja garantido. Este módulo recebe e processa dados provenientes da aplicação e quando recebe um pedido devolve como resposta os dados necessários.
    - Módulo de construção dos mapas de rádio: essencial para a construção dos mapas de rádio, recebe os dados das *fingerprints* recolhidos no dispositivo, processa-os e armazena-os na respetiva base de dados.
    - BD de suporte à aplicação: armazena os dados dos utilizadores da aplicação de calibração dando suporte à aplicação;
    - BD dos mapas de rádio (*fingerprints*): garante o armazenamento dos dados associados aos mapas de rádio.
    - BD das plantas dos edifícios: permite armazenar os dados das plantas de cada piso dos edifícios fornecendo estes dados sempre que for necessário, através do módulo das plantas dos edifícios.

5.2.1.2. Componentes do sistema de calibração

Cada componente do sistema desempenha uma função distinta, garantindo encapsulamento nas operações que são realizadas, evitando erros e falhas no sistema, permitindo deste modo um melhor funcionamento do sistema.

Aplicação de calibração

A aplicação móvel, juntamente com o dispositivo móvel são elementos fulcrais do sistema uma vez que este depende dos valores medidos pelo dispositivo e da aplicação para garantir a interação entre o utilizador e o sistema.

Na secção 5.1 foram definidos os requisitos que a aplicação de calibração deverá implementar e dar ao utilizador capacidade para efetuar as tarefas com facilidade.

O intuito principal da aplicação de calibração é facilitar no processo de calibração, de forma a fornecer ao utilizador as ferramentas adequadas para realizar o processo de recolha de *fingerprints* de forma mais simples e rápida. Para o sistema de posicionamento, a aplicação móvel

é a componente que permite a análise do ambiente de rádio do espaço interior através da interface Wi-Fi do dispositivo e possibilita a construção do mapa de rádio do espaço em questão.

A recolha de *fingerprints* é feita quando o utilizador indica explicitamente o local em que se encontra dentro do edifício e a aplicação faz a recolha dos dados do *scan* aos pontos de acesso disponíveis. Sugere-se que seja feita a recolha de *fingerprints* em vários pontos do mesmo espaço interior para que a localização seja estimada com mais precisão. As *fingerprints* são enviadas para o servidor e são devidamente armazenadas na base de dados sendo associadas a um local.

Servidor

Constituído por três módulos e três bases de dados conceptuais, o servidor recebe e trata pedidos por parte da aplicação móvel, recebe e armazena as plantas dos edifícios e participa na construção do mapa de rádio de cada espaço ao receber as *fingerprints*.

Para o servidor cumprir os requisitos necessita de possuir as seguintes características: ter disponibilidade para atender vários pedidos provenientes de dispositivos diferentes; capacidade para receber vários pedidos ao mesmo tempo e responder aos mesmos; ter implementados protocolos de comunicação capazes de manter a rapidez nas interações entre o servidor e os outros componentes.

Este é o elemento central da arquitetura que mantém ligações com os outros componentes do sistema.

Será utilizada a tecnologia PHP na implementação do servidor devido à sua simplicidade, flexibilidade (apresenta várias *frameworks*), escalabilidade e por ser uma tecnologia *open source*. Além disso esta tecnologia é facilmente integrável com diversas tecnologias devido à sua simplicidade no tratamento de dados (compatibilidade com SQL, JSON ou até HTML).

Módulo das plantas dos edifícios

Para a construção dos mapas de rádio é necessário um serviço que disponibilize as plantas dos edifícios. Deste modo, o módulo das plantas dos edifícios tem as seguintes funcionalidades:

- Receber, processar e armazenar as plantas e informações dos edifícios (na base de dados das plantas dos edifícios);
- Disponibilizar as plantas dos edifícios quando recebe um pedido.

Módulo de suporte à aplicação

O módulo de suporte à aplicação tem a função de receber e responder a pedidos da aplicação móvel com o objetivo de garantir o bom funcionamento da aplicação. Este módulo é responsável por:

- Registar e autenticar os utilizadores;
- Alterar a *password* do utilizador;
- Processar e armazenar os dados pessoais e de configuração dos utilizadores na base de dados do servidor.

Apesar de não ter influência direta no sistema de posicionamento, o módulo de suporte à aplicação garante o funcionamento dos serviços da aplicação e mantém os dados dos utilizadores armazenados.

Módulo de construção dos mapas de rádio

O módulo de construção dos mapas de rádio conecta o servidor à aplicação móvel. Este módulo intervém na construção dos mapas de rádio fazendo a recolha dos dados enviados (*fingerprints*) pela aplicação móvel e armazena-os na base de dados do servidor.

Os dados recolhidos são devidamente processados e armazenados da respetiva estrutura de dados. Cada amostra recolhida é associada ao espaço interior no qual os dados foram recolhidos e ao par de coordenadas absolutas (latitude, longitude) que identificam o local exato onde a amostra foi recolhida.

Este módulo também é responsável pela disponibilização dos dados de calibração de cada piso, para que seja possível verificar o estado de calibração de um piso na aplicação de calibração.

5.2.1.3. Comunicação entre componentes da arquitetura

Na Figura 5.2 juntamente com os componentes da arquitetura estão as ligações entre eles. Cada uma das ligações representa a comunicação entre os dois elementos, ao todo existem oito ligações entre elementos da arquitetura.

(1) Aplicação de calibração ↔ módulo de suporte à aplicação móvel

(1) Define a comunicação HTTP entre a aplicação de calibração e o módulo de suporte à aplicação. A comunicação entre estes dois componentes permite vários serviços descritos na tabela Tabela 5.1.

Tabela 5.1: Serviços disponibilizados através da comunicação entre a aplicação móvel e o módulo de suporte à aplicação móvel

Serviço	Descrição
Registo do utilizador	Na tarefa de registo, o utilizador introduz os seus dados de registo, que incluem um *email* e uma *password*. Os dados são enviados para o módulo de suporte à aplicação móvel que procede ao registo do utilizador.
Autenticação do utilizador	A autenticação do utilizador é feita quando o módulo de suporte à aplicação móvel faz a correspondência dos dados de autenticação introduzidos pelo utilizador com os dados armazenados associados a esse utilizador.
Alteração dos dados pessoais	Os dados pessoais introduzidos aquando do registo podem ser alterados pelo utilizador a qualquer momento.

(2) Aplicação de calibração ↔ módulo das plantas dos edifícios

Tanto na componente de visualização, como na componente de calibração de um edifício é necessário ter acesso à planta do piso em questão. Essa funcionalidade é disponibilizada através da ligação (2) que representa a comunicação HTTP entre a aplicação móvel e o módulo das plantas dos edifícios.

A Tabela 5.2 sintetiza o serviço disponibilizado pela ligação (2).

Tabela 5.2: Serviços disponibilizados através da comunicação entre a aplicação móvel e o módulo das plantas dos edifícios

Serviço	Descrição
Disponibilização da planta do piso	A aplicação móvel através de um pedido indica a planta pretendida. O módulo das plantas dos edifícios recebe o pedido e procede ao envio das informações da planta em questão. Isto é efetuado quando o utilizador pretende recolher *fingerprints* para fazer a construção do mapa de rádio.

(3) Aplicação de calibração ↔ módulo de construção dos mapas de rádio

A comunicação HTTP (3) entre a aplicação móvel e o módulo de construção dos mapas de rádio é fundamental para o sistema de posicionamento, porque é através desta ligação que os dados de posicionamento são recolhidos e analisados, tal como é descrito na Tabela 5.3.

Tabela 5.3: Serviços disponibilizados através da comunicação entre a aplicação móvel e o módulo de construção dos mapas de rádio

Serviço	Descrição
Calibração de local (Envio de *fingerprints*)	Quando o utilizador faz a calibração de um local através da recolha de *fingerprints,* os dados são enviados para o módulo de construção dos mapas de rádio, que realiza o processamento das *fingerprints* e procede ao seu armazenamento.
Obter lista de *fingerprints* do piso	Para apresentar os pontos anteriormente calibrados, o módulo de suporte disponibiliza a lista de locais anotados para apoiar o utilizador no processo de calibração.

(4) Módulo das plantas dos edifícios ↔ BD das plantas dos edifícios

Esta ligação SQL foi descrita devidamente na secção 4.4.1.2.

(5) Módulo de suporte à aplicação móvel ↔ BD de suporte à aplicação

As interações representadas em (5) estão associadas à ligação SQL entre o módulo de suporte à aplicação móvel e a base de dados de suporte à aplicação. Com esta ligação, o servidor tem capacidade para realizar qualquer operação de inserção, remoção ou seleção de dados da base de dados. Para autenticar um utilizador, é necessário fazer a correspondência dos seus dados com os dados armazenados. No caso da alteração das configurações do utilizador, também é necessário fazer as alterações nos dados que estão armazenados na base de dados. As operações sobre a base de dados são feitas quando o módulo de suporte à aplicação móvel recebe um pedido por parte da aplicação e atende esse pedido.

(6) Módulo de construção dos mapas de rádio ↔ BD dos mapas de rádio

A base de dados do servidor permite que o módulo de construção de mapas de rádio adicione nova informação de calibração. A seleção ou inserção de dados é feita pelo módulo de construção dos mapas de rádio diretamente na base de dados dos mapas de rádio.

5.2.1.4. Protocolos de comunicação

Tal como foi analisado anteriormente, os protocolos de comunicação entre componentes do sistema serão baseados em *web services* RESTful utilizando JSON como formato de dados.

5.2.2. Bases de dados

O servidor apresenta uma base de dados, apesar de existirem três bases de dados conceptuais em 5.2.1.2. onde foi feita a distinção entre bases de dados para mostrar a forma como os dados serão armazenados. A base de dados do servidor tem como função armazenar as plantas e dados dos edifícios, a informação das *fingerprints* Wi-Fi recolhida no processo de calibração e as informações dos utilizadores, tendo assim a função de suportar a aplicação móvel e o motor de posicionamento. Esta base de dados será abordada detalhadamente no capítulo seguinte.

A base de dados do dispositivo tem o intuito de armazenar os dados localmente, para que estes sejam enviados para o servidor e para que permitam o modo de funcionamento *offline.*

O sistema depende das seguintes entidades e os seus respetivos dados associados para que funcione devidamente:

- Utilizador:
    - Identificador do utilizador;
    - Nome;
    - Email;
    - Password.
- Dispositivos do utilizador (representa os dispositivos do utilizador):
    - Identificador do utilizador;
    - Endereço MAC do dispositivo;
    - Nome do dispositivo;
    - Descrição.
- Localização (localização do utilizador associada ao espaço interior que este se encontra):
    - Identificador da localização;
    - Identificador do espaço;
    - Latitude;
    - Longitude.

- Informação das *fingerprints* recolhidas (cada *fingerprint* está associada a um espaço interior):
    - Identificador da *fingerprint;*
    - *Timestamp* registado pelo dispositivo no momento da recolha da *fingerprint*;
    - *Timestamp registado* pelo servidor aquando da receção da *fingerprint*;
    - Identificador da localização;
    - Identificador do dispositivo;
    - Identificador do utilizador;
- Pontos de acesso e respetivos níveis de sinal (uma *fingerprint* tem associado a si um conjunto de APs e os seus níveis de sinal):
    - Identificador do ponto de acesso;
    - Endereço MAC do ponto de acesso;
    - RSSI (nível de sinal medido pelo dispositivo);
    - SSID (*string* de texto que identifica a rede);
    - Canal (frequência de funcionamento do ponto de acesso);
    - Identificador da *fingerprint.*

5.2.3. Funcionamento do sistema

Inicialmente, o sistema pressupõe o registo de utilizadores, que ficam associados a um operador no momento do registo. O registo permite o acesso às funcionalidades de calibração e construção do mapa de rádio. Na aplicação os utilizadores podem selecionar a planta do edifício para o qual pretendem construir o mapa de rádio, ou consultar o estado atual do mesmo.

Depois da seleção da planta, o servidor fornece os dados relativos ao mapa de rádio assim como a planta do piso. Este passo permite que a planta seja apresentada na aplicação e permite verificar o estado atual do mapa de rádio através de vários tipos de vistas (densidade de amostras ou antiguidade das mesmas), que facilitam na deteção dos locais que ainda não foram calibrados. Assim os utilizadores podem recolher amostras nos pontos que ainda não foram calibrados e nos pontos que poderão ser recalibrados. No fim, o utilizador pode enviar os dados para o servidor, onde ficarão armazenados e poderão ser consultados para verificação do estado do mapa de rádio na aplicação de calibração.

Informação necessária para o funcionamento do sistema

O sistema depende de um conjunto de informações que são necessárias para que os serviços estejam disponíveis e funcionais. As informações necessárias para o funcionamento do sistema de calibração são:

- Informações dos utilizadores (e-mail, *password*, nome e operador);
- Plantas dos edifícios (plantas de cada piso do edifício com as respetivas salas/espaços devidamente identificados);
- Informações dos edifícios (morada completa, número total de pisos, piso mínimo e piso máximo, coordenadas geográficas, área, operador);
- Informações de cada sala/espaço interior (nome, piso, tipo de espaço interior);
- *Fingerprints* (informações dos pontos de acesso e respetivos níveis de sinal recolhidos pelo dispositivo e associados ao local do espaço interior).

5.2.4. Requisitos técnicos

A análise de requisitos engloba todos os constituintes do sistema e divide-se em requisitos funcionais e requisitos não funcionais.

Os requisitos funcionais estão relacionados com as interações do utilizador com o sistema através da aplicação de calibração onde estão as principais capacidades do sistema. Uma interação do utilizador com a aplicação desencadeia uma comunicação entre a aplicação e o servidor, por isso o servidor também tem associado a si requisitos funcionais. Por sua vez a componente de visualização apresenta requisitos relativos às funcionalidades que deverá apresentar.

Os requisitos não funcionais são as características de furor mais técnico sem as quais o sistema não funcionaria corretamente.

Requisitos da aplicação de calibração

Requisitos funcionais

- Registo do utilizador e respetiva autenticação;
- Apresentar a estrutura interior de cada piso através da planta do mesmo;
- Permitir que o utilizador faça a calibração de espaços dentro de um edifício com recurso à planta do piso;

- Recolher (através a interface Wi-Fi) e enviar *fingerprints* para o servidor aquando do processo de calibração;
- Fornecer informações relativas à construção do mapa de rádio (antiguidade das amostras e densidade das mesmas);
- Alterar a *password*;
- Recuperar a *password.*

Requisitos não funcionais

- Funcionamento em modo *offline* através do armazenamento de dados no dispositivo a fase de calibração poderá ser realizada sem ligação a uma rede.
- Uma vez que o utilizador introduz os seus dados pessoais através da aplicação, é fundamental que o sistema, por questões de segurança, integridade e privacidade tenha mecanismos que garantam estes requisitos aos dados que serão armazenados. O bom funcionamento do sistema também exige que haja disponibilidade dos dados;
- Impedir o acesso de utilizadores não autenticados;
- Garantir que o utilizador mantém a interface Wi-Fi do dispositivo ligada de forma a poder recolher as informações necessárias;
- Consumo reduzido de energia do dispositivo, principalmente quando a aplicação se encontra em *background*;
- Rapidez na obtenção de resposta aos pedidos efetuados ao servidor;
- Desempenho e fluidez nas interações com o utilizador;
- Conexão ao servidor via internet e comunicação entre ambos (dispositivo móvel e servidor).

Requisitos sistema central (server-side)

Requisitos funcionais

- Receber, processar e armazenar as informações dos utilizadores;
- Fornecer os dados das plantas dos edifícios à aplicação móvel;
- Receber, processar e armazenar as informações de posicionamento (*fingerprints*) associadas ao local onde foram recolhidas;
- Construção do mapa de rádio do edifício com recurso às *fingerprints;*
- Atender aos pedidos provenientes da aplicação móvel e responder aos mesmos.

Requisitos não funcionais

- Tal como na aplicação móvel, no servidor é necessário garantir a privacidade, integridade, autenticidade e confidencialidade dos dados armazenados;
- Garantir a disponibilidade dos serviços e capacidade para atender diversos pedidos de dispositivos diferentes;
- Garantir qualidade das comunicações entre os componentes do sistema.

5.3. Implementação do sistema de calibração

A presente secção é relativa ao processo de implementação do sistema de calibração. Nela são abordadas as tecnologias de cada componente e é apresentado o resultado final da implementação de cada componente do sistema de calibração.

5.3.1. Aplicação de calibração

A presente secção é relativa à aplicação de calibração e nela está justificada a escolha da plataforma de desenvolvimento e a descrição da implementação e das funcionalidades da aplicação.

5.3.1.1. Plataforma de desenvolvimento

A aplicação será desenvolvida para a plataforma Android com otimização para *tablets.* Dentro das opções do mercado de dispositivos móveis e tendo em consideração o volume de vendas os sistemas operativos, foi selecionado o sistema operativo Android porque apresenta maior flexibilidade do SO e superioridade na quota de mercado em *smartphones* [75] e *tablets* [76]. A linguagem de programação deste sistema é o Java e o ambiente de desenvolvimento mais adequado é o Android Studio [77] (havendo como alternativa o Eclipse [78]).

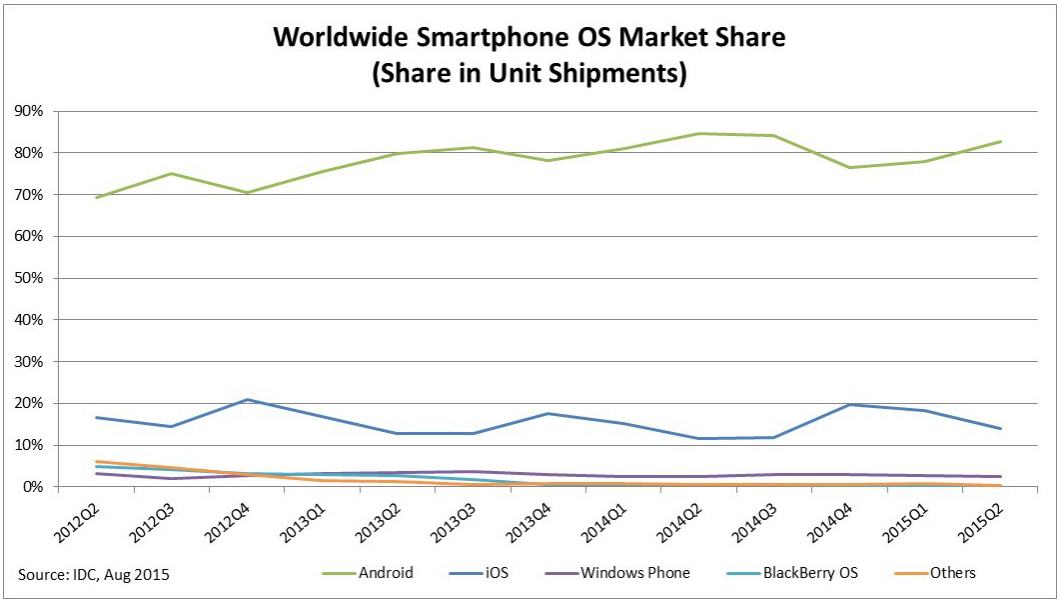


Figura 5.3: Quotas de mercado dos sistemas operativos de *Smartphones* no mundo [75]

O sistema iOS não fez parte das alternativas uma vez que as aplicações que recolhem dados dos pontos de acesso Wi-Fi foram banidas pela Apple [79][80]. Como o sistema que se pretende desenvolver depende diretamente da análise do ambiente Wi-Fi dos espaços, esta restrição por parte da Apple impede o desenvolvimento de soluções deste género para o sistema operativo iOS. Além disso o desenvolvimento para iOS exige a criação de uma conta de desenvolvedor e o pagamento de 99$ para que seja possível publicar aplicações na App Store [81]. Por sua vez, na plataforma Android, o custo para inscrição de um *developer* é de 25$.

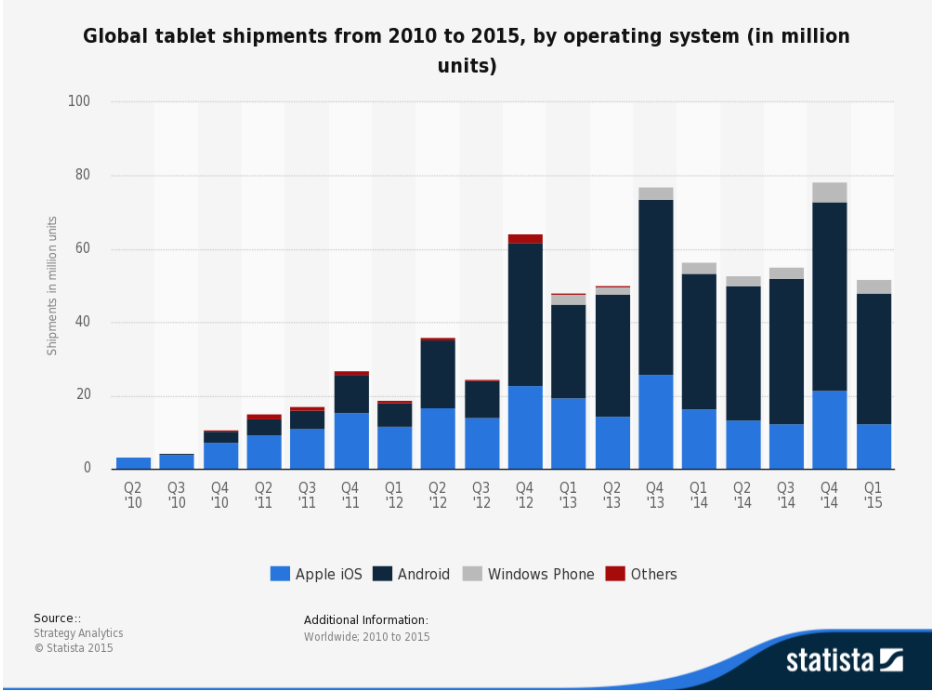


Figura 5.4: Vendas globais de *tablets* de 2010 até 2015, por sistema operativo [76]

Foram enumeradas várias razões pelas quais a plataforma de desenvolvimento Android é mais vantajosa, e por isso, será esta a plataforma de desenvolvimento da aplicação de calibração.

Foi desenvolvido um estudo prévio sobre o modo de funcionamento do sistema Android, esse trabalho pode ser consultado no Anexo B.

5.3.1.2. Componentes da Aplicação de Calibração

Depois de estar concluído o estudo inicial sobre a plataforma Android, foi possível criar um esquema com a arquitetura da aplicação, estando as componentes mais importantes deste sistema e a forma como estas se relacionam entre si. A Figura 5.5 mostra cada componente de aplicação de calibração.

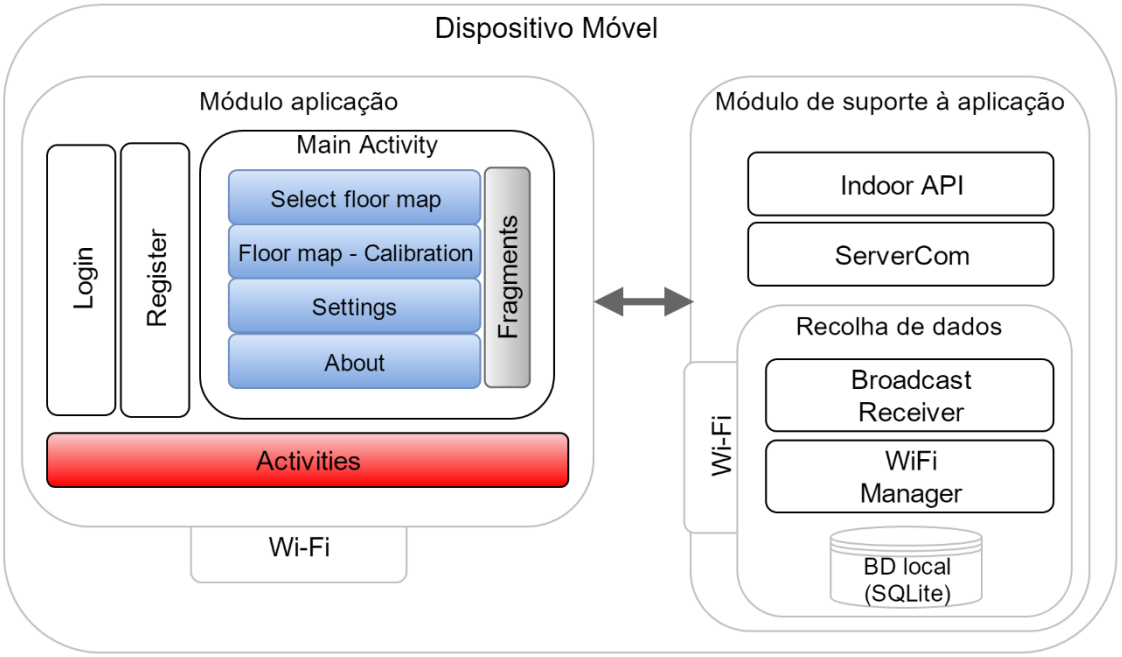


Figura 5.5: Componentes da aplicação de calibração

Distinguem-se dois módulos na aplicação móvel, uma vez que têm funções distintas:

- Módulo aplicação – apresenta as *activitys* onde são geridas todas as interações com o utilizador;
- Módulo de suporte à aplicação – fornece ao módulo da aplicação as principais funcionalidades da aplicação. Este módulo está subdividido da seguinte forma:
    - Indoor API – é a API que foi desenvolvida no âmbito da solução para as plantas dos edifícios. O seu objetivo é processar os dados de uma planta e apresenta-la no ecrã do dispositivo;
    - ServerCom – fornece a comunicação entre a aplicação e o servidor;
    - Recolha de dados – participa na recolha de dados da interface Wi-Fi do dispositivo e faz o seu armazenamento na base de dados local.

Versão Android

A aplicação tem suporte para dispositivos Android com a versão 4.4 (KitKat) ou superior tendo já suporte para a versão Android mais recente, Marshmallow 6.0. Isto abrange um total de 70,8% dos dispositivos de acordo com os dados recolhidos pela Google num período de 7 dias terminado no dia 1 de fevereiro de 2016 [82].

Permissões de utilização

Cada aplicação Android tem associado a si um conjunto de permissões que são apresentadas ao utilizador antes da instalação. As permissões de utilização permitem que a aplicação tenha várias funcionalidades, por exemplo, ter acesso à localização do dispositivo, ou à interface Wi-Fi. A lista de permissões, na Tabela 5.4, é declarada num ficheiro XML (*AndroidManifest.xml*), em que são apresentadas várias informações da aplicação essenciais para o sistema Android.

Tabela 5.4: Permissões da aplicação de calibração

Permissão	Descrição
INTERNET	Permite que a aplicação aceda à internet.
ACCESS_NETWORK_STATE	Permite que a aplicação aceda a informação sobre redes.
ACCESS_FINE_LOCATION	Permite que a aplicação aceda à localização precisa.
ACCESS_COARSE_LOCATION	Permite que a aplicação aceda à localização aproximada.
ACCESS_WIFI_STATE	Permite que a aplicação aceda a informação sobre redes Wi-Fi.
CHANGE_WIFI_STATE	Permite que a aplicação altere o estado da conetividade Wi-Fi.

As permissões de acesso à localização são necessárias para que seja possível fazer a recolha dos dados dos pontos de acesso através da classe WiFiManager 5.3.1.4.

5.3.1.3. Apresentação das plantas no ecrã

No âmbito da calibração é necessário obter os dados de calibração e a planta de modo a apresentar os dados relativos ao mapa de rádio. Desta forma, o fluxograma da Figura 5.6 ilustra todo o processo que resulta no desenho da planta no dispositivo móvel dedicado à aplicação de calibração.

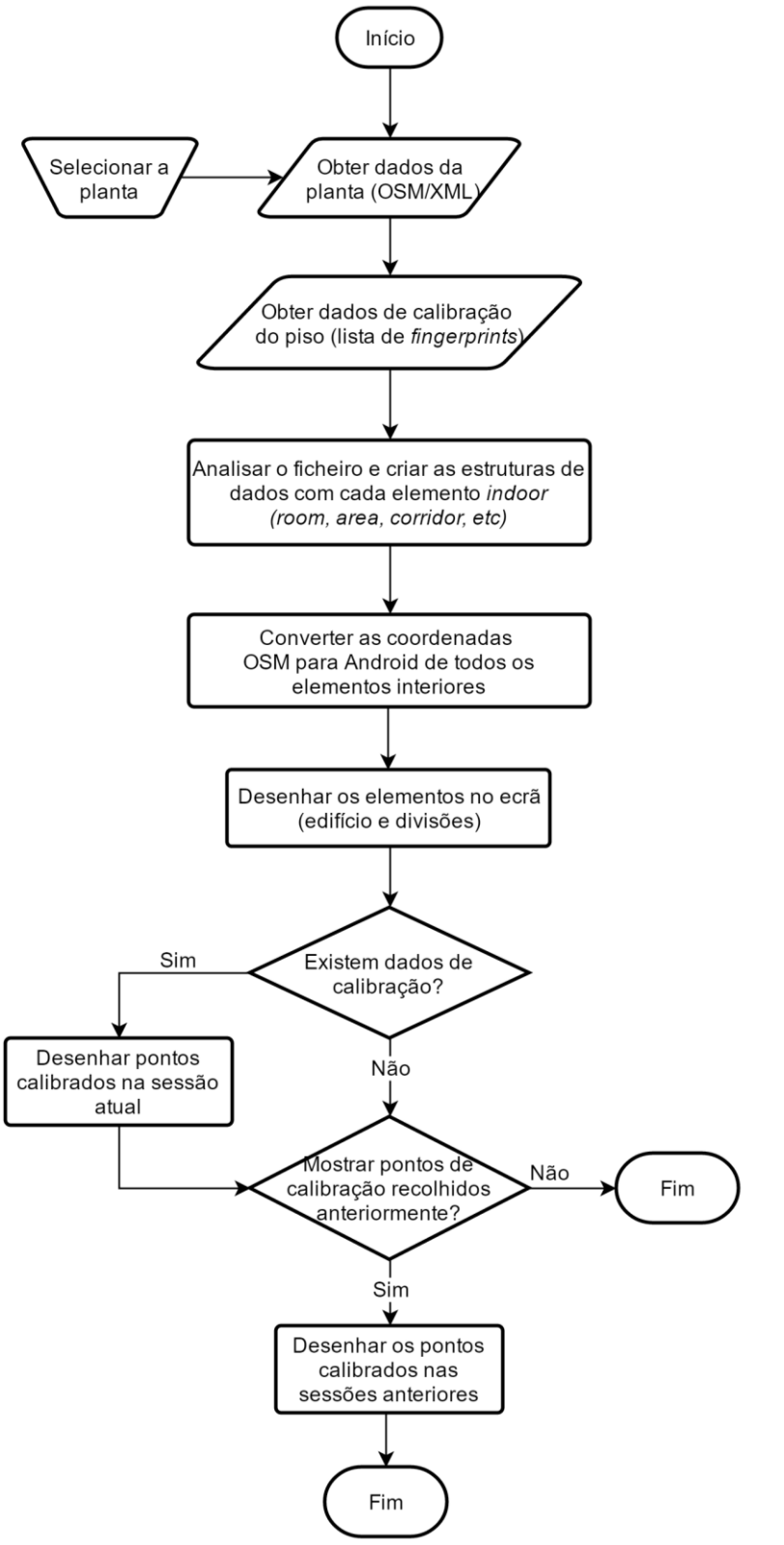


Figura 5.6: Apresentação da planta de um piso na aplicação de calibração

5.3.1.4. Recolha dos dados do ambiente rádio Wi-Fi

O principal intuito da aplicação móvel é fazer a recolha de *fingerprints* nos espaços interiores de um edifício permitindo assim a construção do mapa de rádio. A fase de calibração depende dos dados recolhidos através da interface Wi-Fi do dispositivo móvel, para isso existe o BroadcastReceiver que tem a função de receber os dados dos pontos de acesso Wi-Fi, recolhidos pela interface Wi-Fi na análise ao ambiente rádio, e disponibiliza os dados ao serviço que o registou.

A recolha de dados do ambiente rádio Wi-Fi não necessita de ligação à internet, e por isso os dados são sempre armazenados na base de dados do dispositivo. O utilizador quando pretender fazer o envio dos dados para o servidor, estes dados são enviados e apagados na base de dados local do dispositivo. A aplicação permite ao utilizador recolher as *fingerprints* quando quiser, mesmo estando *offline* uma vez que só é necessário ter a interface Wi-Fi ligada.

WifiManager

Através da classe WiFiManager [83] é possível gerir todos os aspetos da conectividade Wi-Fi. Esta classe permite obter várias informações:

- A lista de redes configuradas, caso existam;
- A rede Wi-Fi ativa no momento, caso exista. A conectividade pode ser estabelecida ou cancelada, e pode-se obter informação sobre a conexão atual;
- Resultados de *scans* aos pontos de acesso disponíveis na área de alcance da interface Wi-Fi do dispositivo.

BroadcastReceiver

As instâncias da classe BroadcastReceiver permitem receber dados provenientes de elementos do sistema. O BroadcastReceiver é utilizado com a função de receber e disponibilizar os dados dos pontos de acesso obtidos através da interface Wi-Fi do dispositivo com a instância da classe WiFiManager. Os dados obtidos são disponibilizados pelo BroadcastReceiver ao serviço que o registou.

A classe WiFiManager permite que o BroadcastReceiver faça *broadcast* quando os resultados dos *scans* estão disponíveis através das *tags* SCAN_RESULTS_AVAILABLE_ACTION ou RSSI_CHANGED_ACTION. Essencialmente os resultados do *scan* aos pontos de acesso só são disponibilizados quando o *scan* estiver terminado.

Base de dados SQLite

A base de dados local (SQLite) é utilizada para armazenar os dados de calibração dos espaços. Como a aplicação permite que os dados sejam recolhidos mesmo estando *offline,* é necessário armazená-los de forma estruturada. A base de dados SQLite tem a vantagem de manter os dados sempre disponíveis para a aplicação, e por isso, o utilizador não precisa de fazer o *upload* dos dados no fim de cada sessão de calibração, porque sempre que iniciar sessão os dados estão armazenados na base de dados do dispositivo e podem ser enviados para o servidor a qualquer momento.

Na aplicação existe uma classe (DbHelper) onde é definida a base de dados e onde estão descritas todas as operações sobre a base de dados: inserção de elementos, deleção de elementos e a seleção de elementos. Esta classe estende a classe nativa do Android SQLiteOpenHelper [84] que tem o intuito de gerir a criação de bases de dados e gerir as versões da base de dados.

A Figura 5.7 apresenta o diagrama de entidades e relacionamentos da base de dados, representado por duas entidades:

- Fingerprint – apresenta as informações sobre cada amostra recolhida, nomeadamente o instante da recolha (*timestamp*) o identificador do espaço interior onde foi feita a recolha (id_room, que é o identificador do elemento na planta do piso) e o par de coordenadas latitude, longitude que descrevem o local exato onde a *fingerprint* foi recolhida;
- Ap – associado a cada *fingerprint* está uma lista de pontos de acesso detetados pelo dispositivo. A tabela "Ap" descreve cada um dos pontos de acesso associados a uma *fingerprint*, que são caracterizados pelo seu endereço MAC, SSID (nome da sua rede), RSSI (nível de sinal) e o canal de operação do ponto de acesso (descrito por um valor inteiro).

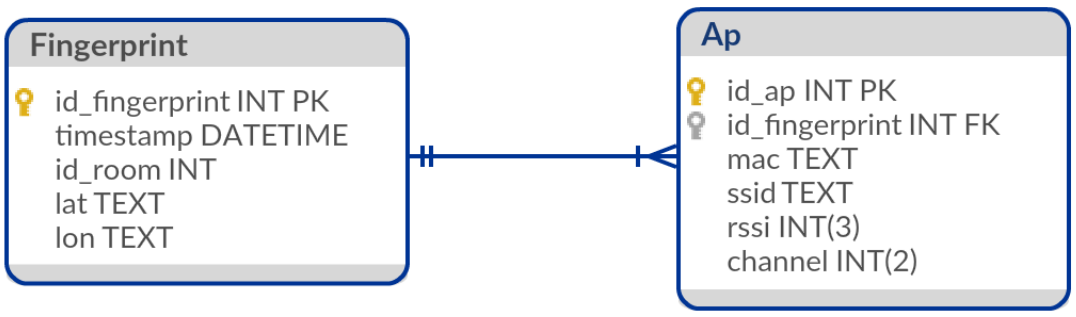


Figura 5.7: Diagrama de entidades e relacionamentos da base de dados SQLite

Algoritmo de recolha de dados

O módulo da recolha de dados utiliza o algoritmo da Figura 5.8 para a recolha de dados. Trata-se de um processo simples que inicialmente verifica que a interface Wi-Fi do dispositivo está ligada, no caso afirmativo procede à recolha dos dados e ao respetivo armazenamento dos dados na base de dados local. Caso o Wi-Fi do dispositivo esteja desligado, é pedido ao utilizador para alterar o estado da interface Wi-Fi.

Com recurso à classe WiFiManager é possível ligar e desligar o Wi-Fi sem o conhecimento do utilizador, mas esta abordagem não é honesta e vai contra as regras de conduta de programação. Assim os dados só são recolhidos quando o utilizador pretender, e o estado da interface Wi-Fi só é alterado quando o utilizador quiser.

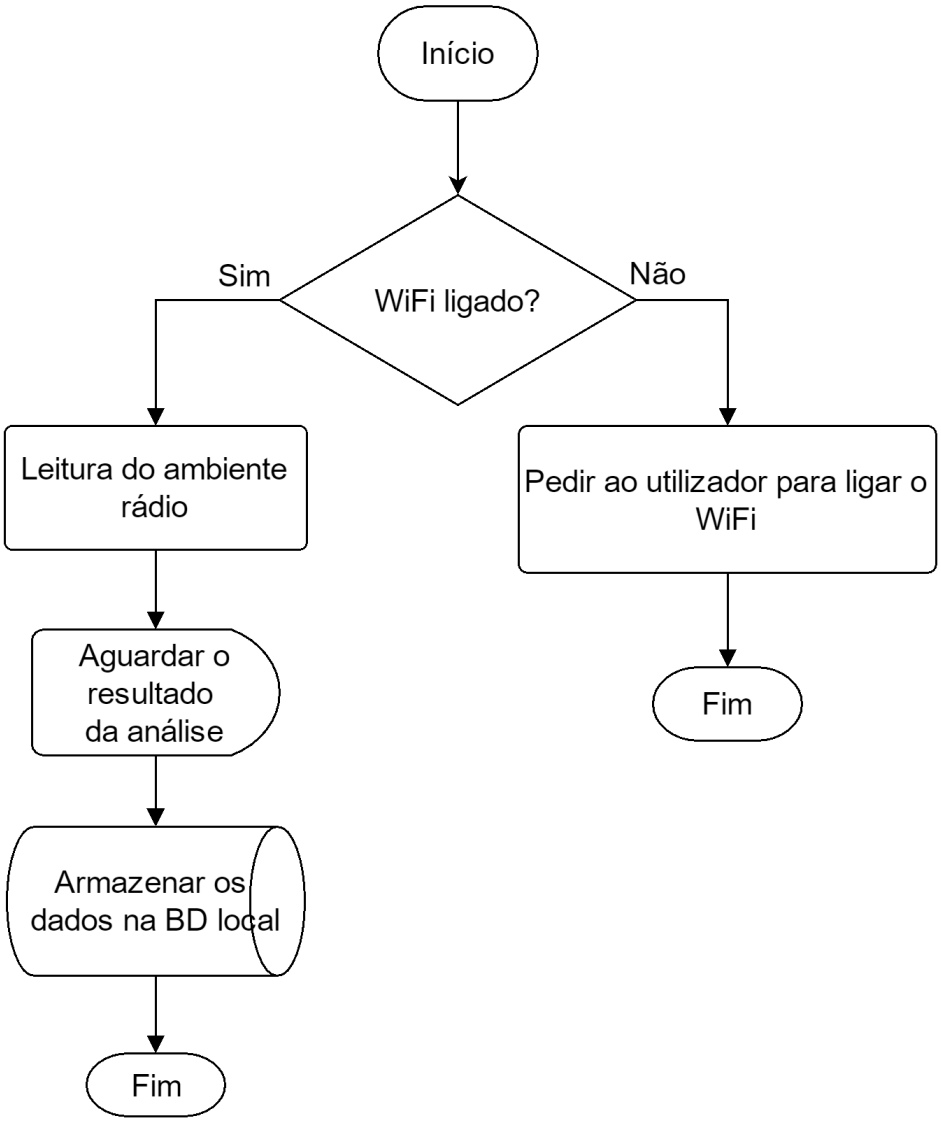


Figura 5.8: Processo realizado quando se recolhem amostras Wi-Fi

Algoritmo de *upload* dos dados para o servidor

Como a aplicação permite fazer a calibração sem ligação à internet, é o utilizador determina quando é adequado fazer o envio dos dados para o servidor uma vez que depende da ligação à internet.

Quando o utilizador seleciona a opção para fazer o *upload* dos dados é executado um algoritmo, representado na Figura 5.9, onde inicialmente é feita uma verificação do estado da ligação do dispositivo à internet. Se o dispositivo não tiver ligação à internet, é pedido ao utilizador que se ligue a uma rede com acesso à internet. Caso o dispositivo esteja ligado a uma rede com acesso à internet, os dados da *fingerprint* são preparados e enviados no formato de dados JSON. De seguida é obtida uma resposta do servidor informando se os dados foram recebidos com sucesso. Quando os dados de uma *fingerprint* são recebidos com sucesso no servidor, estes são removidos da base de dados local, mantendo assim a base de dados vazia depois de serem enviadas todas as amostras. As *fingerprints* mais recentes são enviadas primeiro, seguindo a filosofia LIFO (*Last In, First Out*).

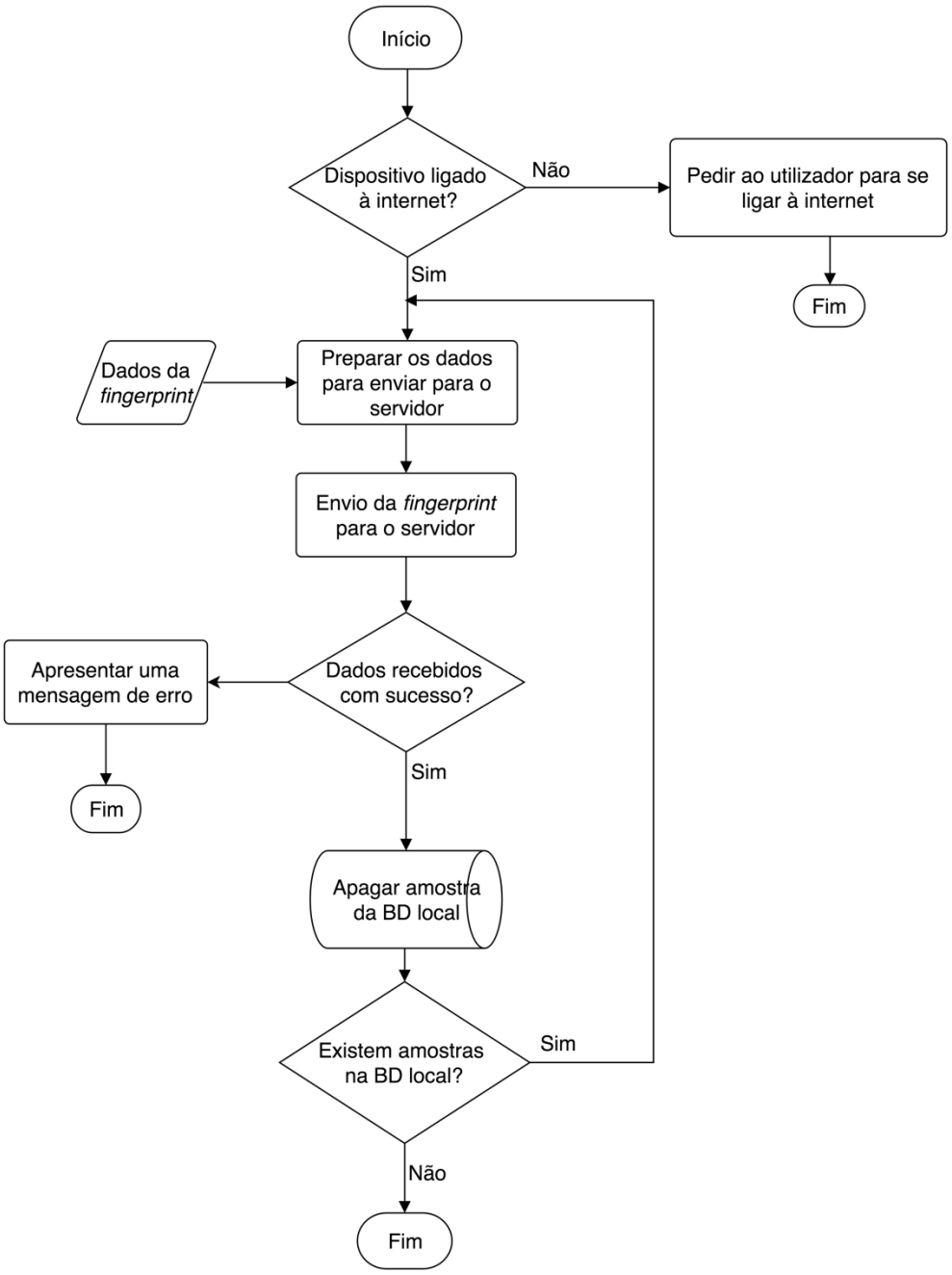


Figura 5.9: Algoritmo realizado quando é feito o *upload* de dados

Os dados de cada *fingerprint* são preparados para serem enviados. A Figura 5.10 apresenta o formato de uma *fingerprint* recolhida e enviada para o servidor.

```
{
   "email":"A65239@alunos.uminho.pt",
   "id_device":"00:04:4b:2c:be:3a",
   "id_room":-231,
   "id_floor":1,
   "lat":"41.452876200000",
   "lon":"-8.286353000000",
   "timestamp":"2015-10-17 15:51:00",
   "fingerprint":[
      {
         "mac":"f0:25:72:cb:b4:2f",
         "rssi":"-52",
         "ssid":"eduroam",
         "channel":44
      },
      {
         "mac":"f0:25:72:cb:b4:2b",
         "rssi":"-53",
         "ssid":"",
         "channel":44
      },
      {
         "mac":"f0:25:72:cb:b4:20",
         "rssi":"-59",
         "ssid":"eduroam",
         "channel":1
      }
   ]
}
```


Figura 5.10: Dados de uma fingerprint no formato JSON

Quando surgem falhas no envio de *fingerprints* para o servidor, é apresentada uma mensagem de erro, e não se apagam da BD local quaisquer *fingerprints* que não tenham sido recebidas com sucesso por parte do servidor. O algoritmo termina quando surge um erro ou quando todas as amostras no dispositivo são enviadas com sucesso.

5.3.1.5. Funcionalidades e ecrãs

De seguida estão listadas as funcionalidades e ecrãs da aplicação de calibração em conjunto com uma explicação de cada funcionalidade. Em alguns casos a informação é complementada com diagramas de sequência que ilustram da melhor forma a interação entre o utilizador e a aplicação, assim como a aplicação com o servidor.

Registo

O processo de registo, apresentado na Figura 5.11, é bastante simples para o utilizador, começando pela escolha do operador ao qual pertence, de seguida preenche o seu nome, o e-mail e por fim é introduzida a palavra passe que deverá ser igual nas duas entradas do formulário. Antes de ser feito qualquer pedido ao servidor, a aplicação verifica sempre que todos os campos estão preenchidos e que as *passwords* coincidem.

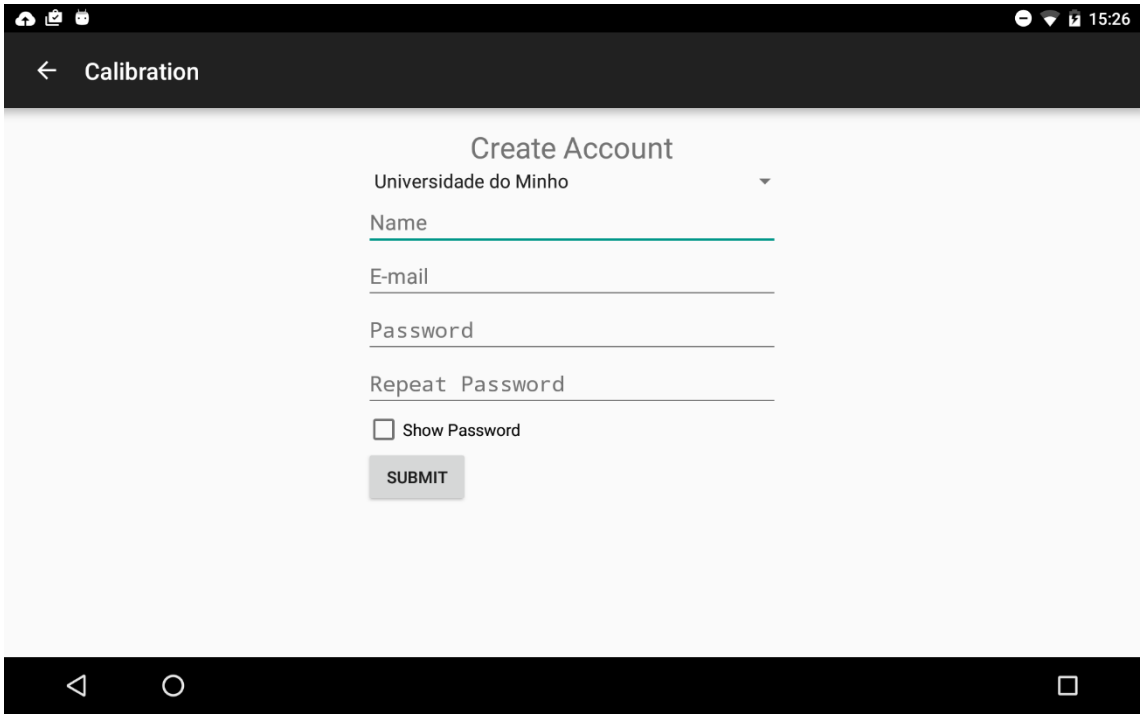


Figura 5.11: Ecrã de registo

Estando os dados do formulário verificados, é criado um objeto JSON com os dados de registo assim como o endereço MAC do dispositivo. Os dados quando são recebidos no servidor é verificada a existência do utilizador e procede-se à sua inserção caso seja um novo utilizador. A *password* antes de ser armazenada sofre um processo que permite o armazenamento da mesma na base de dados numa forma ilegível, uma vez que é armazenado o *hash* da *password*. Através da função *crypt* do PHP, é devolvido o *hash* da *password*. Esta função utiliza um algoritmo baseado no método criptográfico DES e tem como parâmetro um *salt* que se trata de uma *string* aleatória na qual o processo de *hash* se baseia. Este processo protege contra vários ataques: *rainbow tables, lookup*, *dictionary attacks, brute-force attacks* (torna-os significativamente mais lentos, nos casos

dos ataques de *brute-force* e *dictionary*). Além disso, este método permite que não existam *passwords* duplicadas na base de dados.

Por fim, o dispositivo é registado e associado ao utilizador e o servidor devolve o resultado da operação passando para a página principal onde o utilizador pode fazer *login*.

O processo de registo está ilustrado no diagrama de sequência da Figura 5.12.

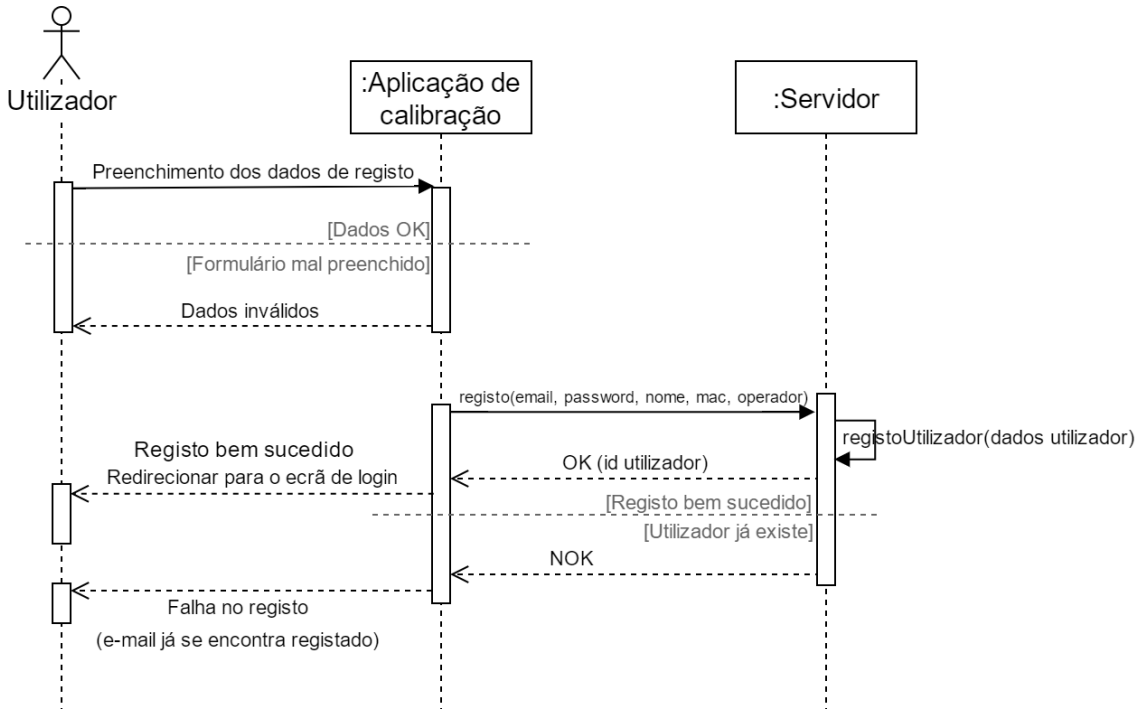


Figura 5.12: Diagrama de sequência - registo do utilizador

Login

O ecrã inicial da aplicação, na Figura 5.13, permite ao utilizador fazer o *login* na plataforma, para isso apenas necessita de introduzir as suas credenciais de acesso (e-mail e *password* introduzidos no processo de registo), e depois o sistema reencaminha o utilizador para o ecrã de seleção da planta do piso.

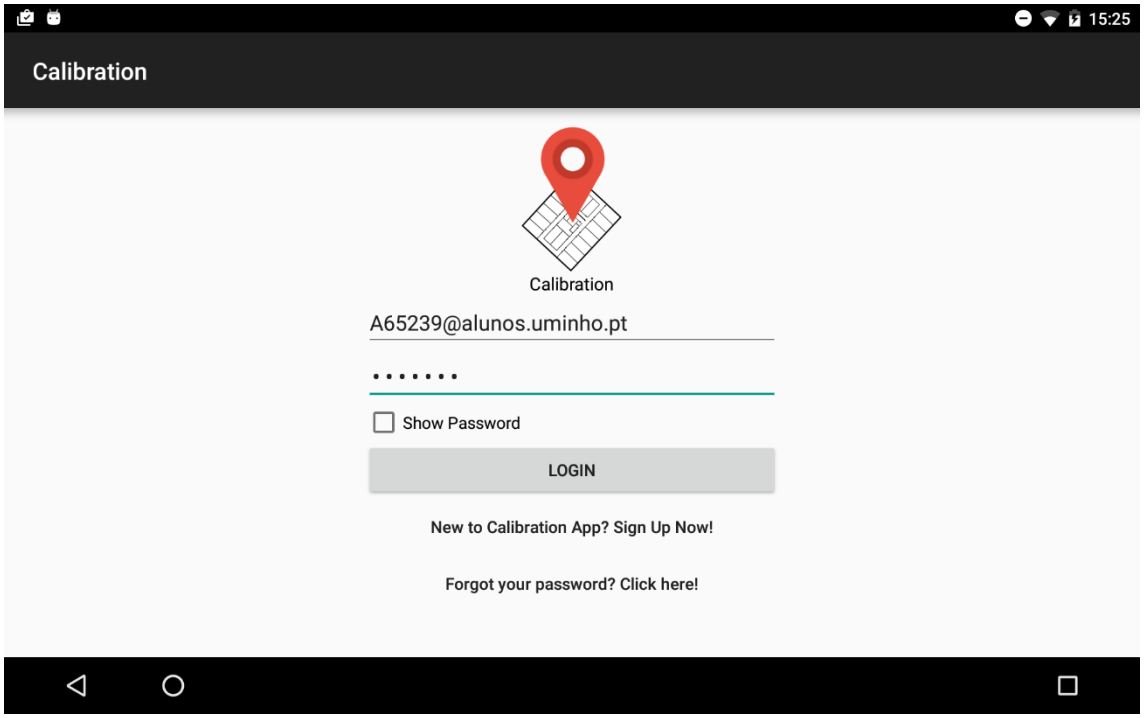


Figura 5.13: Ecrã de login na aplicação

O diagrama de sequência da Figura 5.14 representa as operações entre o utilizador, aplicação móvel e servidor para o processo do *login*.

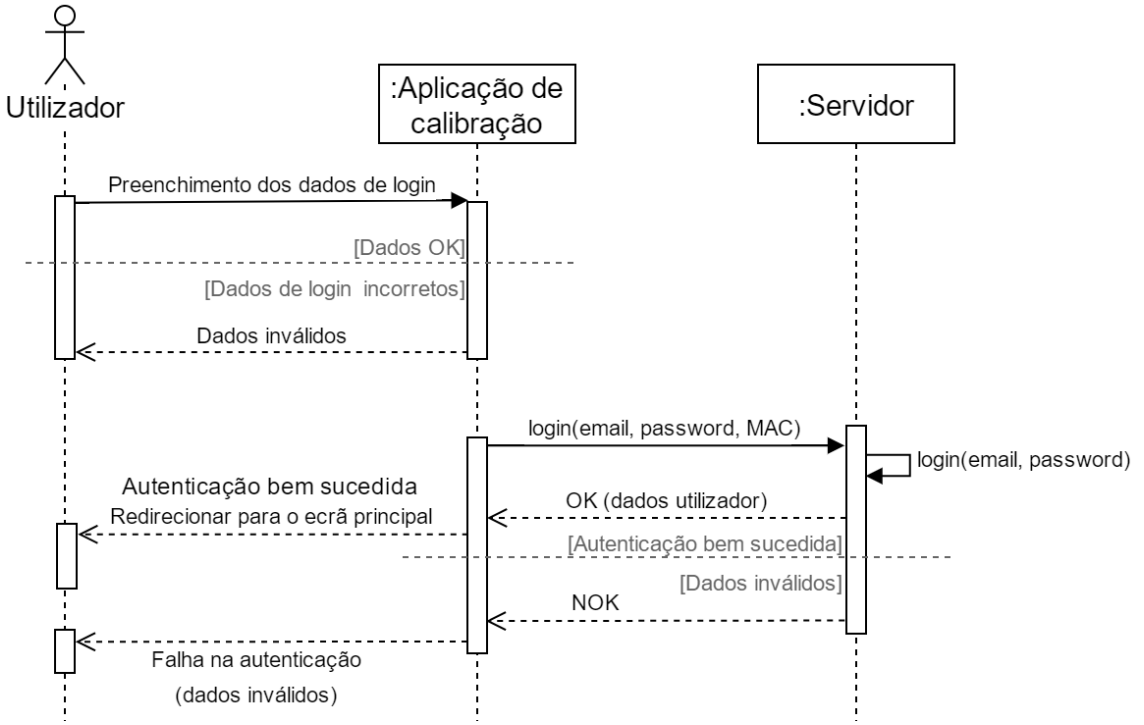


Figura 5.14: Diagrama de sequência - autenticação do utilizador

Seleção de uma planta

No ecrã de seleção da planta (Figura 5.15) o utilizador deverá selecionar a área, edifício e respetivo piso. Adicionalmente, o utilizador pode selecionar uma das seguintes opções para apresentar os dados de calibração:

- "Yes (this user)" – apresenta os dados de calibração recolhidos pelo utilizador no ecrã de calibração;
- "Yes (all users)" – apresenta os dados de calibração recolhidos por todos os utilizadores no ecrã de calibração;
- "No" – não são apresentados os dados de calibração no ecrã de calibração.

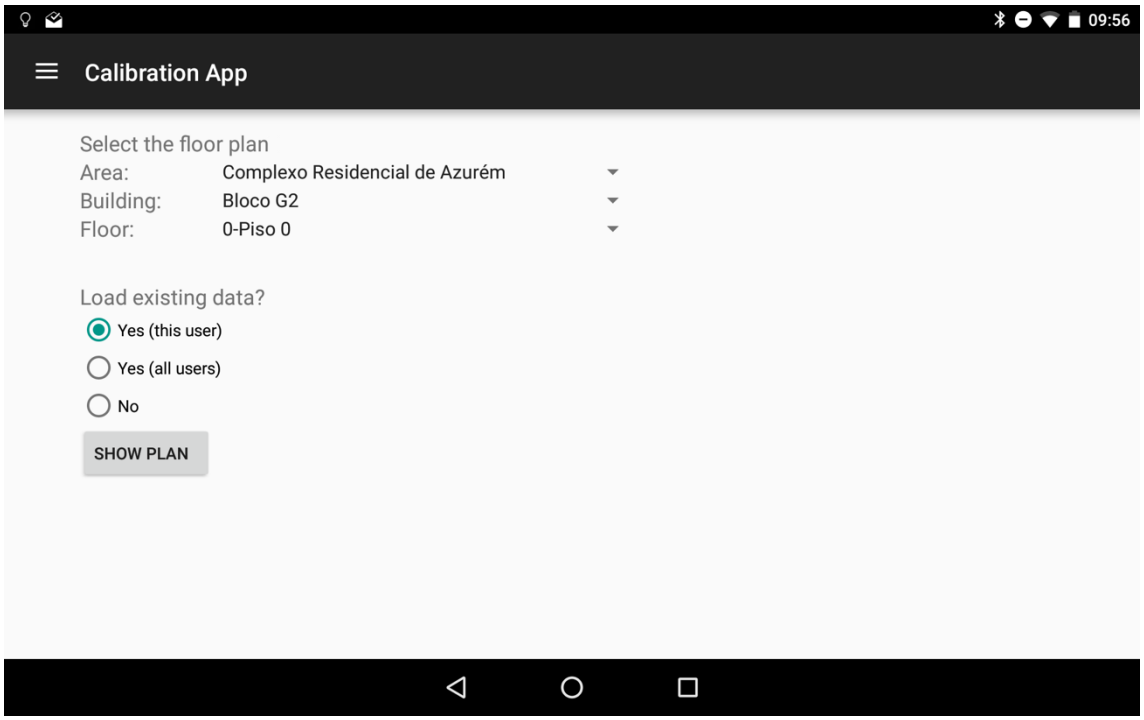


Figura 5.15: Ecrã de seleção da planta

Estando todos os campos devidamente preenchidos, o botão "Show Plan" leva o utilizador para o ecrã de calibração onde é apresentada a planta do piso juntamente com os dados de calibração de acordo com a opção selecionada pelo utilizador.

O botão "Show Plan" inicia um processo de obtenção da planta do piso através de um pedido ao servidor. De seguida é feito outro pedido ao servidor onde são obtidos e os dados de calibração e armazenados temporariamente no dispositivo, para que estejam disponíveis sempre que o utilizador desejar consultar o estado da calibração no piso em questão.

O diagrama de sequência ilustrado na Figura 5.16 descreve o processo de seleção de uma planta que resulta na apresentação do ecrã de calibração com a planta do piso selecionado.

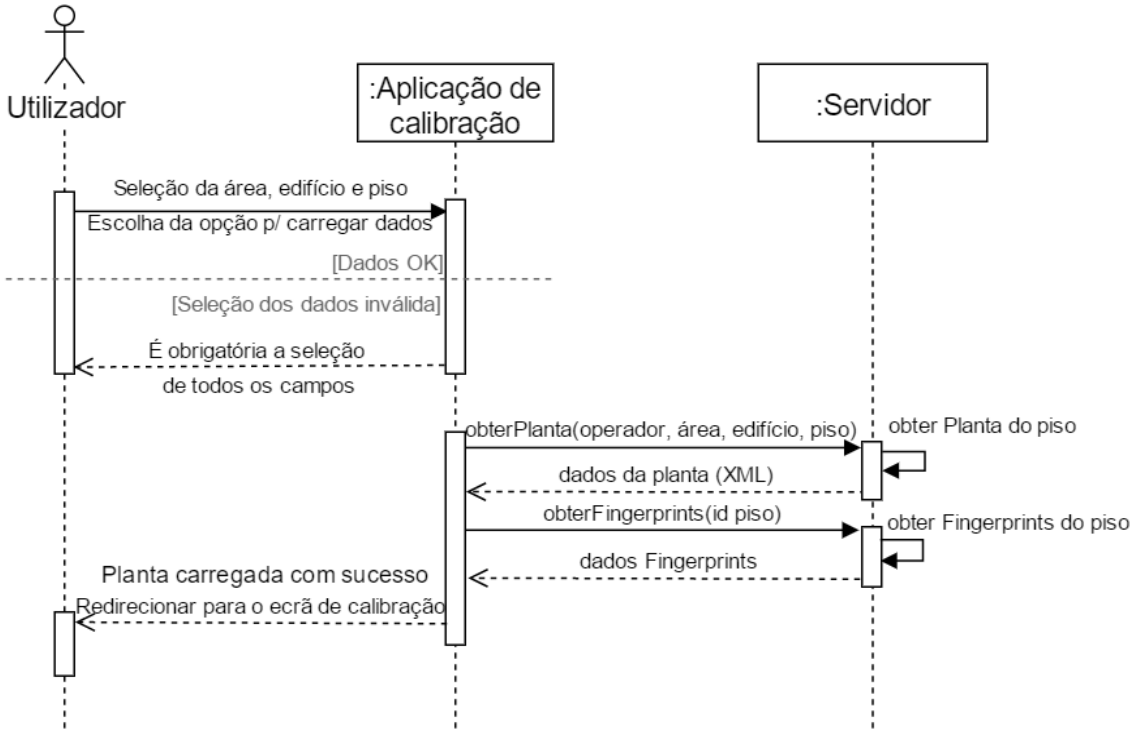


Figura 5.16: Diagrama de sequência - seleção da planta de um piso

Processo de calibração

Após a seleção da planta e da obtenção dos dados necessários, a aplicação mostra o ecrã de calibração (Figura 5.17) com a planta do piso tendo em conta a opção selecionada pelo utilizador para os dados recolhidos anteriormente. No canto superior direito do ecrã está o nome do edifício e o piso. Na barra superior da aplicação existe um menu de opções que oferece algumas funcionalidades que serão abordadas posteriormente.

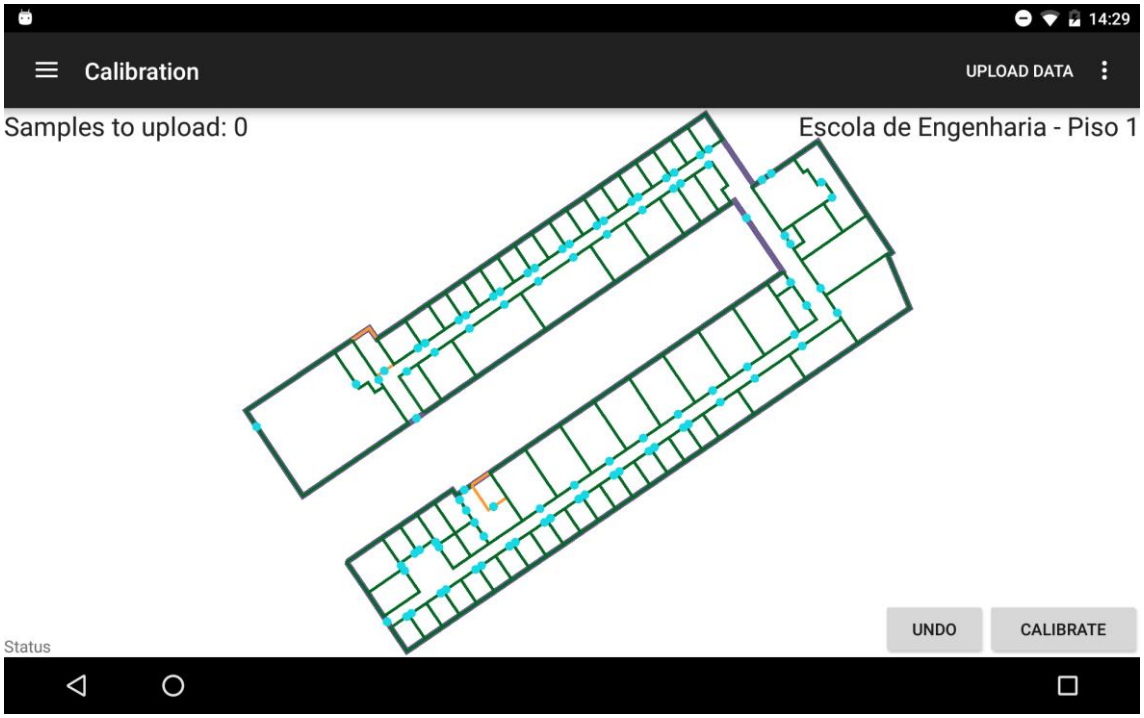


Figura 5.17: Ecrã de calibração

No ecrã de calibração no canto superior esquerdo é apresentado o número de amostras que estão armazenadas atualmente no dispositivo, que podem ser enviadas para o servidor em qualquer momento desde que o dispositivo tenha acesso à internet. No canto inferior esquerdo aparecem as mensagens de estado que informam o utilizador das operações efetuadas e quando são realizadas.

O processo de calibração é uma tarefa bastante simples para o utilizador: basta colocar o ponto vermelho no local em que o utilizador se encontra e carregar no botão "*Calibrate*", sendo aconselhado que o utilizador recolha várias amostras em cada local, basta apenas utilizar o botão "*Calibrate*" sem alterar a posição do ponto vermelho. Caso o utilizador se engane a calibrar, tem a possibilidade de anular amostras utilizando o botão "*Undo*". O botão "*Undo*" remove sempre a *fingerprint* mais recente da base de dados local, tendo em conta as *fingerprints* que foram recolhidas na sessão atual. Ou seja, este botão permite remover todas as amostras recolhidas numa sessão, uma a uma. A Figura 5.18 ilustra o processo de calibração de vários pontos.

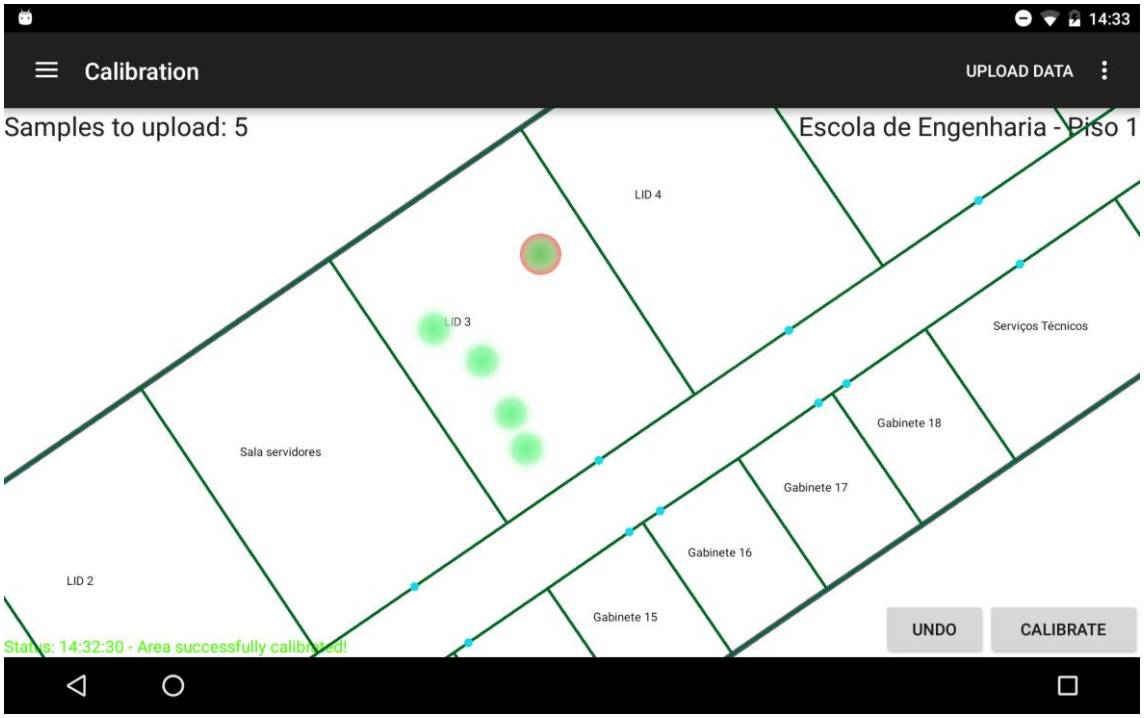


Figura 5.18: Ecrã de calibração com zoom - calibração de vários pontos

Consulta da última amostra recolhida

No momento da calibração, quando o utilizador faz a recolha de uma amostra do ambiente rádio do local, é possível que este consulte os dados da última amostra recolhida a partir do menu de opções.

Basta selecionar a opção "*Show latest sample data*" do menu de opções (Figura 5.19), e é apresentada a informação (SSID, RSSI, canal, endereço MAC) da *fingerprint* numa caixa de diálogo (Figura 5.20).

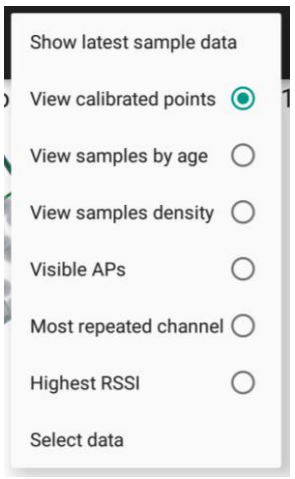
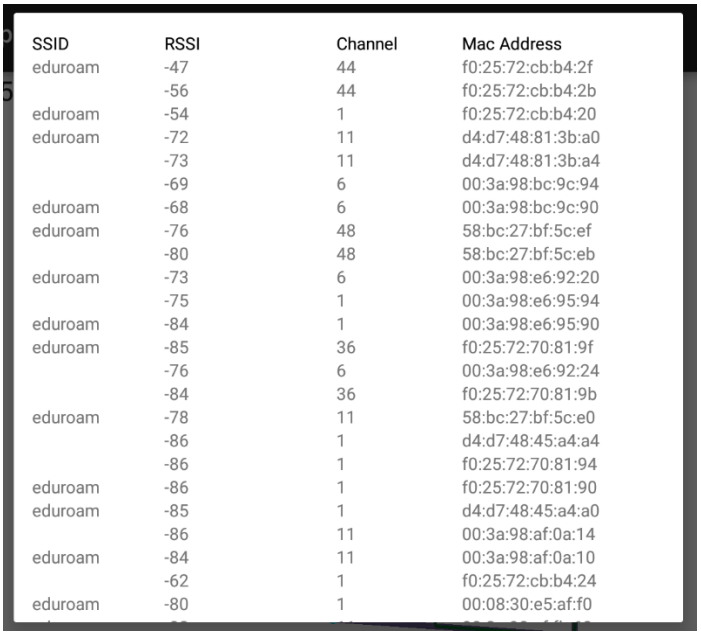


Figura 5.19: Menu de opções do ecrã de calibração



SSID	RSSI	Channel	Mac Address
eduroam	-47	44	f0:25:72:cb:b4:2f
	-56	44	f0:25:72:cb:b4:2b
eduroam	-54	1	f0:25:72:cb:b4:20
eduroam	-72	11	d4:d7:48:81:3b:a0
	-73	11	d4:d7:48:81:3b:a4
	-69	6	00:3a:98:bc:9c:94
eduroam	-68	6	00:3a:98:bc:9c:90
eduroam	-76	48	58:bc:27:bf:5c:ef
	-80	48	58:bc:27:bf:5c:eb
eduroam	-73	6	00:3a:98:e6:92:20
	-75	1	00:3a:98:e6:95:94
eduroam	-84	1	00:3a:98:e6:95:90
eduroam	-85	36	f0:25:72:70:81:9f
	-76	6	00:3a:98:e6:92:24
	-84	36	f0:25:72:70:81:9b
eduroam	-78	11	58:bc:27:bf:5c:e0
	-86	1	d4:d7:48:45:a4:a4
	-86	1	f0:25:72:70:81:94
eduroam	-86	1	f0:25:72:70:81:90
eduroam	-85	1	d4:d7:48:45:a4:a0
	-86	11	00:3a:98:af:0a:14
eduroam	-84	11	00:3a:98:af:0a:10
	-62	1	f0:25:72:cb:b4:24
eduroam	-80	1	00:08:30:e5:af:f0

Figura 5.20: Dados da última *fingerprint*

Opções de visualização

A aplicação apresenta vários modos de visualização dos dados das amostras recolhidas anteriormente. No menu de opções da aplicação estão seis modos de visualização distintos:

- Visualização normal – apresenta todos os pontos onde foram recolhidas *fingerprints;*
- Antiguidade das *fingerprints* – apresenta sob a forma de uma escala de cores a antiguidade das *fingerprints* recolhidas em cada local, o que permite determinar se já existem locais onde o mapa de rádio está desatualizado;
- Densidade das *fingerprints* – permite observar quantas amostras foram recolhidas em cada ponto através de uma escala de cores, ajudando na compreensão do estado do mapa de rádio e da qualidade do mesmo, partindo do principio que o maior número de amostras recolhidas em cada local permite a construção de um mapa de rádio com mais qualidade;
- Número de APs detetados – esta funcionalidade tem como objetivo apresentar os locais onde existe mais sobrecarga de APs, deste modo será possível determinar quais são os locais onde existe uma sobrecarga de APs;
- Canal mais repetido – com o objetivo de identificar os pontos onde existe várias sobreposições de canais, este modo de visualização ajuda a detetar os locais onde existe uma sobreposição excessiva de canais o que prejudica a infraestrutura Wi-Fi do edifício;

- Nível de sinal (RSSI) mais elevado – este modo de visualização permite descobrir os locais do edifício onde o sinal Wi-Fi é mais fraco e deste modo o reforço da infraestrutura Wi-Fi permite oferecer melhor cobertura e níveis de sinal mais adequados para a utilização das redes Wi-Fi.

Visualização normal – na Figura 5.21 aparecem todos os locais onde foram recolhidas amostras marcados com um ponto cinzento.

Figura 5.21: Modo de visualização normal no ecrã de calibração

Antiguidade das *fingerprints* – a Figura 5.22 mostra todas as *fingerprints* com uma escala de cores, ao contrário da visualização normal, em que aparecem as amostras a cinza, neste modo de visualização, as *fingerprints* são apresentadas com uma escala de cores tendo em conta a sua idade:

- Verde – *novas* – as amostras têm no máximo 3 meses de idade;
- Amarelo – *recentes* – as amostras têm entre 3 e 6 meses de idade;
- Laranja – *antigas* – as amostras têm entre 6 e 9 meses de idade;
- Vermelho – *muito antigas* – as amostras têm mais do que 9 meses de idade.

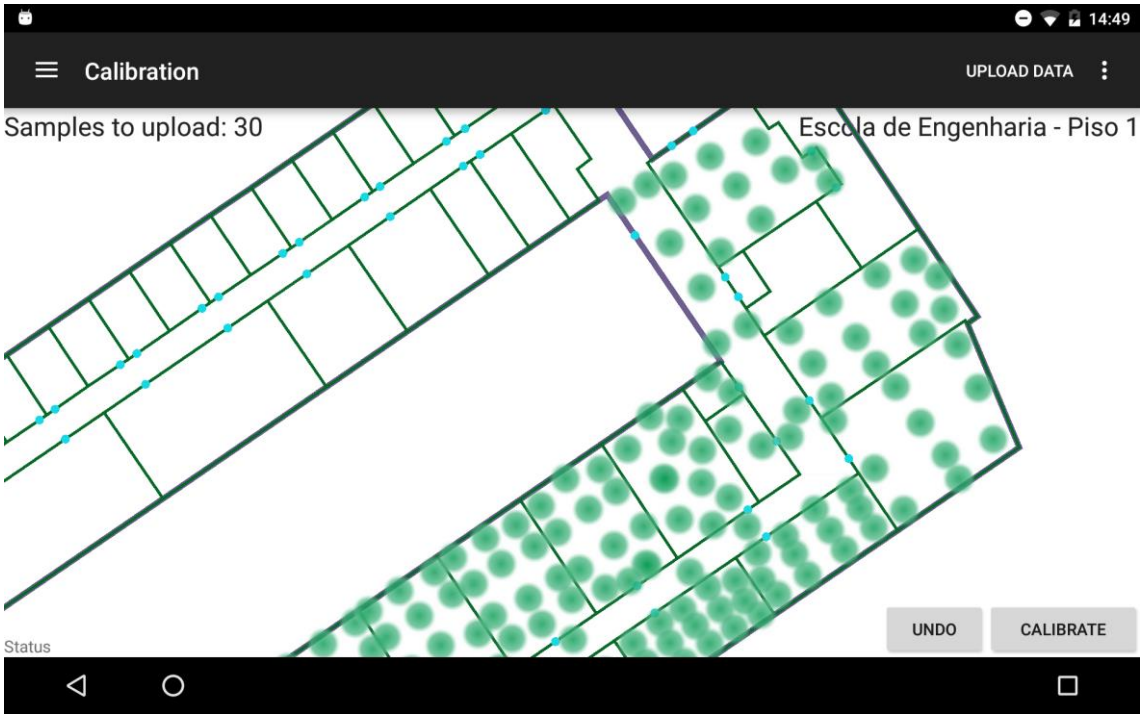


Figura 5.22: Modo de visualização tendo em conta a idade das *fingerprints* com escala de cores

Densidade das *fingerprints* – neste modo de visualização as *fingerprints* são representadas numa escala de cores de acordo com o número de amostras recolhido em cada ponto do piso (Figura 5.23). Tendo em conta o significado geral das cores, o verde considera-se como a cor com mais densidade, e por isso com mais número de amostras no mesmo ponto, por sua vez o vermelho representa os pontos com menos densidade:

- Verde – foram recolhidas 4 ou mais amostras no mesmo local;
- Amarelo – foram recolhidas 3 amostras no mesmo local;
- Laranja – foram recolhidas 2 amostras no mesmo local;
- Vermelho – foi recolhida 1 amostra no local.

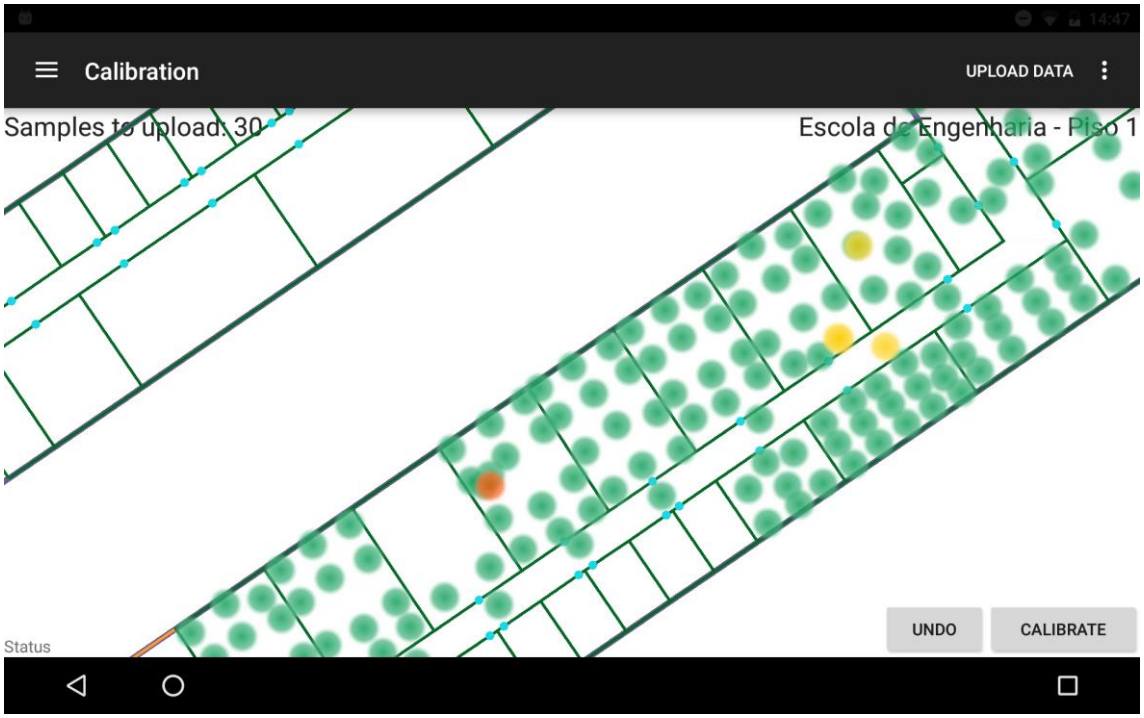


Figura 5.23: Modo de visualização tendo em conta a densidade das *fingerprints* com escala de cores

Número de APs detetados – apresenta numa escala de cores o número de APs detetados em cada ponto. Quanto mais escura for a cor, mais APs foram detetados nesse local. A Figura 5.24 mostra um exemplo do número de APs detetados e quando é feito zoom é possível observar o número exato de APs detetados em cada ponto (Figura 5.25). Quando existem várias amostras obtidas exatamente no mesmo ponto, o valor apresentado é a média de APs detetados nas amostras recolhidas nesse ponto. A escala de cores tem 9 categorias apresentadas na Tabela 5.5.

Tabela 5.5: Escala de cores utilizada para mostrar o nº de APs visíveis

Cor	Nº de APs detetados
	Até 5 APs
	Entre 6 e 10 APs
	Entre 11 e 15 APs
	Entre 16 e 20 APs
	Entre 21 e 25 APs
	Entre 26 e 30 APs
	Entre 31 e 35 APs
	Entre 36 e 40 APs
	Mais do que 40 APs

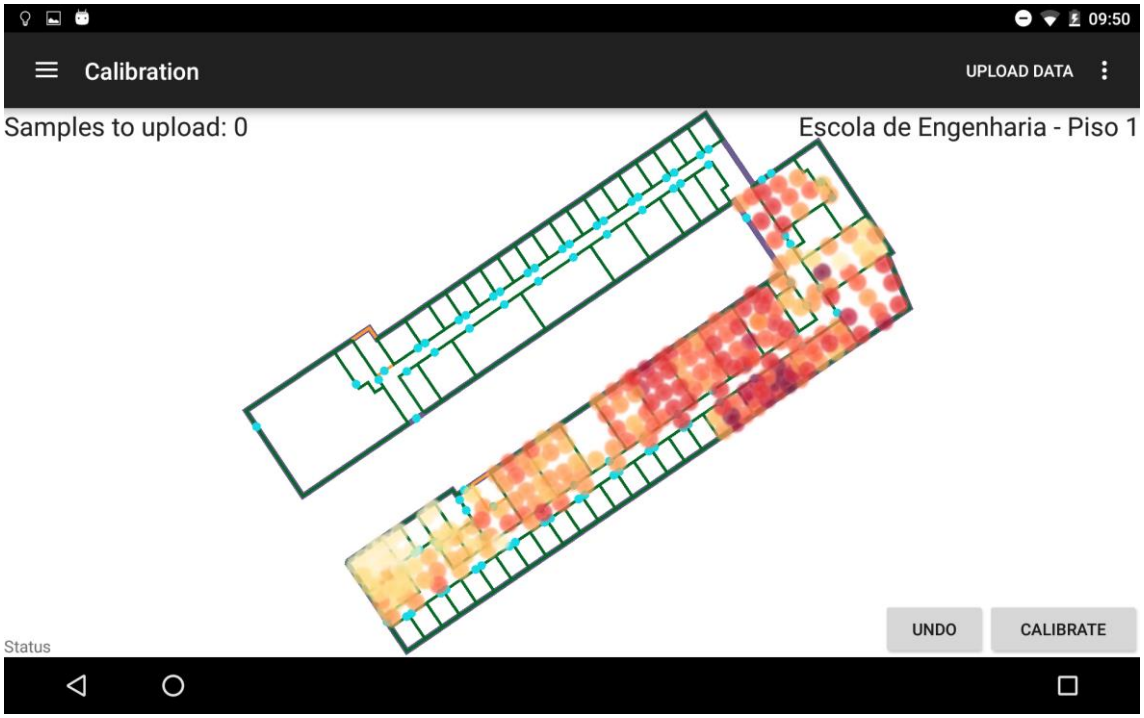


Figura 5.24: Modo de visualização tendo em conta o nº de APs visíveis com escala de cores

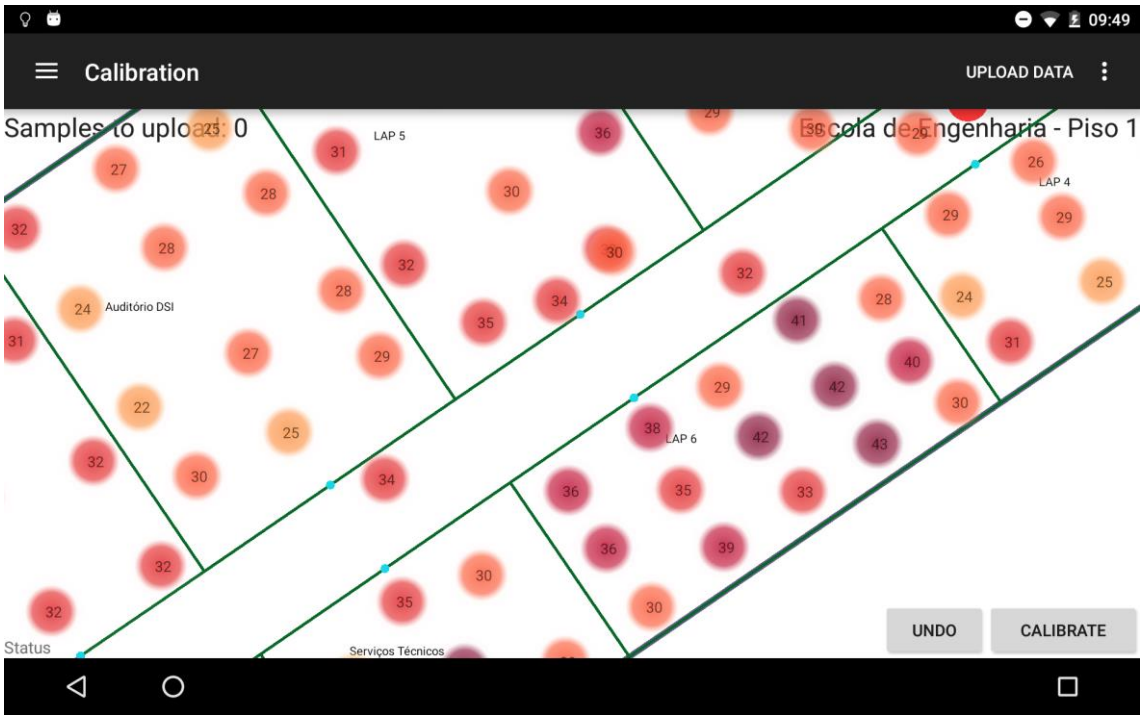


Figura 5.25: Modo de visualização tendo em conta o nº de APs visíveis com valores exatos

Canal mais repetido – apresenta numa escala de cores a quantidade de vezes que um canal se repete em cada *fingerprint* recolhida. Quanto mais escura for a cor, maior é o número de vezes que esse canal é detetado no mesmo local. Quando é feito zoom é possível observar o canal que

aparece repetido mais vezes em cada amostra. As Figura 5.26 ilustra os canais que mais se repetem nas amostras recolhidas e a Figura 5.27 mostra especificamente o canal mais repetido em cada ponto. Nas situações em que existem várias amostras obtidas exatamente no mesmo ponto, é selecionado o canal com mais repetições e a cor apresentada está associada à média de vezes que o canal foi detetado em todas as *fingerprints* recolhidas. A escala de cores tem 9 categorias apresentadas na Tabela 5.6.

Tabela 5.6: Escala de cores utilizada para mostrar os canais mais repetidos

Cor	Nº de repetições
	Igual a 1 repetição
	Entre 1 e 3 repetições
	Entre 4 e 6 repetições
	Entre 7 e 9 repetições
	Entre 10 e 12 repetições
	Entre 13 e 15 repetições
	Entre 16 e 18 repetições
	Entre 19 e 21 repetições
	Mais do que 21 repetições

Figura 5.26: Modo de visualização de acordo com o canal mais repetido em cada local

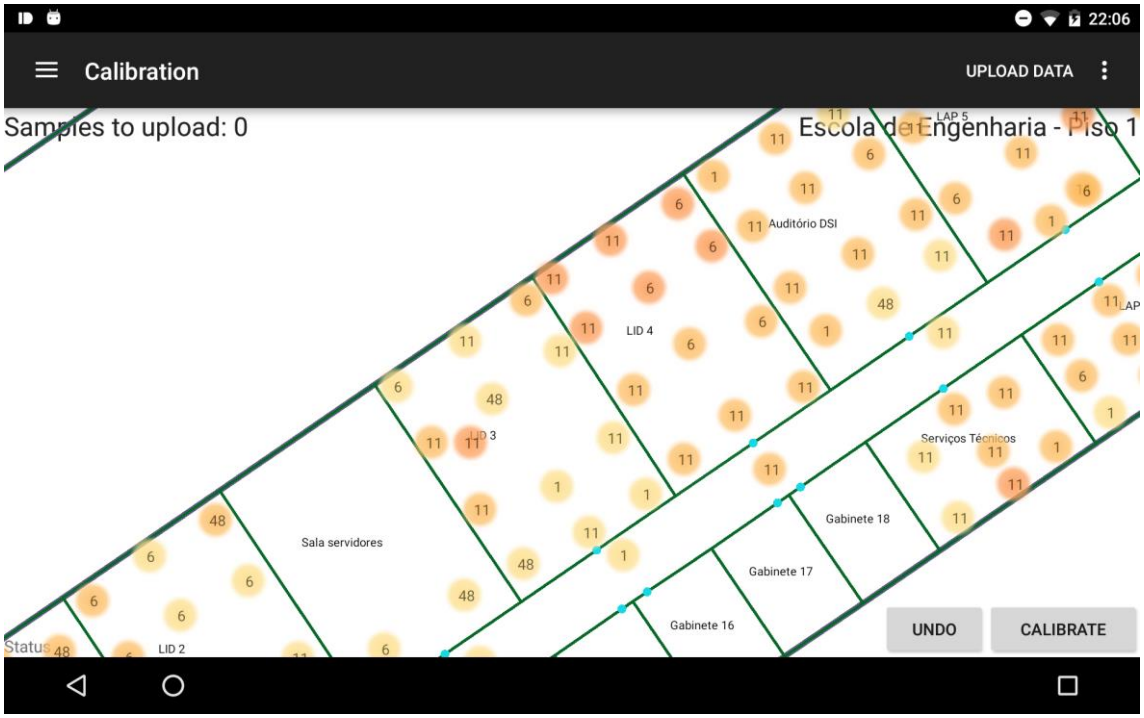


Figura 5.27: Modo de visualização de acordo com o canal mais repetido em cada local - apresentação do canal mais repetido

Nível de sinal (RSSI) mais elevado – é apresentado o valor do nível de sinal mais elevado detetado em cada ponto. Cada *fingerprint* tem um conjunto de APs e os respetivos níveis de sinal, o AP que tiver sinal mais forte permite apresentar sob uma escala de cores o nível de sinal Wi-Fi mais forte em cada local. Nos casos em que são recolhidas amostras no mesmo local é selecionado o nível de sinal mais elevado no conjunto de todas as *fingerprints* recolhidas. Foi necessário definir os níveis de sinal associados à escala de cores apresentada na Tabela 5.7. Quanto mais escura for a cor, melhor é o nível de sinal detetado nesse local. As figuras 5.28 e 5.29 expõem um exemplo dos níveis de sinal associados às *fingerprints* recolhidas no processo de calibração.

Tabela 5.7: Escala de cores utilizada para mostrar o nível de sinal mais forte

Cor	Nível de sinal
	Menor do que -90 dBm
	Entre -90 e -81 dBm
	Entre -80 e -71 dBm
	Entre -70 e -61 dBm
	Entre -60 e -51 dBm
	Entre -50 e -41 dBm
	Entre -40 e -30 dBm
	Maior do que -30 dBm

Figura 5.28: Modo de visualização de acordo com o nível de sinal mais elevado

Figura 5.29: Modo de visualização de acordo com o nível de sinal mais elevado - valores exatos

Seleção dos dados a apresentar

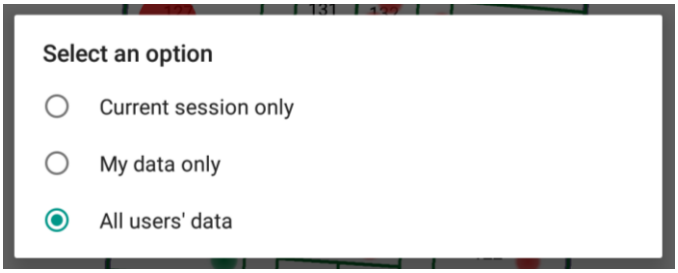


Figura 5.30: Seleção dos dados a apresentar no ecrã de calibração

Além das opções de visualização, o utilizador tem a possibilidade de escolher quais os dados que quer consultar quando está a fazer a calibração (Figura 5.30). Esta opção está presente na fase de seleção da planta, e pode ser alterada sempre que o utilizador desejar no ecrã de calibração a partir do menu de opções.

Esta funcionalidade permite que o utilizador faça uma filtragem dos dados que foram recolhidos anteriormente pelos utilizadores do sistema. No menu estão disponíveis três opções:

- "Current session only" – apresenta apenas as amostras recolhidas na sessão atual;
- "My data only" – apresenta a sessão atual juntamente com os dados de calibração recolhidos pelo utilizador anteriormente;
- "All users' data" – apresenta os dados de calibração recolhidos por todos os utilizadores juntamente com os dados da sessão atual.

Combinando esta opção com os modos de visualização, o utilizador tem acesso a várias informações importantes que o permitem perceber quais são os locais que estão melhor calibrados, quais são as zonas desatualizadas, os pontos com mais densidade de amostras, os pontos com cobertura de mais APs, os locais onde existe mais sobreposição de canais e também os locais onde existe um défice na cobertura de sinal Wi-Fi.

Upload dos dados de calibração

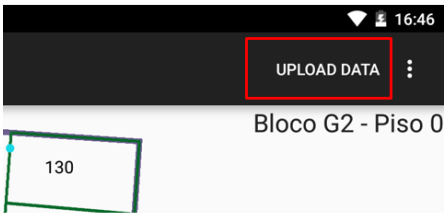


Figura 5.31: Botão de *upload* dos dados recolhidos

Os dados de calibração estão sempre armazenados no dispositivo móvel na base de dados local, mesmo quando o utilizador faz *logout* os dados ficam automaticamente armazenados. No ecrã de calibração existe uma opção para fazer *upload* dos dados para o servidor. O sistema questiona o utilizador se pretende continuar com a operação (Figura 5.32), em caso afirmativo os dados são enviados para o servidor.

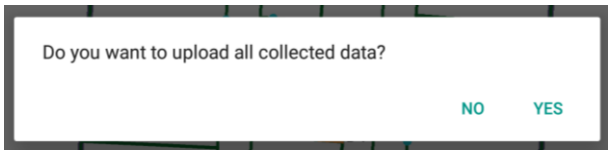


Figura 5.32: Confirmação do *upload* dos dados

Alteração da palavra passe

A Figura 5.33 mostra a secção das definições da aplicação onde o utilizador pode alterar a sua *password* através do preenchimento do formulário com a sua password antiga e a repetição da nova *password.* Verificam-se ambas as *passwords* antes de ser feito o pedido ao servidor e, caso estas coincidam, é feito o pedido ao servidor do qual é obtida a resposta com o resultado da operação da alteração da *password* do utilizador.

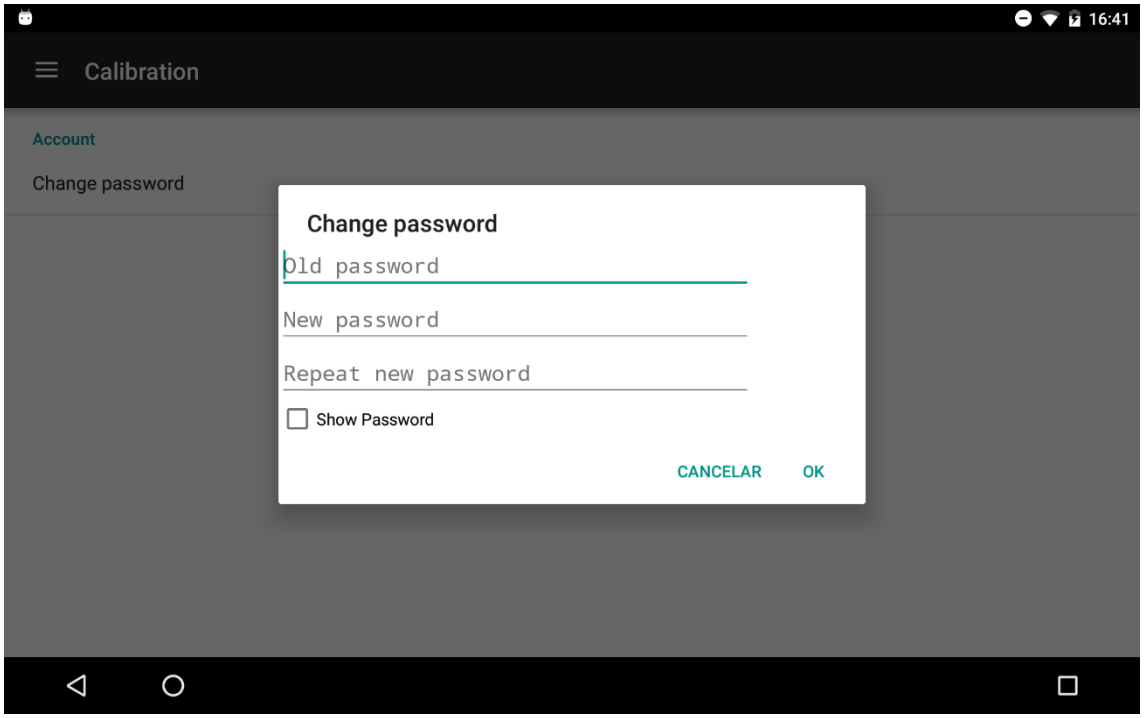


Figura 5.33: Ecrã das definições - alterar password

Recuperação da palavra passe

Caso o utilizador se esqueça da sua palavra passe, esta pode ser recuperada a partir do formulário no qual o utilizador introduz o seu e-mail (Figura 5.34). O sistema verifica a existência do e-mail na base de dados, caso o e-mail esteja associado a um utilizador, é gerada uma nova *password* com 10 caracteres, gerada de forma aleatória, que passa a ser a nova *password* do utilizador. A nova *password* é enviada para o e-mail do utilizador com a informação que deverá alterar a sua *password* logo que possível.

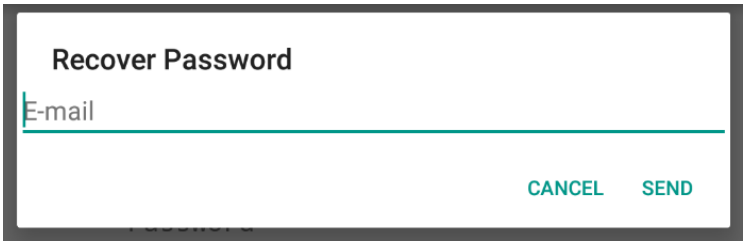


Figura 5.34: Formulário de recuperação da password do utilizador

Sobre

Quando o utilizador consulta as informações da aplicação em "*About*" (Figura 5.35) será possível ver algumas informações sobre a versão da aplicação e o suporte para versões Android. Além disso é mostrada uma nota que explica as cores utilizadas nos diferentes modos de visualização. Descendo no texto estão esclarecimentos para todos os modos de visualização.

Figura 5.35: Ecrã "*About*" na aplicação de calibração

5.3.2. Servidor

A implementação do servidor implica uma análise dos requisitos e das características do sistema.

Dadas as características do sistema, o modelo escolhido para a comunicação entre os componentes do sistema requer a implementação de *web services* REST com integração da linguagem JSON para formato de dados. Esta abordagem em conjunto com a linguagem de *scripting* PHP funcionam bastante bem uma vez que a linguagem PHP é simples, está muito bem documentada e uma das suas principais vantagens é a integração com bases de dados e outras tecnologias.

A extensão PDO (PHP Data Objects) define uma interface para aceder a bases de dados a partir de código PHP. Esta interface permite interagir com a base de dados de uma forma transparente, facilitando as operações CRUD sobre as tabelas.

5.3.2.1. Componentes do servidor

A Figura 5.36 demonstra cada componente do servidor e a tecnologia que o caracteriza. A implementação em PHP consiste em dois ficheiros distintos:

- floormaps.php – Apresenta os módulos de suporte à aplicação e das plantas dos edifícios, uma vez que oferece as funcionalidades necessárias para a aplicação de calibração incluindo os serviços de registo do utilizador e o fornecimento dos dados das plantas de cada piso;
- calibration.php – Trata-se do módulo de construção de mapas de rádio onde são recolhidos os dados de calibração por parte da aplicação. Este módulo também intervém na obtenção dos dados de calibração recolhidos anteriormente, para a apresentação dos mesmos ao utilizador aquando do processo de calibração.

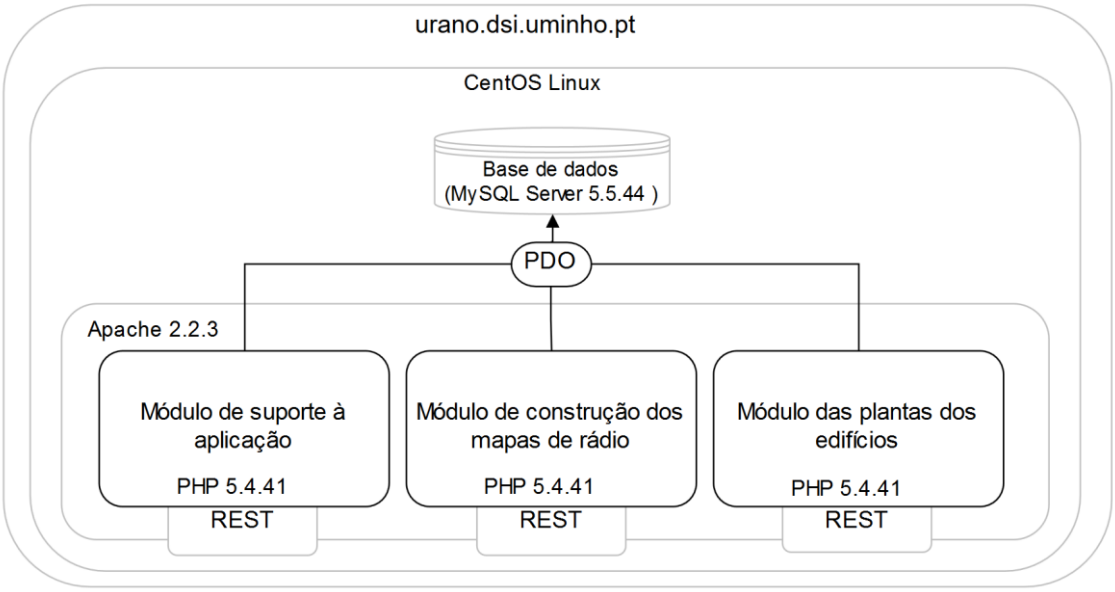


Figura 5.36: Componentes e tecnologias do servidor

5.3.2.2. Base de Dados

A base de dados apresentada nas figuras 5.37, 5.38 e 5.39 apresenta o diagrama de entidades e relacionamentos com todas as entidades do sistema. A base de dados tem como principais funções armazenar os dados de utilizadores, *fingerprints* recolhidas e locais suportados e também tem como função suportar a aplicação de calibração. As entidades representadas nas figuras 5.37, 5.38 e 5.39 não têm relações com chave estrangeira entre si porque assim permitem dar mais flexibilidade ao sistema de forma a ter independência entre cada componente da base de dados:

- Componente de suporte à aplicação móvel;
- Componente dos mapas de rádio;
- Componente das plantas dos edifícios.

Na Figura 5.37 está apresentada a componente das plantas dos edifícios, tal como apresentado em 4.5.2.1. Esta figura mostra as entidades que estão associadas às plantas dos pisos suportados pela aplicação de calibração. Estão também as relações entre cada entidade que permitem associar dados entre as entidades nas entidades com chave estrangeira.

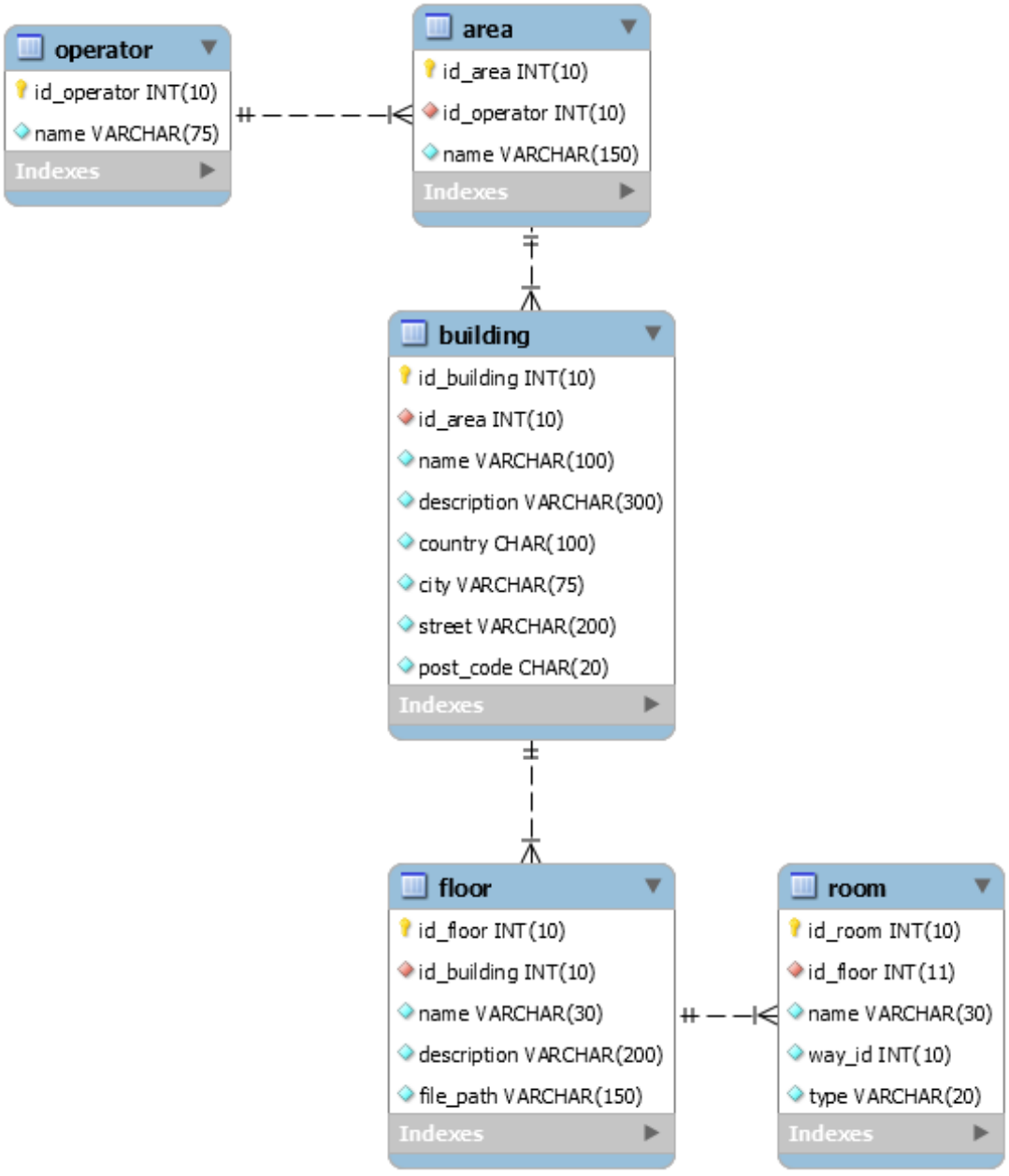


Figura 5.37: Modelo de dados das plantas dos edifícios

A Figura 5.38 representa a componente dos mapas de rádio. Esta componente foca-se nas entidades que armazenam a informação relativa às *fingerprints* recolhidas no processo de calibração. Cada *fingerprint* é recolhida pelo dispositivo do utilizador num dos espaços suportados pela aplicação, permitindo que o mapa de rádio de cada espaço seja construído.

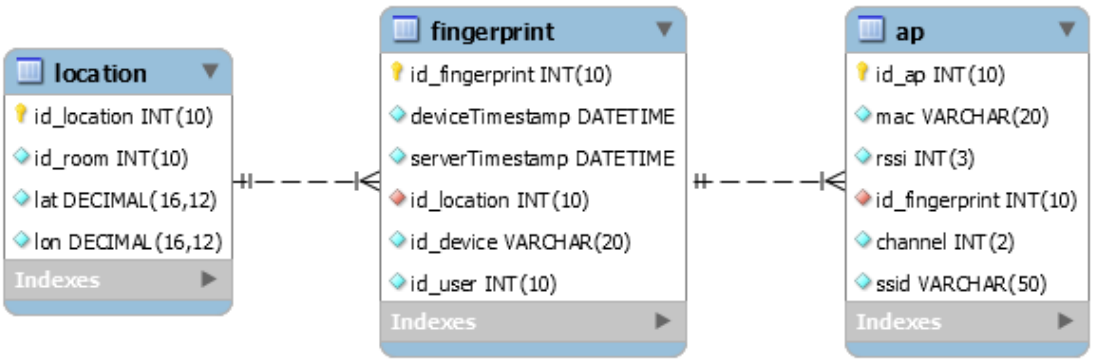


Figura 5.38: Modelo de dados dos mapas de rádio

Tabela localização (location)

A localização é a entidade que representa o local onde o utilizador se encontra. Esse local é caracterizado por um espaço interior específico e um par de coordenadas (latitude e longitude). Quando é feita a recolha de uma *fingerprint* no local, são recolhidas as seguintes informações de localização: identificador do espaço, latitude e longitude. Cada localização tem também um identificador único.

Tabela fingerprint

No processo de calibração são recolhidas várias *fingerprints* nos espaços interiores de cada piso, quando o utilizador pretende fazer a recolha dos dados do ambiente rádio do local, são armazenadas várias informações: *timestamps* tanto do dispositivo como do servidor quando recebe os dados, um identificador da localização que expressa o sítio em que o utilizador fez a recolha dos dados e os identificadores do utilizador e do dispositivo.

Tabela Access Point (ap)

Na fase de calibração, os dados dos pontos de acesso são obtidos em cada análise do ambiente de rádio onde o objetivo é detetar os pontos de acesso e os respetivos níveis de sinal. Os dados dos pontos de acesso que são associados a cada *fingerprint* são: endereço MAC, RSSI

(*received signal strength indicator*), canal (frequência de operação do ponto de acesso) e o SSID (nome da rede do ponto de acesso).

Por último, a Figura 5.39 apresenta a componente dedicada aos utilizadores e dispositivos. São estas entidades que permitem dar suporte à aplicação de calibração onde os dados do utilizador estão associados às *fingerprints* recolhidas, participando assim no processo de construção dos mapas de rádio.

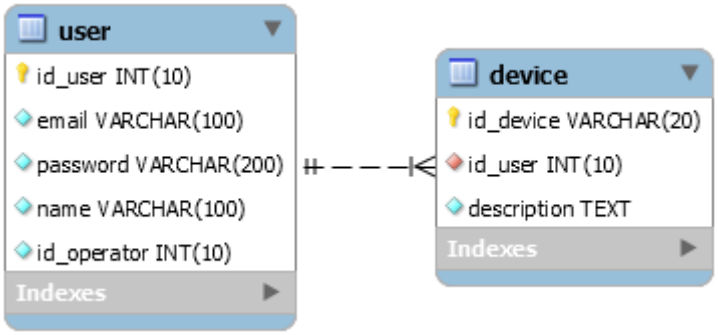


Figura 5.39: Modelo de dados associado aos utilizadores e respetivos dispositivos

Tabela Utilizador (user)

Os dados dos utilizadores profissionais que realizarão a fase de calibração são armazenados na tabela dos utilizadores (user). Cada utilizador deverá ter uma conta à qual está associado um e-mail e uma *password* que são as credenciais de acesso na aplicação de calibração. Cada utilizador tem que estar associado a um operador, que é a entidade que tem domínio sobre os edifícios. Da mesma forma, os dispositivos do utilizador estão associados a si através da chave estrangeira na entidade "*device*". Um utilizador pode ter vários dispositivos que estes são registados devidamente na base de dados.

Tabela Dispositivo (device)

Representa os dispositivos dos utilizadores com os seguintes dados: identificador do dispositivo (endereço MAC do dispositivo móvel), identificador do utilizador que tem o dispositivo e uma descrição do dispositivo.

5.3.3. Comunicação entre os componentes do sistema (Web Services)

Na secção 4.4.1.3 foi apresentada a justificação para a escolha das tecnologias utilizadas nos *web services*. A implementação destes serviços resultou num conjunto de pedidos que podem

ser feitos ao servidor para consultar, inserir ou remover informação. No Anexo C estão descritos todos os pedidos, com exemplos de pedidos efetuados e a resposta obtida.

5.3.3.1. Comunicação com o servidor

As funcionalidades da aplicação de calibração que dependem da comunicação entre a aplicação e o servidor são as seguintes:

- Registo de um utilizador – permite à aplicação móvel de calibração efetuar o registo de um utilizador. Os dados são enviados no formato de dados JSON (e-mail, *password*, nome, operador, e endereço MAC do dispositivo).
- Recuperação da *password* do utilizador – é um pedido que recebe o e-mail do utilizador e procede à geração de uma nova palavra-chave que é enviada via e-mail para o utilizador.
- Alteração da *password* do utilizador – na aplicação de calibração o utilizador pode alterar a sua *password.* Esta operação permite que a *password* do utilizador seja alterada através do envio dos dados (e-mail, *password* nova*, password* antiga).
- Obter os dados do utilizador – devolve os dados do utilizador num objeto JSON (identificador do utilizador, e-mail e o identificador do operador).
- Apagar utilizador – apaga um utilizador dado o seu identificador.
- Receber os dados de uma *fingerprint* – recebe os dados de uma *fingerprint* no formato de dados JSON e armazena esses dados nas respetivas entidades da base de dados.
- Obter os dados das *fingerprints* recolhidas num piso – retorna uma lista de *fingerprints* e as suas informações básicas (id da *fingerprint*, id do utilizador e as coordenadas (lat, lon)) num objeto JSON.

É fundamental garantir que o dispositivo tenha ligação à internet antes de se fazer qualquer um destes pedidos e, por isso, sempre que o dispositivo não estiver ligado à internet, será apresentada uma caixa de diálogo a sugerir ao utilizador que faça a ligação a uma rede com acesso à internet.

Qualquer pedido efetuado ao servidor recebe uma resposta com uma mensagem de estado constituída por um código de estado e uma descrição. Existem vários códigos de estado, dos quais se destacam: 200 "*OK*", 201 "*CREATED*", 400 "*Bad Request*", 404 "*Not Found*". Estes códigos permitem perceber imediatamente se o pedido foi bem-sucedido.

Na Figura 5.40 está representado o modelo de comunicação com o servidor onde é possível observar os pedidos que podem ser feitos ao servidor.

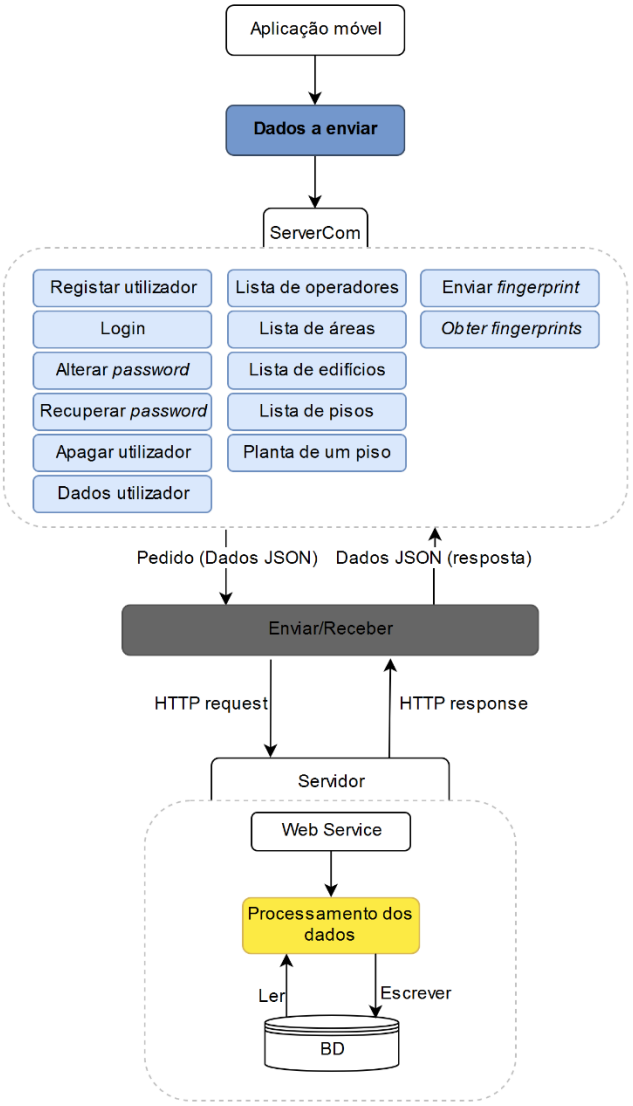


Figura 5.40: Modelo de comunicação com o servidor

5.4. Testes efetuados na Aplicação de Calibração

Ao longo do desenvolvimento da aplicação foram efetuados testes a cada funcionalidade à medida que se adicionavam novas funcionalidades. No final, foi feito um teste global no qual foram testadas todas as funcionalidades da aplicação assim como a sua estabilidade.

A Tabela 5.8 apresenta uma lista dos testes efetuados com o seu respetivo resultado. É importante realçar que este teste foi feito depois de várias iterações onde foram corrigidos diversos erros. No momento deste teste, todas as funcionalidades analisadas passaram com sucesso.

Tabela 5.8: Testes efetuados na aplicação de calibração

Teste	Resultado
Autenticação de utilizador	Passou
Apresentação da lista de áreas e os respetivos edifícios e pisos.	Passou
Apresentação de uma caixa de diálogo com um aviso para o utilizador se ligar à internet quando pretende fazer uma operação que depende da conexão à internet.	Passou
Apresentação de mensagens de aviso/erro: • Na seleção da planta, quando não existe um piso selecionado; • Quando o utilizador tenta calibrar num local fora do espaço interior do edifício; • Quando o utilizador faz "undo" para remover as *fingerprints* sendo que já não existem quaisquer *fingerprints* registadas na sessão atual; • Perguntar se o utilizador realmente pretende fazer *logout*; • Quando o utilizador tenta fazer o *upload* dos dados e não existe qualquer *fingerprint* armazenada no dispositivo.	Passou
Calibração de vários pontos com recurso à planta do piso.	Passou
Diferentes modos de visualização dos pontos previamente calibrados.	Passou
Envio dos dados de calibração para o servidor	Passou
Recuperação da *password.*	Passou
Alteração da *password.*	Passou

A aplicação de calibração também foi sujeita a vários testes que verificaram o gasto de bateria do dispositivo tendo em conta a utilização da aplicação.

Nos testes iniciais na universidade, o objetivo foi recolher várias *fingerprints* e fazer o envio das mesmas para o servidor, fazendo a análise dos gastos de bateria no dispositivo. Os resultados obtidos foram reunidos na Tabela 5.9 e na Tabela 5.10.

Tabela 5.9: Monitorização dos níveis de bateria do dispositivo

Hora	% bateria	Nº de Amostras (Acumulado)
10h00	94%	0
10h05	92%	40
10h16	90%	120
10h35	86%	280
10h45	84%	280

Tabela 5.10: *Upload* dos dados de calibração

Nº Amostras	Resultado da operação	Tempo
20	*Bem-sucedido*	10s
40	*Bem-sucedido*	23s
80	*Bem-sucedido*	2m02s
160	*Bem-sucedido*	1m30s

Os testes foram desenvolvidos ao longo de 45 minutos, onde a fase inicial passou pela recolha de *fingerprints* em vários espaços da escola de Engenharia do Campus de Azurém da Universidade do Minho. O dispositivo utilizado nos testes foi um *tablet* (Nvidia Shield) com a versão Android 6.0. O teste foi desenvolvido com o brilho do ecrã a 50% (ter em atenção que o ecrã é dos elementos que mais contribuem no consumo de bateria nos dispositivos móveis).

Nos minutos iniciais, entre as 10h00 e as 10h35 foram recolhidas amostras do ambiente de rádio de vários espaços. O objetivo é foi simular a utilização da aplicação em contexto real, e por isso foram recolhidas cerca de 5 amostras em cada ponto. Depois de recolhidas as primeiras 20 amostras, foi feito um teste para determinar o tempo que demora a fazer o envio dos dados para o servidor (10s). Recolheram-se mais 40 amostras e fez-se o envio das mesmas, que demorou 23s para que todos os dados tenham sido enviados com sucesso. Fez-se o mesmo procedimento com a recolha de 80 amostras, que demorou 2m02s a fazer o envio dos dados para o servidor. Por fim, recolheram-se 160 amostras e procedeu-se ao envio das mesmas, que desta vez demorou 1m30s.

Por volta das 10h35 tinham sido recolhidas 260 amostras no total. Desde o início do teste até às 10h35 a bateria teve um consumo de 8% através de uma utilização intensiva com a recolha de dados quase constante. Nos 10 minutos finais, apenas se utilizou a aplicação para testar e

verificar os modos de visualização, e por isso não se recolheram mais dados. No final do teste, ao longo de 45 minutos, a bateria desceu cerca de 10%.

Os resultados obtidos não permitem tirar conclusões relativamente à influencia da quantidade de *fingerprints* no tempo que estas demoram a ser enviadas para o servidor. O tempo de *upload* provavelmente depende da conexão à internet e da qualidade da mesma uma vez que o *upload* dos dados foi feito em locais diferentes onde a ligação à internet varia na sua qualidade.

Tendo em conta os consumos de energia normais de um *tablet*, o consumo de energia ao longo do teste efetuado é considerado normal uma vez que o ecrã do dispositivo esteve sempre ligado contribuindo para o aumento do consumo de energia.

5.4.1. Experiência no mundo real

Com o objetivo de utilizar a aplicação para construir o mapa de rádio do piso 1 da Escola de Engenharia do Campus de Azurém da Universidade do Minho, foram recolhidas *fingerprints* nos espaços interiores do DSI (Departamento de Sistemas de Informação).

Foram apenas recolhidas amostras nos laboratórios de desenvolvimento, nas salas de aula e salas de reuniões. Cada espaço interior foi visto como uma matriz o que levou a uma recolha de amostras uniforme, salvo raras exceções devido à estrutura interior da sala. Foram também recolhidas amostras nos corredores e espaços comuns dentro do edifício. Em cada ponto obteve-se cerca de 6 amostras com o intuito de obter dados com mais qualidade.

Ao todo foram recolhidas 1717 *fingerprints* às quais estão associados 42202 registos de pontos de acesso e o seu respetivo nível de sinal.

A Tabela 5.11 apresentam capturas de ecrã com o resultado do processo de calibração do DSI de acordo com os diferentes modos de visualização.

Tabela 5.11: Resultados da calibração no DSI da Escola de Engenharia

Modo visualização	Captura de ecrã
Normal	

Figura 5.41: Modo de visualização normal

Antiguidade das *fingerprints*	

Figura 5.42: Modo de visualização pela antiguidade de amostras

Densidade

Figura 5.43: Modo de visualização por densidade de amostras

APs detetados

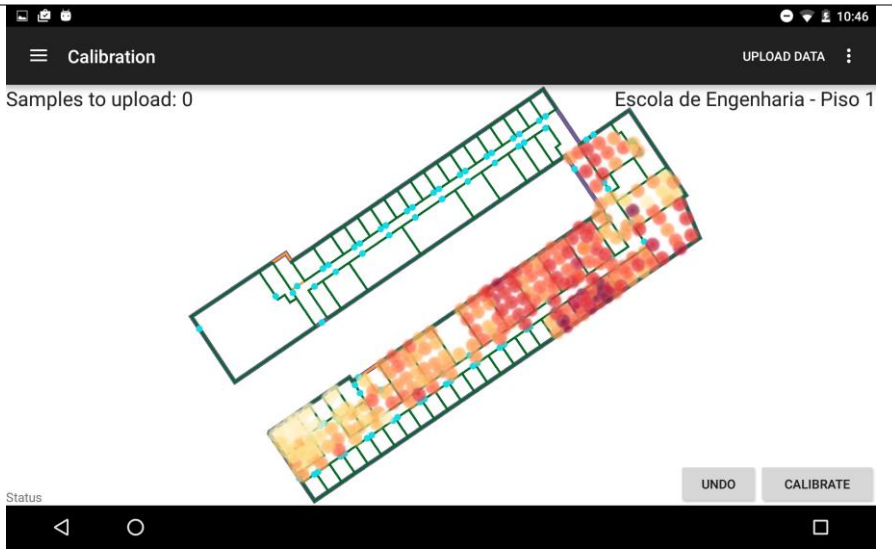


Figura 5.44: Modo de visualização pelo número de APs detetados

Canal mais repetido

Figura 5.45: Modo de visualização pelo canal mais repetido

Nível de sinal mais elevado

Figura 5.46: Modo de visualização pelo nível de sinal mais elevado

Ao longo de 2 dias, num total de 5h35m foi possível obter o mapa de rádio do DSI com recurso à aplicação de calibração. Sem o auxílio da aplicação que apresenta a planta do piso, a tarefa de calibração seria bastante mais demorada e complicada. Estes resultados mostram que em média foram recolhidas 5 amostras por minuto.

5.4.2. Análise dos dados recolhidos

Pode ser feita uma análise aos dados recolhidos através da aplicação de calibração que permite a visualização das amostras recolhidas, da sua idade e da densidade (número de amostras recolhidas em cada ponto).

Como se pode observar na Figura 5.42 todas as amostras recolhidas são recentes e por isso são apresentadas a verde quando se utiliza o modo de visualização relativo à idade das *fingerprints.*

Quanto à densidade de amostras recolhidas, a Figura 5.43 mostra que o mapa de rádio está maioritariamente a verde, indicando que cada ponto verde tem pelo menos 4 amostras associadas. Os pontos amarelos estão associados a pontos com 3 amostras recolhidas.

Através dos outros modos de visualização é possível determinar os espaços onde a cobertura Wi-Fi é melhor, os espaços onde existe elevada densidade de APs e os locais onde os níveis de sinal Wi-Fi são baixos para utilização destas redes.

Foram identificados espaços onde existe uma densidade elevada de APs, principalmente no lado exterior do edifício no LAP 6 e nos serviços técnicos. Trata-se de salas que estão viradas para outro edifício onde poderão ser detetados APs desse edifício. Existem casos onde foram detetados mais do que 40 APs no mesmo local, sendo que isto prejudica seriamente a infraestrutura Wi-Fi. As extremidades do edifício são as zonas onde são detetados menos APs em cada ponto, o que se deve à densidade mais reduzida de pontos de acesso nessas zonas.

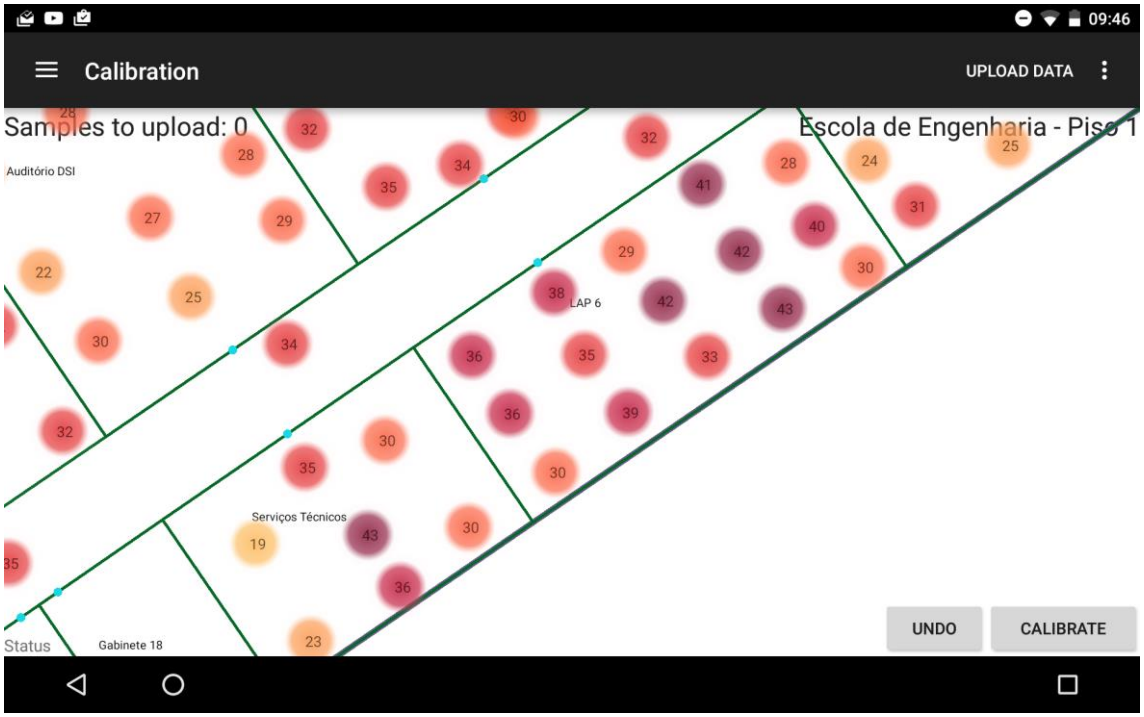


Figura 5.47: Visualização dos espaços com mais densidade de APs

Quanto à sobreposição de canais, os canais com mais sobreposição são os canais 1, 6 e 11. Estes valores são de prever uma vez que esses são os canais cujas frequências não se sobrepõem entre si. De qualquer forma, existem zonas do edifício onde existem várias redes com sobreposição do mesmo canal. Na Figura 5.48 verifica-se que no LAP 6 existem vários pontos onde existem entre 7 e 9 redes com o canal 11. Tal como foi verificado anteriormente, esta é a zona mais afetada pela densidade de APs e com mais sobreposição de canais, existindo um caso onde existem entre 13 e 15 redes configuradas para transmissão no canal 11. A Figura 5.45 demonstra que as extremidades do edifício são as zonas onde a sobreposição de canais é menor.

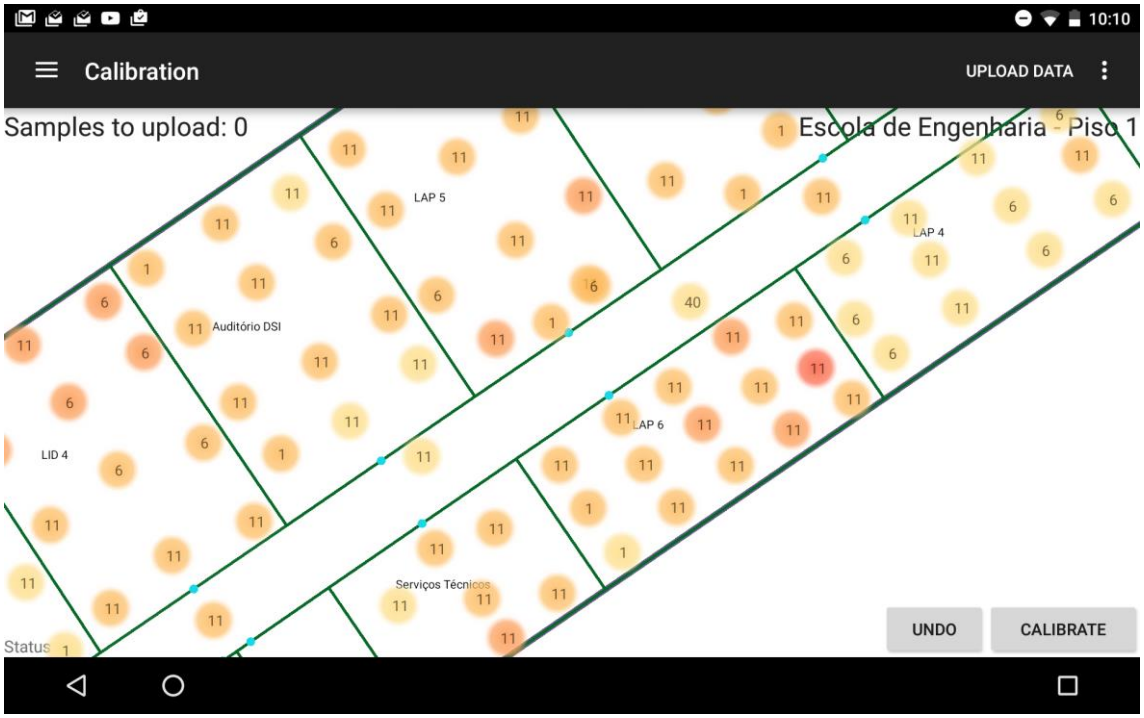


Figura 5.48: Visualização dos locais com mais sobreposição de canais

Relativamente aos níveis de sinal, os locais onde são detetados mais APs estão associados aos níveis de sinal mais fortes dentro do edifício. Nos limites do edifício estão os locais onde os níveis de sinal são mais fracos, esses locais surgem com as cores mais claras na Figura 5.46. Mais uma vez, a zona do LAP 6, LAP 4 e Serviços Técnicos surgem como as zonas que se destacam, desta vez com os níveis de sinal mais fortes (Figura 5.49).

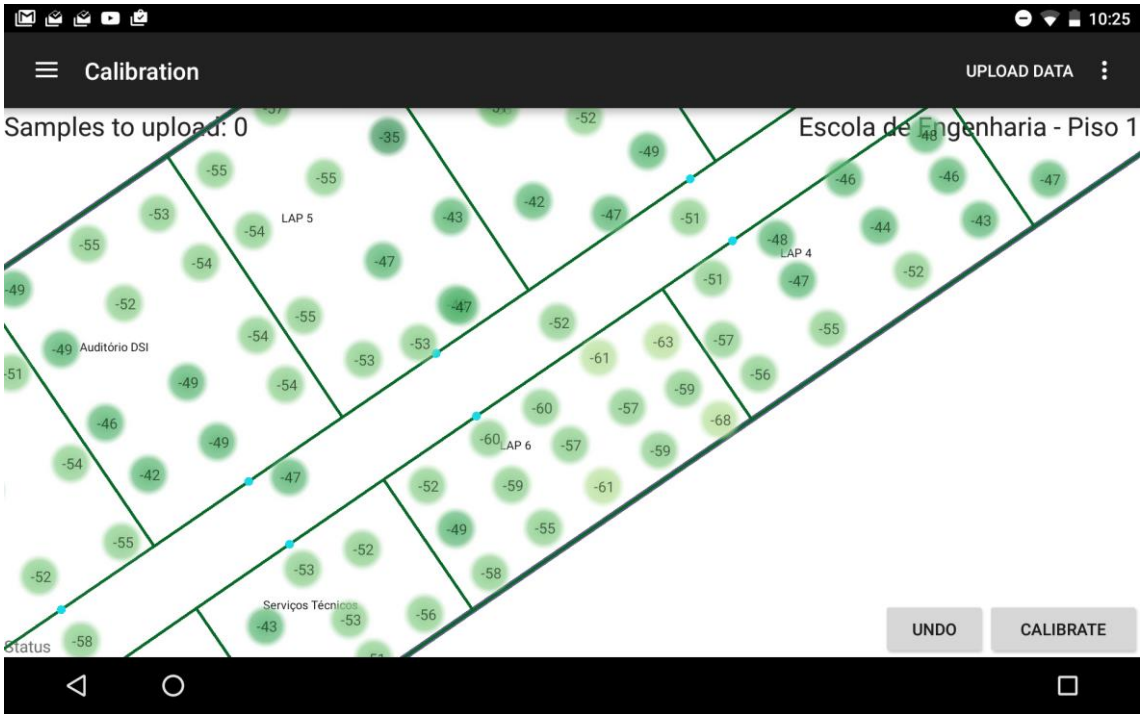


Figura 5.49: Visualização dos locais com níveis de sinal mais fortes

5.4.3. Conclusões

A construção do mapa de rádio do DSI foi feita com sucesso através da aplicação de calibração, que facilita o processo através da recolha de várias *fingerprints*. Foram recolhidas várias amostras em cada local de modo a fornecer mais dados criando mapas de rádio mais ricos.

A aplicação de calibração também permite ver o estado atual de calibração, os locais que ainda não foram calibrados, as zonas em que os dados recolhidos estão associados a *fingerprints* antigas e ainda os locais calibrados com mais densidade de amostras. À medida que é feita a calibração o utilizador pode gerir da melhor forma os locais a calibrar, através deste *feedback* que o sistema fornece.

Os modos de visualização também dão a capacidade de analisar a cobertura Wi-Fi do edifício. Foram detetadas zonas onde existe uma sobrecarga de redes Wi-Fi que prejudica os utilizadores principalmente porque nos locais com mais pontos de acesso existe mais sobreposição de canais. Também foram identificados os espaços onde a cobertura de redes Wi-Fi é mais reduzida, existe menos densidade de APs que consequentemente apresenta menor sobreposição de canais. Nessas zonas estão associadas aos pontos onde foram detetados sinais Wi-Fi mais fracos.

CAPÍTULO 6 – O SISTEMA WHERE@UM

O sexto capítulo descreve o desenho e implementação da componente de visualização da localização integrada no sistema Where@UM. Este sistema estima a localização através da correspondência das *fingerprints* recolhidas com os dados obtidos pelos sensores do dispositivo em tempo real.

O módulo integrado na aplicação Where@UM será parte central do sistema sendo dedicado à visualização das posições estimadas em tempo real. Os utilizadores da aplicação Where@UM podem consultar a sua própria localização e a localização dos seus amigos ao nível de uma sala/espaço interior através da componente de visualização (módulo integrado).

A apresentação das plantas no dispositivo Android é feita através da API Android desenvolvida na solução das plantas dos edifícios.

6.1. Apresentação do sistema Where@UM

O sistema Where@UM foi introduzido anteriormente na secção 2.4.8 onde foram analisadas as suas principais funcionalidades como sistema de posicionamento de interiores colaborativo.

O funcionamento do sistema depende da colaboração dos utilizadores que podem adicionar novos locais que ainda não são suportados pelo sistema. A estimação da localização dos utilizadores é feita através de Wi-Fi *fingerprinting,* e permite que os utilizadores consultem a sua localização e a dos seus amigos.

Atualmente o sistema mostra a localização dos utilizadores no formato textual, tal como mostra a Figura 6.1 e a Figura 6.2. Além disso, não existe forma de perceber exatamente onde é que os amigos do utilizador se encontram, não permite consultar a localização dos amigos com um mapa.

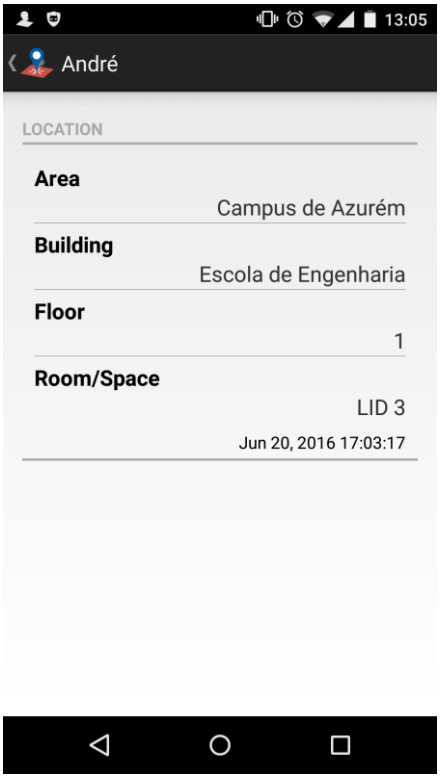


Figura 6.1: Localização dentro da UM

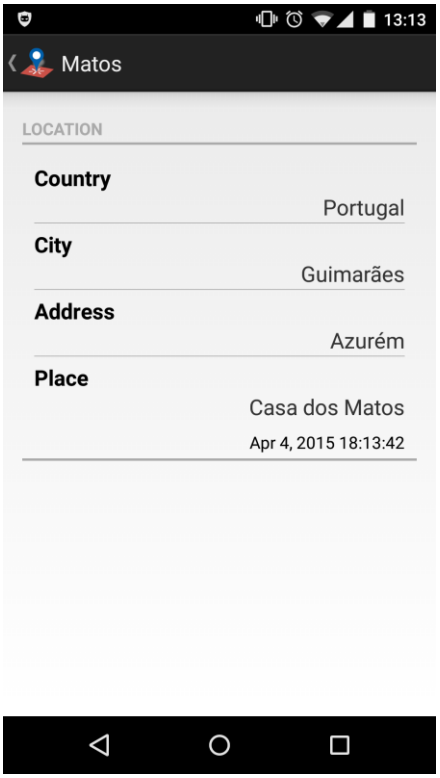


Figura 6.2: Localização fora da UM

No âmbito dos sistemas de posicionamento, a apresentação da localização através das plantas dos edifícios é mais rica e interativa do que a representação em forma de texto. Com as plantas dos edifícios os utilizadores podem perceber com facilidade a sua localização dentro de um edifício. Para a perceção do estado atual da localização dos amigos, uma vista geral com recurso a um mapa fornece aos utilizadores a possibilidade de saberem a sua localização juntamente com a localização dos seus amigos distribuídos num mapa.

Relativamente ao servidor do sistema Where@UM (Figura 2.17), a comunicação fornecida pelo servidor não apresenta qualquer tipo de segurança na troca de dados efetuada entre a aplicação e o servidor. Os dados dos utilizadores e das *fingerprints* recolhidas são trocados sem verificação da integridade e confidencialidade dos mesmos e sem mecanismos de encriptação. A implementação de mecanismos de controlo de sessão permite aumentar a segurança na comunicação e nos dados trocados entre aplicação e servidor. Também é importante garantir que os dados críticos (*passwords*) não são enviados "às claras" na comunicação porque podem estar sujeitos a ataques *man-in-the-middle*.

6.2. Requisitos

O módulo de visualização para a app Where@UM deverá permitir que os utilizadores consultem a sua localização e a localização de cada utilizador (amigo) ao nível de uma sala/espaço de um edifício ou de um modo geral onde aparecem os utilizadores amigos espalhados num mapa. Esta componente exige um serviço que apresente ao utilizador as informações da sua localização e dos outros utilizadores (amigos).

A visualização da localização na app Where@UM deverá ser feita de duas formas: a primeira é uma *overview* dos utilizadores num mapa através de um par de coordenadas (latitude, longitude); a segunda permite observar a localização ao nível de um edifício em que é mostrada a localização do utilizador e dos seus amigos como pontos na planta do espaço interior do piso.

Este módulo necessita de acesso às plantas dos espaços interiores e da API Android que permite fazer o *rendering* das plantas no dispositivo móvel.

6.3. Desenho do módulo de visualização

O desenho do sistema de visualização aborda as informações necessárias para o funcionamento do sistema, a análise dos requisitos e termina na arquitetura do sistema. Este processo facilita a passagem para a fase de implementação.

6.3.1. Informação necessária para a componente de visualização

O funcionamento do módulo integrado na aplicação Where@UM depende das seguintes informações:

- Informações dos utilizadores (nome, e-mail, *password* e a localização do utilizador através de um par de coordenadas (latitude, longitude), área, edifício e o piso);
- Informações dos utilizadores amigos (nome de utilizador, localização através de um par de coordenadas (latitude, longitude), área, edifício e o piso);
- Plantas dos edifícios.

6.3.2. Requisitos técnicos

Requisitos funcionais

- Visualizar a posição do dispositivo estimada pelo sistema, em tempo real, num mapa onde aparece marcada a localização como o ponto no mapa;

- Visualizar a posição do dispositivo estimada pelo sistema num espaço interior de um edifício com recurso à planta do piso;
- Visualizar a localização dos amigos como pontos num mapa (vista geral ou *overview*);
- Visualizar a localização dos amigos em espaços interiores de um edifício com recurso à planta do piso.

Requisitos não funcionais

- Rapidez na obtenção de resposta aos pedidos efetuados ao servidor;
- Rapidez na obtenção e renderização das plantas no dispositivo;
- Desempenho e fluidez nas interações com o utilizador, nomeadamente na secção onde é apresentada a localização através da planta do piso;
- Conexão ao servidor via internet e comunicação entre ambos (dispositivo móvel e servidor).

6.3.3. Arquitetura do sistema de visualização

No sistema Where@UM, a componente de visualização da aplicação obtém os dados de localização dos utilizadores estimados pelo sistema de posicionamento permitindo que a localização dos utilizadores seja apresentada com recurso a um mapa ou à planta do edifício.

6.3.3.1. Arquitetura geral do sistema

Na Figura 6.3, o sistema Where@UM é constituído por uma aplicação móvel (*client-side*) onde são feitas todas as interações com o utilizador, e por um sistema central constituído pelo servidor e base de dados.

Por um lado, a aplicação é o elemento que recolhe as *fingerprints* deste sistema de posicionamento *indoor* colaborativo, além disso fornece várias funcionalidades aos utilizadores que utilizam a plataforma, nomeadamente a partilha da sua localização com os amigos e permite o chat entre utilizadores amigos. Por outro lado, o sistema central é responsável pela recolha e processamento dos dados da aplicação, tanto os dados dos utilizadores como os dados de localização obtidos a partir das *fingerprints.* O sistema central também inclui a componente que foi desenvolvida para fornecer as plantas dos edifícios à aplicação. O servidor do sistema Where@UM é um elemento fulcral para o sistema de posicionamento uma vez que realiza a estimação da localização dos utilizadores.

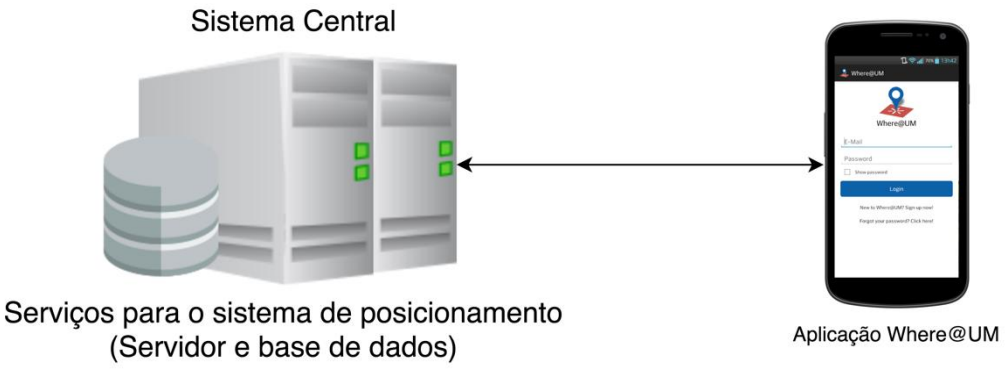


Figura 6.3: Arquitetura geral do sistema Where@UM

A Figura 6.4 apresenta os componentes da arquitetura do sistema Where@UM. O *client-side* é representado pela aplicação Where@UM instalada num dispositivo móvel. Por sua vez, o *server-side* é composto por dois módulos independentes que servirão como suporte à aplicação móvel.

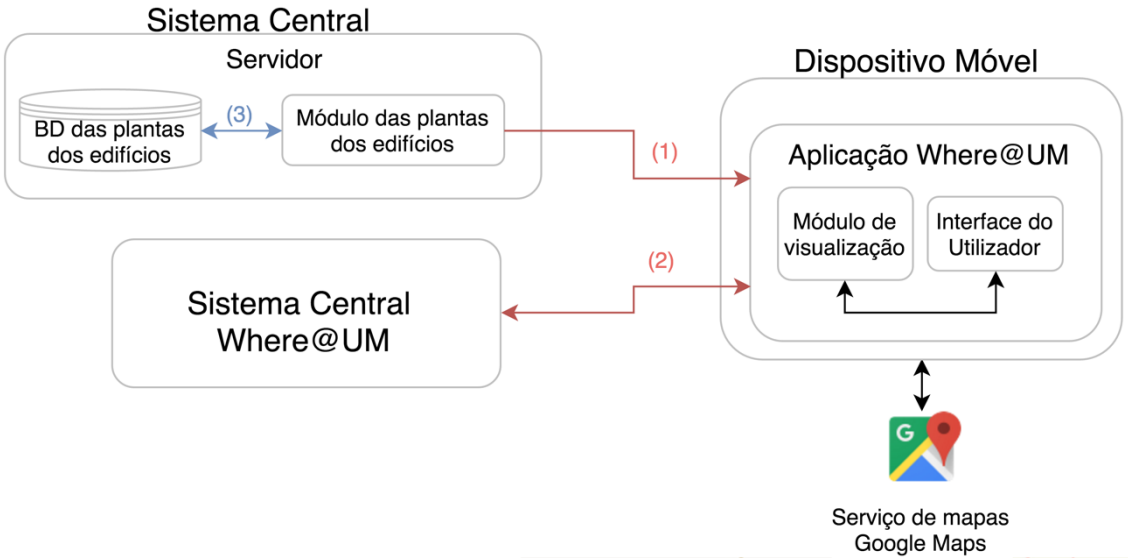


Figura 6.4: Arquitetura do sistema Where@UM - componentes

6.3.3.2. Componentes do sistema Where@UM

Os três principais componentes do sistema Where@UM são a aplicação móvel, o sistema central Where@UM e o sistema que fornece as plantas dos edifícios. Este último é um componente novo com a função de dar suporte à aplicação, além disso pode ser utilizado por outros serviços se for necessário.

Aplicação de Where@UM

A aplicação Where@UM surge como meio de inclusão da componente de visualização, uma vez que se trata de uma aplicação constituinte de um sistema de posicionamento *indoor* colaborativo.

De forma a funcionar como esperado, a aplicação necessita de duas ligações distintas, nomeadamente, o módulo de suporte à aplicação e o módulo das plantas dos edifícios. O primeiro faz parte do sistema central Where@UM tem a função de fornecer as operações de funcionamento à aplicação. O segundo é responsável por fornecer a planta do piso, caso exista, de modo a que esta seja apresentada na aplicação.

A funcionalidade "*overview*" depende do acesso à API do Google Maps que permite a utilização deste serviço em aplicações Android.

Sistema Central

O sistema central foi descrito com detalhe na secção 4.5.2 tem como principal função fornecer as plantas dos edifícios à aplicação Where@UM com o intuito de enriquecer o conteúdo apresentado na aplicação onde é substituída a representação textual pela representação gráfica da planta do piso.

Sistema Central Where@UM

Sendo o elemento central do sistema Where@UM este elemento representa o servidor, bases de dados e motor de posicionamento do sistema. Além disso fornece suporte à aplicação tanto nas funcionalidades necessárias para o bom funcionamento, como nas funcionalidades associadas ao posicionamento.

Este componente encontra-se devidamente documentado em [43] e [44].

6.3.3.3. Comunicação entre componentes da arquitetura

Na Figura 6.4 estão apresentadas as ligações entre as componentes da arquitetura. Nesta secção são descritas todas as ligações, tendo em conta as funções por si suportadas.

(1) Aplicação Where@UM ↔ módulo das plantas dos edifícios

(1) Define a comunicação HTTP entre a aplicação Where@UM e o módulo das plantas dos edifícios. A comunicação entre estes dois componentes tem o intuito de obter a planta do piso.

(2) Aplicação Where@UM ↔ módulo de suporte à aplicação

(2) Sintetiza as ligações efetuadas ao sistema central Where@UM. Essas ligações estão descritas em [43], [44].

6.4. Implementação da componente de visualização na aplicação Where@UM

A presente secção apresenta a implementação das principais alterações na aplicação Where@UM que possibilitaram a integração da componente de visualização.

6.4.1. Aplicação Where@UM

A Figura 6.5 mostra a arquitetura da aplicação Where@UM incluindo os módulos novos que permitem a apresentação gráfica da localização dos utilizadores.

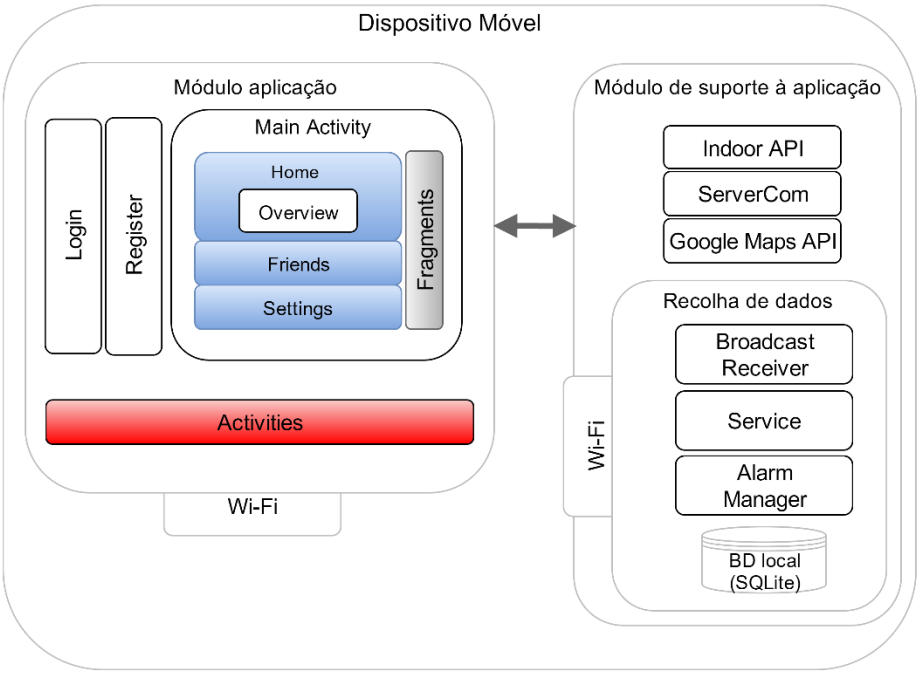


Figura 6.5: Componentes da aplicação Where@UM

Distinguem-se dois módulos na aplicação móvel, uma vez que têm funções bastante diferentes:

- Módulo aplicação – apresenta as *activitys* e *fragments* onde são geridas todas as interações com o utilizador;
- Módulo de suporte à aplicação - é dedicado ao suporte à aplicação, e divide-se em vários sub-módulos com funções distintas:
    - Indoor API – (novo sub-módulo) responsável por receber os dados da planta de um piso (no formato XML), processa-los e depois apresentar a planta no ecrã.

Este sub-módulo desenvolvido para a solução das plantas dos edifícios é semelhante ao sub-módulo Indoor API da aplicação de calibração, mas distingue-se pela possibilidade de apresentar os utilizadores dentro de espaços interiores (salas/escritórios).

- ServerCom – (sub-módulo com novas funcionalidades) a comunicação com o servidor é assegurada por este sub-módulo que realiza os pedidos e obtém as respostas devolvidas pelo servidor.
- Google Maps API – (novo sub-módulo) fornece as funcionalidades que permitem utilizar os serviços Google Maps e colocar marcadores nos pontos desejados. Permite que a localização dos utilizadores seja apresentada com recurso aos marcadores.
- Recolha de dados – sub-módulo existente responsável pela recolha de dados Wi-Fi a partir da interface de rede do dispositivo. Estes dados são armazenados localmente quando o dispositivo não tem acesso à internet. O *broadcast receiver* responde a anúncios de *broadcast*, neste caso está associado ao scan Wi-Fi. O *service* é um serviço que corre dentro de intervalos temporais para recolher *fingerprints*, que são enviadas para o servidor ou armazenadas localmente. O *alarm manager* é necessário para executar o *service* quando o dispositivo é ligado. A base de dados local (BD local) permite armazenar os dados recolhidos quando não existe conectividade à internet.

6.4.2. Servidor

O servidor sofreu algumas alterações de modo a aumentar a segurança na comunicação com a aplicação móvel. Para permitir a retro-compatibilidade com as versões anteriores é necessário disponibilizar a versão anterior dos *web services*, de modo a não afetar o funcionamento das aplicações.

Foi desenvolvida uma nova versão para o servidor de modo a suportar aplicações de diversas plataformas e aumentar a segurança nos serviços de comunicação. As alterações efetuadas foram também ao nível do modelo de dados.

Em termos estruturais e tecnológicos, o servidor mantém a mesma estrutura. Essencialmente são dois ficheiros PHP (asmAPI.php e peAPI.php) que lidam com os pedidos das

aplicações e retornam as respostas aos mesmos. A estrutura do servidor é composta por um módulo de suporte à aplicação móvel e um módulo de construção de mapas de rádio [43].

O módulo das plantas dos edifícios na Figura 6.4, introduzido 4.5.2 em, é independente do servidor Where@UM mas deverá ser considerado como um elemento deste sistema principalmente porque a sua função é fornecer as plantas dos edifícios à aplicação Where@UM.

A Figura 6.6 apresenta as tecnologias da base de dados e do servidor que permitem obter a planta do piso de um edifício. A comunicação é assegurada através da interface REST, através de um pedido o servidor devolve a planta do piso.

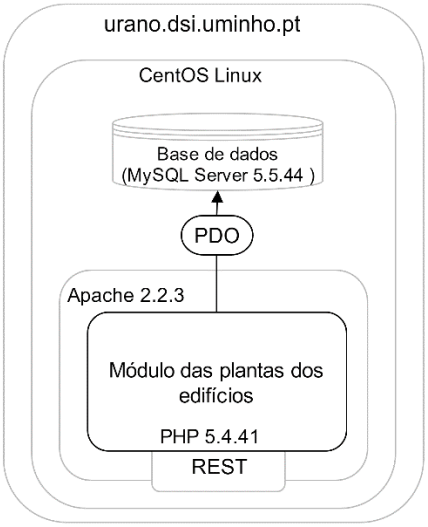


Figura 6.6: Novo componente no servidor Where@UM e respetivas tecnologias

6.4.3. Obtenção da planta do piso

Quando a aplicação faz um pedido de uma planta, o servidor verifica a existência da planta do piso e caso ela exista, devolve o conteúdo da planta no formato XML.

A Figura 6.7 apresenta o diagrama de sequência explicativo do processo de obtenção da planta de um piso. Inicialmente o utilizador seleciona o amigo que pretende consultar, os dados relativos à localização do amigo (operador, área, edifício e piso) são enviados para o servidor, que por sua vez verifica a existência da planta do piso que corresponda aos dados enviados, e devolve o conteúdo da mesma caso ela exista. Depois da receção da planta, a aplicação processa os dados e faz a representação gráfica da planta no ecrã.

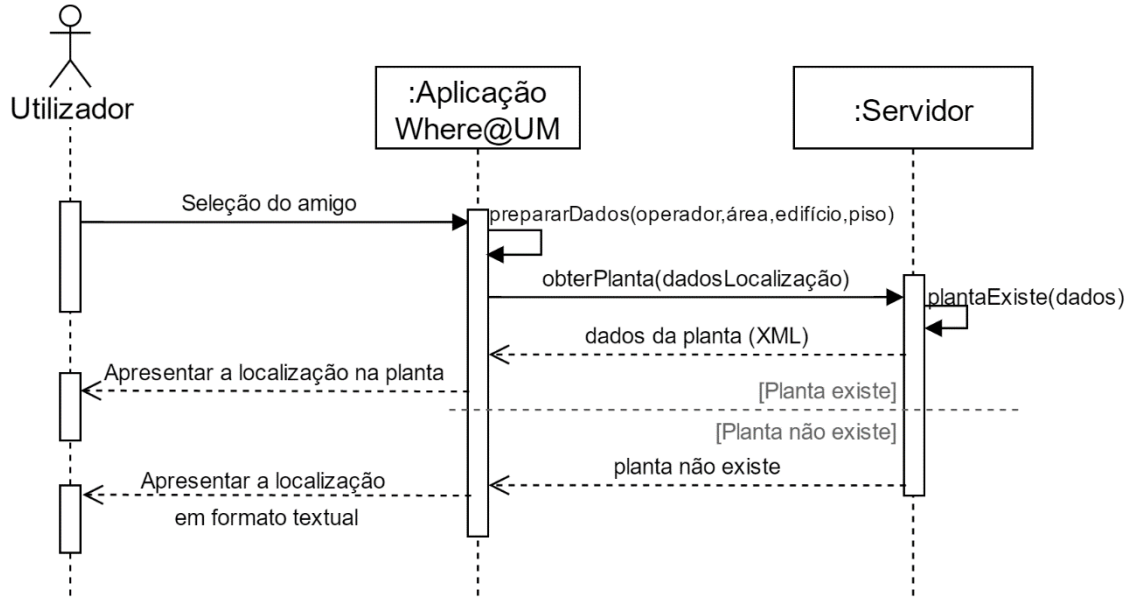


Figura 6.7: Obtenção da planta do piso na aplicação Where@UM

O fluxograma da Figura 6.8 apresenta o conjunto de operações que são realizadas no servidor quando é feito o pedido para obtenção da planta de um piso.

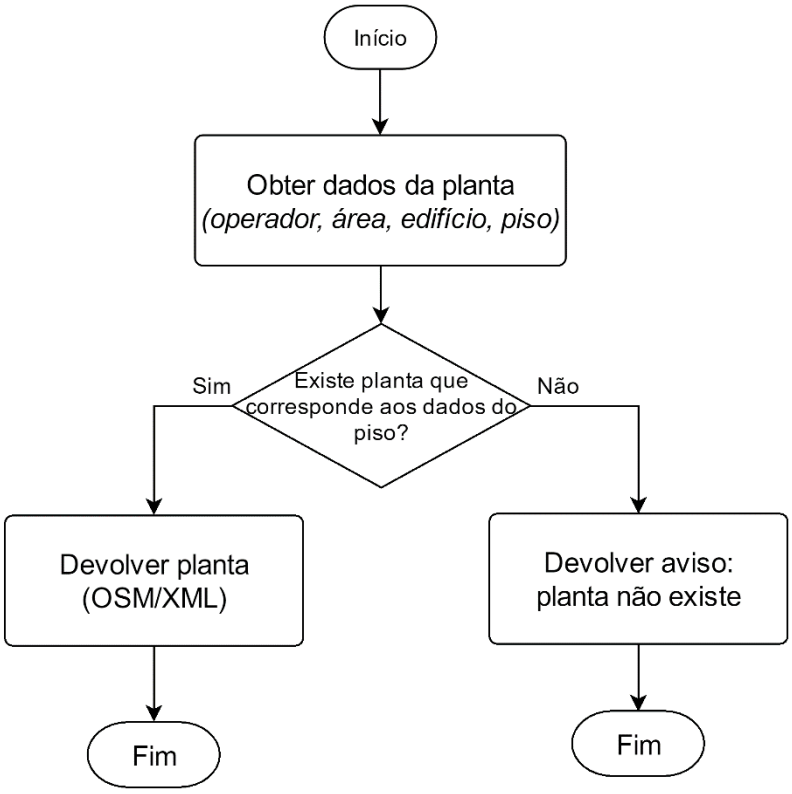


Figura 6.8: Algoritmo realizado no servidor quando é pedida uma planta

6.4.4. Novos ecrãs e funcionalidades

A implementação da componente de visualização aborda a criação de novos ecrãs na aplicação Where@UM. É possível mostrar a *overview* com recurso à API do Google Maps e as plantas de interiores são apresentadas com recurso ao módulo das plantas dos edifícios.

6.4.4.1. Overview

A *overview* de todos os amigos é apresentada no mapa Google Maps que permite a representação de vários marcadores (amigos) que podem ser personalizados. Esta funcionalidade é mostrada nas figuras 6.9 e 6.10.

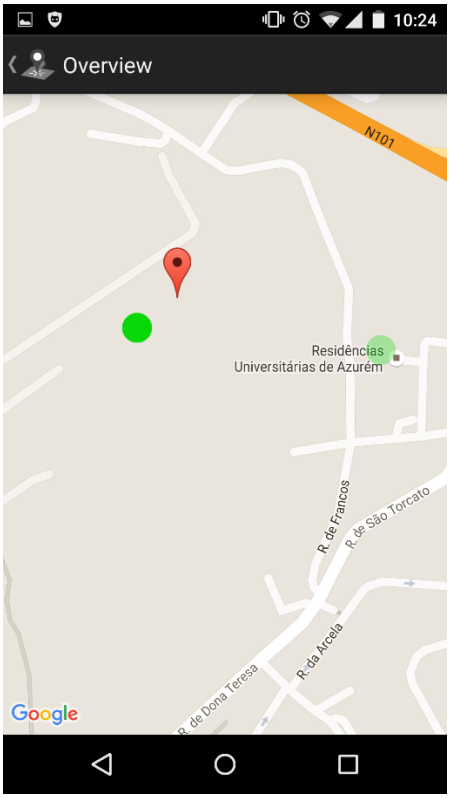


Figura 6.9: *Overview* - localização do utilizador e dos amigos

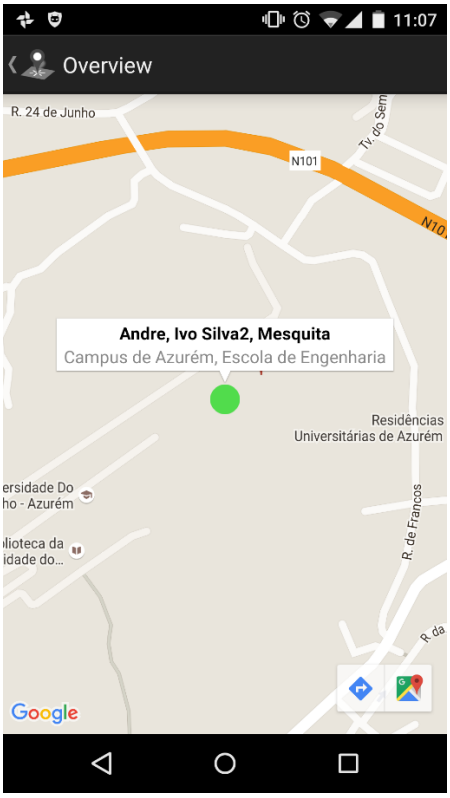


Figura 6.10: *Overview* - localização dos amigos "Andre", "Ivo Silva2" e "Mesquita"

Os pontos verdes são os amigos, dispersos pelo mapa. Quando existem vários utilizadores no mesmo edifício, a sua localização é representada num só ponto verde, onde é possível verificar quais são os utilizadores no edifício através de um toque no ponto verde, tal como é demonstrado na Figura 6.10. A transparência de cada ponto verde varia de acordo com a última atualização da localização do amigo, ou seja, quanto mais viva for a cor, mais recente é a informação

apresentada. Nos casos em que existem vários utilizadores representados no mesmo ponto, a cor apresentada está associada ao utilizador cuja informação de localização é mais recente.

O marcador vermelho representa a localização GPS do dispositivo, permitindo indicar a localização do utilizador dentro ou fora da Universidade do Minho. Como o objetivo da *overview* é uma vista geral de todos os utilizadores, a localização GPS do dispositivo será suficiente.

6.4.4.2. Consulta da localização (Amigo)

No ecrã "*Home*" os utilizadores podem consultar a localização dos seus amigos. Para isso, basta tocar no amigo que pretendem consultar na lista de amigos. Nas situações em que existe a planta do piso, a informação é apresentada graficamente com recurso à planta. Quando a planta não existe, a informação é apresentada em forma textual.

As figuras 6.11 e 6.12 mostram a localização de um amigo. A área, edifício e piso aparecem na parte superior do ecrã, uma vez que são informações que complementam a planta. A localização do amigo é mostrada como um ponto verde dentro do espaço interior, que neste caso é o LID3. O ponto verde funciona como um *switch* para apresentar o nome do amigo, por isso, basta clicar para apresentar o nome e fazer o mesmo para esconder o nome. Tal como na *overview,* a cor verde do ponto está associada à atualidade da informação apresentada.

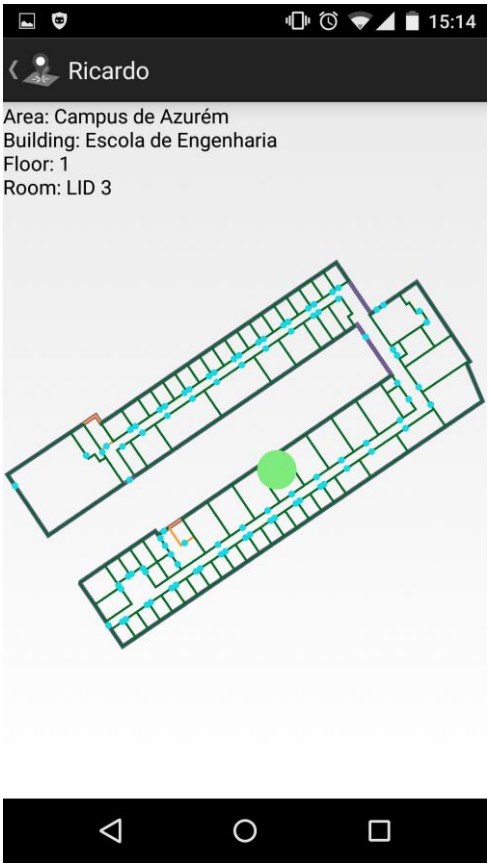


Figura 6.11: Localização de um amigo na planta do piso

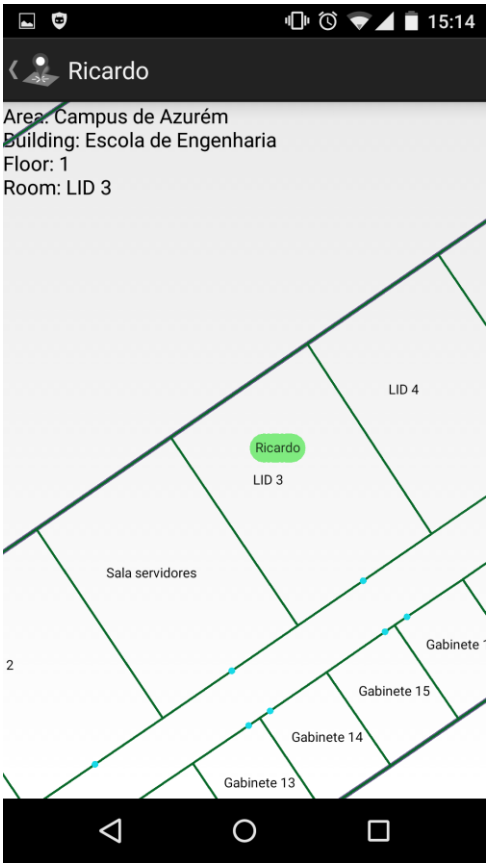


Figura 6.12: Apresentação do nome do amigo na sua localização indoor

6.4.4.3. Consulta da localização (Utilizador e Amigos)

O rodapé no ecrã "*Home*" informa o utilizador da sua posição atual. Quando se pretende alterar ou consultar a informação relativa à posição atual, basta clicar no rodapé para apresentar a informação. Tal como no caso anterior, a planta do piso só é apresentada se existir no servidor. Além da posição do utilizador, este ecrã apresenta todos os amigos que estejam presentes no mesmo piso.

Na Figura 6.13 a localização do utilizador e dos seus amigos é mostrada com recurso à planta da Escola de Engenharia do Campus de Azurém. A principal funcionalidade deste ecrã é a apresentação da localização dos amigos através dos pontos verdes.

A visualização com zoom, na Figura 6.14, tem mais detalhes com os nomes de cada espaço interior. O ponto vermelho representa o utilizador, e os pontos verdes (cuja cor pode variar tendo em consideração as localizações mais recentes) são os amigos do utilizador.

O utilizador pode editar a sua localização, caso se encontre noutro local, através do botão "*Edit my location*" onde pode selecionar um local dentro ou fora da Universidade do Minho.

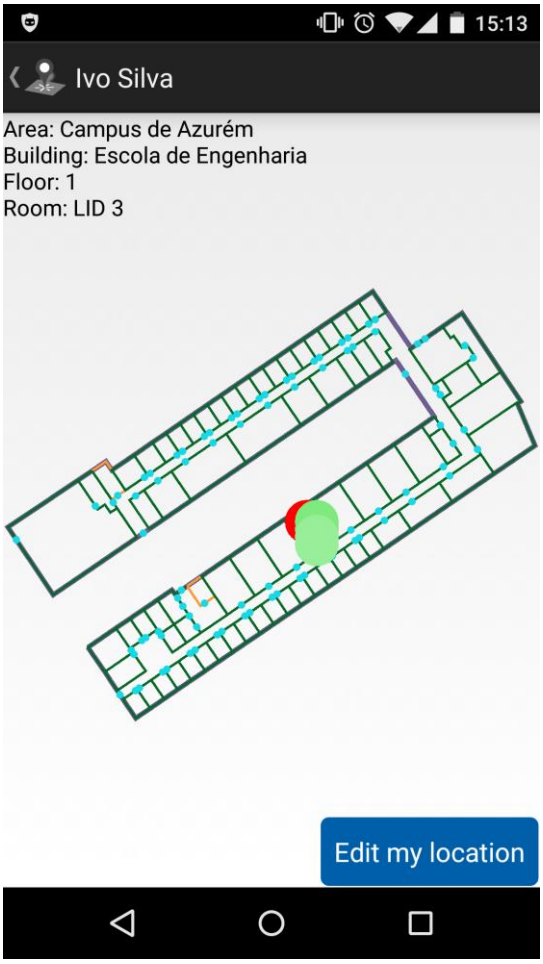


Figura 6.13: Localização do utilizador e dos seus amigos (no mesmo piso)

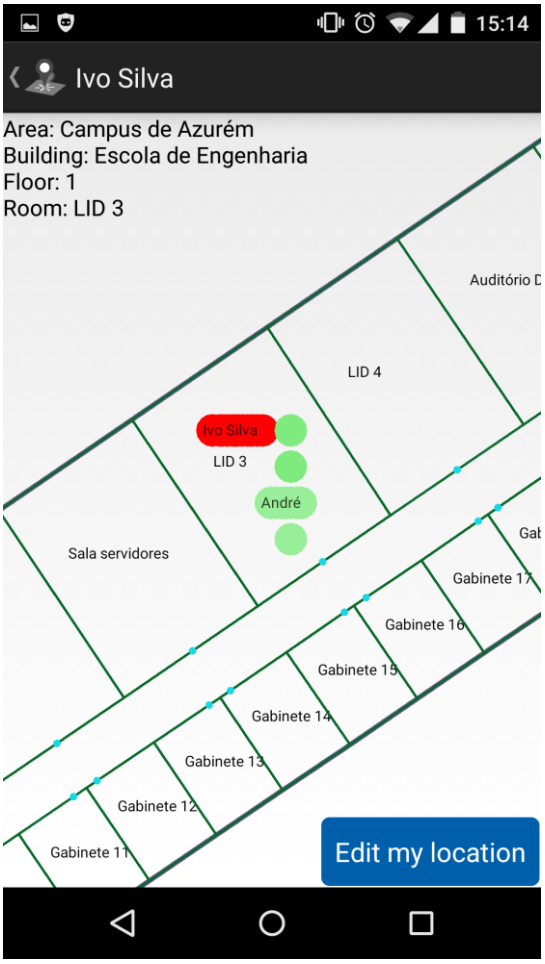


Figura 6.14: Consulta das *tags* dos nomes do utilizador e dos seus amigos

6.4.5. Alterações efetuadas no sistema Where@UM

Juntamente com a implementação da componente de visualização na aplicação Where@UM, foram feitas algumas alterações no sistema com o intuito de melhorar alguns aspetos.

Deste modo, são listadas algumas das alterações que foram feitas:

- Segurança:
  - Implementação de um *token* de sessão enviado em todos os pedidos feitos ao servidor;
  - Renovação do *token* de sessão automática;
  - *Logout* automático quando o *token* de sessão expira.
- Posicionamento:
  - Adicionados os campos "SSID" e "canal" às informações dos pontos de acesso que são obtidas na recolha de *fingerprints*;

- Correção de alguns *bugs* no código fonte.

6.4.5.1. Modelo de dados

As alterações efetuadas resultaram em algumas adições ao modelo de dados da base de dados Where@UM e da base de dados local SQLite da aplicação Android.

Os campos SSID e canal foram adicionados na base de dados local da aplicação Android. A Figura 6.15 representa o novo modelo de dados.

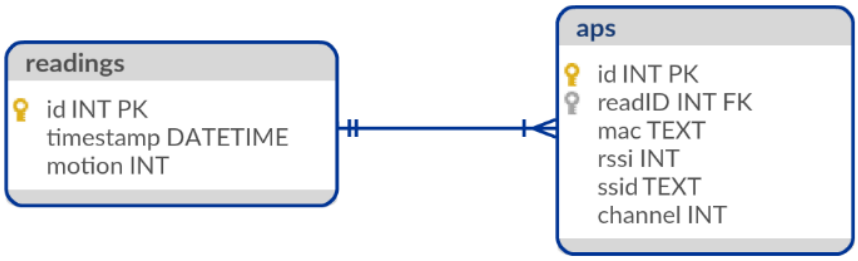


Figura 6.15: Modelo de dados da base de dados SQLite

Na implementação do *token* de sessão foi necessário adicionar uma nova tabela, chamada "*sessions*", que tem os dados das sessões ativas. Esta nova tabela é caracterizada pelos seguintes campos:

- idSession – identificador da sessão;
- idUser – identificador do utilizador associado à sessão;
- mac – endereço MAC da interface de rede do dispositivo;
- platform – plataforma associada à sessão, pode ter os seguintes valores: "android", "windowsphone" e "windowsdesktop";
- timestamp – data/hora do momento no qual a sessão foi criada;
- token – *string* de 50 caracteres, este valor tem que ser único para cada sessão;
- activeToken – pode tomar os valores "*Yes*" ou "*No*", indicando se o *token* está ativo ou não. Isto é necessário porque a sessão é criada antes do utilizador se autenticar. Para que o *token* seja ativado é necessário que o utilizador faça o login com sucesso.

Relativamente aos dados das *fingerprints* e pontos de acesso, foram adicionados os campos "SSID" e "canal" às tabelas **aps** e **apsFingerprintsHistory**.

Por fim, na tabela **users** foi adicionado o campo "password2" que contém o *hash* da password em MD5. Esta alteração foi necessária porque o objetivo da camada de segurança passa por enviar todos os dados de passwords sob a forma de um *hash*.

O novo modelo de dados está representado na Figura 6.16, onde estão as novas adições. A descrição de todas as tabelas pode ser consultada em [44].

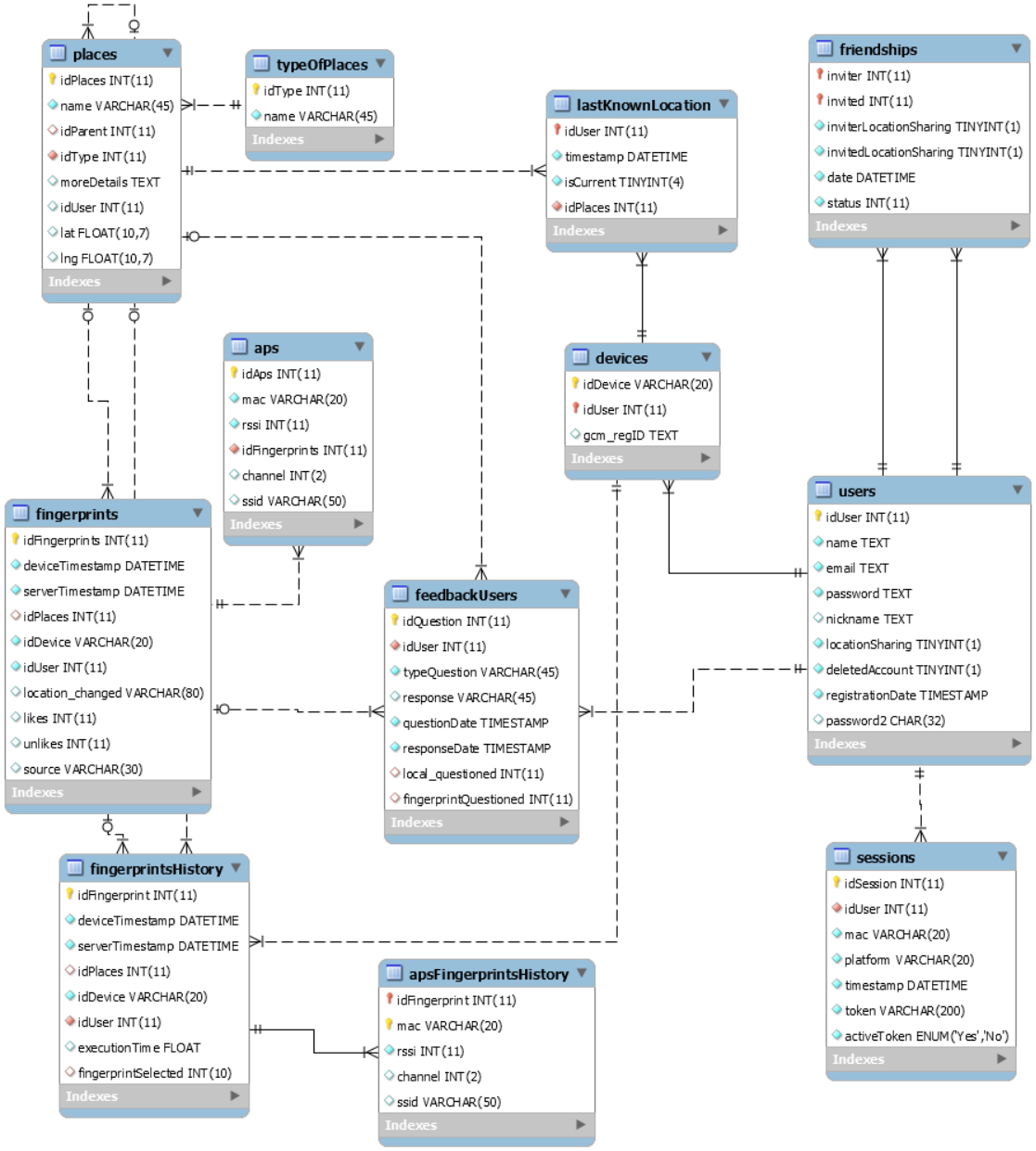


Figura 6.16: Modelo de dados do sistema Where@UM

6.4.5.2. Nova *fingerprint*

As alterações no modelo de dados permitiram uma reformulação às *fingerprints* que são recolhidas e enviadas para o servidor. Foi definida uma nova estrutura para o objeto JSON associado à *fingerprint* como demonstra a Figura 6.17. O objeto JSON contém vários parâmetros associados ao utilizador, local e data/hora onde a *fingerprint* foi recolhida. A *fingerprint* é um *array*

de vários objetos JSON sendo que cada objeto JSON representa as informações de um AP detetado.


```
{
        "idUser": "171",
        "idDevice": "64:BC:0C:7F:AC:7B",
        "timestamp": "2016-03-18 19:37:00",
        "motion": "",
        "place": "",
        "fingerprint": [{
                "mac": "2C:3E:CF:0B:78:BF",
                "rssi": "-37",
                "ssid": "eduroam",
                "channel": 1
        }, {
                "mac": "2C:3E:CF:0B:78:B4",
                "rssi": "-28",
                "ssid": "eduroam",
                "channel": 1
        }, {
                "mac": "2C:3E:CF:0B:78:BB",
                "rssi": "-94",
                "ssid": "",
                "channel": 1
        }]
}
```


Figura 6.17: Novo formato da *fingerprint* Where@UM

6.4.5.3. Novas medidas de segurança

Com o objetivo de melhorar a segurança na comunicação entre aplicação e servidor foi implementado o conceito de *token* de sessão. Tal como foi introduzido na secção anterior, o *token* de sessão é gerado quando a aplicação cliente faz um pedido para a criação da sessão (quando o utilizador faz o *login* ou cria uma nova conta). O servidor cria uma sessão e um *token* associado à mesma, que é enviado para a aplicação cliente mas nesta fase ainda não está ativo. Depois, é enviado para o servidor um novo pedido por parte da aplicação com os dados de autenticação do utilizador, que são validados pelo servidor caso estejam corretos. Em caso afirmativo, o *token* é ativado, e a partir deste momento, a aplicação cliente envia o *token* como *cookie* no *header* de cada pedido HTTP feito ao servidor.

O diagrama de sequência da Figura 6.18 apresenta as operações efetuadas entre a aplicação e o servidor depois do utilizador introduzir os dados de *login.*

O processo é simples, inicialmente é feito um pedido para obter o *token*. O servidor devolve o *token* de sessão associado ao utilizador e ao dispositivo. A aplicação recebe o *token* e faz um pedido de login no qual envia o *token* como cookie no pedido com o valor "token='valor do token'", os dados enviados neste pedido são o e-mail do utilizador, o valor *tokenpassword*, o endereço MAC e a plataforma. O valor *tokenpassword* é obtido da seguinte forma:

$$tokenpassword = md5(md5(password) + token)$$

Essencialmente, é calculado o md5, do valor md5 da password concatenado com o *token*. Isto permite que os dados sejam enviados para o servidor sem a necessidade de a *password* estar às claras.

O servidor quando efetua o login, verifica se os dados recebidos coincidem com os valores da base de dados, retornando uma resposta. Caso os dados estejam corretos, o servidor ativa o *token* e os dados do utilizador são enviados para a aplicação. Caso os dados de autenticação não estejam corretos, é devolvida uma mensagem de erro.

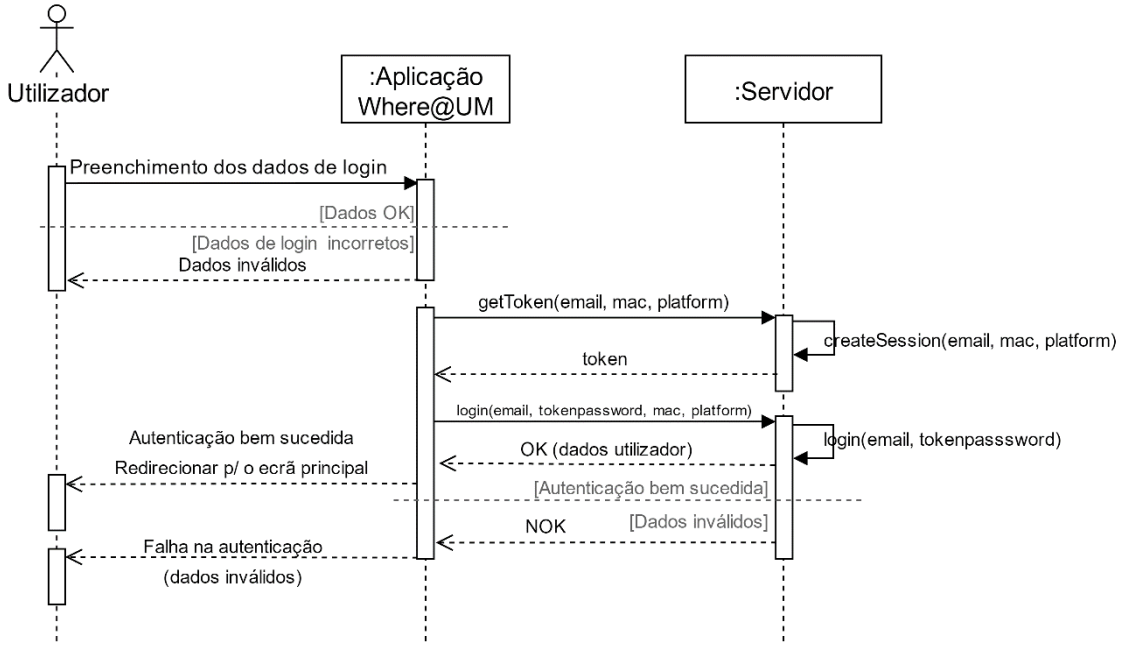


Figura 6.18: Processo de criação de uma sessão após login

O *token* expira 48h após a sua criação, condição que é verificada na aplicação Where@UM que faz a renovação do *token* automaticamente. É seguido um conjunto de verificações que permitem fazer um pedido ao servidor caso o *token* esteja quase a expirar.

O fluxograma da Figura 6.19 representa as operações realizadas na aplicação cliente para a renovação do *token* de sessão. O algoritmo é executado em *background* quando são recolhidas *fingerprints* no dispositivo. É feita a diferença em horas entre a data atual e a data de criação do *token*. Caso a diferença seja entre 36 e 48 horas, ou seja, quando faltarem cerca de 12h para o *token* expirar, é feito um pedido de renovação do *token.* Caso o *token* tenha ultrapassado as 48h, este expirou e por isso é efetuado o *logout*.

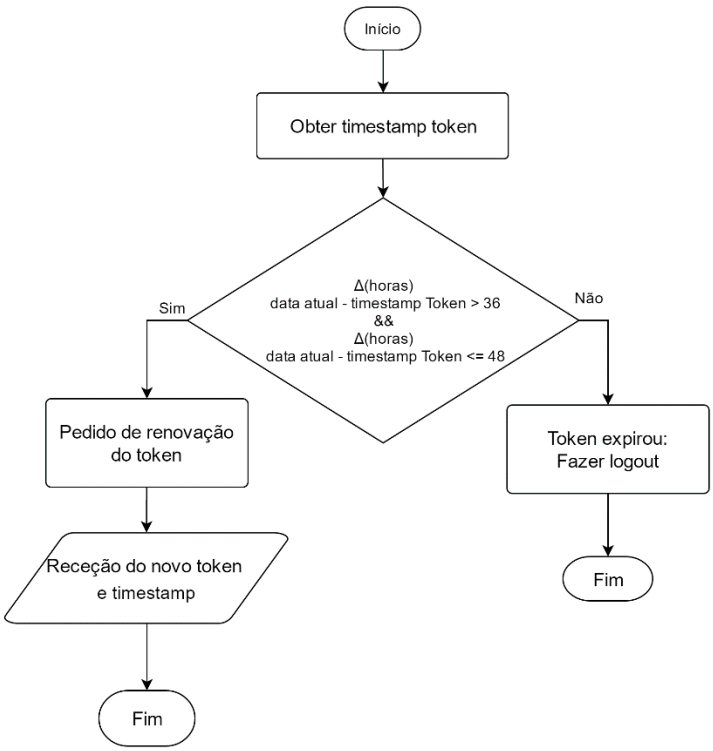


Figura 6.19: Renovação automática do *token* de sessão

O *token* de sessão permite evitar alguns ataques (por exemplo ataques de *brute force*) e aumenta a segurança uma vez que nesta nova versão dos *web services* as *passwords* não são enviadas às claras e existe um *token* associado a cada sessão.

No Anexo E – Web Services Where@UM é possível consultar a nova versão da API dos *web services* utilizada na comunicação entre a aplicação cliente e o servidor.

6.5. Testes efetuados na aplicação Where@UM

Depois de várias iterações e correções de *bugs* durante a implementação, foram testadas as novas funcionalidades para determinar se a aplicação está pronta para passar da versão de

desenvolvimento para a versão de produção. Os testes foram realizados em vários dispositivos e os resultados obtidos encontram-se descritos na Tabela 6.1.

Tabela 6.1: Testes efetuados na aplicação Where@UM

Teste	Resultado
Overview com a apresentação da localização do utilizador e dos seus amigos num mapa (Google Maps).	Passou
Consultar a localização amigo: apresentação da planta do piso, com o ponto verde no respetivo espaço interior onde o amigo se encontra.	Passou
Consultar a localização do utilizador e amigos: apresentação da planta do piso com o ponto vermelho (utilizador), e os seus amigos (pontos verdes) que se encontram no mesmo piso.	Passou
Apresentação de uma caixa de diálogo com um aviso para o utilizador se ligar à internet quando pretende fazer uma operação que depende da conexão à internet.	Passou
Armazenamento local das *fingerprints* no novo formato quando não existe ligação à internet.	Passou
Login com recurso ao *token* de sessão.	Passou
Renovação automática do *token* de sessão.	Passou
Logout automático quando o *token* de sessão expira.	Passou

É possível concluir que a nova versão da aplicação Where@UM está pronta a ser lançada no Google Play como uma nova atualização para os utilizadores uma vez que não surgiram problemas nos testes efetuados.

CAPÍTULO 7 – CONCLUSÕES

No sétimo capítulo são apresentadas as conclusões do trabalho desenvolvido através de uma análise aos resultados obtidos.

Por fim, os trabalhos futuros são discutidos com o objetivo de definir alguns aspetos que podem melhorar tanto o sistema de calibração como o sistema Where@UM.

7.1. Conclusões

O objetivo principal desta dissertação era a criação de um sistema para a construção de mapas de rádio para sistemas de posicionamento Wi-Fi, que permitisse a construção de mapas de rádio de uma forma simples e eficaz. O processo de calibração manual é uma tarefa morosa que exige tempo e paciência ao utilizador. Os sistemas de posicionamento que possuem funcionalidades de calibração manual são muitas vezes pouco precisos nas recolhas de dados e requerem bastante tempo ao utilizador.

Partindo do estudo do estado da arte foi considerado o desenvolvimento de uma aplicação móvel depois do reconhecimento dos sistemas atuais que realizam a calibração através da recolha de *fingerprints* Wi-Fi. A aplicação móvel foi desenvolvida com o intuito de suportar qualquer instituição e qualquer tipo de edifício, para que possa ser utilizada em diversos contextos no âmbito da calibração. Antes da implementação foi necessário criar o desenho da solução passando pela análise do sistema, da sua arquitetura e das tecnologias de cada componente.

Um dos desafios desta dissertação prende-se no desenho e criação das plantas dos edifícios. O processo de criação de uma planta foi alvo de um estudo pormenorizado nesta dissertação. Depois da análise às soluções possíveis, o formato OSM revelou-se a escolha mais acertada. Através da ferramenta JOSM é possível a criar as plantas, tratando-se de uma tarefa que deverá ser realizada por um utilizador avançado e requer precisão e concentração. A solução para as plantas dos edifícios incluí um serviço para disponibilização dos dados das plantas através de *web services.* O servidor fornece as plantas do piso sempre que pedido. Também apresenta um módulo que recebe uma planta, faz o seu *upload* e introduz as informações relativas à mesma na base de dados. A comunicação com o servidor é assegurada através de *web services* REST que permitem a comunicação simples com recurso ao formato de dados JSON. Este serviço foi o primeiro passo na fase de desenvolvimento uma vez que vem dar suporte à aplicação de calibração e à solução de visualização implementada na aplicação Where@UM.

A aplicação de calibração foi desenvolvida para a plataforma Android com recurso ao IDE Android Studio. A aplicação depende da comunicação com o servidor (suportada por *web services* REST), mas permite o funcionamento *offline* depois do utilizador obter a planta do piso, dando flexibilidade na recolha de amostras em qualquer local onde não existir conexão à internet. Existe um módulo de suporte à aplicação no servidor que recebe e responde aos pedidos da aplicação móvel. O módulo das plantas dos edifícios do servidor é fulcral para o funcionamento de todo o sistema, porque através da planta do piso é possível associar uma *fingerprint* a um ponto da planta, permitindo assim a construção do mapa de rádio.

A construção do mapa de rádio de um edifício é facilmente realizada através da aplicação de calibração com acesso à planta de cada piso. O sistema vai dando *feedback* relativamente ao estado do mapa de rádio à medida que as amostras são enviadas para o servidor, porque é possível observar os locais que não foram calibrados e os locais onde foram recolhidas poucas amostras. Além da construção dos mapas de rádio o sistema desenvolvido permite fazer a gestão do mesmo, sabendo os locais que não estão devidamente calibrados e os locais que estão desatualizados, tendo em consideração que a infraestrutura Wi-Fi sobre alterações e que as flutuações dos sinais são constantes. Também pode ser feita uma análise sobre a cobertura Wi-Fi dentro do edifício através de funcionalidades de visualização que mostram as zonas com densidade elevada de APs, as zonas com sobreposição de canais e os locais de acordo com os níveis de sinal mais elevados.

O módulo de visualização passou pela integração das plantas dos edifícios na aplicação Where@UM. Os utilizadores agora usufruem de uma funcionalidade que os permite interagir com os mapas e consultar a sua localização dentro de edifícios, como é o caso da Escola de Engenharia (edifício suportado, juntamente com o Bloco G2 do Complexo Residencial de Azurém). A apresentação visual da posição em ambientes interiores é fundamental num sistema deste género uma vez que é informação facilmente interpretada pelo utilizador. A adição da *overview*, implementada com recurso à API do Google Maps também enriquece bastante a experiência do utilizador uma vez fornece informação visual e interativa relativamente às posições dos utilizadores dando suporte a qualquer local do mundo.

No âmbito dos sistemas de posicionamento, este trabalho permitiu o desenvolvimento de *software* que interveio nas duas fases de um sistema de posicionamento. A aplicação de calibração permite realizar a fase de treino e a componente de visualização oferece uma visualização gráfica na fase *real-time* do sistema Where@UM.

7.2. Trabalhos Futuros

O trabalho desenvolvido ao longo desta dissertação apresenta alguns aspetos que podem ser alvo de melhorias no futuro.

Partindo da aplicação de calibração, existe a possibilidade de dar mais funcionalidades para gestão e controlo do mapa de rádio. Seria interessante filtrar as *fingerprints* de acordo com um determinado intervalo temporal, dando uma data de início e uma data final. Para efeitos de atualização do mapa de rádio, seria interessante introduzir um sistema de alertas que informa o utilizador que existem áreas desatualizadas que precisam de ser recalibradas. A consulta de uma *fingerprint* específica utilizando a aplicação móvel é uma funcionalidade interessante em termos de construção do mapa de rádio, que poderá ser introduzida numa versão futura da aplicação. Apagar *fingerprints* antigas para manter o mapa de rádio atualizado seria uma funcionalidade pertinente para espaços que são atualizados frequentemente. A comunicação com o servidor não utiliza uma ligação segura e confidencial, que poderá ser assegurada através do protocolo SSL. Este protocolo oferece confidencialidade, integridade aos dados transmitidos, que apesar de não serem críticos deverão ser transmitidos de uma forma segura. Este será um dos aspetos mais importantes a nível de segurança para implementações futuras.

Tanto para permitir a criação de mapas de rádio de edifícios da Universidade do Minho como para enriquecer a experiência Where@UM, a adição de mais plantas dos edifícios da universidade seria bastante benéfica. Atualmente apenas são suportados dois edifícios do campus de Azurém, mas no futuro poderão ser criadas plantas de outros edifícios. O sistema de calibração e o sistema Where@UM fazem uso das plantas de interiores, deste modo, quantas mais existirem mais vantajoso será para ambos os sistemas.

Atualmente o sistema de calibração não está a ser utilizado em conjunto com um sistema de posicionamento. A sua integração em sistemas como o sistema Where@UM auxiliará na estimação da localização com dados mais ricos e com maior detalhe, facilitando principalmente na fase *online* onde se estima a localização do utilizador. Tendo as plantas de novos edifícios permitirá a criação dos mapas de rádio através da aplicação de calibração e consequentemente a integração com outros sistemas permitirá a utilização dos dados de calibração na fase *online.* Considerando que os dados de calibração manual foram devidamente recolhidos, a fase *online* poderá utilizar dados com mais qualidade para estimar a posição do utilizador.

No âmbito da aplicação Where@UM, a adição de plantas de outros edifícios da Universidade do Minho oferece uma experiência mais rica e interativa para os utilizadores. A

utilização das plantas poderá introduzir novas funcionalidades, como por exemplo a indicação de um espaço *indoor* a que o utilizador se pretenda deslocar, permitindo assim auxiliar na navegação pelo edifício. A aplicação também pode mostrar as plantas dos edifícios da universidade para consulta dos espaços interiores de cada edifício. A integração com redes sociais, por exemplo o *Facebook*, seria interessante porque facilitaria os utilizadores na autenticação e a encontrarem os seus amigos.

BIBLIOGRAFIA

[1] R. Monteiro and A. Moreira, "CRC'2010 - 10ª Conferência sobre Redes de Computadores," Braga, Portugal: Universidade do Minho, 2010, pp. 167–172.

[2] A. Lamarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit, "Place Lab: Device Positioning Using Radio Beacons in the Wild," in *Pervasive Computing,Third International Conference*, Munique, Alemanha, 2005, pp. 116–133.

[3] P. Bahl and V. N. Padmanabhan, "RADAR: An In-building RF-based User Location and Tracking System," in *Proc. IEEE INFOCOM 2000. The 19th annual conference on Computer Communications*, 2000, vol. 2, pp. 775–784.

[4] "Real-Time Location Systems." [Online]. Available: http://www.centrak.com/. [Accessed: 23-Sep-2015].

[5] A. Lamarca, J. Hightower, I. Smith, and S. Consolvo, "Self-Mapping in 802 . 11 Location Systems," pp. 87–104, 2005.

[6] J. Krumm, G. Cermak, and E. Horvits, "RightSPOT: A Novel Sense of Location for a Smart Personal Object," in *UbiComp 2003*, 2003, pp. 36–43.

[7] "IndoorAtlas." [Online]. Available: https://www.indooratlas.com/. [Accessed: 23-Sep-2015].

[8] "Accuware." [Online]. Available: http://www.accuware.com/. [Accessed: 23-Sep-2015].

[9] "Guardly." [Online]. Available: https://www.guardly.com. [Accessed: 23-Sep-2015].

[10] "Wifarer." [Online]. Available: http://www.wifarer.com/. [Accessed: 23-Sep-2015].

[11] "Mapas de interiores da Google." [Online]. Available: http://www.google.com/maps/about/partners/indoormaps/. [Accessed: 23-Sep-2015].

[12] S. Feldmann, K. Kyamakya, A. Zapater, and Z. Lue, "An indoor Bluetooth-based positioning system : concept , Implementation and experimental evaluation," *Int. Conf. Wirel. Networks*, pp. 109–113, 2003.

[13] S. S. Saad and Z. S. Nakad, "A standalone RFID indoor positioning system using passive tags," *IEEE Trans. Ind. Electron.*, vol. 58, no. 5, pp. 1961–1970, 2011.

[14] L. M. Ni and A. P. Patil, "LANDMARC: indoor location sensing using active RFID," *Proc. First IEEE Int. Conf. Pervasive Comput. Commun.*, pp. 407–415, 2003.

[15] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of Wireless Indoor Positioning Techniques and Systems," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 6, pp. 1067–1080, 2007.

[16] J. F. M. Carvalho, "Localização de Dispositivos Móveis em Redes Wi-Fi," Universidade de Trás-os-Montes e Alto Minho, 2007.

[17] C. Savarese, J. M. Rabaey, and J. Beutel, "LOCATIONING IN DISTRIBUTED AD-HOC WIRELESS SENSOR NETWORKS," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, 2001, pp. 2037–2040.

[18] R. J. Fontana, "ULTRA WIDEBAND PRECISION GEOLOCATION SYSTEM," 2000.

[19] R. Mautz, "Overview of current indoor positioning systems," *Geod. Cartogr.*, vol. 35, no. 1, pp. 18–22, 2009.

[20] H. Koyuncu and S. H. Yang, "A Survey of Indoor Positioning and Object Locating Systems," *Int. J. Comput. Sci. Netw. Secur. (IJCSNS '10)*, vol. 10, no. 5, pp. 121–128, 2010.

[21] a K. M. M. Hossain, H. N. Van, Y. Jin, and W.-S. Soh, "Indoor Localization Using Multiple Wireless Technologies," *2007 IEEE Internatonal Conf. Mob. Adhoc Sens. Syst.*, pp. 1–8, 2007.

[22] D. A. da S. Matos, A. Moreira, and F. Meneses, "Wi-Fi fingerprint similarity in collaborative radio maps for indoor positioning," pp. 184–194.

[23] P. Bolliger, "Redpin - Adaptive, Zero-Configuration Indoor Localization through User Collaboration," in *Melt'08*, 2008, pp. 55–60.

[24] M. Paciga and H. Lutfiyya, "Herecast : An Open Infrastructure for Location- Based Services Using WiFi," *Access*, vol. 9, 2004.

[25] B. N. Schilit, A. LaMarca, G. Borriello, W. G. Griswold, D. McDonald, E. Lazowska, A. Balachandran, J. Hong, and V. Iverson, "Challenge: Ubiquitous location-aware computing and the Place Lab initiative," in *Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, 2003, pp. 29–35.

[26] "Redpin Android Client." [Online]. Available: http://redpin.org/android.html. [Accessed: 12-Oct-2015].

[27] "Redpin iPhone Client." [Online]. Available: http://redpin.org/iphone.html. [Accessed: 12-Oct-2015].

[28] T. King, S. Kopf, T. Haenselmann, and C. Lubberger, "COMPASS : A Probabilistic Indoor Positioning System Based on 802 . 11 and Digital Compasses," in *WiNTECH '06 Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization*, 2006, no. September, pp. 34–40.

[29] "WiFiSLAM," 2015. [Online]. Available: https://angel.co/wifislam. [Accessed: 07-Oct-2015].

[30] "IndoorAtlas MapCreator," *aplicação Android*. [Online]. Available: https://play.google.com/store/apps/details?id=com.indooratlas.mapcreator.main. [Accessed: 08-Oct-2015].

[31] "IndoorAtlas MapCreator," *aplicação iOS*. [Online]. Available: https://itunes.apple.com/pt/app/indooratlas-mapcreator/id720005234?mt=8. [Accessed: 08-Oct-2015].

[32] "Q-TRACK." [Online]. Available: http://q-track.com/. [Accessed: 07-Oct-2015].

[33] "Ekahau." [Online]. Available: http://www.ekahau.com/. [Accessed: 07-Oct-2015].

[34] Y. Gu, A. Lo, and I. Niemegeers, "A survey of indoor positioning systems for wireless personal networks," *IEEE Commun. Surv. Tutorials*, vol. 11, no. 1, pp. 13–32, 2009.

[35] "Demonstração do Sistema de Posicionamento Indoor WiFiSlam," *vídeo youtube*. [Online].

Available: https://www.youtube.com/watch?v=gtjBMZTW5Fg. [Accessed: 09-Oct-2015].

[36] "Navizon." [Online]. Available: https://www.navizon.com/. [Accessed: 07-Oct-2015].

[37] "Navizon Indoors," *aplicação Android*. [Online]. Available: https://play.google.com/store/apps/details?id=com.navizon.NavizonIndoors. [Accessed: 08-Oct-2015].

[38] "Navizon Indoors," *aplicação iOS*. [Online]. Available: https://itunes.apple.com/pt/app/navizon-indoors/id881016365?mt=8. [Accessed: 08-Oct-2015].

[39] "infsoft." [Online]. Available: http://www.infsoft.com/. [Accessed: 08-Oct-2015].

[40] "Calibration," *aplicação Android*. [Online]. Available: https://play.google.com/store/apps/details?id=com.infsoft.android.wificalib. [Accessed: 08-Oct-2015].

[41] "Demonstração do Sistema de Posicionamento Indoor Guardly," *vídeo youtube*. [Online]. Available: https://www.youtube.com/watch?v=w4IWTb8G8IY&feature=youtu.be. [Accessed: 08-Oct-2015].

[42] "Where@UM," *aplicação Android*. [Online]. Available: https://play.google.com/store/apps/details?id=com.whereum. [Accessed: 08-Oct-2015].

[43] D. A. da S. Matos, "where @ UM - Aplicação móvel de posicionamento," 2014.

[44] V. P. B. Barroso, "Posicionamento colaborativo em redes Wi-Fi - Where @ UM2," Universidade do Minho, 2015.

[45] "Wifarer," *aplicação Android*. [Online]. Available: https://play.google.com/store/apps/details?id=com.wifarer.android. [Accessed: 08-Oct-2015].

[46] "Wifarer," *aplicação iOS*. [Online]. Available: https://itunes.apple.com/pt/app/wifarer/id445140849?mt=8. [Accessed: 08-Oct-2015].

[47] "Maps Inside," *aplicação Android*. [Online]. Available: https://play.google.com/store/apps/details?id=de.mapsinside.android. [Accessed: 08-Oct-2015].

[48] "Indoor GPS," *aplicação Android*. [Online]. Available: https://play.google.com/store/apps/details?id=com.ladiesman217.indoorgps. [Accessed: 08-Oct-2015].

[49] "WiFi Indoor Localization," *aplicação Android*. [Online]. Available: https://play.google.com/store/apps/details?id=com.hfalan.wifilocalization. [Accessed: 08-Oct-2015].

[50] "Mapas de interiores da Apple," 2015. [Online]. Available: https://mapsconnect.apple.com/indoor/signup. [Accessed: 06-Oct-2015].

[51] "Google Maps." [Online]. Available: https://www.google.pt/maps. [Accessed: 12-Oct-2015].

[52] "Here Maps," *aplicação Android*. [Online]. Available: https://play.google.com/store/apps/details?id=com.here.app.maps. [Accessed: 16-Oct-

2015].

[53] "Mapas de Interiores Heidelberg Mobil." [Online]. Available: https://www.heidelberg-mobil.com/loesungen/indoor-karten.html. [Accessed: 12-Oct-2015].

[54] "Exemplo de Mapas de Interiores OpenStreetMap." [Online]. Available: http://clement-lagrange.github.io/osmtools-indoor/#lat=50.60952&lon=3.13773&z=20&id_building=3440826&id_level=3440825. [Accessed: 12-Oct-2015].

[55] "JOSM - extensible editor for OSM." [Online]. Available: https://josm.openstreetmap.de/. [Accessed: 23-Nov-2015].

[56] "OpenStreetMap." [Online]. Available: https://www.openstreetmap.org/. [Accessed: 23-Nov-2015].

[57] "Indoor Mapping." [Online]. Available: http://wiki.openstreetmap.org/wiki/Indoor_Mapping. [Accessed: 23-Nov-2015].

[58] "IndoorOSM - Tagging Schema." [Online]. Available: http://wiki.openstreetmap.org/wiki/IndoorOSM. [Accessed: 23-Nov-2015].

[59] "Simple Indoor Tagging - Tagging Schema." [Online]. Available: http://wiki.openstreetmap.org/wiki/Simple_Indoor_Tagging. [Accessed: 23-Nov-2015].

[60] "Project Tango do Google." [Online]. Available: https://www.google.com/atap/project-tango/. [Accessed: 13-Oct-2015].

[61] P. Robertson, M. Angermann, and B. Krach, "Simultaneous localization and mapping for pedestrians using only foot-mounted inertial sensors," in *Proceedings of the 11th international conference on Ubiquitous computing Ubicomp 09*, 2009, pp. 93–96.

[62] "Google Maps Android API." [Online]. Available: https://developers.google.com/maps/documentation/android-api/. [Accessed: 22-Oct-2015].

[63] "Google Places API for Android." [Online]. Available: https://developers.google.com/places/android-api/. [Accessed: 22-Oct-2015].

[64] G. Mulligan and D. Gračanin, "A COMPARISON OF SOAP AND REST IMPLEMENTATIONS OF A SERVICE BASED INTERACTION INDEPENDENCE MIDDLEWARE FRAMEWORK," in *Proceedings of the 2009 Winter Simulation Conference*, 2013, vol. 53, pp. 1423–1431.

[65] K. Wagh and R. Thool, "A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host," *J. Inf. Eng. Appl.*, vol. 2, no. 5, pp. 12–16, 2012.

[66] C. Pautasso, "SOAP vs. REST Bringing the Web back into Web Services." pp. 1–19, 2007.

[67] C. G. Pendão, "Recolha de Dados de Movimento em Dispositivos Móveis Pessoais," Universidade do Minho, 2012.

[68] "SOAP Version 1.2." [Online]. Available: http://www.w3.org/TR/soap12/. [Accessed: 10-Dec-2015].

[69] "What Are RESTful Web Services?" [Online]. Available: https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html. [Accessed: 10-Dec-2015].

[70] "JSON." [Online]. Available: http://json.org/. [Accessed: 10-Dec-2015].

[71] "Why REST + JSON is preferred over SOAP for mobile web services." [Online]. Available: http://www.bamboorocketapps.com/rest-json-vs-soap-xml/. [Accessed: 30-Oct-2015].

[72] "RFC2616: 9. Method Definitions." [Online]. Available: http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html. [Accessed: 11-Dec-2015].

[73] "Android - Creating a View Class." [Online]. Available: https://developer.android.com/training/custom-views/create-view.html. [Accessed: 20-Jun-2016].

[74] "Creating Custom Views." [Online]. Available: https://developer.android.com/training/custom-views/index.html. [Accessed: 20-Jun-2016].

[75] "Smartphone OS Market Share, 2015 Q2." [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Accessed: 27-Oct-2015].

[76] "Global tablet shipments from 2010 to 2015, by operating system (in million units)." [Online]. Available: http://www.statista.com/statistics/273268/worldwide-tablet-sales-by-operating-system-since-2nd-quarter-2010/. [Accessed: 27-Oct-2015].

[77] "Android Studio." [Online]. Available: http://developer.android.com/sdk/index.html. [Accessed: 08-Feb-2016].

[78] "Eclipse." [Online]. Available: https://eclipse.org/. [Accessed: 08-Feb-2015].

[79] K. Marsal, "Apple removes Wi-Fi scanners, 'minimum functionality' iPhone apps." [Online]. Available: http://appleinsider.com/articles/10/03/04/apple_removes_wi_fi_scanners_minimum_functionality_iphone_apps. [Accessed: 04-Feb-2016].

[80] J. Newman, "Apple Abolishes Wi-Fi Scanners From App Store." [Online]. Available: http://www.pcworld.com/article/190789/wifi_scanners_banned.html. [Accessed: 04-Feb-2016].

[81] "Choosing a Membership." [Online]. Available: https://developer.apple.com/support/compare-memberships/. [Accessed: 04-Feb-2016].

[82] "Android Platform Versions." [Online]. Available: http://developer.android.com/about/dashboards/index.html#Platform. [Accessed: 22-Feb-2016].

[83] "WifiManager." [Online]. Available: http://developer.android.com/reference/android/net/wifi/WifiManager.html. [Accessed: 19-Feb-2016].

[84] "SQLiteOpenHelper." [Online]. Available: http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html. [Accessed: 19-Feb-2016].

[85] "Android Interfaces and Architecture." [Online]. Available: http://source.android.com/devices/index.html. [Accessed: 22-Feb-2016].

[86] "Android - Architecture." [Online]. Available:

http://www.tutorialspoint.com/android/android_architecture.htm. [Accessed: 22-Feb-2016].

[87] "Activities." [Online]. Available: http://developer.android.com/guide/components/activities.html. [Accessed: 22-Feb-2016].

[88] "Services." [Online]. Available: http://developer.android.com/guide/components/services.html. [Accessed: 22-Feb-2016].

[89] "Content Providers." [Online]. Available: http://developer.android.com/guide/topics/providers/content-providers.html. [Accessed: 22-Feb-2016].

[90] "App Fundamentals." [Online]. Available: http://developer.android.com/guide/components/fundamentals.html. [Accessed: 22-Feb-2016].

[91] "Android - Application Components." [Online]. Available: http://www.tutorialspoint.com/android/android_application_components.htm. [Accessed: 22-Feb-2016].

ANEXO A – COMPARAÇÃO ENTRE APLICAÇÕES ANALISADAS

No Anexo A está representada uma tabela com informações específicas acerca das aplicações móveis estudadas relativamente à plataforma, funcionalidades, técnicas de localização, testes e considerações sobre a aplicação.

Tabela A.1: Comparação entre as aplicações móveis analisadas

Aplicação	Funcionalidades	Técnica(s) de localização	Testado	Observações
IndoorAtlas MapCreator Android iOS	• Criação de mapas de rádio a partir da planta do edifício. • Introdução da planta do piso e alinhamento com o edifício no Google Maps. • Calibração através do movimento do utilizador. • Posicionamento com recurso a vários sensores do dispositivo. • Capacidade para adicionar plantas de cada piso de vários edifícios. • Apresenta as plantas de edifícios introduzidos por outros utilizadores que podem ser usados para a localização nesses mesmos edifícios.	GPS Wi-Fi Sensores (magnetómetro, acelerómetro, Giroscópio)	Sim	• Sistema eficaz, capaz de detetar a posição e movimentação dentro de um espaço pequeno. • A fase de calibração é simples.
Navizon Indoors Android iOS	• Selecionar o edifício a partir do Google Maps. • Modo de treino permite que uma rota de treino seja definida no edifício e depois faz-se a fase de treino dessa rota. • Modo de navegação, a aplicação apresenta a localização do utilizador de acordo com o seu movimento em tempo	GPS Bluetooth GSM Wi-Fi Sensores do dispositivo	Sim	• Não permite que o utilizador introduza as plantas do edifício (funcionalidade apenas disponível aos clientes do serviço). • Muitas funcionalidades restritas a

	real, na área de treino (calibrada previamente).			subscritores do serviço. • Calibração simples e fácil através do movimento do utilizador de um ponto de origem até um ponto de destino.
Calibration Android iOS	• Calibração de locais com a tecnologia de navegação *indoor* da infsoft.	Wi-Fi GSM 3G/4G Sensores do dispositivo (campos magnéticos, bússola, pressão do ar, barómetro, acelerómetro e giroscópio)	Não	• A aplicação é restrita a utilizadores do serviço disponibilizado pela infsoft. • A introdução das plantas do edifício é feita a partir de uma aplicação específica, também desenvolvida pela infsoft.
Where@UM Android	• Partilha da localização entre os utilizadores dentro e fora dos campi da Universidade do Minho. • Adicionar amigos. • Criação dos mapas de rádio de forma colaborativa quando o sistema não reconhece a localização. • Editar ou adicionar uma nova localização dentro ou fora dos campi da Universidade do Minho.	Wi-Fi	Sim	• Utiliza a API do Foursquare para a sugestão de pontos de interesse fora dos campi da Universidade. • Não necessita de calibração. • Depende da informação que os utilizadores introduzem para que o sistema desenvolva e mantenha os mapas de rádio atualizados.

Wifarer Android iOS	• Localização em ambientes interiores com precisão. • Navegação de um ponto de origem para um ponto de destino. • Mapas precisos atualizados. • Disponibiliza uma série de pontos de interesse aos quais o utilizador pode aceder quando se localiza num deles.	Wi-Fi GPS Bluetooth	Não	• Limitado aos locais que disponibiliza.
Maps Inside Android	• Localização *indoor*. • Utilização de mapas inseridos pelo utilizador • Acompanhamento do movimento do utilizador em tempo real • Não é necessária a conexão a uma rede Wi-Fi	Wi-Fi	Sim	• Aplicação com funcionalidades limitadas a utilizadores com subscrição ao serviço.
Indoor GPS Android	• O utilizador pode introduzir a planta do piso através de uma imagem ou uma fotografia tirada no momento. • Construção do mapa de rádio através da recolha de *fingerprints* com uma trajetória definida pelo utilizador. • Localização do utilizador automaticamente no espaço interior, sem necessidade de conexão à internet.	Wi-Fi GPS	Sim	• Algumas funcionalidades apenas disponíveis numa versão da aplicação paga. • O processo de recolha de *fingerprints* é simples.
WiFi Indoor Localization Android	• Podem ser adicionadas plantas do piso a partir de um ficheiro.	Wi-Fi	Sim	• Configuração simples. • O processo de recolha de

	• Define-se o tamanho da célula adequado para a planta do edifício. • Fase de calibração através da recolha de *fingerprints* em várias células. • Localização do utilizador. • Exportação dos dados sobre os pontos de acesso para um ficheiro de texto.		*fingerprints* é o mais intuitivo mas é mais demorado do que nas outras aplicações testadas.

ANEXO B – INTRODUÇÃO AO ANDROID

O desenvolvimento da aplicação de calibração exige o conhecimento do modo de funcionamento das aplicações, da arquitetura do sistema e dos componentes disponibilizados pela plataforma Android.

O Android surge como líder do mercado no que diz respeito aos sistemas operativos para dispositivos móveis, sendo esta temática abordada anteriormente neste documento. Esta plataforma aberta é baseada em Linux permite que sejam desenvolvidas aplicações com diversas finalidades capazes de tirar proveito do *hardware* dos dispositivos.

As aplicações normalmente são desenvolvidas na linguagem de programação Java através ambiente de desenvolvimento oficial, Android Studio [77]. Este IDE baseado em IntelliJ IDEA surgiu recentemente e veio substituir o Eclipse [78] uma vez que traz várias melhorias principalmente ao nível da experiência que proporciona ao utilizador. É necessária a instalação do IDE Android Studio em conjunto as ferramentas SDK (*Software Development Kit*) que oferecem várias funcionalidades para o desenvolvimento das aplicações: criação das interfaces gráficas, *debug* do código desenvolvido, a geração dos ficheiros executáveis apk, download das APIs de várias versões Android, entre outras.

A principal ferramenta de difusão das aplicações Android é a Google Play que não se foca apenas nas aplicações, mas também apresenta vários conteúdos multimédia (música, filmes e livros).

Quanto à distribuição das versões Android, existe fragmentação elevada uma vez que ainda há uma quantidade expressiva de dispositivos com versões bastante antigas. De acordo com os dados recolhidos num período de 7 dias terminado a 1 de fevereiro de 2016 [82], a versão KitKat (4.4) tem maior expressão estando presente em 35,5% dos dispositivos, a versão Lollipop (5.0 e 5.1) abrange cerca de 34% dos dispositivos. A versão mais recente, Marshmallow (6.0) apenas está instalada em 1,2% dos dispositivos. Os restantes valores estão distribuídos pelas versões mais antigas do Android.

Arquitetura de baixo nível

A arquitetura do sistema Android presente na Figura B.1 é composta por várias camadas que comunicam entre si.

A camada superior, denominada *Application Framework* é a camada utilizada mais frequentemente pelos *developers.* Esta camada fornece aos *developers* vários serviços de alto nível através de classes Java que estes podem utilizar nas suas aplicações[85][86].

Os *proxies* da camada Binder IPC permitem a intercomunicação entre a camada superior (*Application Framework*) e a camada inferior (*Android System Services*). No nível da *Application Framework* esta comunicação está escondida do desenvolvedor e permite que a comunicação entre a camada superior e a camada *Android System Services* seja feita sem o seu conhecimento[85].

A camada *Android System Services* comunica com os serviços do sistema para aceder ao *hardware.* Os serviços dividem-se em dois grupos: *system* e *media.* Os *system services* envolvem todos os serviços do sistema, por sua vez os *media services* englobam a reprodução e gravação de multimédia [85].

Mais abaixo, a camada HAL (*Hardware Abstraction Layer*) permite aceder ao *hardware* permitindo que o sistema Android seja agnóstico relativamente às implementações de baixo nível [85].

Por fim, a camada Linux Kernel é a última camada da arquitetura e caracteriza-se pela utilização do Linux Kernel com algumas adições, nomeadamente os *wake locks* (sistema de gestão de memória mais agressivo na preservação de memória), a driver do Binder IPC e outras funcionalidades importantes para a plataforma móvel [85].

Figura B.1: Arquitetura do Sistema Android [85]

Componentes da Aplicação

A *framework de aplicação Android* apresenta vários componentes, e permite a criação de aplicações utilizando como base vários componentes que interagem entre si.

Existe um ficheiro (AndroidManifest.xml) onde são declarados todos os componentes da aplicação e a forma como interagem entre si. Além disso o AndroidManifest.xml também apresenta as permissões de utilização da aplicação (por exemplo, aceder à memória interna), o nível da API utilizada na aplicação as bibliotecas utilizadas e os elementos de *hardware* necessários para a aplicação (Wi-Fi, Bluetooth, etc).

Componentes da aplicação dividem-se em quatro elementos distintos:

- Activity – Este componente fornece um ecrã através do qual os utilizadores podem interagir para fazer qualquer tarefa. A cada *activity* é associado um *layout* que contém o interface do utilizador. A aplicação é composta por várias *activitys* relacionadas entre si que permitem fazer operações diferentes. Existe uma *stack* (chamada "*back stack*") onde se insere a *activity* anterior sempre que se inicia uma nova *activity*, por isso,

quando se inicia uma nova *activity*, para-se a anterior. A *stack* permite voltar atrás (*activity* anterior) sempre que o utilizador carregue no botão "*back*". A *activity* tem um ciclo de vida descrito na Figura B.2, onde está representada a transição de cada estado através do respetivo método [87].

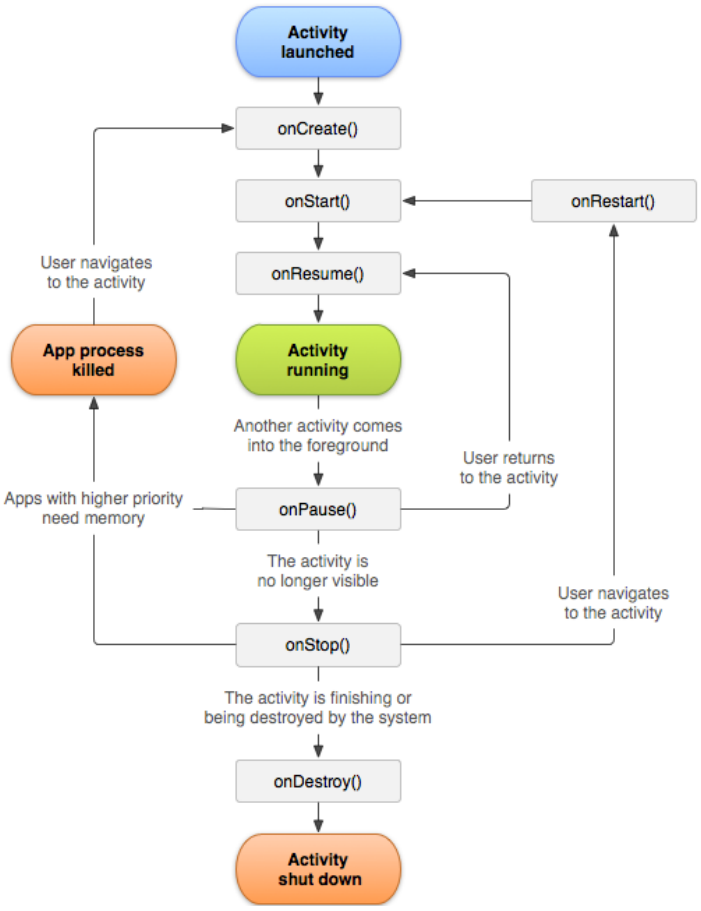


Figura B.2: Ciclo de vida de uma Activity [87]

- Service – São componentes capazes de efetuar operações de longa execução em *background* sem qualquer interface para o utilizador. Um *Service* pode ser executado por outro componente de aplicação e continua em execução mesmo que o utilizador continue em execução em *background.* Um componente pode-se ligar a um *service* para interagir com ele e também pode comunicar com esse mesmo *service* (*interprocess communication IPC*). Existem duas formas diferentes pode assumir: Started ou Bound. No primeiro caso, o serviço é iniciado por um componente de aplicação (por exemplo, uma *activity*) e a partir desse momento fica a correr em *background* por um tempo indeterminado. Normalmente, um serviço deste género apenas realiza uma operação, não retorna qualquer valor e termina quando a tarefa for realizada. Por sua vez, os *services* do tipo *bound* utilizam o modelo cliente-servidor

permitindo que os componentes de aplicação interajam com o *service* através do envio de pedidos, obtenção de resultados, também é possível comunicar com o *service* (IPC). O serviço mantém-se em execução enquanto a componente que o iniciou estiver ligada ao mesmo. Podem existir vários componentes a fazer *bind* ao serviço, mas quando todos estes fizerem *unbind* ao serviço, o serviço é destruído [88]. Tal como as *activities,* os *services* também apresentam um ciclo de vida:

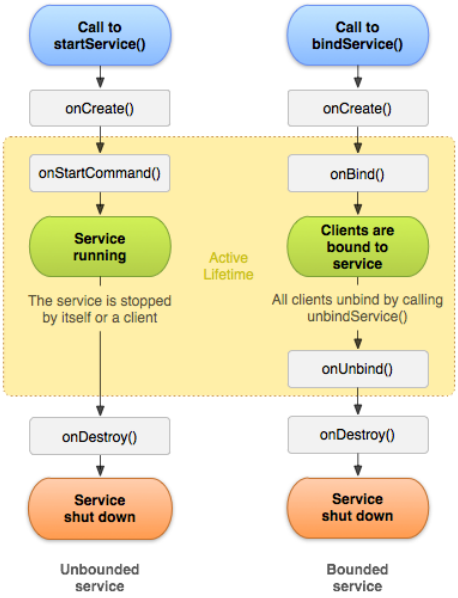


Figura B.3: Ciclo de vida de um Service [88]

- Content Providers – A gestão do acesso para um conjunto de dados estruturado é feita pelos *content providers*, que fazem o encapsulamento dos dados e providenciam mecanismos para a definição da segurança dos dados. São a interface *standard* que conecta os dados de um processo com código em execução noutro processo [89].
- Broadcast Receivers – É um componente que responde a anúncios *broadcast* de todo o sistema. Muitos destes anúncios *broadcast* têm origem no sistema (por exemplo: quando o ecrã é desligado, quando a bateria está fraca ou quando é tirada uma fotografia). As aplicações também podem iniciar *broadcasts* (por exemplo quando o download de um ficheiro é terminado e a aplicação pretender notificar outra aplicação). Os *broadcast receivers* não apresentam interface, mas são capazes de criar uma notificação na barra de estado para informar o utilizador quando ocorre um evento *broadcast* [90].

Existe também um conjunto de componentes adicionais que são utilizados na construção dos componentes acima descritos, na sua lógica e na interligação entre eles. Estes componentes são [91]:

- Fragments – representa uma parte do interface do utilizador numa *activity.* Uma *activity* pode ter vários fragmentos, sendo que cada fragmento tem associado um *layout* específico;
- Views – elementos do interface do utilizador que são desenhados no ecrã, incluindo botões, listas, formas, etc;
- Layouts – hierarquia de *views* que controla o formato do ecrã e a aparência das *views;*
- Intents – mensagens que interligam as componentes;
- Resources – todos os elementos externos tais como *strings,* constantes, imagens;
- Manifest – é o ficheiro AndroidManifest.xml descrito anteriormente, essencialmente é o ficheiro de configurações para a aplicação.

ANEXO C – WEB SERVICES DAS PLANTAS DOS EDIFÍCIOS E SISTEMA DE CALIBRAÇÃO

Especificação dos *web services* que permitem a comunicação entre o servidor e as aplicações de calibração e Where@UM.

Os *web services* distinguem-se em três componentes:

- Componente das plantas dos edifícios;
- Componente de suporte à aplicação de calibração;
- Componente dos mapas de rádio.

Componente das plantas dos edifícios

1. Obter a lista de *operators* armazenados na base de dados

Método HTTP	GET
URI	{server}/floormaps/operators
Parâmetros	Sem parâmetros
Resposta (formato)	{"operators": {"id":"1", "name":"Universidade do Minho"}, {"id":"2", "name":"Universidade do Porto"} }

Descrição: Devolve a lista de *operators* armazenados na base de dados.

2. Obter a lista de áreas de um *operator*

Método HTTP	GET
URI	{server}/floormaps/operators/:*id_operator*/areas
Parâmetros	• id_operator – identificador do operador
Resposta (formato)	{"areas": {"id":"1", "name":"Campus de Gualtar"}, {"id":"2", "name":"Campus de Azurém"} }

Descrição: Retorna a lista de áreas de um *operator* dado o seu id, o resultado é um *array* com a lista de *areas*(id, nome).

3. Obter a lista de edifícios de uma área de um operador.

Método HTTP	GET
URI	{server}/floormaps/operators/:*id_operator*/areas/:*id_area*/buildings
Parâmetros	• id_operator – identificador do operador • id_area – identificador da área
Resposta (formato)	{"buildings": {"id":"1", "name":"Escola de Engenharia"}, {"id":"2", "name":"Campus de Azurém"} }

Descrição: Devolve a lista de edifícios (id, nome) tendo em conta a *area* a que pertencem a partir dos ids da *area* e do *operator*.

4. Obter a lista de pisos de um edifício.

Método HTTP	GET
URI	{server}/floormaps/operators/*:id_operator*/areas/*:id_area*/buildings/*:id_bulding*/floors
Parâmetros	• id_operator – identificador do operador • id_area – identificador da área • id_building – identificador do edifício
Resposta (formato)	{"floors": {"id":"1", "name":"Floor0"}, {"id":"2", "name":"Floor1"}, {"id":"3", "name":"Floor-1"} }

Descrição: Devolve a lista de pisos (plantas) do edifício dados os ids do *building*, *area* e *operator*.

5. Obter a planta de um piso.

Método HTTP	GET
URI	{server}/floormaps/operators/*:id_operator*/areas/*:id_area*/buildings/*:id_building*/floors/*id_floor*
Parâmetros	• id_operator – identificador do operador • id_area – identificador da área • id_building – identificador do edifício • id_floor – identificador do piso
Resposta (formato)	Conteúdo do ficheiro da planta (XML/OSM)

Descrição: Devolve a planta do piso no formato XML. Este conteúdo é obtido através dos ids do *floor*, *building*, *area* e *operator*.

6. Obter planta de um piso (aplicação Where@UM)

Método HTTP	GET
URI	{server}/floormaps/maps?operator=Universidade%20do%20Minho&area=Campus%20de%20Azurém&building=Escola%20de%20Engenharia&floor=1
Parâmetros	• Operador – Nome do operador • Área – Área geográfica • Edifício – nome do edifício • Piso – nome do piso (por exemplo, 0, 1, 2, etc)
Resposta (formato)	Conteúdo do ficheiro da planta (XML/OSM)

Descrição: Este pedido foi desenvolvido especificamente para servir a aplicação Where@UM, mas pode ser utilizado no futuro por qualquer serviço. Através dos nomes dos parâmetros é possível obter a planta de cada piso da mesma forma que se obtém a planta no pedido 5. Como os nomes permitem identificar cada um dos elementos (operador, área, edifício e piso), é possível obter a planta do piso utilizando estas informações

Componente de suporte à aplicação

1. Efetuar o *login* na aplicação de calibração.

Método HTTP	POST
URI	{server}/floormaps/login
Parâmetros	Sem parâmetros

***Payload* pedido**	{
	"email":"A65239@alunos.uminho.pt",
	"password":"ABC1234",
	"device_mac":"00:04:4b:2c:be:ff"
	}

Resposta (formato)	{"status":
	{
	"value":true,
	"msg":"Login successfull"
	}
	}

Descrição: O pedido POST com os parâmetros e-mail e *password*, permite que o serviço dê acesso ao utilizador com a resposta devolvida (true/false). Essencialmente, a *password* do

utilizador é comparada com a *password* do utilizador armazenada na BD, e o resultado da comparação permite fazer o login do utilizador. Também é enviado o endereço MAC do dispositivo e este é registado caso seja a primeira vez que o utilizador utiliza o dispositivo.

2. Registo de um utilizador.

Método HTTP	POST
URI	{server}/floormaps/users
Parâmetros	Sem parâmetros

***Payload* pedido**	<pre>{
	"email":"teste@alunos.uminho.pt",
	"password":"ABC1234",
	"name":"Ivo Silva",
	"operator_name":"Universidade do Minho",
	"device_mac":"00:04:4b:2c:be:ff"
	}</pre>

Resposta (formato)	<pre>{"status":
	{
	"value":true,
	"msg":" User registered with success "
	}
	"idUser":"5"
	}</pre>

Descrição: Tem a função de receber os parâmetros de registo de um utilizador (*operator*, *name*, *password*, *email* e endereço MAC do dispositivo) e efetua o registo do utilizador e do dispositivo caso se trate de um novo utilizador, devolvendo como resposta um objeto JSON que apresenta o resultado da operação.

3. Recuperação da *password* do utilizador.

Método HTTP	POST
URI	{server}/floormaps/users/:id_user/password
Parâmetros	• id_user – identificador do utilizador

***Payload* pedido**	<pre>{
	"email":"A65239@alunos.uminho.pt"
	}</pre>

Resposta (formato)	<pre>{"status":
	{
	"value":true,
	"msg":"Password recovered successfully"
	}
	}</pre>

Descrição: Este pedido é feito quando o utilizador pretende recuperar a *password*. É gerada uma nova *password* aleatória e enviada para o email do utilizador. Como resultado é devolvido um objeto JSON que descreve o estado da operação.

4. Alteração da *password* do utilizador.

Método HTTP	PUT
URI	{server}/floormaps/users/*:id_user*/password
Parâmetros	• id_user – identificador do utilizador

***Payload* pedido**	{
	"email":"A65239@alunos.uminho.pt",
	"oldpassword":"ABC1234",
	"newpassword":"1234ABC"
	}

Resposta (formato)	{"status":
	{
	"value":true,
	"msg":"Password updated successfully"
	}
	}

Descrição: Este pedido permite atualizar a *password* do utilizador. Os dados enviados no pedido são a *password* antiga, a nova password e o email. Como resultado é devolvido um objeto JSON com o estado da operação.

5. Obter dados de um utilizador dado o seu e-mail.

Método HTTP	GET
URI	{server}/floormaps/users?email=A65239@alunos.uminho.pt
Parâmetros	• E-mail – e-mail do utilizador

Resposta (formato)	{
	"id_user":"1",
	"email":"A65239@alunos.uminho.pt",
	"name":"Ivo Silva",
	"id_operator":"1"
	}

Descrição: Permite obter os dados de um utilizador a partir do seu email que é passado como argumento no URI do pedido. O resultado é um objeto JSON com os dados do utilizador.

6. Remover utilizador.

Método HTTP	DELETE
URI	{server}/floormaps/users/*:id_user*
Parâmetros	• id_user – identificador do utilizador

Resposta (formato)	{"status":
	{
	"value":true,
	"msg":"User deleted with success"
	}
	}

Descrição: Apagar um utilizador dado o seu identificador.

Componente dos mapas de rádio

1. Envio de uma *fingerprint.*

Método HTTP	POST
URI	{server}/calibration/fingerprints
Parâmetros	Sem parâmetros

***Payload* pedido**	{
	"email":"A65239@alunos.uminho.pt",
	"id_device":"00:04:4b:2c:be:3a",
	"id_room":-203,
	"id_floor":1,
	"lat":"41.452918695431364",
	"lon":"-8.286296829769856",
	"timestamp":"2016-02-16 10:41:21",
	"fingerprint":[
	{"mac":"d4:d7:48:0c:8d:40","rssi":"-58","ssid":"eduroam","channel":11},
	{"mac":"00:3a:98:ef:fa:50","rssi":"-61","ssid":"eduroam","channel":1},
	{"mac":"b4:14:89:d1:b3:8b","rssi":"-94","ssid":"","channel":36}
	]
	}

Resposta (formato)	{"status":
	{
	"value":true,
	"msg":"Fingerprint saved in the database"
	}
	}

Descrição: Este pedido do tipo POST recebe um conjunto de dados associados à *fingerprint*, sendo que estes são recebidos, devidamente processados e armazenados na base de dados.

2. Obter lista de *fingerprints* recolhidas num piso.

Método HTTP	GET
URI	{server}/calibration/floors/*:id_floor*/fingerprints
Parâmetros	• id_floor– identificador do piso

Resposta (formato)	{
	"5": {
	"id_user": "1",
	"lat": "41.452906963240",
	"lon": "-8.286312124060",
	"timestamp": "2016-01-18 12:03:47"
	},
	"6": {
	"id_user": "1",
	"lat": "41.452906911110",
	"lon": "-8.286312122220",
	"timestamp": "2016-01-18 12:09:09"
	},
	"7": {
	"id_user": "1",

```
                "lat": "41.452906911110",
                "lon": "-8.286312122220",
                "timestamp": "2016-01-21 16:12:46"
            }
        }
```

Descrição: Pedido que permite obter o estado de calibração de um piso através das informações das *fingerprints* armazenadas.

Uma API deverá suportar pelo menos alguns dos códigos de estado HTTP. Estes códigos no corpo da resposta informam a aplicação cliente qual será a ação que deverá ser tomada com a resposta. Por exemplo, o código de resposta 200 significa que o pedido foi processado com sucesso do lado do servidor. Por outro lado, códigos de resposta como o 400 ou o 401 são respostas associadas a maus pedidos ou a pedidos não autorizados. Na tabela seguinte estão alguns dos códigos de estado HTTP que devem ser suportados pelos *web services*.

Tabela C.1: *Alguns dos códigos de estado HTTP*

Código	Valor
200	*OK*
201	*Created*
304	*Not Modified*
400	*Bad Request*
401	*Unauthorized*
403	*Forbidden*
404	*Not Found*
422	*Unprocessable Entity*
500	*Internal Server Error*

ANEXO D – TUTORIAL JOSM

O Anexo D apresenta um tutorial relativo à criação de plantas de edifício utilizando a ferramenta JOSM.

Requisitos

- Programa JOSM (https://josm.openstreetmap.de/wiki/Download)
- *Plugin* PicLayer no JOSM (*Edit->Preferences->Modules->PicLayer*)

Download do Mapa OSM

Inicialmente é necessário obter o mapa da área que se pretende mapear através da opção para descarregar o mapa (Figura D.1), e selecionar a área desejada, como ilustra a Figura D.2.

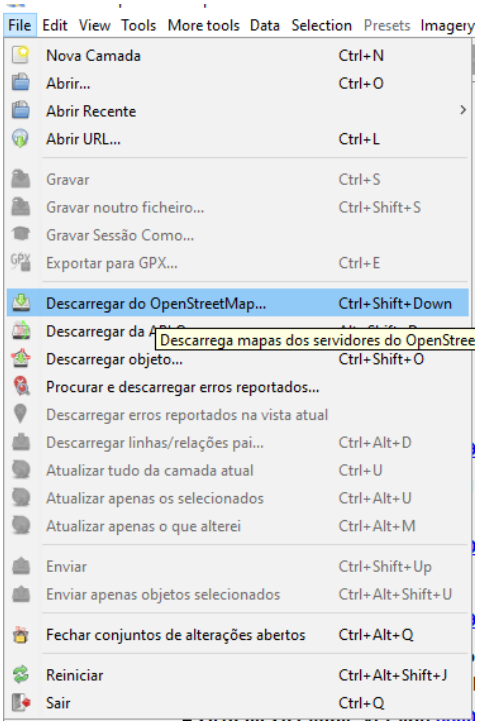


Figura D.1: Selecionar a opção para descarregar uma parte do mapa

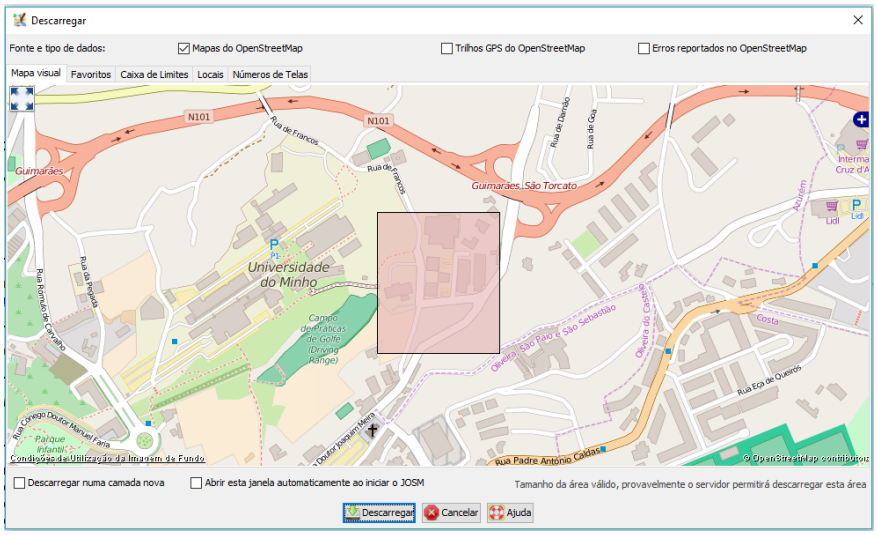


Figura D.2: Selecionar a área do mapa

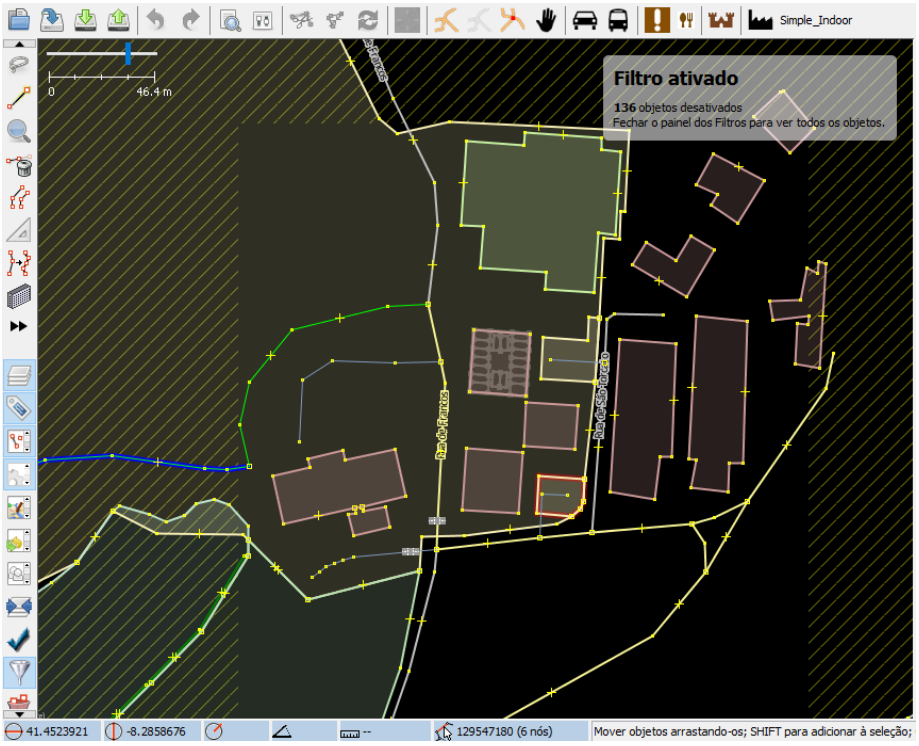


Figura D.3: Mapa descarregado

O objetivo agora é adicionar a imagem da planta através do PicLayer, como demonstra a Figura D.4.

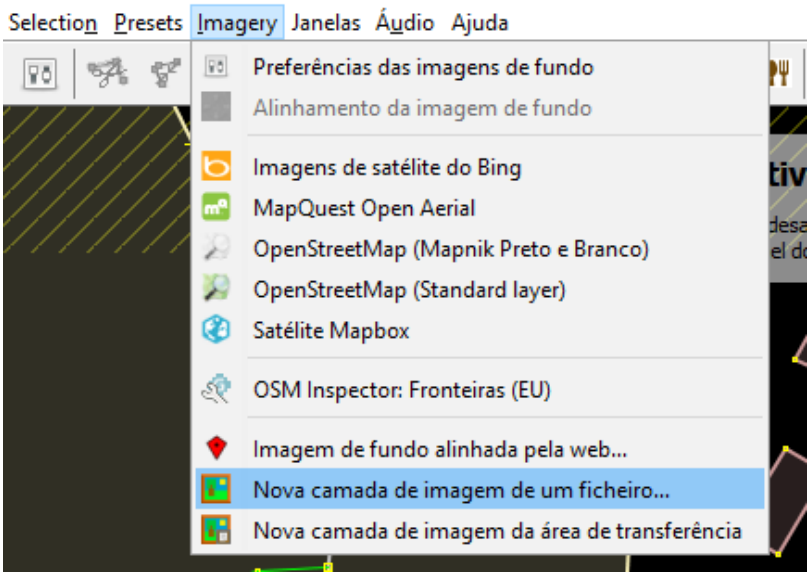


Figura D.4: Seleção da opção para selecionar a imagem da planta

Selecionar a imagem da planta e depois selecionar a imagem exemplificado na Figura D.5.

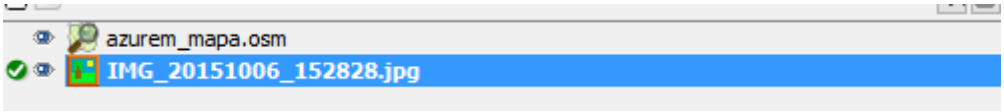


Figura D.5: Selecionar a planta para que possa ser editada com o PicLayer

Usar as seguintes ferramentas para redimensionar a imagem:

Redimensionar a imagem 

Rodar a imagem 

Mover a imagem 

Para criar "pregos" na imagem, utilizar:  (os pregos permitem que a imagem seja fixa em três pontos de forma a que ela seja encaixada na estrutura do edifício) É aconselhável que sejam selecionados os 3 pontos dominantes da planta (por exemplo, numa planta retangular são selecionados 3 cantos), a Figura D.6 demonstra esse processo.

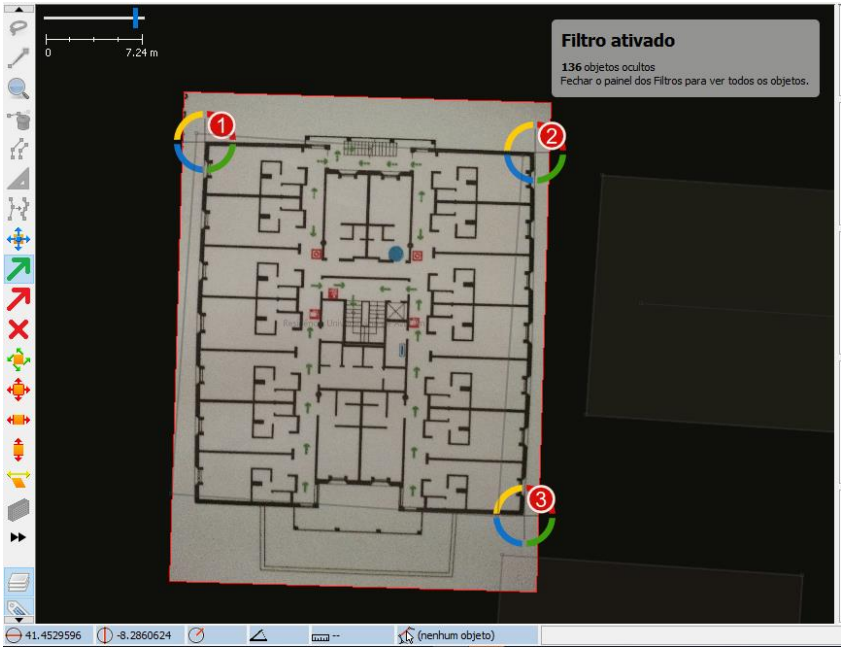


Figura D.6: Marcação dos pontos a fixar na planta do piso

Depois os "pregos" são deslocados para cada um dos cantos do edifício, ajustando a planta à estrutura do edifício, através da seguinte ferramenta:  .

É necessário colocar o cursor exatamente no ponto em que o "prego" foi marcado para que a imagem possa ser devidamente deslocada.

A Figura D.7 apresenta o resultado na colocação da planta exatamente no mesmo local que o edifício.

Figura D.7: Resultado do ajuste da imagem da planta com o mapa

Modelação

O próximo passo tem como objetivo a marcação de cada um dos espaços interiores.

Como base utiliza-se o elemento do edifício que já está definido e adicionam-se algumas *tags* no seguinte menu da Figura D.8:

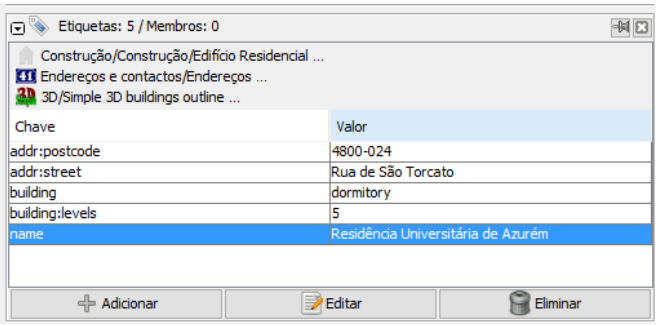


Figura D.8: Menu de etiquetas

O objetivo é ter os seguintes pares chave-valor:

- *max_level*=piso mínimo
- *min_level*=piso máximo
- *building*=tipo de edifício (por exemplo "*residential*")
- *name*=nome do edifício
- *operator*=Entidade que possui o edifício

Além destes valores é necessário ter também os valores associados à morada do edifício, normalmente já se encontram preenchidos:

- *addr:country*=PT (exemplo)
- *addr:district*=*
- *addr:city*=*
- *addr:street*=*
- *addr:place*=*
- *addr:postcode*=*

O "*" representa qualquer valor.

O processo de criação de chaves resulta no conjunto de *tags* para o edifício, tal como mostra a Figura D.9.

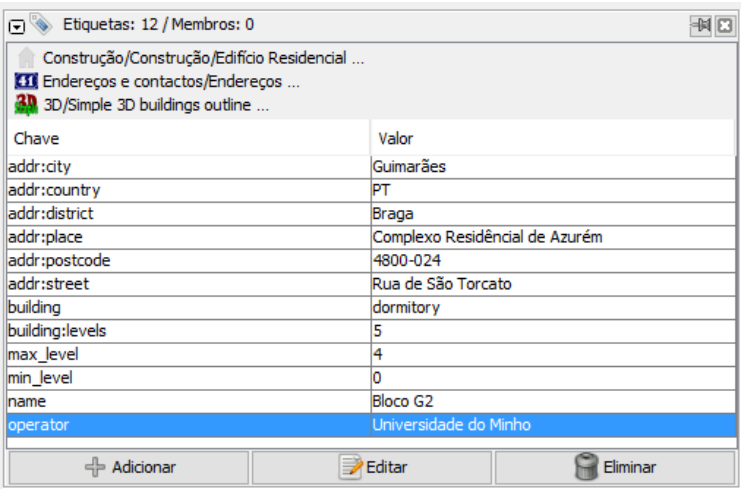


Figura D.9: Menu resultante depois de se adicionarem todos os pares chave-valor

De seguida procede-se ao desenho de cada divisão interior utilizando a ferramenta:  .

Cada divisão deverá ser uma forma fechada e não pode incluir outras divisões dentro da sua forma. Pode mesmo ser necessário desenhar duas formas para a mesma divisão, por exemplo um corredor.

Podem ser desenhados os seguintes elementos:

- *room (way)*
- *area (way)*
- *corridor (way)*
- *stairs (way)*
- *elevator (way)*
- *wall (way)*
- *door (node)*

Way é representado por uma forma (um conjunto de nós unidos, representando um polígono fechado), demonstrado na Figura D.10.

Node é representado como um ponto, é um simples nó.

Cada forma tem que ser um polígono fechado, deste modo quando se desenham divisões juntas, é necessário desenhar a sua forma completa independentemente da forma da outra divisão vizinha (Figura D.11).

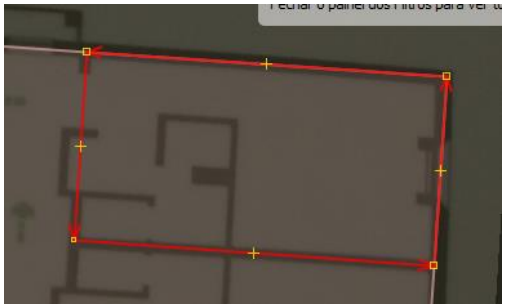


Figura D.10: Desenho de um quarto

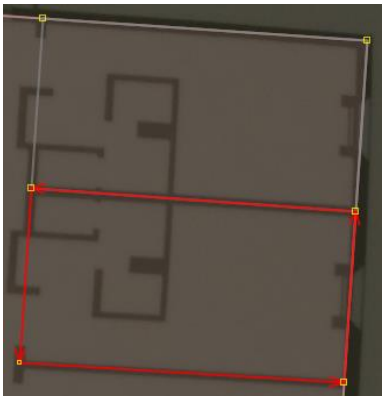


Figura D.11: Modelação de um quarto adjacente

Para se marcar um node, basta marcar o ponto no local e utilizar a tecla "*Escape*" para que o ponto inicial fique marcado como "*node*" no local (Figura D.12).

Figura D.12: Marcação de um nó (porta)

Depois de marcados todos os espaços interiores com exceção da estrutura base do edifício, é possível selecionar todos os elementos *indoor* de forma a associar os pares chave-valor (Figura D.13):

- *level=**

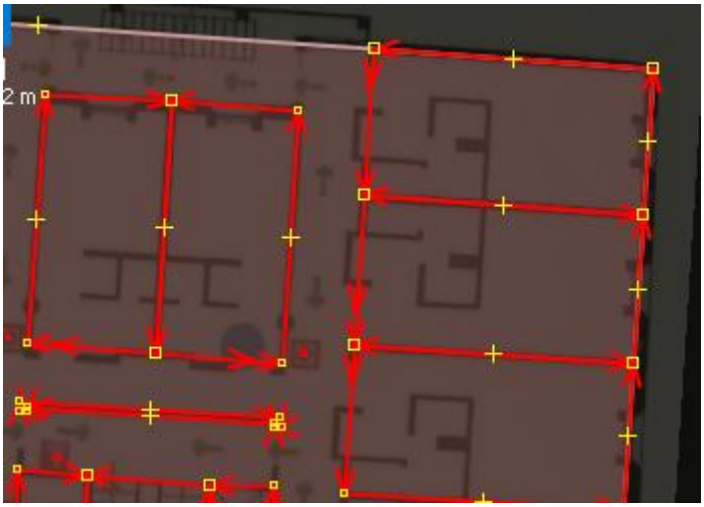


Figura D.13: Seleção de todos os elementos *indoor* exceto a estrutura exterior do edifício

Todos os elementos *indoor* têm a chave "*level*" e o valor do respetivo piso.

Além disso cada um dos elementos tem os seguintes pares chave-valor:

- *room*
  - *name=**
  - *indoor=room*
- *area*
  - *name=** (não é obrigatório este par)
  - *indoor=area*
- *corridor*
  - *name=** (não é obrigatório este par)
  - *indoor=corridor*
- *wall*
  - *indoor=wall*
- *stairs*
  - *indoor=room*
  - *stairs=yes*
- *elevators*
  - *highway=elevator*
  - *indoor=room*
- *doors*
  - *door=hinged*
- *porta de entrada do edifício*
  - *door=yes*
  - *entrance=main*

Relations

Depois de se criarem todos os espaços interiores e de se atribuírem os pares chave-valor, é necessário criar relações, uma que define a estrutura exterior do edifício "*shell*" e outra que define todas as partes do edifício que constituem o piso.

Inicialmente basta selecionar a estrutura do edifício, e cria-se a relação como ilustra a Figura D.14.

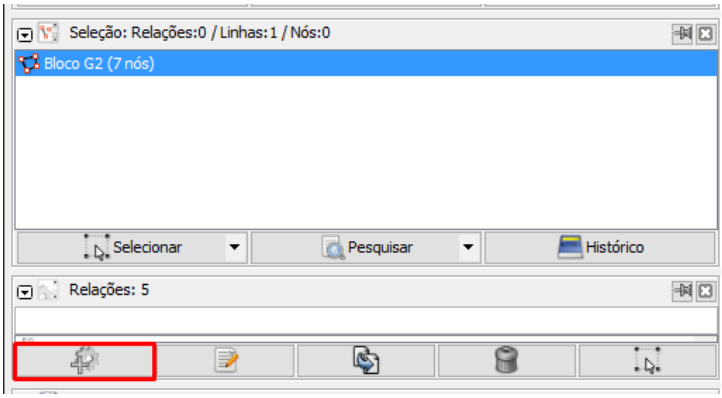


Figura D.14: Criar uma nova relação

Os pares chave-valor da relação podem ser editados no menu da criação da relação (Figura D.15).

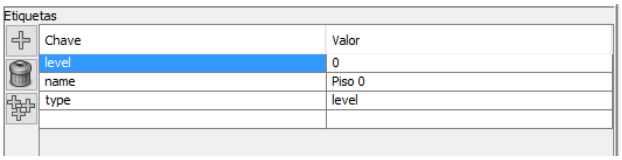


Figura D.15: Menu de criação de relação

A seleção adiciona-se como mostra a Figura D.16.

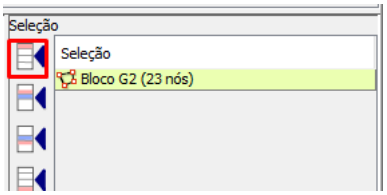


Figura D.16: Adicionar seleção à relação

A seleção deverá ter a função "*shell*" (Figura D.17).

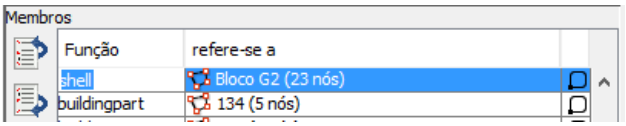


Figura D.17: Adicionar a função do elemento adicionado à relação

Depois selecionam-se todos os elementos *indoor*, utilizando a tecla "*Ctrl*" e o botão esquerdo do rato, aos quais se atribui a função de "*buildingpart*". Este processo está representado nas figuras D.18 e D.19.

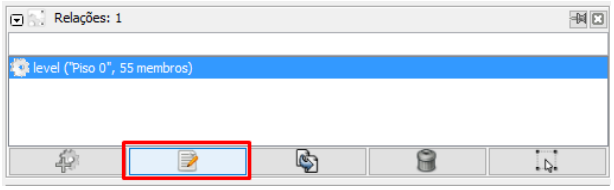


Figura D.18: Editar a relação

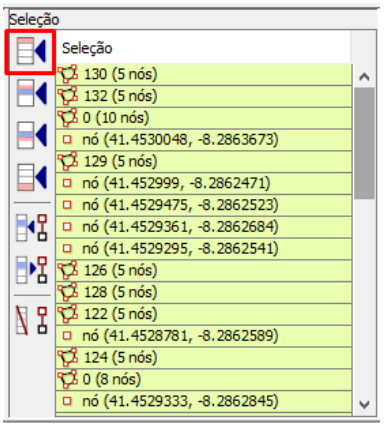


Figura D.19: Adicionar a seleção à relação

Cada elemento adicionado terá a função "*buildingpart*" (Figura D.20).

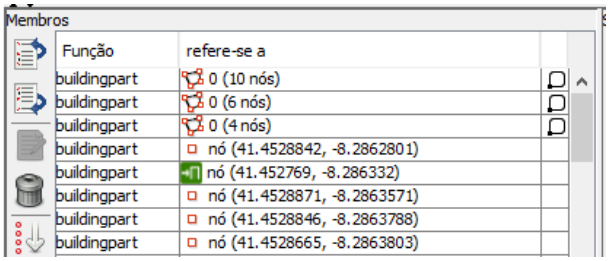


Figura D.20: Atribuição da função "*buildingpart*" aos elementos adicionados à relação

Guardar ficheiro OSM

Por fim, seleciona-se todo o edifício e copia-se o conteúdo (Figura D.21).

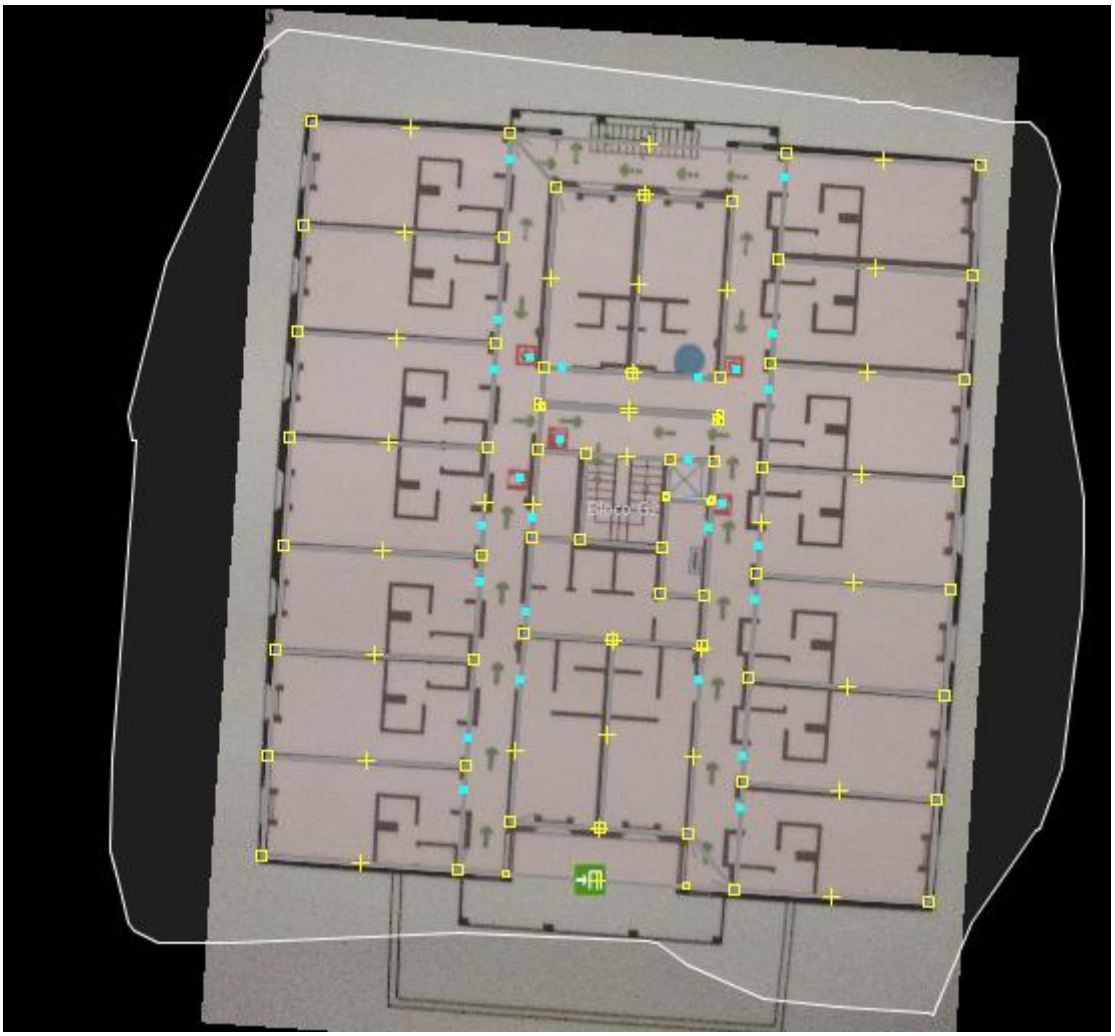


Figura D.21: Seleção de todo o piso

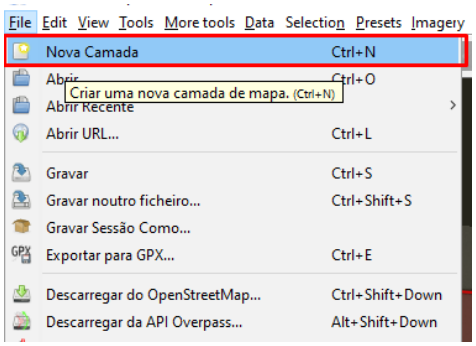


Figura D.22: Criação de uma camada nova

Na nova camada de dados basta colar o conteúdo selecionado (Figura D.22), que está mostrada na Figura D.23.

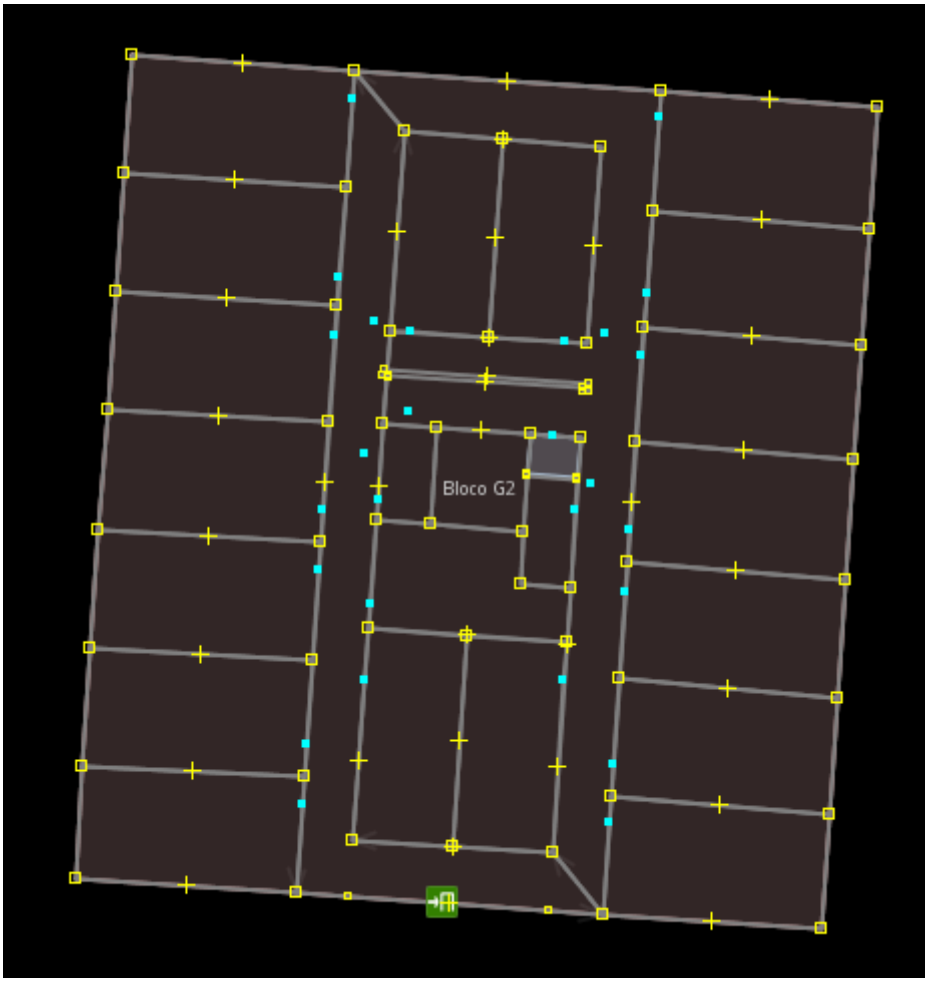


Figura D.23: Nova camada com a planta do piso modelada

Para guardar o ficheiro, basta selecionar Ficheiro → Gravar.

Basta fazer o mesmo processo, para cada piso, dando os respetivos nomes. Também é possível aproveitar as plantas nos casos em que todos os pisos têm a mesma estrutura.

ANEXO E – WEB SERVICES WHERE@UM

O Anexo E especifica os *web services* do sistema Where@UM que permitem a comunicação entre a aplicação móvel e o servidor. Estes serviços foram sofreram alterações com o intuito de melhorar a segurança.

ASM
{server}=http://urano.dsi.uminho.pt/whereatumdev/asm2

1. Pedido de *token* de sessão

Método HTTP	**POST**
URI	{server}/users/self/token
Token como Cookie	Não
Parâmetros	• *String email* – endereço eletrónico do novo utilizador. • *String mac* – endereço MAC. • *String platform* – plataforma proveniente (android, windowsdesktop, windowsphone).
***Payload* pedido**	{ "email":"teste1@email.com", "mac": "64:BC:0C:7F:AC:7B", "platform": "windowsdesktop" }
Resposta (formato)	Sucesso – devolve objeto com os dados do perfil de utilizador. • *String token.* Insucesso: 400 Bad Request

Descrição: Pedido efetuado antes de iniciar a sessão. Como resposta é devolvido o *token* para o utilizador se autenticar na plataforma.

2. Autenticação de um utilizador

Método HTTP	**POST**
URI	{server}/users/self/login
Token como Cookie	Sim
Parâmetros	• *String email* – endereço eletrónico do novo utilizador. • *String tokenpassword* – Hash MD5 da *password* com o *token*. Este campo é o MD5(*token*+MD5(*password*)) no novo método de autenticação, caso o utilizador ainda não tenha a password como MD5, o valor *token* password é uma string com o valor "*token+password*". • *String mac* – endereço MAC.

	• *String platform* – plataforma proveniente (android, windowsdesktop, windowsphone).
***Payload* pedido**	{ "email":"teste1@email.com", "tokenpassword":"3db3545816dbfdb548863a15a41865ff", "mac": "64:BC:0C:7F:AC:7B", "platform": "windowsdesktop" }
Resposta (formato)	Sucesso – devolve objeto com os dados do perfil de utilizador. • *Int idUser* • *String name* • *String nickname* • *String email* • *String locationSharing* • *Int deletedAccount* • *String registrationDate* • *String token* • *String tokenTimestamp* 400 Bad Request – Em caso de insucesso { "idUser": "186", "name": "Ivo Silva2", "email": "ivo_silva@outlook.com", "nickname": "Ivo (Outlook)", "locationSharing": "1", "deletedAccount": "0", "registrationDate": "2016-03-12 10:38:09", "tokenTimestamp": "2016-05-09 17:00:12" }

Situação extraordinária:

Como na base de dados ainda existem passwords guardadas noutro tipo de cifra que não MD5, devolve uma mensagem específica.

```
{
     "status": "Failed",
     "msg": "Please use the old authentication method or wrong email-password"
}
```

Neste caso o campo *tokenpassword* deve conter o *token+password* sem qualquer tipo de *hashing*.

Descrição: Depois de obter o *token* de sessão, é este pedido para verificar os dados de autenticação do utilizador. Caso os dados estejam corretos, é devolvido um objeto JSON com os dados de utilizador.

3. *Logout* de um utilizador

Método HTTP	**POST**
URI	{server}/users/self/logout
Token como Cookie	Sim
Parâmetros	
***Payload* pedido**	
Resposta (formato)	Sucesso: 200 OK Insucesso: 400 Bad Request

Descrição: Quando o utilizador faz *logout* este pedido é despoletado com o intuito de apagar a sessão ativa no sistema.

4. *Refresh* do *token* de sessão

Método HTTP	**POST**
URI	{server}/users/{idUser}/refreshtoken
Token como Cookie	Sim
Parâmetros	
***Payload* pedido**	

Resposta (formato)	{
	"token": "uKioww1Kxf5TsKQAWR5yNyPl5HSxDgYacfXs4ioWeomjjqTq03",
	"tokenTimestamp": "2016-05-07 13:39:44"
	}

Descrição: Para renovar o *token* de sessão antes dela expirar, é necessário fazer este pedido para que seja devolvido um novo *token* atualizado.

5. Registo de um utilizador

Método HTTP	**POST**
URI	{server}/users
Token como Cookie	Não
Parâmetros	• *String name* – nome do novo utilizador. • *String nickname* (opcional) – alcunha do novo utilizador. • *String email* – endereço eletrónico do novo utilizador.

	• *String password* – valor md5 da *password.* • *String idDevice* – *idDevice* do dispositivo. • *String platform* – plataforma proveniente (android, windowsdesktop, windowsphone).

***Payload* pedido**	{
	"name":" Ivo Gmail ",
	"nickname":"",
	"email":" ivomiguelmsilva@gmail.com",
	"password":"03d36b1fc29fffb01b9a0715c795ba9c",
	"idDevice": "64:BC:0C:7F:AC:7B",
	"platform": "windowsdesktop"
	}

Resposta (formato)	{
	"idUser": "203",
	"name": "Ivo Gmail ",
	"email": "ivomiguelmsilva@gmail.com",
	"nickname": null,
	"locationSharing": "1",
	"deletedAccount": "0",
	"registrationDate": "2016-05-09 16:38:49",
	"token": "1dzkBxBzwbmAVzH3Bi71AX4Zdf2haeK5ZZzkK4UDvwMGOOVYKD",
	"tokenTimestamp": "2016-05-09 16:38:49"
	}

Descrição: Pedido efetuado quando é feito um registo na aplicação cliente. Como resultado é devolvido um objeto JSON com as informações do utilizador.

6. Obter a planta de um piso

{server}=http://urano.dsi.uminho.pt/whereatumdev/cal

Método HTTP	GET
URI	{server}/floormaps/maps?operator=Universidade%20do%20Minho&area=Campus%20de%20Azurém&building=Escola%20de%20Engenharia&floor=1
Parâmetros	• *String* operador – nome do operador. • *String* área – área geográfica. • *String* edifício – nome do edifício. • *String* piso – nome do piso (por exemplo, 0, 1, 2, etc).
Resposta (formato)	Conteúdo do ficheiro da planta (XML/OSM). Caso a planta não exista, é devolvido o código 404 (*not found*).

Descrição: Este pedido foi desenvolvido especificamente para servir a aplicação Where@UM, mas pode ser utilizado por qualquer serviço. Através dos nomes dos parâmetros é possível obter

a planta de cada piso, caso exista. Como os nomes permitem identificar cada um dos elementos (operador, área, edifício e piso), é possível obter a planta do piso utilizando estas informações.

7. Mensagens do utilizador

Método HTTP	**POST**
URI	{server}/users/{idUser}/messages
Token como Cookie	Sim
Parâmetros	• *String regid* – identificador de registo do serviço GCM do destinatário; • *String message* – mensagem a enviar.
***Payload* pedido**	{ "regid":"APA91bHPpzbLGrRwJMjGVuq0q02jx5bf6hBHKVcjX347LjMcWFaMkQsK6PHIV0X3o-wP5CxRFvPa6tc2CO0BCshzDWcgceBW0O_EVBEZMlXQYEbTgXp-1iA", "message":"olá" }
Resposta (formato)	{ "multicast_id": 6417630731239064644, "success": 1, "failure": 0, "canonical_ids": 0, "results": [{ "message_id": "0:1464080873442675%aa711c20f9fd7ecd" }] }

Descrição: Pedido responsável por receber a mensagem juntamente com o identificador de registo no serviço GCM do destinatário e o identificador do utilizador que enviou a mensagem. Envia estes dados para o servidor do GCM (invocando a função "*send_push_notifications*") que se certificará de enviar a mensagem ao utilizador destinatário.

8. Envio do identificador GCM do utilizador

Método HTTP	**POST**
URI	{server}/users/{idUser}/gcm
Token como Cookie	Sim
Parâmetros	• *String regID* – identificador de registo do serviço GCM do utilizador. • *String idDevice* – endereço MAC do dispositivo.

***Payload* pedido**	{
	"regID":"APA91bEDiRO-63QzK0zZVsVd3IJVpRa__P1eVYdu5b9OzHCFWcnFWNrQpS8n2f-lnDDOr19vgH8pBquUr7FcPunP3FkSsyw0QFovpXS0utJs7UC54IxOLps",
	"idDevice": "00:04:4B:2C:BE:3A"
	}

Descrição: Envia o *regID* (fornecido pelo GCM) e o *idDevice* da aplicação cliente para o servidor da Where@UM.

9. Obter o identificador GCM do utilizador

Método HTTP	**Get**
URI	{server}/users/{idUserDestination}/Regid
Token como Cookie	Sim
Parâmetros	
***Payload* pedido**	

Resposta (formato)	{
	"gcm_regID":"APA91bHX1vrwQNBUphE6kgGkvzOSTJjyK1JYsSxRGX0X…(continua)"
	}

Descrição: Sempre que um utilizador deseja enviar uma mensagem a outro utilizador (amigo) a aplicação móvel solicita a esta função o identificador de registo no serviço GCM referente ao amigo a que pretende enviar uma mensagem.

10. Enviar pergunta 1

Método HTTP	**Get**
URI	{server}/send/{idUser}/question1
Token como Cookie	Sim
Parâmetros	
***Payload* pedido**	
Resposta (formato)	Sucesso: 200 OK Insucesso: 400 Bad Request

Descrição: Pedido responsável por enviar uma notificação aos utilizadores questionando se concordam com a localização exibida pela aplicação. O servidor depois faz um envio de uma notificação para o dispositivo móvel através da função "*send_push_notification*".

11. Enviar pergunta2

Método HTTP	**Get**
URI	{server}/send/{idUser}/question2
Token como Cookie	Sim
Parâmetros	
***Payload* pedido**	
Resposta (formato)	Sucesso: 200 OK Insucesso: 400 Bad Request

Descrição: Pedido responsável por enviar uma notificação aos utilizadores alertando que estão numa localização desconhecida, e perguntando se desejam inserir a sua localização atual.

12. Enviar resposta1

Método HTTP	**POST**
URI	{server}/send/{idUser}/responseQ1
Token como Cookie	Sim
Parâmetros	• *String response* – valor da resposta. • *String date* – data da resposta.

***Payload* pedido**	{
	"response":"YES",
	"date_e":"24/05/2016 16:05:55"
	}
	Nota: o valor da response pode ser "*YES*" ou "*NO*"

Resposta (formato)	Sucesso: 200 OK Insucesso: 400 Bad Request

Descrição: Sempre que um utilizador responder a uma questão, enviada pela função "*sendquestion1*", esta função é invocada e tem como função guardar na base de dados, na tabela

"*feedbackUsers*", as respostas emitidas pelos utilizadores e atualizar os campos "*likes*" e "*unlikes*" da tabela "*fingerprints*".

13. Enviar resposta 2

Método HTTP	**POST**
URI	{server}/send/{idUser}/responseQ2
Token como Cookie	Sim
Parâmetros	• *String response* – valor da resposta. • *String date* – data da resposta.

***Payload* pedido**	{
	"response":"YES",
	"date_e":"24/05/2016 16:09:10"
	}
	Nota: o valor da response pode ser "YES" ou "NO"

Resposta (formato)	Sucesso: 200 OK Insucesso: 400 Bad Request

Descrição: Quando um utilizador responde a uma questão enviada pela função "sendquestion2", esta é invocada e tem como função guardar na base de dados as respostas emitidas pelos utilizadores.

14. Obter informação relativa aos amigos do utilizador

Método HTTP	**GET**
URI	{server}/users/{idUser}/friends
Token como Cookie	Sim
Parâmetros	
***Payload* pedido**	
Resposta (formato)	Sucesso – devolve objeto com os dados os amigos 400 Bad Request – Em caso de insucesso { "pending": null, "friends": [{ "idUser": "4", "date": "2016-02-17 12:45:10", "name": "Adriano Moreira",

```
                "email": "ajcm2appls@gmail.com",
                "nickname": "ajcmoreira"
            }
        ],
        "requests": [
            {
                "idUser": "173",
                "date": "2016-02-24 23:28:32",
                "name": "Ricardo Outlook",
                "email": "ricardomesquita@outlook.pt",
                "nickname": null
            },
            {
                "idUser": "180",
                "date": "2016-02-24 23:26:59",
                "name": "joaoteste@teste.com",
                "email": "joaoteste@teste.com",
                "nickname": null
            }
        ]
    }
```


Descrição: Este pedido permite obter a informação relativa aos amigos do utilizador.

15. Obter localização do utilizador e seus amigos

Método HTTP	**GET**
URI	{server}/users/{idUser}/checkins
Token como Cookie	Sim
Parâmetros	
***Payload* pedido**	

Resposta (formato)	Sucesso – devolve objeto com os dados do perfil de utilizador.
	{
	"self": {
	"deviceID": "64:BC:0C:7F:AC:7B",
	"timeStamp": "Mar 17, 2016 12:15:06",
	"xPos": 0,
	"yPos": 0,
	"building": "null",
	"floor": "null",
	"room": "null",
	"confidence": 0,
	"raio": -1,

```
                "lat": "0.00000",
                "lon": "0.00000"
        },
        "friends": [{
                "idUser": "4",
                "name": "Adriano Moreira",
                "nickname": "ajcmoreira",
                "idDevice": "cc:fa:00:b4:90:63",
                "location": {
                        "roomID": "76",
                        "room": "LID 3",
                        "floorID": "49",
                        "floor": "1",
                        "buildingID": "22",
                        "building": "Escola de Engenharia",
                        "areaID": "10",
                        "area": "Campus de Azurém",
                        "timeStamp": "Mar 16, 2016 11:09:42",
                        "lat": "0.00000",
                        "lon": "0.00000"
                }
        }]
}
```


Nota: O *timestamp* do self é atualizado sempre que que este pedido é feito. Ao invés de receber de responder com o último que foi enviado.

Descrição: Para obter a localização mais recente do utilizador e dos seus amigos é feito o pedido de obtenção destas informações.

16. Alterar permissão de partilha da localização

Método HTTP	**PUT**
URI	{server}/users/{idUser}/permission
Token como Cookie	Sim
Parâmetros	• *Int permission* – 0 indica que não é permitida a partilha da localização. 1 indica que a partilha da localização é permitida.

***Payload* pedido**	{
	"permission":"0"
	}

Resposta (formato)	Sucesso: 200 OK Insucesso: 400 Bad Request

Descrição: A alteração da permissão da partilha da localização é feita através deste pedido que recebe um parâmetro JSON para configurar a partilha da localização.

17. Remover amigo

Método HTTP	**POST**
URI	{server}/users/{idUser}/unfriend
Token como Cookie	Sim
Parâmetros	• *Int idUser* – identificador do utilizador que se pretende remover.
***Payload* pedido**	{ "idUser":"123" }
Resposta (formato)	Sucesso: 200 OK Insucesso: 400 Bad Request

Descrição: Enviando o identificador do utilizador que se pretende remover da lista de amigos do utilizador é possível remover esse amigo. O identificador do amigo é enviado como um parâmetro JSON.

18. Enviar pedido de amizade a um utilizador

Método HTTP	**POST**
URI	{server}/users/{idUser}/request
Token como Cookie	Sim
Parâmetros	• String email – Email do utilizador que pretende convidar
***Payload* pedido**	{ "email":"ivo_silva11@hotmail.com" }
Resposta (formato)	Sucesso: OK Insucesso: 400 Bad Request

Descrição: Um pedido de amizade é feito a partir deste pedido, como parâmetro é enviado o e-mail do amigo que se pretende convidar.

19. Cancelar pedido de amizade efetuado

Método HTTP	**POST**
URI	{server}/users/{idUser}/cancel
Token como Cookie	Sim
Parâmetros	• *Int idUser* – identificador do utilizador associado ao pedido efetuado
***Payload* pedido**	{ "idUser":"171" }
Resposta (formato)	Sucesso: OK Insucesso: 400 Bad Request

Descrição: Este pedido permite cancelar um pedido de amizade através do envio do identificador do utilizador convidado.

20. Aceitar pedido de amizade

Método HTTP	**POST**
URI	{server}/users/{idUser}/approve
Token como Cookie	Sim
Parâmetros	• *Int idUser* – identificador do utilizador que se pretende aceitar.
***Payload* pedido**	{ "idUser":"171" }
Resposta (formato)	Sucesso: 200 OK Insucesso: 400 Bad Request

Descrição: Quando um utilizador aceita um pedido de amizade este pedido é despoletado. Para que o pedido seja aceite é enviado o identificador do utilizador que fez o convite.

21. Recusar pedido de amizade

Método HTTP	**POST**
URI	{server}/users/{idUser}/deny
Token como Cookie	Sim

Parâmetros	• *Int idUser* – identificador do utilizador que se pretende recusar
***Payload* pedido**	{ "idUser":"171" }
Resposta (formato)	Sucesso: 200 OK Insucesso: 400 Bad Request

Descrição: A rejeição de um pedido de amizade é feita da mesma forma do que no pedido anterior. Neste caso, o pedido recusa o convite efetuado.

22. Reenvio de palavra-chave, gerada automaticamente pelo servidor

Método HTTP	**POST**
URI	{server}/users/new/password
Token como Cookie	Não
Parâmetros	• *String email* – E-mail do utilizador.
***Payload* pedido**	{ "email":"ivo_silva11@hotmail.com" }
Resposta (formato)	Sucesso: OK Insucesso: 400 Bad Request

Descrição: Quando o utilizador pretende receber uma nova *password* este pedido recebe o e-mail do utilizador e faz um *reset* à *password*.

23. Mudar Alcunha

Método HTTP	**PUT**
URI	{server}/users/{idUser}/nickname
Token como Cookie	Sim
Parâmetros	• *String nickname* – *nickname* que se deseja (tem que ser diferente do atual senão o servidor dá erro).
***Payload* pedido**	{ "nickname":"teste" }
Resposta (formato)	Sucesso: OK

	Insucesso: 400 Bad Request

Descrição:. A alteração da alcunha é feita através deste pedido que recebe o novo valor como parâmetro JSON.

24. Mudar Password

Método HTTP	**PUT**
URI	{server}/users/{idUser}/password
Token como Cookie	Sim
Parâmetros	• String currentTokenPassword – md5(token+md5(password)), onde a password, é o valor atual do utilizador • String newPassword –md5(password) nova password que o utilizador deseja.
***Payload* pedido**	{ "currentTokenPassword":"gaOfl", "newPassword":"teste" }
Resposta (formato)	Sucesso: OK Insucesso: 400 Bad Request

NOTA: Caso seja devolvida uma mensagem de aviso semelhante à do pedido de *Login*, é necessário fazer o pedido novamente com os seguintes parâmetros:

- *currentTokenPassword* = "*token+password*"
- *newPassword* = "*newPassword*" (conteúdo às claras)

Descrição: Este pedido permite a alteração da *password* do utilizador. Através dos parâmetros enunciados na tabela é possível alterar este valor na base de dados.

Posicionamento
{server}=http://urano.dsi.uminho.pt/whereatumdev/pe2

1. Obter lista de locais UM

Método HTTP	**GET**
URI	{server}/places/um
Token como Cookie	Sim
Parâmetros	

***Payload* pedido**	
Resposta (formato)	JSON com os dados dos locais da UM

2. Obter lista de locais fora da UM

Método HTTP	**GET**
URI	{server}/places/{coordenadas GPS}/search Exemplo: {server}/places/41.5924797,-8.7857590/search
Token como Cookie	Sim
Parâmetros	
***Payload* pedido**	

Resposta (formato)	{
	"places": [{
	"id": "51d1f929498e1352a72316b6",
	"name": "HOOL Restaurante",
	"location": {
	"address": "Largo da Oliveira",
	"lat": 41.442932653653,
	"lng": -8.2928088075987,
	"distance": 4,
	"cc": "PT",
	"city": "Guimar\u00e3es",
	"state": "Braga",
	"country": "Portugal",
	"formattedAddress": ["Largo da Oliveira", "Guimar\u00e3es", "Portugal"]
	}
	}, {
	"id": "486",
	"name": "Largo da Oliveira",
	"checkinsCount": "1",
	"location": {
	"country": "Portugal",
	"cc": "PT",
	"city": "Guimar\u00e3es",
	"address": "Largo da Oliveira",
	"distance": 9,
	"lat": "41.4430008",
	"lng": "-8.2929325"
	}
	}, {
	"id": "84",
	"name": "Manifestis Probatum",
	"checkinsCount": "2",

```
                "location": {
                        "country": "Portugal",
                        "cc": "PT",
                        "city": "Guimar\u00e3es",
                        "address": "R. Egas Moniz, 57-63",
                        "distance": 97,
                        "lat": "41.4421005",
                        "lng": "-8.2928705"
                }
        }]
}
```


Descrição: Quando se pretende alterar a localização dentro da UM é necessário obter a lista de locais suportados pelo sistema. Este pedido devolve a lista de locais sob a forma de um objeto JSON.

3. Enviar *fingerprint*

Método HTTP	**POST**
URI	{server}/fingerprints
Token como Cookie	Sim
Parâmetros	• *String idUser* – identificador do utilizador associado à *fingerprint.* • *String idDevice* – Endereço MAC da interface Wi-Fi do dispositivo do utilizador. • *Array fingerprint*: *String MAC* – endereço MAC de cada AP; *Int RSSI* – nível de sinal do AP em dBm; *Int channel* – canal associado à frequência de funcionamento do AP; *String SSID* – nome da rede do AP; • *Int motion* – valor que indica se o utilizador está em movimento (opcional). • *String place* – informação relativa ao local (opcional). • *String timestamp* – data e hora da recolha da *fingerprint* (`yyyy-mm-dd HH:mm:ss`).

***Payload* pedido**	{
	`"idUser": "171",`
	`"idDevice": "64:BC:0C:7F:AC:7B",`
	`"timestamp": "2016-03-18 19:37:00",`
	`"motion": "",`

<table>
<tr><td></td><td><pre>
"place": "",
"fingerprint": [{
"mac": "2C:3E:CF:0B:78:BF",
"rssi": "-37",
"ssid": "eduroam",
"channel": 1
}, {
"mac": "2C:3E:CF:0B:78:B4",
"rssi": "-28",
"ssid": "eduroam",
"channel": 1
}, {
"mac": "2C:3E:CF:0B:78:BB",
"rssi": "-94",
"ssid": "",
"channel": 1
}]
}
</pre></td></tr>
<tr><td>Resposta (formato)</td><td>Sucesso: 200 OK
Insucesso: 400 Bad Request</td></tr>
</table>

Descrição: O envio de *fingerprints* para o servidor é feito através deste pedido. Na tabela estão apresentados todos os parâmetros da *fingerprint.*

4. Introduzir local que pertence à Universidade do Minho

Método HTTP	**POST**
URI	{server}/places/um
Token como Cookie	Sim
Parâmetros	• Int idUser – identificador do utilizador. • String idDevice – Endereço MAC da interface Wi-Fi do dispositivo do utilizador. • *Array fingerprint*: *String MAC* – endereço MAC de cada AP; *Int RSSI* – nível de sinal do AP em dBm; *Int channel* – canal associado à frequência de funcionamento do AP; *String SSID* – nome da rede do AP; • *Array location*: *String campus* – nome do campus;

Int campusID – identificador do campus;

String building – nome do edifício;

Int buildingID – identificador do edifício;

String floor – nome do piso;

Int floorID – identificador do piso;

String room – nome do quarto/espaço;

Int roomID – identificador do quarto/espaço;

Long lng – latitude;

Long lat – longitude;

Int location_changed – tem o valor 0 ou 1 dependendo da alteração da localização.

***Payload* pedido**


```
{
"idUser": "171",
"idDevice": "64:BC:0C:7F:AC:7B",
"timestamp": "2016-03-21 09:43:00",
"fingerprint": [{
        "mac": "d4:d7:48:0c:8d:40",
        "rssi": "-70",
        "ssid": "eduroam",
        "channel": 11
}, {
        "mac": "00:3a:98:ef:fa:50",
        "rssi": "-62",
        "ssid": "eduroam",
        "channel": 1
}, {
        "mac": "b4:14:89:d1:b3:8b",
        "rssi": "-95",
        "ssid": "",
        "channel": 36
}],
"location": {
        "campus": "Campus de Azurém",
        "campusID": "1",
        "building": "Bloco A",
        "buildingID": "1",
        "floor": "1",
        "floorID": "1",
        "room": "teste",
        "roomID": "570",
        "lat": "",
        "lng": "",
        "location_changed": "1"
```

	} } **NOTA:** {roomID:"new"} - quando se pretende inserir um novo local não definido até ao momento.
Resposta (formato)	Sucesso: 200 OK Insucesso: 400 Bad Request

Descrição: Quando o utilizador adiciona um novo local associado à UM, é necessário utilizar este pedido para enviar a informação para o servidor. É enviada uma *fingerprint* juntamente com a informação do local.

5. Introduzir local fora da Universidade do Minho

Método HTTP	**POST**
URI	{server}/places
Token como Cookie	Sim
Parâmetros	• *Int idUser* – identificador do utilizador. • *String idDevice* – Endereço MAC da interface Wi-Fi do dispositivo do utilizador. • *Array fingerprint*: *String MAC* – endereço MAC de cada AP; *Int RSSI* – nível de sinal do AP em dBm; *Int channel* – canal associado à frequência de funcionamento do AP; *String SSID* – nome da rede do AP. • *Array location*: *String campus* – nome do campus; *Int campusID* – identificador do campus; *String building* – nome do edifício; *Int buildingID* – identificador do edifício; *Long lng* – latitude; *Long lat* – longitude;

	Int location_changed – tem o valor 0 ou 1 dependendo da alteração da localização.

***Payload* pedido**	{
	"idUser": "171",
	"idDevice": "64:BC:0C:7F:AC:7B",
	"timestamp": "2016-03-21 09:52:00",
	"fingerprint": [{
	"mac": "d4:d7:48:0c:8d:40",
	"rssi": "-58",
	"ssid": "eduroam",
	"channel": 11
	}, {
	"mac": "00:3a:98:ef:fa:50",
	"rssi": "-60",
	"ssid": "eduroam",
	"channel": 1
	}, {
	"mac": "b4:14:89:d1:b3:8b",
	"rssi": "-94",
	"ssid": "",
	"channel": 36
	}],
	"location": {
	"country": "Portugal",
	"city": "Guimarães",
	"address": "Largo da Oliveira",
	"place": "",
	"lat":" 41.442784",
	"lng":", -8.292726",
	"location_changed":"1"
	}
	}

Resposta (formato)	Sucesso: 200 OK Insucesso: 400 Bad Request

Descrição: Quando é introduzido um local fora da UM este pedido permite enviar as informações relativas à localização e uma *fingerprint* que permite associar o local à *fingerprint.*