PM

Wi-Share
Partilha de configurações Wi-Fi

PROJETO DE MESTRADO

Lino Fernandes Oliveira
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

Fevereiro | 2016

Wi-Share
Partilha de configurações Wi-Fi

PROJETO DE MESTRADO

Lino Fernandes Oliveira
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTADORA
Lina Maria Pestana Leão de Brito

CO-ORIENTADOR
Eduardo Miguel Dias Marques

Resumo

A popularidade dos dispositivos móveis tem vindo a aumentar significativamente nos últimos anos e, com isso, surge a necessidade de aceder à Internet nos *smartphones* e *tablets*, quer para fins laborais quer para lazer. Devido às limitações de tráfego nas redes móveis, como 3G ou 4G, as pessoas procuram conectar-se aos pontos de acesso nas suas proximidades para poupar tráfego móvel. Os pontos de acesso também são uma outra forma de se conseguir conectar à Internet no estrangeiro, mesmo quando não se tem disponível um plano de dados móveis.

As soluções existentes, que visam conectar os seus utilizadores à Internet através de pontos de acesso, requerem o pagamento de uma taxa elevada ou violam a privacidade das redes Wi-Fi ao permitir que todos os utilizadores se consigam conectar sem a devida autorização dos proprietários e que consumam tráfego e largura de banda sem quaisquer restrições.

Com este trabalho pretende-se permitir que os proprietários das redes possam limitar os recursos de quem acede às suas redes (tráfego, largura de banda e/ou número de utilizadores conectados) usando apenas uma aplicação *Android* para fazer todo o controlo de acesso e limitação de recursos. Além de limitar os recursos pretende-se possibilitar a interoperabilidade entre pontos de acesso de diferentes plataformas para permitir que utilizadores de diferentes operadores de telecomunicações possam partilhar as suas redes mutuamente. Para se atingir estes objetivos foi desenvolvido um sistema composto por uma aplicação *Android* e um servidor *web*.

O teste da solução foi feito através de testes com utilizadores, identificando-se que os participantes partilharam maioritariamente as suas próprias redes. A maioria dos utilizadores optou por partilhar as suas redes de forma pública (com todos os utilizadores) e limitar o número de utilizadores conectados para salvaguardar o desempenho da sua ligação. Com este trabalho, consegue-se concluir que é possível incentivar os utilizadores a partilhar as suas redes caso estejam presentes mecanismos que consigam manter a privacidade da rede e que lhes consigam dar controlo sobre a partilha.

Palavras-chave

Android

Pontos de acesso

Redes móveis

Wi-Fi

Abstract

The popularity of mobile devices has increased significantly in recent years and, with that, comes the need to gain access to the Internet on *smartphones* and *tablets*, whether for work purposes or for leisure. Because of the limited traffic on mobile networks such as 3G or 4G, people try to connect to access points in their vicinity to save mobile traffic. Access points are also another way to access the Internet from abroad, even when there is no mobile data plan available.

Existing solutions that aim to connect their users to the Internet through access points require the payment of a large fee or violate the privacy of Wi-Fi networks by allowing all users to connect without the permission of the owners and consume traffic and bandwidth without any restrictions.

This work aims to allow access point owners to limit the resources (traffic, bandwidth and/or number of connected users) to whoever connects to their networks using only an *Android* application to manage all access control and resources limitation. In addition to limiting the resources, this work intends to enable interoperability between access points of different platforms to allow users of different telecommunications operators to share their networks with each other. To achieve these goals it was developed a system composed of an *Android* application and a *web* server.

To test the solution user testing was conducted, being identified that participants mostly shared their own networks. Most users have chosen to share their networks publicly (with all users) and limited the number of users connected to safeguard the performance of their connection. With this work it can be concluded that it is possible to encourage users to share their networks if there are mechanisms that can maintain the privacy of the network and are able to give them control over the sharing.

Keywords

Android

Access points

Mobile networks

Wi-Fi

Agradecimentos

Agradeço, em primeiro lugar, à Professora Lina Brito e ao Professor Eduardo Marques por terem aceite a orientação deste trabalho e por toda a sua compreensão e conselhos dados com rumo à conclusão desta importante etapa.

Especiais agradecimentos à minha mãe e madrinha, que sempre me deram força e ânimo quando mais precisava, não esquecendo certamente todo o esforço financeiro que foi feito para possibilitar a conclusão deste último objetivo académico.

Agradeço também a todos os meus colegas e amigos que me acompanharam ao longo de todos estes anos, especialmente ao Rúben Henriques e Eduardo Vasconcelos que sempre me apoiaram incondicionalmente desde o início ao fim desta longa jornada académica.

Um grande obrigado ao André Freitas por me incentivar à realização deste trabalho através das suas ideias inovadoras.

E, finalmente, agradeço a todos aqueles que se voluntariaram e participaram nos testes da aplicação desenvolvida.

Índice

Lista de figuras

Lista de tabelas

Abreviaturas e símbolos

AES – Advanced Encryption Standard

ER – Entidade-Relação

HTTP – Hypertext Transport Protocol

HTTPS – Hypertext Transport Protocol Secure

IETF – Internet Engineering Task Force

IMEI – International Mobile Equipment Identity

JSON – JavaScript Object Notation

JVM – Java Virtual Machine

MMS – Multimedia Message System

NFC – Near Field Communication

ORM – Object Relational Mapping

QR – Quick Responde

SDK – Software Development Kit

SMS – Short Message System

SSID – Service Set Identifier

SSL – Secure Sockets Layer

TLS – Transport Layer Security

XML – Extensible Markup Language

Capítulo 1

Introdução

1.1. Motivação

A popularidade dos dispositivos móveis tem vindo a aumentar significativamente nos últimos anos e, com isso, surge a necessidade de obter acesso à Internet nos *smartphones* e *tablets*, quer para fins laborais quer para lazer. Devido às limitações de tráfego nas redes móveis, como 3G ou 4G, as pessoas procuram conectar-se aos pontos de acesso nas suas proximidades para poupar tráfego móvel. Os pontos de acesso também são uma outra forma de se conseguirem conectar à Internet no estrangeiro, mesmo quando se tem disponível um plano de dados móveis.

Os pontos de acesso Wi-Fi utilizam a norma 802.11 para a criação de redes sem fios e interligar dipositivos móveis com interfaces sem fios (vulgo, *wireless*). Esta norma inclui alguns protocolos de segurança para proteger as comunicações. Tais protocolos de segurança são, nomeadamente, o WEP (*Wired Equivalent Privacy*), introduzido em 1997, e o WPA (*Wi-Fi Protected Access*), introduzido em 2003 com o objetivo de resolver as falhas de segurança presentes no WEP [1].

Embora as pessoas se conectem aos pontos de acesso para poupar tráfego móvel, as velocidades que se conseguem atingir são também superiores em comparação com as velocidades permitidas pelas redes 3G ou 4G. O protocolo 802.11ac usado nos pontos de acesso mais recentes prevê que se obtenha até 6.93Gbps de velocidade de transmissão máxima, enquanto a tecnologia 4G usada nos dispositivos móveis prevê que se atinjam velocidades de pelo menos 100Mbps [2], [3]. Comparando tais especificações com as velocidades reais praticadas, consegue-se verificar que se obtêm velocidades muito inferiores. Um ponto de acesso usando a tecnologia 802.11ac consegue atingir velocidades a rondar os 300Mbps a uma distância de 33 metros do dispositivo recetor, enquanto a tecnologia 4G, dependendo da zona e do

operador, consegue atingir velocidades médias a rondar os 16Mbps e máximas a rondar os 60Mbps, muito inferior aos 100Mbps pretendidos [2], [4].

Tanto para poupar tráfego móvel como para obter velocidades superiores, as pessoas procuram se conectar aos pontos de acesso disponíveis nas suas proximidades. Exemplo disso são os ambientes domésticos, nos quais é frequente ser pedido acesso à rede Wi-Fi doméstica quando se recebem pessoas em casa. Um problema que aqui se põe é a dificuldade de dar acesso à rede. Muitas das pessoas não têm as credenciais da rede na ponta da língua e têm de consultar as definições da rede no seu computador ou deslocar-se até ao ponto de acesso para fornecer as definições de segurança que estão presentes na etiqueta. Outro problema que se identifica é a vontade de não divulgar a senha da rede Wi-Fi para preservar a sua privacidade.

Outro exemplo é quando um grupo de amigos se reúne num café ou restaurante. Aqui o problema aparece quando é feita a tentativa de se conectar a uma rede Wi-Fi nas proximidades e se depara com redes privadas às quais não se tem forma de obter as credenciais. Algumas aplicações existentes visam resolver este problema ao incentivar as pessoas a partilharem as credenciais das redes privadas a que vão tendo acesso no seu dia-a-dia para que, com a ajuda de todos, se crie um mapa com todas estas redes partilhadas e se consiga aceder gratuitamente em praticamente todo o lado.

Ao resolver um problema, estas aplicações criam um outro; estas estão a quebrar a privacidade das redes partilhadas ao divulgar e comprometer as suas credenciais de segurança. Esta violação da privacidade das redes permite que todos os utilizadores de tais aplicações tenham acesso às redes partilhadas a qualquer momento e sem o consentimento dos proprietários. Existe, ainda, o risco de um número elevado de utilizadores se conectarem a uma rede e consumirem uma elevada quantidade de largura de banda, degradando severamente o desempenho da ligação do proprietário.

Do ponto de vista de quem está a aceder às redes partilhadas, não existem quaisquer adversidades, basta se conectar e começar a navegar na Internet sem gastar tráfego móvel. Do ponto de vista do dono da rede, esta partilha apenas lhe trás prejuízo, uma vez que quem está a aceder tem total acesso à largura de banda e tráfego da sua rede, podendo degradar o desempenho da ligação caso se esgote a largura de banda disponível.

Assim, identifica-se a necessidade de simplificar o processo de partilhar as credenciais de redes Wi-Fi com terceiros, simplificar o seu processo de configuração e encontrar um modelo de partilha e acesso às redes privadas que consiga manter a sua privacidade e controlar o tráfego e a largura de banda utilizados por terceiros.

As necessidades acima mencionadas são também aplicáveis a empresas que queiram fornecer acesso à Internet aos seus clientes através de pontos de acesso nos seus estabelecimentos comerciais. Na área da restauração, por exemplo, ao utilizar um sistema que consiga conectar os clientes mal estes cheguem ao estabelecimento seria muito mais cómodo tanto para os clientes como para os funcionários, que não precisariam de estar constantemente a informar todos os clientes que chegam ao estabelecimento acerca das configurações de segurança da rede. Assim, as empresas não só simplificariam o processo de conexão dos seus clientes, como também salvaguardariam a qualidade da ligação ao limitar o desempenho de cada cliente para impedir que uns consumam mais largura de banda que outros, por exemplo.

Por estas razões, surgiu a ideia de criar uma aplicação móvel que permitisse aos seus utilizadores partilhar as configurações das suas redes entre si, simplificando o processo de conexão e mantendo a privacidade das redes partilhadas.

1.2. Objetivos

Para se conseguir responder às necessidades de partilha de redes privadas identificadas anteriormente, este projeto vem desenvolver uma aplicação *Android* com os seguintes objetivos:

- Partilhar configurações de redes Wi-Fi com terceiros
- Partilhar as configurações *offline*
- Simplificar o processo de partilha das configurações
- Simplificar o processo de conexão às redes Wi-Fi
- Ocultar as credenciais a terceiros
- Limitar largura de banda a terceiros
- Limitar tráfego consumido por terceiros
- Limitar utilizadores conectados a uma rede

Os objetivos propostos visam contribuir para que se consiga dar acesso às redes privadas a alguém sem serem reveladas diretamente as informações de segurança, conseguindo-se assim manter a privacidade da rede. Outra contribuição é a de salvaguardar o bom desempenho das redes às quais se fez a partilha das configurações, dando-se controlo aos proprietários sobre os parâmetros de desempenho da sua rede.

Para que se consiga cumprir tais objetivos levanta-se uma série de desafios os quais se terá de ultrapassar. Tais desafios estão associados ao facto de não se conseguir controlar os pontos de acesso aos quais os utilizadores se vão conectar. Esta barreira dificulta a limitação do tráfego, largura de banda e número de utilizadores conectados a uma determinada rede, sendo necessário encontrar mecanismos alternativos que consigam efetivamente controlar tais parâmetros. A partilha de configurações *offline* será também um desafio interessante, uma vez que terão de ser encontradas formas de conseguir utilizar os mecanismos do *smartphone* do utilizador para transmitir os dados da rede entre o proprietário e o recetor da partilha de forma relativamente simples e sem conexão à Internet. Outro desafio será a gestão da propriedade de uma rede, uma vez que qualquer utilizador que saiba as configurações de segurança de uma rede pode adicioná-la ao sistema como sendo sua. Para corrigir tal situação, terão de ser fornecidos mecanismos para possibilitar que o real dono da rede consiga tomar controlo sobre a rede caso tal venha a acontecer.

Outro objetivo, além do desenvolvimento da aplicação, é a realização de testes com utilizadores de forma a obter *feedback* e tentar perceber se a aplicação desenvolvida e as suas funcionalidades são relevantes para quem tem necessidade de partilhar as configurações da sua rede Wi-Fi com terceiros.

1.3. Estrutura

Este documento é composto por 7 outros capítulos que abordam o desenvolvimento do sistema, desde a fase de análise das soluções existentes até à análise dos resultados obtidos. Descrevem-se os demais capítulos como se segue:

Capítulo 2: Trabalhos relacionados – é feito um levantamento e comparação entre as soluções existentes. Identificam-se as vantagens e desvantagens de cada solução.

Capítulo 3: Especificação – consolidam-se as ideias iniciais com os pontos pertinentes identificados no capítulo anterior e é feito um breve questionário a possíveis utilizadores. Com isto é criada uma especificação do sistema a desenvolver.

Capítulo 4: Implementação – são estudadas as tecnologias a utilizar e são descritos os vários passos da implementação o sistema, assim como as demais ferramentas e tecnologias utilizadas ao longo deste processo.

Capítulo 5: Testes e resultados – aqui descrevem-se os testes efetuados ao sistema assim como os resultados obtidos.

Capítulo 6: Conclusões e trabalho futuro – apresentam-se então as reflexões acerca das várias etapas do projeto e tiram-se conclusões a partir dos resultados obtidos. Com isto apresentam-se propostas para uma possível evolução do projeto.

Capítulo 2

Trabalhos relacionados

2.1. Introdução

No capítulo anterior contextualizou-se e apresentou-se a motivação do presente projeto, identificando a necessidade de se conseguir conectar aos pontos de acesso através de dispositivos móveis. Foi também identificada a necessidade de melhorar a facilidade de partilha das configurações das redes Wi-Fi privadas e a manutenção da privacidade destas.

Devido à necessidade de obter acesso à Internet através dos dispositivos móveis, existem várias soluções que visam dar acesso à Internet aos seus utilizadores através de pontos de acesso espalhados por todo território. De todas estas, identificam-se dois grupos distintos: sistemas proprietários e sistemas comunitários.

A secção 2.2 foca-se nos sistemas proprietários, que são disponibilizados por empresas que utilizam os pontos de acesso dos seus clientes ou colocam pontos de acesso dedicados nas áreas com mais afluência para fornecer Internet a quem se queira conectar.

A secção 2.3 aborda os sistemas comunitários, que são aplicações gratuitas desenvolvidas por entidades independentes que visam facilitar o acesso à Internet aos seus utilizadores através dos pontos de acesso privados existentes em zonas urbanas. Estes sistemas baseiam-se na comunidade, ou seja, nos seus utilizadores, para partilharem as credenciais das redes Wi-Fi privadas para que outros utilizadores se possam conectar posteriormente de forma gratuita.

Na última secção faz-se uma comparação entre os sistemas mencionados anteriormente de forma a perceber qual o modelo mais vantajoso para o utilizador final.

2.2. Sistemas proprietários

Os sistemas proprietários que visam dar acesso à internet aos seus utilizadores através de pontos de acesso são, normalmente, disponibilizados por operadores locais de telecomunicações, estando disponíveis em pontos de interesse com uma certa afluência de pessoas, como por exemplo, em aeroportos, em centros comerciais e em espaços públicos urbanos.

Para usar estes sistemas é necessário se conectar ao ponto de acesso do operador, efetuar o registo e adquirir um voucher com um determinado *plafond* de utilização (ex. 30 min). Para efetuar o registo é necessário abrir um *browser* de Internet onde será apresentada uma página do operador com as instruções necessárias a seguir. Concluído o registo é necessário selecionar um *voucher* com um *plafond* a gosto e efetuar o pagamento do voucher selecionado. Uma vez na posse de um *plafond* positivo o utilizador passa a ter acesso à Internet sempre que se conecte a um ponto de acesso da plataforma que se registou. Alguns operadores locais como a MEO e a NOS têm já aplicações para as plataformas móveis que visam automatizar o processo de conexão e autenticação dos utilizadores assim que esteja disponível um ponto de acesso da sua plataforma nas proximidades.

Martin Varsavsky fundou a empresa FON com o objetivo de criar uma comunidade global de pessoas que partilhem e usem os seus pontos de acesso mutuamente de forma a obterem acesso gratuito à Internet fora de casa. Este modelo torna-se possível através da utilização de pontos de acesso fornecidos pela própria FON, que correm um *software* baseado na distribuição Linux OpenWRT. Este *software* permite que se difundam duas redes Wi-Fi, uma pública e outra privada, ficando a rede pública para dar acesso a terceiros e a privada para uso pessoal [5]. A FON indica também que o tráfego entre a rede pública e privada é separado e que é dada prioridade ao tráfego do proprietário do ponto de acesso para que o desempenho da sua conexão não seja afetado [6]. Para monetizar este modelo de partilha dos pontos de acesso, aos membros FON são atribuídas três categorias distintas:

- Linus: um utilizador registado que partilha a sua largura de banda com a comunidade FON em troca de acesso gratuito a outros pontos de acesso FON [5], [7].

- Bill: um utilizador registado que partilha a sua largura de banda em troca de uma compensação monetária. Para aceder a outros pontos de acesso FON terá também de efetuar uma compensação monetária [5], [7].

- Alien: um utilizador registado que não partilha a sua largura de banda e que se conecta aos pontos de acesso dos Linus e Bills da comunidade FON através da compra de *vouchers* [5], [7].

A FON além de fornecer os seus próprios equipamentos também estabelece várias parcerias internacionais com os operadores de comunicações a nível mundial, incentivando os seus parceiros a incorporar a tecnologia da FON nos seus pontos de acesso. As vantagens desta parceria advêm do facto dos clientes dos parceiros da FON se tornarem também elegíveis a usufruir do modelo de acesso gratuito mencionado anteriormente. Este acesso gratuito é também válido no estrangeiro, ficando assim disponíveis mais de 17 milhões de pontos de acesso a nível mundial pelos quais se pode aceder à Internet de forma gratuita [8], [9].

Uma abordagem um pouco diferente é a da MEO, embora sirva o mesmo propósito. Através dos seus pontos de acesso a MEO atribui tráfego ilimitado aos clientes M5O e de banda larga, deixando os restantes clientes com um *plafond* de 200MB mensais. Para dar acesso aos seus utilizadores no estrangeiro, a MEO associou-se com os demais operadores dos vários países em todo o mundo em vez de se associar com uma entidade global como a FON. Uma das consequências é que a MEO consegue fornecer acesso gratuito no estrangeiro apenas em alguns países [10]–[12].

Os problemas destas soluções são o seu eventual custo de acesso e o facto de serem sistemas fechados, não podendo os clientes da NOS aceder aos pontos de acesso da MEO e vice-versa.

Contudo, existem sistemas que visam aproveitar todos os pontos de acesso de todas as plataformas a nível mundial de forma gratuita. Tais propostas são abordadas no ponto seguinte.

2.3. Sistemas comunitários

O principal objetivo destas soluções é dar acesso gratuito à Internet aos seus utilizadores para onde quer que estes vão.

Estes sistemas são geralmente aplicações para *smartphones* que incentivam os seus utilizadores a partilharem as credenciais dos pontos de acesso Wi-Fi a que vão tendo acesso no seu dia-a-dia, sejam estes de cafés, restaurantes ou residências.

É óbvio que ao seguir um modelo deste género, os pontos de acesso partilhados pelos utilizadores são usualmente de pessoas alheias, ou seja, os utilizadores estão a partilhar senhas de pontos de acesso que não são seus por direito.

Para aceder a um ponto de acesso muitas destas aplicações fornecem as credenciais das redes em texto legível para que os utilizadores as usem para se conectar ao ponto de acesso pretendido.

Existem várias aplicações que funcionam sob este modelo, analisando-se de seguida algumas delas.

2.3.1. Aplicações existentes

Existem diversas aplicações disponíveis na *Play Store*, fornecedora oficial de aplicações da plataforma *Android*, que operam segundo este modelo de partilha.

Uma vez que existem inúmeras aplicações que operam segundo este modelo, escolheram-se apenas as que estavam providas com um nível mais elevado tanto em termos de número de transferências como de satisfação dos utilizadores. Como existem também aplicações populares com várias réplicas, como o *osmino Wi-Fi*, deixaram-se tais réplicas fora dos testes. As aplicações testadas foram: *Mandic magiC*, *WiFi Free*, *osmino Wi-Fi*, *WiFi Map*, *Instabridge*, *WifiPass*, *Wi-Fi Finder*, *Friendly Wifi* e *Hotspotio* [13]–[21].

Usualmente, para se estabelecer conexão a um ponto de acesso doméstico que esteja protegido, apenas é necessário introduzir a senha correta quando se tenta ligar à rede Wi-Fi. O modelo de partilha de todas estas aplicações consiste apenas nisso, fazendo com que os utilizadores partilhem o SSID (nome) da rede, a senha e a localização da mesma. Essas informações são depois disponibilizadas a todos os utilizadores através de aplicações *Android* para que estes se possam conectar às redes Wi-Fi partilhadas pela comunidade.

Para que se consigam identificar as funcionalidades oferecidas por cada uma destas aplicações, são apresentadas, na tabela 1, todas as funcionalidades oferecidas pelas aplicações analisadas e, na tabela 2, quais destas funcionalidades são disponibilizadas por cada aplicação.

Tabela 1 - Funcionalidades

N.º	*Funcionalidade*
1	Mapa das redes
2	Senhas ocultas
3	Seleção da melhor rede
4	Partilha pública
5	Partilha seletiva

6	Acesso *offline*
7	Testes de desempenho
8	Conexão automática

Tabela 2 - Comparação de funcionalidades

Aplicação/Funcionalidade	1	2	3	4	5	6	7	8
Mandic magiC	X			X		X		
WiFi Free	X			X		X		
Osmino Wi-Fi	X	X		X		X	X	X
WiFi Map	X			X		X		
Instabridge	X	X		X		X	X	X
WifiPass	X	X		X		X	X	X
WiFi Finder	X	X	X	X		X		X
Hotspotio	X	X		X	X			

As funcionalidades mais populares nestas aplicações são as de apresentar as redes partilhadas num mapa, ocultar as senhas, partilhar redes Wi-Fi publicamente e possibilitar que se aceda às redes partilhadas quando se está *offline* (sem acesso à Internet no dispositivo).

A visualização da localização das redes com recurso a um mapa permite que as pessoas saibam em que zonas estão disponíveis as redes partilhadas e possam decidir onde tomar um café com base nisso, por exemplo.

A ocultação das senhas permite manter a privacidade da rede, impossibilitando que outros utilizadores mal-intencionados partilhem as credenciais noutras aplicações concorrentes. As aplicações que ocultam as senhas das redes, excluindo o *Hotspotio*, oferecem também a funcionalidade de conexão automática às redes nas proximidades, deixando de ser necessário que o utilizador tenha de fazer a configuração manual das redes.

Ao fazer a partilha pública de uma rede, esta fica acessível a todos os utilizadores da aplicação. Esta funcionalidade tem o objetivo incentivar os utilizadores a partilhar as redes Wi-Fi com todos os utilizadores da aplicação na esperança de que os restantes também o façam, permitindo que se crie uma área mais alargada de redes partilhadas para ser possível obter acesso à Internet para onde quer que se vá.

A única aplicação que permite partilhar uma rede seletivamente com outros utilizadores é o *Hotspotio*. Esta funcionalidade é útil se quer partilhar uma rede com alguém mas não se deseja dar acesso a todos os utilizadores. O *Hotspotio* usa os amigos do *Facebook* para efetuar esta partilha seletiva. A desvantagem desta aplicação é a necessidade de estar ligado à Internet para se poder conectar a uma rede Wi-Fi partilhada connosco.

A maioria das aplicações permitem que se consiga conectar às redes partilhadas mesmo que estejamos sem ligação à Internet no dispositivo, sendo as redes partilhadas nas proximidades descarregadas para o dispositivo na primeira execução das aplicações. Esta é uma funcionalidade importante porque se o objetivo é aceder às redes Wi-Fi porque não se tem acesso à Internet, não faz sentido ser necessário estar ligado à Internet para conseguir aceder a uma rede Wi-Fi.

Apenas algumas aplicações permitem que se façam testes de desempenho, como largura de banda e tempo de resposta. Estas informações são importantes para que, na presença de várias redes no mesmo local, se consiga escolher a rede com melhor desempenho de forma a obter a melhor conexão possível. Curiosamente, a única aplicação que sugere a rede com melhor desempenho a que se deve estabelecer conexão, o *WiFi Finder*, não faz uso de informações de largura de banda nem tempo de resposta, analisando apenas a o sinal das redes disponíveis.

Além das aplicações mencionadas anteriormente existem outras que visam uma partilha mais pessoal das configurações de segurança das redes Wi-Fi utilizando *tags* NFC ou códigos QR. Uma *tag* NFC é essencialmente uma etiqueta que consegue armazenar uma pequena quantidade de informação que pode ser lida por outros dispositivos NFC, como por exemplo um *smartphone*. Um código QR é um código de barras bidimensional que consegue representar informação textual e binária. No contexto das redes Wi-Fi, os códigos QR são usados para armazenar as configurações de segurança de uma rede para que ao serem lidos por uma aplicação apropriada esta se consiga conectar à rede. Uma aplicação que faz uso destas duas tecnologias para partilhar as configurações de uma rede Wi-Fi é o *InstaWifi* [22].

Nesta aplicação, para se partilhar uma rede através de uma *tag* NFC é requerido que o utilizador introduza os dados da rede a partilhar e que aproxime uma *tag* NFC ao seu *smartphone* para que esta seja devidamente configurada

pela aplicação. Uma vez configurada a *tag* NFC, para se estabelecer a conexão basta aproximar a *tag* NFC com um dispositivo que utilize a tecnologia NFC e que tenha a aplicação *InstaWifi* instalada, sendo a configuração e conexão à rede efetuada automaticamente.

Para se partilhar uma rede através de um código QR é necessário introduzir as informações da rede manualmente num formulário e a aplicação irá construir automaticamente o código QR com base nessas informações, possibilitando a sua partilha através de uma imagem que pode ser enviada por correio eletrónico, *Dropbox*, *Facebook*, ou outro método à escolha. Para efetuar a conexão basta fazer a leitura da imagem (código QR) através de uma aplicação apropriada (ex. *Barcode Scanner*) e a rede será configurada e estabelecida a conexão de forma automática.

Apresentadas estas três aproximações diferentes ao problema, faz-se no ponto seguinte uma comparação entre eles de forma a identificar qual o mais vantajoso para o utilizador final.

2.4. Comparação

Temos, então, três sistemas que abordam o problema de formas completamente diferentes.

Os sistemas proprietários operam segundo um modelo que obriga os utilizadores a terem de pagar uma taxa para obter acesso às redes Wi-Fi da sua plataforma. Estes sistemas também permitem que se aceda gratuitamente a outros pontos de acesso da plataforma caso também se partilhe o nosso ponto de acesso.

Os sistemas comunitários possibilitam, essencialmente, que se partilhem as redes Wi-Fi dos pontos de acesso com todos os utilizadores de forma a incentivar que se partilhe o maior número possível de redes Wi-Fi se conseguir obter acesso à Internet para onde quer que se vá. Existe também uma vertente mais pessoal destas aplicações gratuitas, nomeadamente o *InstaWifi*, que se foca numa partilha mais pessoal usando *tags* NFC e códigos QR.

As vantagens dos sistemas proprietários são a sua disponibilidade em locais públicos, a priorização do tráfego do dono da rede (no caso da FON) e a possibilidade de o acesso se poder tornar gratuito caso se faça também a partilha do ponto de acesso pessoal. As desvantagens são o custo e a falta de interoperabilidade entre as diferentes plataformas, impossibilitando que clientes de diferentes operadoras se consigam conectar mutuamente aos pontos de acesso partilhados.

Os sistemas comunitários têm como vantagem a simplicidade de partilhar e aceder às redes privadas de forma gratuita através da disponibilização das credenciais necessárias para se efetuar a conexão. A desvantagem é que, sem os mecanismos de priorização de tráfego presentes na plataforma da FON, o desempenho da ligação do proprietário pode ser afetado. Outra desvantagem é também a falta de controlo que o proprietário tem sobre a partilha, não tendo este controlo algum sobre quem pode aceder, uma vez que nesta abordagem as credenciais são partilhadas com todos os utilizadores da aplicação.

Ainda dentro dos sistemas comunitários, o *InstaWifi*, que se foca em métodos de partilha mais pessoais como *tags* NFC e códigos QR, tem como vantagem a simplicidade de como é feita a partilha e o acesso das configurações de rede. Uma vez configurada uma *tag* NFC, basta aproximá-la do dispositivo com o qual se quer partilhar a rede e a conexão é automaticamente estabelecida. Uma das desvantagens é que, para se conseguir partilhar as configurações de uma rede através de uma *tag* NFC, o recetor tem de possuir um telemóvel com a tecnologia NFC. A partilha através de um código QR é também muito versátil podendo este ser enviado através de correio eletrónico, Bluetooth ou até mesmo MMS. Uma desvantagem de usar códigos QR é necessidade de uma outra aplicação para fazer a sua leitura e permitir a conexão à rede Wi-Fi. Outra desvantagem, e à semelhança do que foi dito anteriormente, o *InstaWifi* também não possui quaisquer mecanismos que consigam priorizar o tráfego do proprietário da rede.

Tendo em conta estes aspetos, torna-se necessário identificar qual das soluções é a melhor. Os sistemas proprietários pecam pelo custo elevado. Por exemplo, o serviço MEO WiFi oferece um *voucher* de 2 euros com 30 minutos de *plafond*, cerca de 7 cêntimos por minuto. Isto não torna muito apelativo, por exemplo, ir tomar café, pagar o café mais 2 euros pela taxa do Wi-Fi.

Os sistemas comunitários gratuitos pecam pela falta de controlo sobre a rede partilhada. As redes, ao serem partilhadas, todos conseguem ter acesso sem quaisquer limitações. Pode-se assim afirmar que apenas quem beneficia é quem acede, trazendo prejuízo para o proprietário da ligação partilhada. É claro que, se todos fizerem o mesmo e partilharem a sua ligação, acresce o benefício de uma maior cobertura. Mas, é claro que sem restrições nem priorização de tráfego, quem se encontra nos locais mais movimentados e partilha o seu ponto de acesso sairá obviamente mais prejudicado em termos de desempenho.

A vertente mais pessoal dos sistemas gratuitos, representada pelo *InstaWifi*, embora não tenha quaisquer limitações nem priorização de tráfego, não é tão agravante uma vez que a partilha feita através de *tags* NFC e códigos QR é uma abordagem mais pessoal e seletiva. Mesmo assim, para se conseguir partilhar as configurações de uma rede através de uma *tag* NFC o recetor tem de possuir um telemóvel com a tecnologia NFC. A alternativa às *tags* NFC são

os códigos QR, mas estes requerem que se possua uma outra aplicação para efetuar a sua leitura e assim conseguir efetuar a conexão à rede. Assim, teríamos de andar sempre com duas aplicações instaladas para partilhar e receber configurações de amigos, o que não se torna muito prático.

Em suma, de todos os sistemas analisados, a solução apresentada pela FON é a mais vantajosa para os seus utilizadores, uma vez que estes ao fazerem a partilha do seu ponto de acesso passam a poder aceder de forma gratuita a todos os pontos de acesso partilhados mundialmente na plataforma da FON. A sua vasta cobertura, a possibilidade do proprietário cancelar a partilha e a priorização do tráfego da ligação pessoal para que o desempenho da ligação não seja afetado são os pontos decisivos que dão vantagem à FON sobre as outras soluções analisadas.

2.5. Conclusões

Neste capítulo, identificaram-se e analisaram-se os diversos tipos de sistemas que, como o sistema proposto neste projeto, visam conectar os seus utilizadores à Internet através de pontos de acesso.

Com essa análise foram identificadas várias vantagens e desvantagens destes sistemas, efetuando-se uma comparação entre eles. Nessa comparação conclui-se que a solução da FON é a mais vantajosa para os seus utilizadores, uma vez que o tráfego da ligação pessoal do proprietário é priorizado e é concedido acesso gratuito aos membros a nível internacional com uma cobertura muito aliciante.

A solução ideal para os utilizadores seria aquela que possibilitasse a partilha e acesso às redes Wi-Fi de forma gratuita e que preservasse o desempenho das redes partilhadas à semelhança da solução da FON, embora abolindo a necessidade de se usar *hardware* proprietário de forma a permitir a interoperabilidade entre pontos de acesso de diferentes operadores de telecomunicações.

No próximo capítulo pretende-se identificar quais as funcionalidades pertinentes a incluir no sistema a ser desenvolvido, especificando-se o sistema com recurso a casos de uso com base na metodologia apresentada por Ivar Jacobson, denominada de Use Case 2.0 [23].

Capítulo 3

Especificação

3.1. Introdução

No capítulo anterior analisaram-se os diversos tipos de aplicações existentes e fez-se uma comparação entre elas com o intuito de identificar qual o modelo mais vantajoso para o utilizador final. Concluiu-se que a solução da FON consegue ser a mais vantajosa dado que esta consegue priorizar o tráfego dos proprietários das ligações e possibilita o acesso gratuito, embora apenas a pontos de acesso da sua plataforma.

Passa-se agora neste capítulo à especificação do sistema proposto para este projeto de mestrado tendo em conta os objetivos definidos no capítulo 1. Definem-se quais as funcionalidades a serem implementadas e exploram-se tais funcionalidades através de casos de uso. Estes casos de uso serão abordados com base na metodologia Use Case 2.0 sugerida por Ivar Jacobson [23], definindo-se também as restrições com que teremos de lidar na implementação do projeto.

Com base nos casos de uso e nas restrições do sistema é então definida a arquitetura do sistema, que tem de ser capaz de suportar as demais funcionalidades estipuladas e lidar com as restrições impostas.

Assim, começa-se por estudar no próximo ponto quais as funcionalidades que deverão estar presentes no sistema a desenvolver.

3.2. Funcionalidades

Através de um *brainstorming* inicial e com base nas análises anteriores, surgiram algumas ideias no que toca a possíveis funcionalidades a incluir na aplicação. Tais funcionalidades são:

- Gerir redes Wi-Fi
    - Adicionar redes à aplicação para serem partilhadas
    - Editar informações da rede
    - Dar/remover acesso à rede
- Cópia de segurança das redes
- Sincronização com todos os dispositivos do utilizador
- Partilhar redes
    - Com utilizadores específicos
    - Publicamente (com todos os utilizadores)
- Partilha da ligação móvel 3G/4G
- Definir restrições de partilha
    - Limite de tráfego
    - Limite de largura de banda.
    - Limite de utilizadores conectados em simultâneo
- Autenticação transparente ao utilizador
    - Não é necessário fornecer credenciais para se conectar a redes partilhadas
- Conexão automática
- Deteção inteligente do alcance da ligação
- Testar a desempenho da rede
- Informações sobre a rede
- *Chat* com quem está conectado nas proximidades
- Ver quem se conectou mais vezes a si e a quem se você se conectou mais vezes.

Para uma melhor compreensão, descrevem-se no ponto seguinte cada uma destas funcionalidades em mais detalhe.

3.2.1. Descrição

Começando pela gestão, adicionar redes, editar e garantir/remover acesso a outros utilizadores, estas são as operações essenciais que um

utilizador tem de ter ao seu dispor para conseguir partilhar efetivamente a sua rede Wi-Fi.

A cópia de segurança serve para garantir que as redes adicionadas pelo utilizador e que, eventualmente, serão partilhadas, não são perdidas caso haja algum contratempo no dispositivo do utilizador que leve à perda de dados.

A sincronização entre os dispositivos dá a possibilidade ao utilizador de manter a sua experiência de utilização consistente, seja qual for o dispositivo que esteja a usar. Isto significa que todas as suas redes e definições serão consistentes em todos os seus dispositivos.

Partilhar uma rede significa dar a possibilidade a uma determinada pessoa de se conectar a uma rede Wi-Fi. Esta partilha poderá ser feita especificamente com outros utilizadores do sistema ou então com todos estes. Uma das intenções da partilha é o ocultamento das credenciais da rede partilhada a quem se for conectar, garantindo a privacidade da rede.

Normalmente, a partilha da ligação 3G/4G é feita por outras aplicações, através da criação de um ponto de acesso Wi-Fi no dispositivo do utilizador. A ideia seria também possibilitar a partilha deste ponto de acesso.

De modo a dar mais confiança a quem vai partilhar as suas redes Wi-Fi, é sugerido a imposição de restrições de desempenho a terceiros que acedam às redes partilhadas, isto para que o desempenho da ligação partilhada não seja severamente afetada. Tais restrições podem ser de limitação da largura de banda disponível, tráfego diário e/ou máximo de utilizadores conectados à rede num dado instante.

Para tornar mais prático o acesso às redes partilhadas, pretende-se tornar a autenticação e posterior conexão um processo transparente ao utilizador, sem ser necessário que o utilizador introduza manualmente as credenciais para se conectar às redes.

Para os utilizadores conseguirem estar sempre conectados às demais redes partilhadas, é sugerida a funcionalidade de conexão automática. Esta visa que a conexão seja estabelecida sempre que esteja disponível uma rede partilhada no alcance do dispositivo.

É sugerida também a deteção inteligente do alcance da ligação, para quando o utilizador estiver a sair do alcance da corrente ligação e entrar no alcance de outra, o sistema se conectar automaticamente à segunda. Esta funcionalidade servirá para maximizar o tempo que o utilizador está conectado.

Um fator essencial a ter em conta quando temos que decidir a que rede nos conectar quando múltiplas estão disponíveis é o seu desempenho. Para ser possível fazer uma escolha acertada, sugere-se que sejam feitos testes às

ligações de modo a ser possível ter noção de qual o nível de desempenho que cada uma oferece aos utilizadores.

Em conformidade com o objetivo dos testes de desempenho acima referidos, é proposto que se dê a conhecer tais resultados ao utilizador. A sua utilidade é orientada aos utilizadores que se queiram conectar manualmente a uma rede com base nestes parâmetros.

Correntemente, a vertente social tem-se revelado um fator importante em muitas aplicações, incentivando o seu uso. Deste modo, como temos uma aplicação orientada aos pontos de acesso Wi-Fi, prevê-se que os utilizadores tenham o Wi-Fi quase sempre ligado. Com isto, abre-se a possibilidade de criar um *chat* com outros utilizadores que estejam ligados nas proximidades.

Outra funcionalidade sugerida é a possibilidade dos utilizadores poderem ver quem se conectou mais vezes às suas redes e a quem estes se conectaram mais vezes. Tal funcionalidade servirá também de certa forma como uma componente social, permitindo também aos proprietários controlar quem acede às suas redes.

3.2.2. Inquérito sobre funcionalidades

De modo a avaliar se as funcionalidades propostas eram do interesse dos utilizadores e quais as suas preferidas, foi realizado um inquérito aos membros da Universidade da Madeira.

O objetivo deste inquérito era angariar o maior número de respostas possível de modo a obter um *feedback* razoável. Por isso, tentou-se manter o formulário o mais simples e breve possível para cativar os inquiridos a responder por completo ao inquérito. O público-alvo deste questionário constituiu num total de 167 participantes, todos alunos da Universidade da Madeira, 62 do ramo da engenharia e 105 dos restantes departamentos.

No início deste questionário, foi dada uma breve descrição do propósito da aplicação, em que a primeira questão era se haveria interesse neste tipo de aplicações. Caso a resposta fosse afirmativa, eram mostradas as funcionalidades com uma breve descrição e pedido para as classificar de 1 a 4, onde 1 era pouco interessante e 4 muito interessante. Caso a resposta fosse negativa, era questionado sobre qual a razão de tal desinteresse, sendo as possíveis respostas:

- Razões de privacidade
- Não vejo qual a sua utilidade
- Outro

3.2.2.1. Análise

Pode-se considerar que os resultados foram interessantes, sendo que algumas das funcionalidades que à partida se pensava que não iriam suscitar tanto interesse por parte dos inquiridos até obtiveram um resultado significativo.

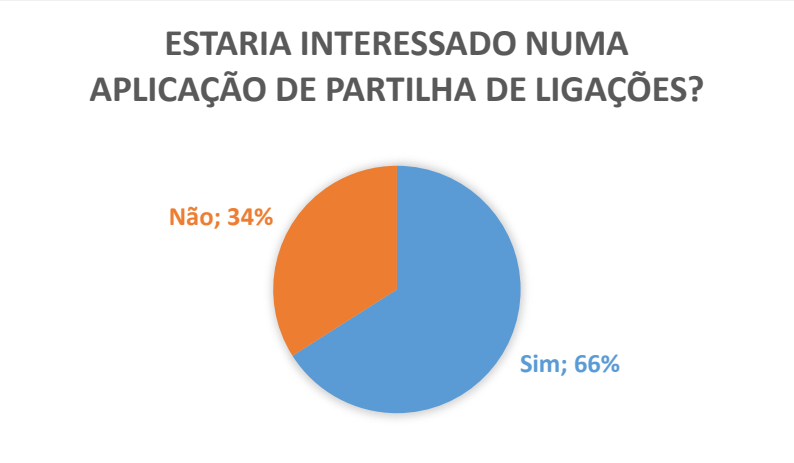

Figura 1 – Interesse dos inquiridos

A figura 1 mostra os resultados à primeira pergunta, revelando que existe de facto um certo interesse por parte dos inquiridos neste tipo de aplicações, uma vez que a vasta maioria (66%) demonstrou interesse e apenas 34% não partilham dessa opinião.

De modo a obter uma classificação para as funcionalidades com base nos resultados do inquérito, foi produzido um gráfico que inclui a pontuação obtida nos demais níveis de interesse. Estas pontuações foram feitas com base no número de respostas dos inquiridos num dado nível de interesse e no valor desse mesmo nível de interesse. A fórmula usada para o cálculo de pontuação foi, então, a seguinte: $P = I \times N$, em que P representa a pontuação, I o nível de interesse (entre 1 e 4) e N o número de inquiridos que demonstrou esse mesmo nível de interesse.

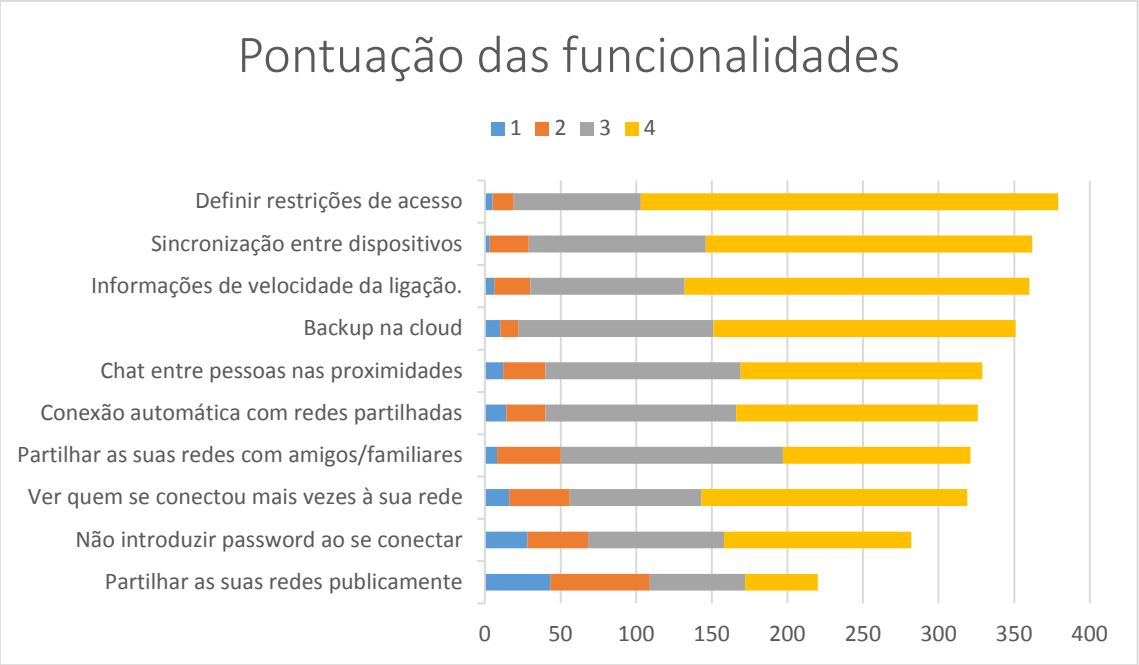

Figura 2- Pontuação de funcionalidades

Consoante nos mostra o gráfico da figura 2, as funcionalidades que suscitaram mais interesse aos inquiridos foram as restrições de acesso, sincronização entre dispositivos e as informações de velocidade da ligação, seguidas pela cópia de segurança.

Uma das funcionalidades que não era esperada que obtivesse um resultado tão significativo foi o *chat* entre pessoas nas proximidades, o que revela de certa forma que a existência de uma componente social poderá cativar as pessoas para a continuarem a utilizar a aplicação.

As restantes funcionalidades já não suscitaram tanto interesse aos inquiridos. Mesmo assim, as funcionalidades de conexão automática, partilha com amigos/familiares e a possibilidade de ver quem se conectou mais vezes às nossas redes ainda obtiveram um resultado aceitável.

O desinteresse acontece mesmo ao nível da partilha de redes publicamente e da não introdução da senha ao nos conectarmos. Em relação à partilha pública este desinteresse provavelmente advirá de preocupações com a privacidade. No que toca à não necessidade de introduzir senha ao nos conectarmos, esta talvez passou despercebida porque se irá revelar um tanto "transparente" ao utilizador, embora represente para mim uma grande vantagem em termos de usabilidade e até de privacidade, uma vez que os terceiros que acedam à rede não ficarão a conhecer qual a senha em uso.

Em suma, quatro das funcionalidades obtiveram um elevado nível de interesse por parte dos utilizadores. Embora as restantes não tenham obtido tanto interesse não quer dizer que as estas sejam totalmente irrelevantes para a aplicação.

Os resultados deste inquérito serviram para consolidar as funcionalidades inicialmente previstas e ter uma visão de que estas eram realmente do interesse dos inquiridos, sendo que apenas duas delas se revelaram menos apelativas.

3.2.3. Conclusões

O *brainstorming* inicial serviu para tentar arranjar novas ideias a integrar na aplicação, e o inquérito para validar se estas eram realmente do interesse dos utilizadores.

Embora os resultados tenham sido positivos nesse aspeto, é necessário refletir em que funcionalidades vale realmente a pena investir o esforço inicial de desenvolvimento. Assim, foram deixadas de fora as seguintes funcionalidades: *chat* entre utilizadores nas proximidades, partilha da ligação 4G, ver, quem se ligou mais a nós e a quem nos ligámos mais vezes.

O *chat* entre utilizadores nas proximidades foi deixado de fora porque esta é uma funcionalidade que já está implementada noutras aplicações populares, nomeadamente no *WiTalky* e no *hike messenger*. Estas possuem mais de 100 mil instalações e mais de 10 milhões, respetivamente. Além de estar a competir com aplicações com centenas de milhares de utilizadores, o esforço requerido para implementar tal funcionalidade juntamente com todas as outras vai para além do possível no contexto deste projeto de mestrado. Por essas razões, foi decidido deixar tal funcionalidade para uma futura expansão.

A partilha da ligação móvel 3G/4G foi deixada de fora pelas mesmas razões acima mencionadas e também porque muitos *smartphones* vêm com esta funcionalidade já embutida. Como exemplo de aplicações com esta funcionalidade, temos o *Foxfi*, com mais de 5 milhões de instalações, e o *EasyTether*, com mais de 1 milhão de instalações. Uma vez que é criado um ponto de acesso Wi-Fi quando partilhamos a nossa ligação móvel 3G/4G, este poderá também ser partilhado através do sistema a desenvolver.

Embora seja sempre uma decisão difícil de tomar, deixar funcionalidades de fora do desenvolvimento inicial do sistema pode se tornar decisivo no momento da entrega do sistema ao cliente. Saber o que a equipa de desenvolvimento consegue suportar num determinado espaço de tempo irá certamente determinar a qualidade final do sistema no momento da sua entrega.

Assim sendo, as funcionalidades a serem implementadas são as seguintes:

- Gerir redes Wi-Fi
- Partilhar redes Wi-Fi
- Definir restrições de partilha
- Cópia de segurança
- Sincronização com todos os dispositivos do utilizador
- Autenticação transparente ao utilizador
- Conexão automática
- Deteção inteligente do alcance da ligação
- Testar o desempenho da rede

Com as funcionalidades definidas, passa-se no próximo ponto à formalização dos requisitos através de casos de uso.

3.3. Requisitos

Para suportar devidamente o processo de desenvolvimento do sistema, tem de haver toda uma documentação que seja capaz de transmitir os requisitos de uma forma clara a todos os *stakeholders*.

Os *stakeholders* são todos os intervenientes no projeto, desde o cliente até à equipa de desenvolvimento. É importante fazer passar bem a mensagem entre todos eles para não haver falhas na definição dos requisitos, evitando mal entendidos e perdas de tempo.

Como o projeto é fruto de uma iniciativa pessoal, os requisitos foram estipulados em torno da visão inicial do conceito, dos resultados do inquérito e da análise de funcionalidades feita no ponto anterior.

A especificação dos requisitos para este projeto será feita com base em casos de uso segundo a metodologia Use Case 2.0 sugerida por Ivar Jacobson. Estes casos de utilização representam uma vista simplificada de todos os requisitos do sistema, sendo descritos em mais detalhe através de uma 'história', tornando mais fácil a perceção dos requisitos entre todos os *stakeholders* [23]. Tal temática é explorada em mais detalhe à medida que se procede à sua definição, elaborada no próximo ponto.

3.3.1. Casos de uso

Os casos de uso são todas as formas de usar um sistema para atingir um determinado objetivo de um utilizador. O conjunto de todos os casos de uso demonstra todas as formas úteis de utilizar o sistema, ilustrando o valor que este irá gerar [23].

Um caso de uso é:

- Uma sequência de ações que um sistema executa resultando em algo observável com valor para o utilizador.
- Um comportamento específico do sistema que colabora com o utilizador para que o resultado tenha valor para o utilizador.
- A unidade mais pequena de atividade que fornece um resultado significante para o utilizador.

Isto torna claro o que o sistema vai fazer e, por omissão intencional, o que não vai fazer, permitindo previsões eficazes, gestão do contexto e desenvolvimento incremental de sistemas de qualquer tipo e tamanho [23].

Os casos de uso são largamente utilizados para o levantamento de requisitos, podendo ser extraídos através de um cenário simples que retrata o que utilizador espera do sistema [24].

Os casos de uso são documentados usando um diagrama de alto nível, identificando os atores e as suas interações com o sistema. Os atores, que podem ser humanos ou sistemas, são representados através de uma figura humana e as interações através de elipses. Cada um destes deve conter uma descrição textual, podendo ser ligado a outros modelos para descrever o cenário em mais detalhe [24].

Com base nas funcionalidades levantadas anteriormente no ponto 3.2, define-se então os casos de uso do sistema:

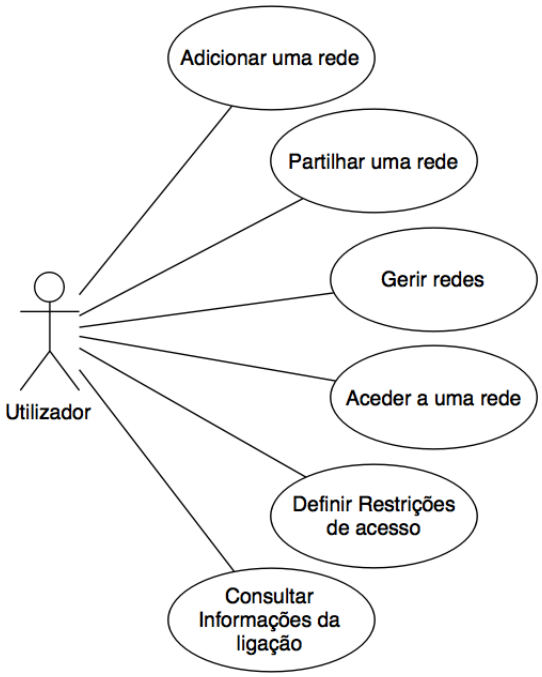

Figura 3 – Diagrama de casos de uso

Na figura 3 está representado o diagrama de casos de uso do sistema a desenvolver. Estes são os casos de uso essenciais e sem muito detalhe, uma vez que cada um será explorado em mais detalhe posteriormente.

De forma a obter "traceability", a possibilidade de identificar a evolução dos requisitos ao longo do projeto, é definida uma tabela de casos de uso devidamente numerada. Esta numeração será usada posteriormente aquando da divisão dos casos de uso.

Tabela 3 - Casos de uso

N.º	*Caso de uso*
1	Adicionar uma rede
2	Partilhar uma rede
3	Gerir redes
4	Aceder a uma rede
5	Definir restrições de acesso
6	Consultar informações da rede

Para dar suporte à tabela de casos de uso definida anteriormente, é necessário explorar cada um através de uma descrição mais detalhada. Esta descrição será feita com recurso a 'histórias', ilustradas através de tabelas com os fluxos esperados dos casos de uso. Isto permite uma melhor compreensão dos requisitos entre todos os *stakeholders* envolvidos [23].

O fluxo é dividido em dois segmentos, o fluxo básico e o alternativo. O fluxo básico representa o comportamento principal do caso de uso, enquanto o fluxo alternativo representa comportamentos excecionais como indisponibilidade de serviços, ocorrência de erros ou escolhas do utilizador.

Como é referido em [23], [24], é conveniente dividir os casos de uso em várias *fatias* para suportar a construção do sistema de forma incremental. Esta divisão será feita após a construção do fluxo do caso de uso, analisando-o e tentado dividi-lo em pequenas *fatias* que representem as suas demais etapas em pequenas porções quantificáveis.

Tais *fatias* serão descritas através de uma tabela com uma breve descrição, importância, critério de teste e complexidade de implementação. Cada *fatia* será numerada com o formato: $C.F$, onde C representa o caso de uso associado à *fatia* número F.

Através deste processo de divisão dos casos de uso em *fatias* consegue-se obter uma melhor visão das tarefas e do esforço a ser realizado em cada caso de uso na fase de desenvolvimento.

Nos pontos seguintes são aprofundados todos os casos de uso consoante a metodologia descrita anteriormente, identificando-se com isto os requisitos do sistema.

3.3.1.1. Adicionar uma rede

Neste ponto explora-se o caso de uso "Adicionar uma rede", começando-se pela identificação do seu fluxo e avançando-se para sua divisão em *fatias*.

Tabela 4 - Fluxo de "Adicionar uma rede"

Fluxo básico	*Fluxo alternativo*
1. Selecionar rede	A1. Credenciais inválidas
2. Selecionar opção adicionar	A2. Localização indisponível
3. Introduzir as credenciais	
4. Conectar	
5. Testar o desempenho da rede	
6. Obter localização	
7. A rede é adicionada	

Na tabela 4 estão representados os fluxos do caso de uso “Adicionar rede” consoante a metodologia descrita anteriormente. Á esquerda encontra-se o fluxo básico com os demais passos necessários à sua conclusão. À direita encontram-se os fluxos alternativos, identificados ao analisar o fluxo básico.

Neste caso, identificam-se dois comportamentos alternativos, um no processo de conexão e outro na obtenção da localização. No processo de conexão as credenciais podem estar erradas, impossibilitando a realização do passo 4 com sucesso. No processo de obtenção da localização o utilizador pode se encontrar num local com mau sinal ou ter a localização desabilitada, impossibilitando a conclusão do passo 6.

Explora-se, agora, o fluxo presente na tabela 4, tentando dividi-lo em várias *fatias* que representem operações ou paços atómicos, de forma a separar as demais tarefas individualmente.

Através do fluxo básico apresentado na tabela 4 identificam-se as seguintes *fatias*: conectar à rede a adicionar, testar o desempenho da rede, obter a sua localização e guardar a rede no sistema. Através do fluxo alternativo identificam-se as *fatias* de tratar credenciais inválidas e localização indisponível.

De forma a dar uma visão geral dos requisitos do caso de uso, representam-se as demais *fatias* devidamente numeradas através da seguinte tabela:

Tabela 5 - Fatias do caso de uso "Adicionar rede"

N.º	*Fatia*
1.1	Conectar à rede
1.2	Testar o desempenho
1.3	Obter localização
1.4	Guardar rede no sistema
1.5	Tratar credenciais inválidas
1.6	Tratar localização indisponível

Com as demais *fatias* identificadas por meio da tabela 5, explora-se agora individualmente cada *fatia* de forma mais detalhada. Cada *fatia* será representada por uma tabela contendo uma breve descrição, a importância desta para o sistema, um critério de teste para torna-la testável na fase de desenvolvimento e a complexidade de implementação estimada. A complexidade de implementação é particularmente importante para ser possível comparar e priorizar as *fatias* a serem desenvolvidas em primeiro lugar.

A descrição das demais *fatias* apresentam-se de seguida, começando-se pela *fatia* 1.1 - Conectar à rede.

Tabela 6 - Fatia 1.1 "Conectar à rede"

Descrição	O sistema deve conseguir conectar-se à rede a adicionar.
Importância	Garantir que as credenciais estão corretas.
Critério de teste	O sistema conecta-se à rede dadas as credenciais corretas introduzidas pelo utilizador. O processo de conexão não deve exceder os 10 segundos.
Complexidade de implementação	Média

Aqui, o foco é garantir que a rede está acessível e que as credenciais providas pelo utilizador estão corretas. É atribuída uma complexidade média de implementação devido à necessidade de detetar e configurar a conexão à rede consoante o seu tipo de segurança.

É de salientar que o processo de estimar a complexidade de implementação depende muito da experiência e do instinto de quem o faz, servindo apenas como uma referência e não como facto garantido.

De seguida apresenta-se a descrição da *fatia* 1.2 - Testar o desempenho.

Tabela 7 - Fatia 1.2 "Testar o desempenho"

Descrição	O sistema deve conseguir testar o desempenho da ligação.
Importância	Prover o utilizador com informações de desempenho. Auxiliar a escolha da melhor rede.
Critério de teste	O sistema obtém informações de tempo de resposta e largura de banda disponível.
Complexidade de implementação	Alta

O principal desafio desta *fatia* concentra-se em encontrar um método razoável para testar o desempenho das redes. É de notar que para possibilitar que múltiplas redes sejam testadas ao mesmo tempo tem de haver largura de banda suficiente para acomodar todos estes testes de forma adequada. Assim, é atribuída uma dificuldade de implementação alta.

Apresenta-se seguidamente a descrição da *fatia* 1.3 - Obter localização.

Tabela 8 – Fatia 1.3 "Obter localização"

Descrição	O sistema deve conseguir obter a localização da rede a adicionar.
Importância	Prover o utilizador da localização das suas redes quando as adiciona.
Critério de teste	O sistema obtém a localização da rede a adicionar. A duração máxima deve ser inferior a 10 segundos.
Complexidade de implementação	Baixa

Obter a localização na plataforma *Android* será, em princípio, relativamente fácil, uma vez que são fornecidos mecanismos nativos e de fácil implementação para obter a localização do utilizador. Assim, atribui-se uma complexidade baixa a esta *fatia*.

Apresenta-se agora a descrição referente à *fatia* 1.4 - Guardar rede no sistema.

Tabela 9 - Fatia 1.4 "Guardar rede no sistema"

Descrição	O sistema deve conseguir armazenar as redes do utilizador.
Importância	Tornar as redes do utilizador persistentes.
Critério de teste	A rede do utilizador é armazenada no sistema.
Complexidade de implementação	Alta

Para guardar efetivamente uma rede no sistema e esta poder depois ser partilhada e acedida por outros utilizadores, têm de ser recolhidas informações pertinentes acerca da sua configuração e só depois se pode proceder ao ser armazenamento. Uma provável solução é a utilização de um servidor *web* ligado a uma base de dados.

Como uma das restrições impostas é a da proteção dos dados contra ataques externos (restrição n.º 5 do ponto 3.3.2), terá de ser estudado o método de proteção mais adequado a ser utilizado. Assim, é atribuída uma complexidade de implementação alta a esta *fatia*.

Passa-se, de seguida, à descrição da *fatia* 1.5 – Tratar credenciais inválidas.

Tabela 10 - Fatia 1.5 "Tratar credenciais inválidas"

Descrição	O sistema deve mostrar uma mensagem ao utilizador quando as credenciais estiverem incorretas.
Importância	Promover a reintrodução das credenciais corretamente.
Critério de teste	É mostrada uma mensagem de erro quando as credenciais são incorretamente introduzidas.
Complexidade de implementação	Baixa

O tratamento de credenciais inválidas deve, em princípio, ser um processo trivial, em que é mostrado uma mensagem ao utilizador caso a conexão à rede falhe devido à autenticação. Atribui-se assim uma complexidade baixa a esta *fatia*.

Descreve-se de seguida a *fatia* 1.6 - Tratar localização indisponível.

Tabela 11 - Fatia 1.6 "Tratar localização indisponível"

Descrição	O sistema deve mostrar uma mensagem ao utilizador quando a localização estiver indisponível.
Importância	Encorajar o utilizador a ativar a recolha da localização no seu dispositivo.
Critério de teste	É mostrada uma mensagem ao utilizador quando a localização estiver indisponível.
Complexidade de implementação	Baixa

Semelhantemente à *fatia* anterior, pretende-se alertar o utilizador através de uma mensagem, o que se estima ser um processo algo trivial. Atribui-se assim uma complexidade baixa.

Com esta última, termina-se a descrição das *fatias* referentes ao caso de uso n.º 1 - Adicionar uma rede.

De seguida analisa-se este caso de uso como um todo, identificando-se quais os elementos do sistema envolvidos e como estes interagem na realização do presente caso de uso. Esta etapa serve para dar mais um passo na tradução da linguagem de negócio para a linguagem de desenvolvimento.

A análise descrita será apresentada por meio de uma vista componente-conetor. Esta vista, embora simples e abstrata, representa os demais elementos do sistema e as suas interações, servindo para guiar o desenho arquitetural numa fase posterior.

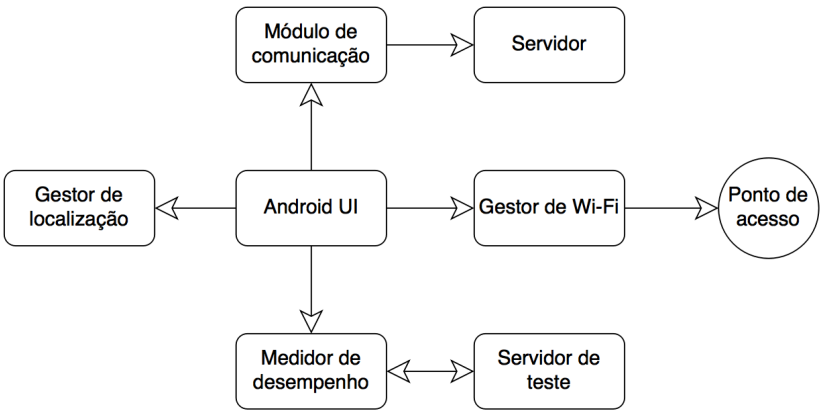

Figura 4 - Vista componente-conetor do caso de uso "Adicionar uma rede"

Com esta vista consegue-se ter uma melhor noção que componentes do sistema estão envolvidos neste caso de uso e quem interage com quem.

Neste caso temos no centro a interface *Android*, que será por onde o utilizador fará as demais interações necessárias para adicionar a rede. A interface irá assim estar ligada aos módulos necessários para recolher as informações da rede, nomeadamente o Gestor de Wi-Fi, Medidor de desempenho e Gestor de localização.

O Gestor de Wi-Fi resulta das *fatias* 1.1 e 1.4, atribuindo-se a responsabilidade de manusear as redes Wi-Fi, quer seja para se conectar quer para extrair as suas informações.

O Medidor de desempenho resulta da *fatia* 1.2, necessitando de comunicar com um servidor de teste e transferir uma certa quantidade de informação para que seja possível testar os parâmetros de tempo de resposta e largura de banda definidos nessa mesma *fatia*.

O Gestor de localização resulta da *fatia* 1.3, ficando entregue à tarefa de obter a localização da rede que está a ser adicionada ao sistema.

Após todos estes os componentes terem servido o seu propósito de extrair as informações da rede a adicionar, é necessária a sua transmissão para o servidor que as irá armazenar. Fica assim encarregue desta tarefa o Módulo de comunicação.

E, assim, termina a análise deste caso de uso, passando-se no ponto seguinte à descrição e análise do caso de uso n.º 2 – Partilhar uma rede.

3.3.1.2. Partilhar uma rede

Seguindo a mesma metodologia, explora-se agora o caso de uso "Partilhar uma rede". Primeiro, é identificado o seu fluxo e, depois, este é dividido em *fatias*. Apresenta-se, de seguida, o fluxo do caso de uso n.º 2 – Partilhar uma rede.

Tabela 12 – Fluxo de "Partilhar uma rede"

Fluxo básico	*Fluxo alternativo*
1. Selecionar rede	A1. Não é o proprietário da rede
2. Verificar se pode ser partilhada	A2. Recetor da partilha inexistente
3. Selecionar opção partilhar	
4. Selecionar com quem partilhar	
5. A rede é partilhada	

Este fluxo concentra-se em partilhar uma dada rede com outro utilizador do sistema.

Analisando o fluxo principal obtém-se dois fluxos alternativos, o A1, que impede a conclusão do passo 3, e o A2, que impede a conclusão do passo 4.

Posto isto, analisam-se os demais fluxos, identificando-se as *fatias* deste caso de uso, apresentadas de seguida na tabela 13.

Tabela 13 - Fatias do caso de uso "Partilhar uma rede"

N.º	*Fatia*
2.1	Partilhar publicamente
2.2	Partilhar seletivamente
2.3	Partilhar estando *offline*
2.4	Tratar utilizador não proprietário
2.5	Tratar remetente inválido

Com as *fatias* identificadas, passa-se agora à descrição individual das mesmas, começando-se pela *fatia* 2.1 – Partilhar publicamente.

Tabela 14 - Fatia 2.1 "Partilhar publicamente"

Descrição	O sistema deve permitir a partilha de redes com todos os utilizadores.
Importância	Permitir que o utilizador torne as suas redes acessíveis a todos os outros.
Critério de teste	Após ter sido partilhada, a rede fica acessível a todos os utilizadores do sistema.
Complexidade de implementação	Baixa

Esta *fatia* concentra-se na partilha com todos os membros da aplicação, sendo que, ao partilhar a rede, todos os utilizadores do sistema podem se conectar a ela.

Atribui-se uma complexidade baixa a esta *fatia*, dado que neste passo apenas deve ser necessária a implementação da interface e ser feito um pedido ao servidor para que se partilhe efetivamente a rede.

Descreve-se de seguida a *fatia* 2.2 – Partilhar seletivamente.

Tabela 15 - Fatia 2.2 "Partilhar seletivamente"

Descrição	O sistema deve permitir a partilha de redes apenas com os utilizadores selecionados.
Importância	Permitir que o utilizador escolha quem pode aceder às suas redes.
Critério de teste	Após a partilha, o acesso é garantido apenas aos utilizadores selecionados.
Complexidade de implementação	Baixa

Esta *fatia* é muito semelhante à *fatia* anterior. A diferença centra-se nos alvos da partilha, que neste caso serão apenas aqueles que o utilizador selecione. Pelas mesmas razões da *fatia* anterior, é atribuída uma complexidade de implementação baixa.

Apresenta-se de seguida a *fatia* 2.3 – Partilhar estando *offline*.

Tabela 16 - Fatia 2.3 "Partilhar estando offline"

Descrição	O sistema deve permitir a partilha de uma rede sem que o utilizador tenha acesso à internet.
Importância	Permitir que os alvos da partilha recebam acesso à rede sem terem ligação à internet.
Critério de teste	Um utilizador consegue partilhar uma rede com outro utilizador estando este sem ligação à internet.
Complexidade de implementação	Alta

Embora o conceito aqui seja o mesmo que nas *fatias* 2.1 e 2.2, partilhar uma rede com outro utilizador, esta possui complexidade adicional. Tal complexidade advém do facto do alvo da partilha estar sem conexão à internet, impossibilitando a comunicação com o servidor. Assim, atribui-se uma complexidade de implementação alta, uma vez que terão de ser encontrados mecanismos que não utilizem uma ligação ao servidor para efetuar a partilha.

Passa-se, de seguida, à descrição da *fatia* 2.4 - Tratar quando o utilizador não é o proprietário.

Tabela 17 - Fatia 2.4 "Tratar utilizador não proprietário"

Descrição	O sistema deve impedir um utilizador de partilhar redes que não adicionou.
Importância	Manter a privacidade das redes.
Critério de teste	Um utilizador não consegue partilhar uma rede que não tenha sido adicionada por ele.
Complexidade de implementação	Baixa

Esta *fatia* centra-se em negar que se partilhem redes adicionadas por outros utilizadores, respeitando os 'direitos de propriedade' de quem adicionou a rede.

Em termos práticos, esta *fatia* passará por validar se o utilizador é o proprietário da rede, impedindo que este a partilhe caso contrário. Tal validação será, em princípio, de simples implementação, atribuindo-se assim uma complexidade baixa.

Descreve-se agora a *fatia* 2.5 - Tratar destinatário inválido.

Tabela 18 - Fatia 2.5 "Tratar destinatário inválido"

Descrição	O sistema deve mostrar uma mensagem ao utilizador se o destinatário estiver incorreto.
Importância	Promover a correta introdução do destinatário.
Critério de teste	É mostrada uma mensagem de erro quando o destinatário da partilha não estiver correto.
Complexidade de implementação	Baixa

O foco desta *fatia* é o de detetar se os dados do alvo da partilha estão corretos, mostrando uma mensagem ao utilizador caso contrário. Uma vez mais atribui-se uma complexidade baixa a esta *fatia*, sendo que tal validação deverá ser algo trivial.

E, com esta última, concluem-se todas as *fatias* do caso de uso n.º 2 – Partilhar uma rede.

Passa-se, agora, à identificação dos componentes envolvidos neste caso de uso e das interações entre eles através de uma vista componente-conetor, apresentada na figura 5.

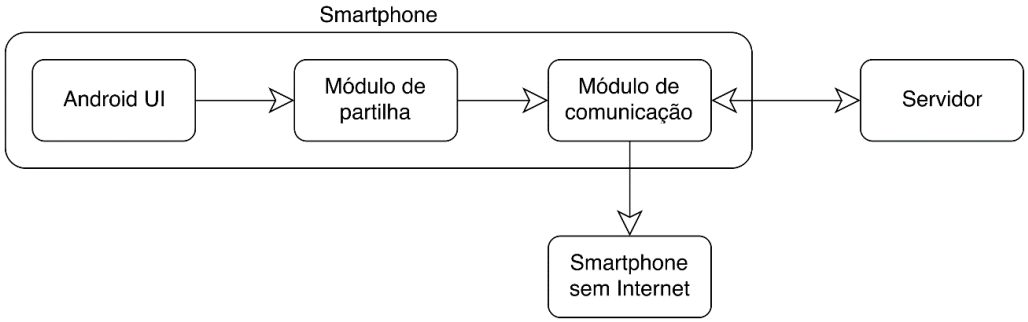

Figura 5 - Vista componente-conetor do caso de uso "Partilhar uma rede"

A interface *Android* será responsável por disponibilizar os mecanismos necessários para que o utilizador consiga prosseguir com a partilha da sua ligação.

O Módulo de partilha resulta das *fatias* 2.1, 2.2 e 2.3, tratando os dados introduzidos pelo utilizador e interagindo com o Módulo de comunicação para que a partilha seja comunicada ao servidor ou ao *smartphone* de outro utilizador através da partilha *offline*, representado pelo elemento *smartphone* sem Internet.

E termina, assim, a descrição do corrente caso de uso - Partilhar uma rede, passando-se no ponto seguinte à descrição e análise do caso de uso n.º 3 – Gerir redes.

3.3.1.3. Gerir redes

Uma vez mais e para descrever efetivamente o corrente caso de uso – Gerir redes, começamos pela descrição do seu fluxo, apresentado através da tabela 19.

Tabela 19 - Fluxo de "Gerir redes"

Fluxo básico	*Fluxo alternativo*
1. Entrar na área de gestão	A3. Remover rede
2. Seleciona rede	A4. Dados inválidos
3. Alterar informação	A5. Cópia de segurança
4. Validar a informação	A6. Sincronizar redes
5. Guardar as novas informações	

O fluxo da gestão de redes centra-se no manuseamento dos demais parâmetros associados às redes do utilizador. Tais ações são representadas pelo fluxo principal que toma uma abordagem mais genérica com o passo 3 – Alterar informação, e pelos fluxos alternativos que tomam uma abordagem mais específica.

Decidiu-se incluir neste caso de uso como comportamentos alternativos a cópia de segurança e a sincronização ao invés da criação de novos casos de uso. Esta decisão tem um impacto positivo no ponto em que simplifica a especificação, mas também contribui para que haja menos granularidade e com isso possam ser omissos detalhes importantes para a correta definição destes dois pontos.

Dada a complexidade do sistema a desenvolver, não se achou relevante aumentar ainda mais o nível de granularidade nestes pontos, tornando assim a documentação um pouco mais leve.

Assim sendo, apresenta-se então as *fatias* para este caso de uso através da tabela 20.

Tabela 20 - Fatias do caso de uso "Gerir redes"

N.º	*Fatia*
3.1	Editar rede
3.2	Tratar dados inválidos
3.3	Remover rede
3.4	Remover partilha pública
3.5	Remover partilha seletiva
3.6	Cópia de segurança
3.7	Sincronizar redes entre dispositivos

Com as *fatias* identificadas passa-se agora à sua descrição individual, começando-se pela 3.1 – Editar rede.

Tabela 21 - Fatia 3.1 "Editar rede"

Descrição	O sistema deve permitir que o utilizador edite as propriedades das suas redes.
Importância	Dar mecanismos de gestão ao utilizador.
Critério de teste	O utilizador consegue modificar o nome e os valores das restrições de partilha da rede.
Complexidade de implementação	Média

Nesta *fatia* pretende-se dar mecanismos ao utilizador para editar as informações das suas redes. Atribui-se uma complexidade de implementação média a esta *fatia* uma vez que terão de ser implementados mecanismos que consigam modificar os parâmetros não só no dispositivo do utilizador como também em todos os dispositivos.

Segue-se agora para a *fatia* 3.2 – Tratar dados inválidos.

Tabela 22 - Fatia 3.2 "Tratar dados inválidos"

Descrição	O sistema deve mostrar uma mensagem ao utilizador se os dados introduzidos forem inválidos.
Importância	Promover a correção dos dados.
Critério de teste	É mostrada uma mensagem de erro quando forem introduzidos dados inválidos.
Complexidade de implementação	Baixa

A intenção desta *fatia* é a de detetar eventuais erros introduzidos pelo utilizador, mostrando uma mensagem para incentivar o utilizador a corrigi-las. Foi atribuída uma complexidade de implementação baixa a esta *fatia*, uma vez que as validações devem ser relativamente simples como impedir os utilizadores de introduzir carateres inválidos.

Apresenta-se, de seguida, a descrição da *fatia* 3.3 – Remover rede.

Tabela 23 - Fatia 3.3 "Remover rede"

Descrição	O sistema deve permitir a remoção das redes.
Importância	Dar mecanismos de gestão ao utilizador.
Critério de teste	A rede é removida do sistema.
Complexidade de implementação	Baixa

O intuito desta *fatia* é o de possibilitar que utilizadores consigam eliminar permanentemente as suas redes do sistema. Aqui atribui-se também um nível de complexidade baixa, uma vez que tal operação se estima ser algo trivial.

Segue-se, agora, a descrição da *fatia* 3.4 – Remover partilha pública.

Tabela 24 - Fatia 3.4 "Remover partilha pública"

Descrição	O sistema deve permitir ao utilizador remover a partilha pública das suas redes.
Importância	Dar mecanismos de gestão ao utilizador.
Critério de teste	É removido o acesso de terceiros à rede.
Complexidade de implementação	Baixa

Esta *fatia* foca-se em retirar a possibilidade de os demais utilizadores do sistema conseguirem aceder à rede previamente partilhada. Estima-se que a remoção desta partilha seja também este um processo simples, sendo-lhe assim atribuída uma complexidade baixa.

Passa-se, agora, à descrição da *fatia* 3.5 – Remover partilha seletiva.

Tabela 25 - Fatia 3.5 "Remover partilha seletiva"

Descrição	O sistema deve permitir ao utilizador remover entidades seletivamente da partilha.
Importância	Dar mecanismos de gestão ao utilizador.
Critério de teste	É removido o acesso às entidades selecionadas.
Complexidade de implementação	Baixa

Muito semelhante à *fatia* 3.3, só que neste caso remove-se a partilha apenas de alguns utilizadores em específico. Estima-se que a complexidade esteja dentro dos valores da *fatia* anterior, atribuindo-se assim uma complexidade de implementação baixa.

Avança-se, agora, para a descrição da *fatia* 3.6 – Cópia de segurança.

Tabela 26 - Fatia 3.6 "Cópia de segurança"

Descrição	O sistema deve manter uma cópia de segurança das redes do utilizador.
Importância	Garantir a persistência dos dados do utilizador em caso de perca de informação no seu dispositivo.
Critério de teste	Ao haver perda de dados no seu dispositivo, o utilizador consegue restaurar as suas redes.
Complexidade de implementação	Média

Esta *fatia* foca-se em possibilitar que os utilizadores possam recuperar as suas redes caso exista algum problema no seu dispositivo que leve à perda da sua informação.

Uma vez que o sistema armazena as redes dos utilizadores na componente do servidor, descrito na *fatia* 1.4, aqui será necessário uma normalização da informação contida na componente do servidor para que esta possa ser transmitida e lida no dispositivo do utilizador, possibilitando a restauração das suas redes. Assim, atribui-se um nível de complexidade médio.

De seguida, descreve-se a *fatia* 3.7 - Sincronizar dispositivos.

Tabela 27 - Fatia 3.7 "Sincronizar dispositivos"

Descrição	O sistema deve ser capaz de sincronizar as redes do utilizador com todos os seus dispositivos.
Importância	Manter os dispositivos do utilizador atualizados.
Critério de teste	Ao serem efetuadas mudanças nas redes do utilizador, este deve conseguir fazer refletir tais alterações em todos os seus dispositivos. Este processo não deve demorar mais de 10 segundos sobre uma ligação média de 5Mbps.
Complexidade de implementação	Média

O foco desta *fatia* é o de sincronizar os dispositivos do utilizador, fazendo que as mudanças efetuadas num dispositivo se façam refletir em todos os outros.

O desafio será fazer o envio de quaisquer alterações de um para todos os outros dispositivos de um determinado utilizador. Assim, atribui-se um nível de complexidade médio a esta *fatia*.

E, com esta última, conclui-se a descrição das demais *fatias* co corrente caso de uso, avançando-se para a identificação dos elementos e respetivas interações entre eles para se realizar este caso de uso. Tal é representado através de uma vista componente-conetor ilustrada na figura 6.

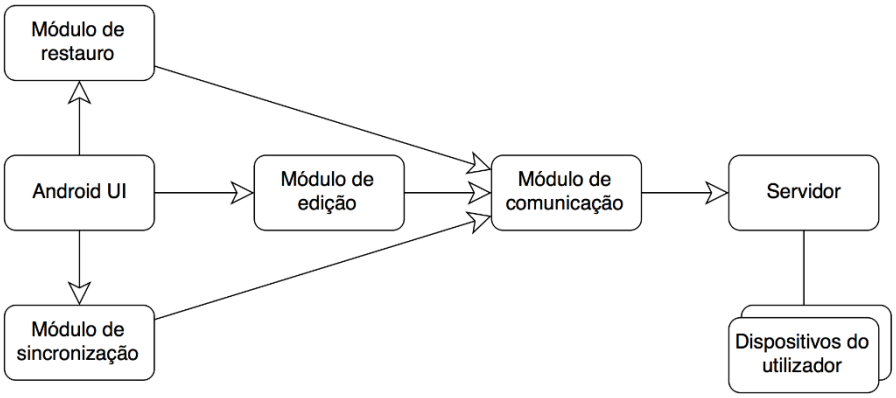

Figura 6 - Vista componente-conetor do caso de uso "Gerir redes"

Nesta vista tem-se uma vez mais como ponto inicial a interface *Android*, que irá comunicar com os demais módulos de sincronização, restauro e edição conforme as intenções do utilizador.

O módulo de restauro origina da *fatia* 3.6, visando a recuperação dos dados do utilizador em caso de perda de extravio.

O módulo de sincronização origina da *fatia* 3.7, com o objetivo de fazer refletir as alterações em todos os dispositivos do utilizador.

O módulo de edição origina das *fatias* 3.1, 3.3, 3.4 e 3.5, que visam alterar certas informações de uma rede em específico.

Finalmente, para fazer refletir as mudanças efetuadas é pedido ao módulo de comunicação que faça a transmissão da informação para o servidor, delegando a este a tarefa de fazer refletir as mudanças nos demais dispositivos do utilizador.

E, com esta, vista termina-se a descrição do corrente caso de uso – Gerir redes, passando-se no ponto seguinte para a descrição e análise do caso de uso n.º 4 – Aceder a uma rede.

3.3.1.4. Aceder a uma rede

Mantendo a mesma filosofia, passa-se à descrição de mais um caso de uso – Aceder a uma rede, começando-se pela definição do seu fluxo através da tabela 28.

Tabela 28 - Fluxo de "Aceder a uma rede"

Fluxo básico	*Fluxo alternativo*
1. Selecionar rede	A1. Rede não partilhada
2. Selecionar opção conectar	A2. Rede fora do alcance
3. A conexão é estabelecida	A3. Conexão automática

O fluxo para este caso de uso, à primeira vista, parece ser relativamente simples, embora a sua análise demonstre o contrário. Tal análise reflete-se através da tabela 29, que contém as demais *fatias* identificadas neste processo.

Tabela 29 - Fatias do caso de uso "Aceder a uma rede"

N.º	*Fatia*
4.1	Conectar manualmente
4.2	Conectar automaticamente
4.3	Autenticar de forma transparente
4.4	Aceder às redes em modo *offline*
4.5	Escolher autonomamente a melhor rede
4.6	Manter a conexão na deslocação entre os pontos de acesso
4.7	Tratar redes não partilhadas
4.8	Tratar redes fora do alcance

Este é um exemplo do quão importante é esta análise mais detalhada. Um caso de uso que inicialmente aparenta ser um tanto simples e que se vem a tornar em muito mais que isso. Depois de analisar o fluxo e identificar as suas ramificações acaba-se com 8 *fatias*, em que a maioria apresenta uma complexidade de implementação média ou alta.

Tais *fatias* são descritas em mais detalhe de seguida, começando-se pela *fatia* 4.1 – Conectar de forma manual.

Tabela 30 - Fatia 4.1 "Conectar de forma manual"

Descrição	O sistema deve permitir que o utilizador possa se conectar a uma rede manualmente.
Importância	Dar liberdade ao utilizador de escolher a rede a utilizar.
Critério de teste	Uma vez selecionada a opção conectar, é estabelecida conexão à rede. A duração do processo não deve exceder os 10 segundos.
Complexidade de implementação	Baixa

Neste caso, pretende-se dar liberdade ao utilizador de escolher a rede a se conectar. A conexão manual pensa-se ser trivial, uma vez que se possui as informações sobre o tipo de segurança e as credenciais da rede a aceder (através da *fatia* 1.4 – Guardar rede no sistema). O esforço inicial de perceber os mecanismos de como se conectar a uma rede já deve ter sido feito com a *fatia* 1.1 – Conectar à rede, requerendo assim menos esforço de implementação, em princípio. Daí a atribuição de uma complexidade de implementação baixa.

Avança-se, agora, para a *fatia* 4.2 – Conectar de forma automática, descrita na tabela 31.

Tabela 31 - Fatia 4.2 "Conectar de forma automática"

Descrição	O sistema deve ser capaz de se conectar a uma rede sem intervenção do utilizador.
Importância	Tornar o processo de conexão mais cómodo.
Critério de teste	Ao ser encontrada uma rede acessível, a conexão é automaticamente estabelecida.
Complexidade de implementação	Média

A intenção desta *fatia* é a de facilitar a conexão às demais redes nas proximidades sem que o utilizador tenha de se conectar manualmente. Atribui-se uma complexidade de implementação média a esta *fatia*, uma vez que o sistema terá de detetar quais as redes ao alcance que estão partilhadas e estabelecer conexão a uma delas sem qualquer intervenção do utilizador.

Segue-se, agora, para a *fatia* 4.3 - Autenticar de forma transparente.

Tabela 32 - Fatia 4.3 "Autenticar de forma transparente"

Descrição	O sistema deve ser capaz de gerir o processo de autenticação a um ponto de acesso de forma transparente ao utilizador.
Importância	Tornar o processo de autenticação mais cómodo.
Critério de teste	Aquando da conexão a uma rede, o sistema autentica-se a esta sem interação do utilizador.
Complexidade de implementação	Média

Esta *fatia* consiste em "esconder" o processo de autenticação do utilizador para que este não tenha de introduzir as credenciais das redes partilhadas a que acede de forma manual. Atribui-se a esta *fatia* uma complexidade de implementação média.

Apresenta-se, agora, a *fatia* 4.4 - Aceder às redes em modo *offline*.

Tabela 33 - Fatia 4.4 "Aceder às redes em modo offline"

Descrição	O sistema deve ser capaz de aceder às redes a que o utilizador tem acesso sem ligação à internet.
Importância	Permitir que o utilizador se possa conectar quando não tem forma de obter ligação à internet.
Critério de teste	Encontrando-se sem ligação à internet, o sistema consegue conectar-se às redes partilhadas.
Complexidade de implementação	Média

Esta *fatia* consiste em permitir que os utilizadores consigam se ligar às demais redes partilhadas sem que estejam conectados à Internet. As redes terão de ser armazenadas de alguma forma no dispositivo do utilizador para que este acesso *offline* seja possível. Por esse motivo, atribui-se uma complexidade de implementação média.

Segue-se, agora, para a *fatia* 4.5 - Escolher autonomamente a melhor rede.

Tabela 34 - Fatia 4.5 "Escolher automáticamente a melhor rede"

Descrição	O sistema deve ser capaz de determinar qual a melhor rede a que o utilizador se deve conectar sem a intervenção do mesmo.
Importância	Prover o utilizador da melhor ligação possível.
Critério de teste	Dada uma lista de redes disponíveis, o sistema determina qual delas oferece melhor desempenho.
Complexidade de implementação	Alta

Esta *fatia* consiste em escolher a rede com mais qualidade à qual o utilizador se deve conectar. Este processo deverá ser um pouco mais complexo uma vez que existem muitas variáveis a ter em conta ao comparar duas redes, como a força do sinal, largura de banda e canal de difusão. Por isso, é-lhe atribuída uma complexidade de implementação alta.

Passa-se, agora, à descrição da *fatia* 4.6 - Manter a conexão na deslocação entre os pontos de acesso.

Tabela 35 - Fatia 4.6 " Manter a conexão na deslocação entre os pontos de acesso "

Descrição	O sistema deve ser capaz de manter o utilizador conectado à medida que este entra e sai do alcance das demais redes nas proximidades.
Importância	Manter o utilizador sempre conectado à medida que este se desloca.
Critério de teste	Ao sair do alcance de uma rede e entrar no alcance de outra, o sistema consegue estabelecer a conexão à última.
Complexidade de implementação	Alta

Este processo consiste em manter o utilizador conectado à medida que este se desloca de um local para outro. A sua implementação pode ser algo desafiante uma vez que o sistema terá de ter noção de quais as redes estão onde o utilizador se encontra, detetar quando o utilizador sai do alcance de uma rede e tentar estabelecer conexão a uma outra que esteja no seu alcance. Assim, atribui-se uma complexidade de implementação alta.

Avança-se, agora, para a *fatia* 4.7 - Tratar redes não partilhadas.

Tabela 36 - Fatia 4.7 "Tratar redes não partilhadas"

Descrição	O sistema deve prevenir a conexão às redes que não estão partilhadas com o utilizador.
Importância	Respeitar as definições de partilha das demais redes. Não incentivar a conexão a redes estranhas ao sistema.
Critério de teste	O sistema ao encontrar redes que não estejam partilhadas com o utilizador ou que sejam estranhas à plataforma, não deve tentar estabelecer conexão estas.
Complexidade de implementação	Média

Esta *fatia* consiste em fazer respeitar as regras da partilha e negar o acesso a quem não está autorizado a aceder, assim como não incentivar a conexão a redes estranhas à plataforma. Atribui-se uma complexidade de implementação média, dado que o sistema terá de determinar quais as redes a que se deve conectar.

De seguida, apresenta-se a *fatia* 4.8 - Tratar redes fora do alcance.

Tabela 37 - Fatia 4.8 "Tratar redes fora do alcance"

Descrição	O sistema deve mostrar uma mensagem ao utilizador caso a rede a conectar esteja fora do alcance.
Importância	Dar *feedback* ao utilizador.
Critério de teste	É mostrada uma mensagem ao utilizador caso não seja possível se conectar à rede que selecionou.
Complexidade de implementação	Média

Esta *fatia* visa dar *feedback* ao utilizador caso não seja possível conectar à rede escolhida. Mostrar uma mensagem é algo trivial, mas como são desconhecidos os mecanismos nativos para verificar a conexão, aplica-se uma complexidade de implementação média.

E, assim, termina-se a descrição das *fatias* do corrente caso de uso, passando-se de seguida à identificação dos demais elementos e respetivas interações envolvidas na sua realização através de uma vista componente-conetor, ilustrada na figura 7.

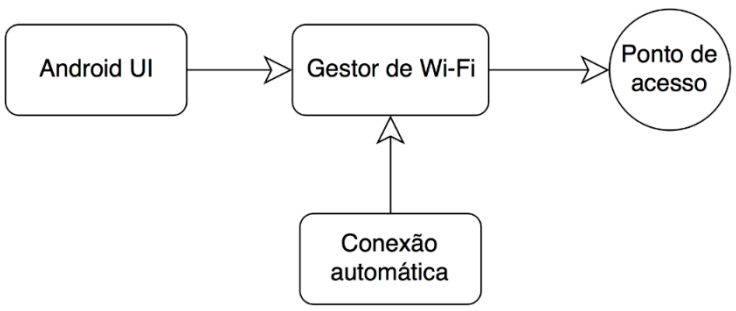

Figura 7 - Vista componente-conetor do caso de uso "Aceder a uma rede"

Nesta vista, além da interface *Android*, o módulo de conexão automática pode também incentivar a conexão a uma dada rede.

Enquanto a interface apenas reflete a intenção do utilizador de se conectar a uma dada rede, descrito através da *fatia* 4.1, o módulo de conexão automática atua autonomamente, combinando as *fatias* 4.2, 4.3 e 4.5, conectando-se a uma rede a seu critério sem qualquer interação por parte do utilizador.

O módulo de gestão de Wi-Fi fica encarregue da tarefa de conectar o utilizador à rede escolhida, quer por meio da interface, quer pelo módulo de conexão automática.

E, com esta vista, termina-se a análise do corrente caso de uso – Aceder a uma rede, passando-se no próximo ponto à descrição e análise do caso de uso n.º 5 – Definir restrições de acesso.

3.3.1.5. Definir restrições de acesso

Neste ponto descreve-se o caso de uso n.º 5 – Definir restrições de acesso, começando-se pelo seu fluxo através da tabela 38.

Tabela 38 - Fluxo de "Definir restrições de acesso"

Fluxo básico	*Fluxo alternativo*
1. Seleciona rede	A1. Tratar rede não partilhada
2. Introduzir valores limite	A2. Tratar valores inválidos
3. Guardar alterações	

O fluxo da definição de restrições de acesso foca-se em definir os valores limite dos demais parâmetros de desempenho da rede partilhada.

Do fluxo básico resultaram dois fluxos alternativos. O A1 - Tratar rede não partilhada, que impede a conclusão do passo 2, e o A2 - Tratar valores inválidos, que impede a conclusão do passo 3.

Com uma posterior análise a estes fluxos, identificaram-se as demais *fatias* deste caso de uso, apresentadas na tabela 39.

Tabela 39 - Fatias do caso de uso "Definir restrições de acesso"

N.º	*Fatia*
5.1	Limitar o tráfego diário
5.2	Limitar a largura de banda
5.3	Limitar utilizadores ligados a um ponto de acesso em simultâneo
5.4	Tratar conexão do proprietário
5.5	Tratar valores inválidos

Uma vez identificadas as *fatias* do caso de uso, passa-se à sua descrição, começando-se peça *fatia* 5.1 - Limitar o tráfego diário.

Tabela 40 - Fatia 5.1 "Limitar o tráfego diário"

Descrição	O sistema deve permitir ao proprietário da rede limitar o tráfego diário a terceiros que se conectem à sua rede.
Importância	Permitir que os proprietários das redes consigam controlar o tráfego feito por terceiros na sua rede.
Critério de teste	Uma vez definido um limite de tráfego diário pelo proprietário, o sistema conta o tráfego de terceiros que acedam à rede, impedindo a sua conexão caso seja atingido o limite imposto.
Complexidade de implementação	Média

Com esta *fatia* pretende-se impor um limite de tráfego a quem acede a uma rede partilhada. A dificuldade passa estudar quais os mecanismos nativos disponíveis para contagem de tráfego e impedir a conexão quando o limite definido pelo proprietário da rede for alcançado. Atribui-se assim uma dificuldade de implementação média.

Passa-se agora à *fatia* 5.2 - Limitar a largura de banda.

Tabela 41 - Fatia 5.2 "Limitar largura de banda"

Descrição	O sistema deve permitir ao proprietário da rede limitar a largura de banda disponível a terceiros que se conectem à sua rede.
Importância	Permitir que os proprietários das redes consigam controlar a largura de banda consumida por quem acedam à sua rede.
Critério de teste	Uma vez definido um limite de largura de banda pelo proprietário, o sistema limita a largura de banda a terceiros que acedam à rede.
Complexidade de implementação	Alta

Esta *fatia* visa a limitação da largura de banda máxima disponível a terceiros que acedam à rede partilhada. Não existem mecanismos nativos na plataforma *Android* que permitam tal limitação, mas como esta é baseada em *Linux* há a possibilidade de utilizar a ferramenta *iptables* para alcançar tal objetivo. Uma vez que não se possui experiência com tal ferramenta estima-se que seja necessário um esforço adicional para a sua implementação, atribuindo-se uma complexidade de implementação alta.

Passa-se, agora, à *fatia* 5.3 - Limitar utilizadores conectados em simultâneo.

Tabela 42 - Fatia 5.3 "Limitar utilizadores conectados em simultâneo"

Descrição	O sistema deve permitir ao proprietário da rede limitar o número de terceiros conectados à sua rede.
Importância	Permitir que os proprietários controlem o congestionamento das suas redes.
Critério de teste	Uma vez definido um limite de utilizadores conectados pelo proprietário, o sistema impede conexões de terceiros caso seja atingido esse limite.
Complexidade de implementação	Alta

Nesta *fatia* pretende-se limitar o número de utilizadores conectados a uma dada rede. Como não teremos controlo algum sobre os pontos de acesso, a tarefa de limitar o número de conexões a este será um desafio algo interessante. Por esse motivo atribui-se um nível de complexidade de implementação alto.

Segue-se a descrição da *fatia* 5.4 - Tratar conexão do proprietário.

Tabela 43 - Fatia 5.4 "Tratar conexão do proprietário"

Descrição	O sistema deve permitir que o proprietário da rede se conecte sem quaisquer limitações.
Importância	Não penalizar os proprietários.
Critério de teste	O sistema não aplica quaisquer restrições quando os proprietários se conectam às suas próprias redes.
Complexidade de implementação	Baixa

Nesta *fatia* dá-se importância ao facto de o proprietário aceder à sua própria rede, sendo injusto que este fique sujeito às limitações que definiu para terceiros. Pensa-se que o processo de detetar que o utilizador é proprietário da rede e impedir que sejam aplicadas as limitações não seja muito complexo, atribuindo assim uma complexidade de implementação baixa.

Apresenta-se, de seguida, a *fatia* 5.5 - Tratar valores inválidos.

Tabela 44 - Fatia 5.5 "Tratar valores inválidos"

Descrição	O sistema deve prevenir a introdução de valores inválidos.
Importância	Definir as restrições corretamente.
Critério de teste	O sistema aceita apenas valores válidos, mostrando uma mensagem de erro caso contrário.
Complexidade de implementação	Baixa

Aqui previne-se que sejam introduzidos valores inválidos para que se possam definir corretamente todas as limitações definidas anteriormente nas *fatias* 5.1, 5.2 e 5.3. A validação deve ser um processo simples, assim como a mostragem de erros, atribuindo-se assim uma complexidade de implementação baixa.

E, assim, termina-se a descrição das *fatias* do corrente caso de uso, passando-se agora à identificação dos demais elementos e respetivas interações envolvidas na realização deste caso de uso, representadas de seguida através de uma vista componente-conetor.

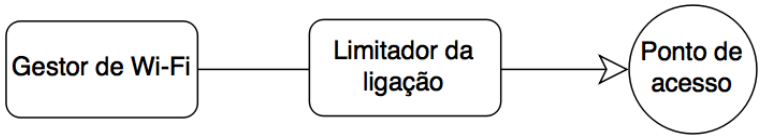

Figura 8 - Vista componente-conetor do caso de uso "Definir restrições de acesso"

Como referido anteriormente, o gestor de Wi-Fi será responsável por estabelecer a conexão às redes. De forma a conseguir restringir os demais parâmetros introduz-se o módulo limitador da ligação, que resulta das *fatias* 5.1, 5.2 e 5.3.

E, com esta vista, conclui-se a análise do corrente caso de uso – Definir restrições de acesso, passando-se de seguida ao caso de uso n.º 6 – Consultar informações da rede.

3.3.1.6. Consultar informações da rede

Descreve-se neste ponto o caso de uso n.º 6 – Consultar informações da rede, começando-se pelo seu fluxo através da tabela 45.

Tabela 45 - Fluxo de "Consultar informações da rede"

Fluxo básico	*Fluxo alternativo*
1. Selecionar rede	A1. Rede não partilhada
2. Verificar se está partilhada	
3. Mostrar informações da rede	

O fluxo deste caso de uso centra-se apenas na visualização das informações das redes Wi-Fi, sejam estas as redes partilhadas ou as redes do utilizador.

Através do fluxo básico identifica-se apenas um fluxo alternativo, o A1, que condiciona a informação a ser apresentada no passo 3.

Analisando o fluxo presente na tabela 45, obtém-se as demais *fatias* presentes na tabela 46.

Tabela 46 - Fatias do caso de uso "Consultar informações da rede"

N.º	*Fatia*
6.1	Visualizar informação das redes ao alcance
6.2	Tratar rede não partilhada

Com as *fatias* identificadas, passa-se à sua descrição individual, começando-se pela *fatia* 6.1 - Visualizar informação das redes ao alcance.

Tabela 47- Fatia 6.1 "Visualizar informação das redes ao alcance"

Descrição	O sistema deve apresentar as informações das redes que se encontram no seu alcance.
Importância	Suportar a tomada de decisão dos utilizadores sobre qual a rede a se conectar.
Critério de teste	O sistema apresenta as informações sobre a rede selecionada. Tais informações são: força do sinal, largura de banda e tempo de resposta.
Complexidade de implementação	Média

Esta *fatia* foca-se em apresentar as informações das redes ao alcance ao utilizador quando este faz uma seleção. As informações a serem mostradas são recolhidas no momento em que a rede é adicionada ao sistema, o que em princípio não deve requerer esforço adicional. A dificuldade estará em recolher

quais as redes que estão no alcance e mapeá-las às redes contidas no sistema. Assim, atribui-se uma complexidade de implementação média.

Apresenta-se, de seguida, a *fatia* 6.2 - Tratar rede não partilhada.

Tabela 48 - Fatia 6.2 "Tratar rede não partilhada"

Descrição	Devem ser mostradas as informações básicas caso a rede não esteja no sistema.
Importância	Dar informações mínimas sobre a rede selecionada.
Critério de teste	O sistema apresenta informação sobre a força do sinal.
Complexidade de implementação	Baixa

O foco desta *fatia* está em mostrar a força do sinal quando não temos informações mais detalhadas acerca de uma rede. Este processo deve ser relativamente simples, daí a atribuição de uma classificação baixa na complexidade de implementação.

E assim conclui-se a descrição das *fatias* deste caso de uso, passando-se agora à identificação dos demais elementos e respetivas interações envolvidas na realização do mesmo, apresentadas por meio de uma vista componente-conetor.

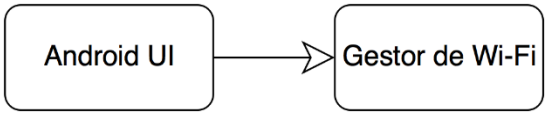

Figura 9 - Vista componente-conetor do caso de uso "Consultar informações da rede"

Este caso de uso é bastante simples, escolhendo-se apenas dois componentes para a sua realização: a interface *Android* e o gestor Wi-Fi. A interface apenas tem de recorrer ao gestor de Wi-Fi para obter as informações da rede e mostrá-las ao utilizador.

E, com esta vista, conclui-se a análise deste caso de usso, assim como a de todos os casos de uso do sistema.

No próximo ponto passa-se à descrição das restrições que terão de ser tidas em conta na construção do sistema.

3.3.2. Restrições

No ponto anterior descreveram-se os casos de uso do sistema juntamente com todas as suas *fatias*, que representam parte dos requisitos do sistema. Neste ponto são apresentadas as restrições sobre as quais o sistema deve operar, complementando o trabalho de documentação feito no ponto anterior.

Semelhantemente a como foram representadas as demais *fatias* dos casos de uso, recorre-se uma tabela para descrever cada restrição do sistema. Tais tabelas contêm, no seu cabeçalho, um número identificador e o tipo de restrição que representam. No seu corpo, inclui-se uma breve descrição do que se pretende, a sua importância para o sistema e finalmente o critério de teste que será usado para validar o seu cumprimento. Assim, apresenta-se a primeira restrição por meio da tabela 49.

Tabela 49 - Restrição n.º 1 "Portabilidade"

Descrição	O sistema deve correr sobre a plataforma *Android* 4.0 ou superior.
Importância	Manter uma base de possíveis utilizadores de pelo menos 90%.
Critério de teste	Um utilizador que tenha um dispositivo que corra sobre plataforma *Android* 4.0 ou superior deve conseguir utilizar a aplicação.

Na tabela anterior apresenta-se uma restrição de portabilidade, definindo-se que o sistema terá de correr sobre a plataforma *Android* 4.0 de modo a obter uma base de possíveis utilizadores de pelo menos 90% [25].

Segue-se agora a restrição n.º 2, descrita na tabela 50.

Tabela 50 - Restrição n.º 2 "Custo"

Descrição	O sistema deve conseguir disponibilizar as suas funcionalidades sem introduzir *hardware* adicional do lado dos utilizadores.
Importância	Manter os custos e complexidade baixos para o utilizador final.
Critério de teste	O utilizador deve conseguir utilizar o sistema recorrendo apenas ao seu dispositivo *Android*.

A restrição n.º 2 vem remover a possibilidade de introduzir qualquer tipo de *hardware* do lado dos utilizadores para que estes consigam conseguir partilhar e aceder às demais redes.

Apresenta-se, de seguida, a restrição n.º 3 através da tabela 51.

Tabela 51 - Restrição n.º 3 "Usabilidade"

Descrição	Os pontos de acesso das ligações a serem partilhadas não devem ser modificados.
Importância	Manter a complexidade do sistema relativamente baixa.
Critério de teste	Um utilizador consegue partilhar uma rede sem serem feitas quaisquer modificações no ponto de acesso.

Com esta restrição pretende-se que o sistema se mantenha com a filosofia de facilitar a partilha de uma rede, fazendo com que este processo seja o mais cómodo possível para o utilizador final.

De seguida, apresenta-se a restrição n.º 4 por meio da tabela 52.

Tabela 52 - Restrição n.º 4 "Usabilidade"

Descrição	Os utilizadores devem conseguir conectar-se a uma ligação partilhada facilmente.
Importância	Tornar este processo simples aos novos utilizadores.
Critério de teste	Um utilizador sem qualquer treino deve conseguir conectar-se a uma ligação partilhada em menos de 20 segundos.

Esta restrição segue a mesma filosofia que a anterior, mas para o processo de conexão, onde se pretende tornar o acesso às redes partilhadas o mais cómodo e intuitivo possível.

Passa-se, agora, à restrição n.º 5, descrita na tabela 53.

Tabela 53 - Restrição n.º 5 "Segurança"

Descrição	O sistema deve preservar a privacidade das informações das demais redes partilhadas.
Importância	Manter as informações seguras e não legíveis a acessos não autorizados.
Critério de teste	Agentes externos à plataforma não devem conseguir aceder aos dados das redes partilhadas. As senhas das demais redes partilhadas não devem estar legíveis aos utilizadores do sistema.

Com esta restrição pretende-se assegurar que apenas os utilizadores da plataforma recebem as informações das redes partilhadas, protegendo-as de ataques externos. Pretende-se também que as senhas não estejam visíveis mesmo até pelos utilizadores que recebam uma partilha.

Segue-se agora a restrição n.º 6, descrita na tabela 54.

Tabela 54 - Restrição n.º 6 "Armazenamento"

Descrição	O espaço reservado ao modo *offline* não deve exceder os 50MB no dispositivo.
Importância	Permitir que o sistema consiga correr numa vasta gama de dispositivos, mesmo naqueles mais limitados em termos de memória.
Critério de teste	Os dados armazenados pela aplicação no dispositivo do utilizador não devem exceder os 50MB.

Com esta restrição pretende-se manter o espaço ocupado pelas informações que tenham de ser armazenadas no dispositivo do utilizador, estipulando um limite de 50MB. Tal critério foi baseado no incentivo da Google aos desenvolvedores para distribuírem as suas aplicações com um tamanho base de até 50MB, distribuindo um pacote de expansão separadamente caso necessitem de mais dados [26].

Descreve-se, de seguida, a restrição n.º 7, através da tabela 55.

Tabela 55 - Restrição n.º 7 "Desempenho"

Descrição	O sistema deve preservar a bateria do dispositivo.
Importância	Permitir que os utilizadores usufruam o mais possível da aplicação.
Critério de teste	Os gastos de energia da aplicação no dispositivo do utilizador devem ser inferiores a 1% por hora.

O objetivo desta restrição é promover a eficiência das operações no dispositivo do cliente, evitando que a longevidade da bateria do dispositivo seja severamente afetada quando a aplicação estiver a correr.

E, com esta restrição, termina-se a descrição de todas as restrições do sistema, passando-se no próximo ponto à descrição da arquitetura proposta para a sua implementação.

3.4. Arquitetura

De modo a acomodar os requisitos acima descritos, é necessária uma arquitetura que consiga ultrapassar os desafios de forma eficaz e com o menor custo possível.

Os requisitos apresentados visam uma arquitetura com ênfase na troca de informação entre dispositivos, uma vez que sempre que um utilizador partilha uma rede com outro utilizador esta informação terá de chegar ao dispositivo do recetor da partilha.

Existe, também, a necessidade de incluir uma vertente de armazenamento persistente para permitir ao utilizador restaurar os seus dados caso haja perdas de informação no seu dispositivo. Esta vertente fica ainda mais evidente quando os requisitos apontam para uma função de sincronização dos demais dispositivos do utilizador, que não poderá tolerar quaisquer perdas de informação permanentes.

Uma vez feita a análise de todos os casos de uso, no final de cada descrição identificaram-se possíveis componentes e respetivas interações entre eles. Olhando para essa etapa de todos os casos de uso, consegue-se ter uma melhor perceção de uma possível arquitetura para o sistema.

Com base na análise dos demais casos de uso e das restrições abordadas anteriormente, chega-se finalmente à arquitetura do sistema, apresentada na figura 10.

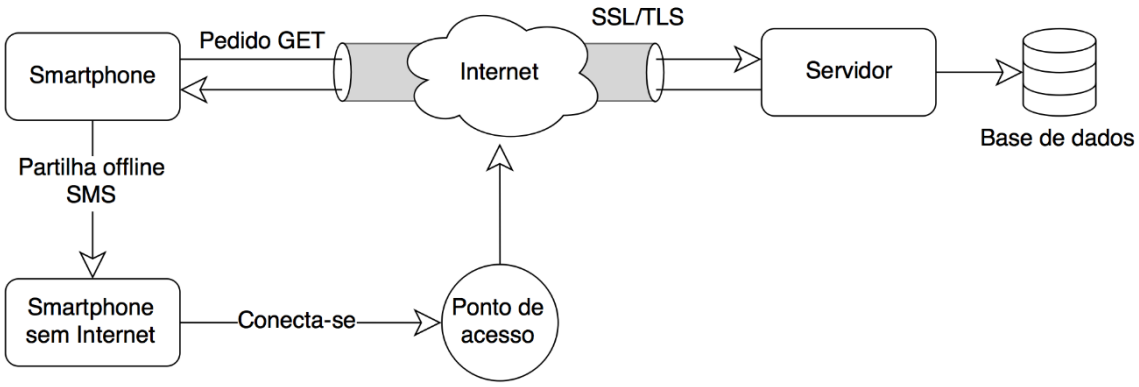

Figura 10 – Arquitetura

A arquitetura ilustrada na figura 10 é baseada numa arquitetura cliente-servidor, que consiste em fornecer dados aos demais utilizadores da plataforma através de pedidos GET HTTP a um servidor remoto, usando o protocolo de segurança TLS para proteger as comunicações.

A utilização de pedidos GET HTTP para transmissão de informação advém da sua versatilidade e simplicidade de transmitir dados entre as demais plataformas. No contexto deste trabalho de mestrado o sistema está a ser

implementado para a plataforma *Android*, mas caso seja necessária a sua expansão para outra plataforma, o método de transmissão não será certamente o ponto fraco. O envio destes dados é também protegido contra os demais ataques externos através da utilização do protocolo TLS.

O TLS (*Transport Layer Security*) é um protocolo de segurança pertencente aos padrões da IETF (*Internet Engineering Task Force*), largamente utilizado para proteger as comunicações feitas através da Internet. Este protocolo é uma evolução do SSL (*Secure Sockets Layer*) originalmente desenvolvido pela Netscape, renomeado para TLS em 1999. Normalmente o TLS é integrado nas aplicações para proteger os dados que são enviados entre clientes e servidores através de HTTP, também conhecido como HTTP sobre TLS (HTTPS) [27], [28].

A filosofia do TLS é a de garantir que as comunicações entre as entidades são privadas e de que os intervenientes na comunicação são realmente quem dizem ser. Para tal são utilizados mecanismos de autenticação baseados em certificados digitais, troca de chaves públicas e confidencialidade de dados usando um algoritmo de encriptação padrão (ex. AES) [28].

Para utilizar este protocolo de segurança apenas é necessário obter um certificado de uma entidade competente e instalá-lo no servidor que irá fornecer os dados.

A base de dados representa a parte persistente da informação dos utilizadores. Este armazenamento persistente irá atuar como cópia de segurança caso existam perdas de informação nos dispositivos dos utilizadores e servirá para difundir as demais redes partilhadas pelos vários utilizadores do sistema.

Para resolver o problema da partilha de uma rede com os utilizadores que estejam sem acesso à Internet, introduz-se o conceito de partilhar uma rede por SMS. Terá de ser desenvolvido um mecanismo que consiga de alguma forma codificar as informações pertinentes de uma rede, enviá-las por SMS e descodifica-las no dispositivo sem acesso à Internet de modo a ser possível este conectar-se à rede partilhada.

Esta arquitetura representa uma visão de alto nível de como o sistema deve operar, sendo esta complementada pela análise feita no fim de cada caso de uso por meio da análise das interações entre os possíveis componentes do sistema. Tal análise é ainda um pouco primordial e irá certamente evoluir à medida que o sistema é implementado, servindo para estabelecer um ponto de partida em rumo à solução final.

3.5. Conclusões

Neste capítulo definiram-se as funcionalidades a implementar tendo em conta o estudo das soluções existentes que foi elaborado no capítulo 3.

Com as funcionalidades estipuladas definiram-se os requisitos com recurso à metodologia Use Case 2.0 de modo a conseguir dividir cada caso de uso em *fatias* que possam ser testáveis na fase de desenvolvimento. O método seguido embora extenso é eficaz na descoberta e identificação de requisitos, usando uma linguagem simples e percetível tanto pelo cliente como pela equipa de desenvolvimento.

Posteriormente definiram-se as restrições e elaborou-se a arquitetura do sistema, que vem dar resposta não só às necessidades das *fatias* de cada caso de uso como também às restrições impostas.

Com base em toda a documentação produzida neste capítulo, dá-se então início ao desenvolvimento do sistema em concreto, descrito no próximo capítulo.

Capítulo 4

Implementação

4.1. Introdução

Com base na especificação feita no capítulo anterior, passa-se neste capítulo à escolha das tecnologias a utilizar e à implementação do sistema em concreto.

Começa-se pela definição da metodologia de desenvolvimento a adotar e à priorização dos casos de uso a implementar.

Depois, estudam-se as tecnologias, a utilizar para receber e enviar mensagens HTTP, mencionando-se que formato será usado na troca de mensagens, que tipo de servidor e *framework web* foram escolhidos para suportar o modelo de comunicação.

Definidas as tecnologias a utilizar segue-se a descrição do processo de desenvolvimento do sistema. Começa-se pela prototipagem da interface, seguindo-se da configuração do servidor, da base de dados e, finalmente, à implementação do sistema. A descrição do desenvolvimento aborda essencialmente as decisões mais importantes que foram tomadas ao desenvolver o sistema, em vez de se criar um guia extensivo da sua construção.

Assim, começa-se pela descrição da metodologia a seguir.

4.2. Metodologia

A metodologia a seguir consiste na utilização das *fatias* resultantes do levantamento de requisitos, feito no capítulo anterior, para planear e testar a implementação do sistema. A divisão dos casos de uso em *fatias* atómicas testáveis permite que se mapeie um determinado requisito (*fatia*) a um determinado *sprint*. Um *sprint* é uma terminologia usada no processo de

desenvolvimento SCRUM que se refere a um período de tempo ao qual se estabelece como meta o desenvolvimento de determinadas funcionalidades do sistema [24]. Assim, faz-se uso do conceito de *sprint* e define-se o número de sprints a realizar, a sua duração e as *fatias* a incluir em cada um deles, resultando daí um plano com metas temporais e objetivos testáveis (*fatias*). Com os *sprints* definidos pode-se, então, começar a implementar o sistema, dando início ao primeiro sprint definido no plano de desenvolvimento.

Tal plano de desenvolvimento é abordado em detalhe no próximo ponto.

4.3. Plano de desenvolvimento

Antes de se começar na implementação, é necessário estudar que tecnologias e ferramentas a utilizar de modo a que se consiga suportar as demais funcionalidades e cumprir com os requisitos estipulados. Para isso dedica-se a primeira tarefa ao estudo e definição das tecnologias que serão utilizadas do lado do servidor, tanto para armazenar as redes dos utilizadores como para efetuar a troca de informação com a plataforma *Android*. Com base na arquitetura, sabe-se que tais tecnologias terão de funcionar juntamente com os protocolos HTTP e TLS (HTTPS) para se efetuarem as comunicações. Com isso em mente, define-se o tipo de servidor, *framework* e base de dados a utilizar. Também se define o formato de dados que será utilizado nas comunicações entre o servidor e a aplicação *Android*.

Depois de definidas as tecnologias a usar, passa-se à instalação e configuração do servidor, da Framework e da base de dados.

Configuradas as ferramentas base para o arranque do desenvolvimento, avança-se para a implementação da aplicação *Android*, que irá requerer também implementação em paralelo do lado do servidor à medida que for necessária a criação de novos serviços. Uma tarefa fundamental nesta etapa de desenvolvimento são os testes unitários, que visam garantir o correto comportamento do sistema.

O processo de implementação do sistema será feito de forma incremental, dando uso à divisão dos casos de uso em *fatias* feita na fase de especificação do sistema, e ao conceito de *sprint* referido na secção anterior. A cada *sprint* decidiu-se atribuir um caso de uso por completo, priorizando-se as *fatias* a serem implementadas dentro de cada sprint de forma a promover a implementação das *fatias* que não dependem de outras. Pretende-se com isto aumentar a eficiência de implementação, tentando evitar que se efetuem mudanças nas implementações já concluídas ao começar noutra *fatia*. Será dado por concluído um *sprint* quando forem concluídas todas as *fatias* do caso de uso atribuído. O processo de implementação é, por sua vez, concluído quando se finalizarem todos os *sprints* definidos.

Assim, definem-se os *sprints* juntamente dos respetivos casos de uso a serem implementados em cada sprint:

1. *Sprint* 1 - Adicionar uma rede
2. *Sprint* 2 - Partilhar uma rede
3. *Sprint* 3 - Aceder a uma rede
4. *Sprint* 4 - Gerir redes
5. *Sprint* 5 - Definir restrições de acesso
6. *Sprint* 6 - Consultar informações da rede

Escolheu-se dar prioridade ao caso de uso “Adicionar uma rede” uma vez que é essencial adicionar redes ao sistema para que seja possível a realização dos casos de uso “Partilhar uma rede” e “Aceder a uma rede”.

De seguida, escolheu-se implementar esses mesmos casos de uso, “Partilhar uma rede” e “Aceder a uma rede”, de modo a ser possível realizar as funções essenciais da aplicação.

Uma vez que as *fatias* do caso de uso “Definir restrições de acesso” são mais complexas, decidiu-se implementar primeiro o caso de uso “Gerir redes”. Embora hajam mais *fatias* a implementar, estas são de menor complexidade, permitindo que sejam implementadas mais funcionalidades num menor espaço de tempo.

Implementados estes dois casos de uso apenas resta um último, “Consultar informações da rede”. Este foi deixado em último lugar na lista porque se achou que o seu impacto na funcionalidade do sistema não seria tão relevante quanto os restantes casos de uso.

Na priorização dos casos de uso descrita anteriormente é de notar que a preocupação inicial foi a de implementar as funcionalidades fulcrais para que o sistema demonstrasse a sua funcionalidade básica. Um sistema deve ser desenvolvido de forma incremental para que o cliente consiga constatar alguma funcionalidade adicional à medida que o sistema vai sendo desenvolvido. Neste caso não existe um cliente real, mas tal fator é essencial no desenvolvimento de um projeto de *software*, resultando daí a utilidade da partição dos casos de uso em *fatias*. Esta partição permite que se decida um plano temporal de implementação de modo a ir apresentando resultados que o cliente possa constatar, evitando que se passe muito tempo sem apresentar nada e se entregue o sistema todo de uma só vez.

Definidos os *sprints* passa-se ao planeamento temporal de conclusão, usando como informação de apoio a documentação produzida na secção 3.3 - Requisitos. Assim, define-se o seguinte plano de conclusão para os *sprints*:

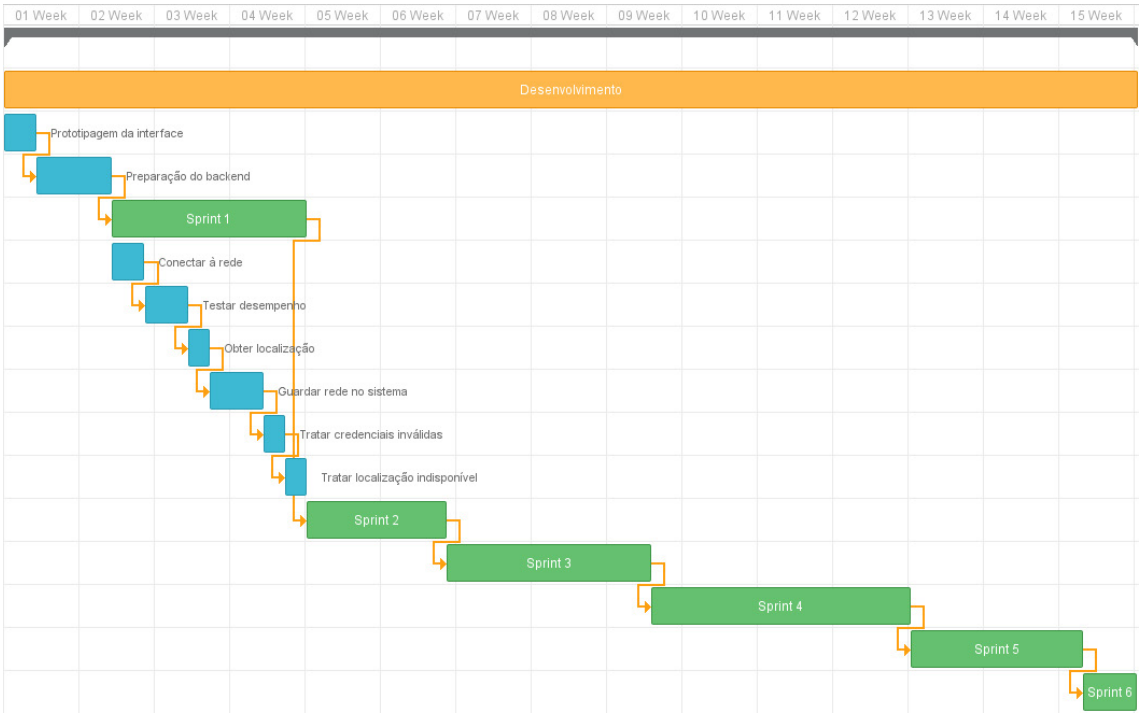

Figura 11 - Plano de desenvolvimento

A duração de cada *sprint* é calculada através da soma da duração estimada de todas as *fatias* atribuídas a cada sprint, sendo atribuídas durações de 2 a 5 dias a cada *fatia* consoante a sua dificuldade de implementação estimada. A título de exemplo, o *sprint* 1 tem a duração de 18 dias, em que a *fatia* 1.1 tem a duração de 3 dias, a *fatia* 1.2 de 4 dias, a 1.3 de 2 dias, a 1.4 de 5 dias e a 1.5 e 1.6 de 2 dias.

Uma vez terminados todos os *sprints*, faz-se a transição para a fase de produção do sistema, publicando-se a aplicação na *Play Store*, distribuidora oficial do mercado *Android*.

Assim, passa-se então à primeira etapa de desenvolvimento definida anteriormente, estudar e definir as tecnologias a utilizar para armazenar e trocar informação entre o servidor e o cliente *Android*.

4.4. Tecnologias

Com base na arquitetura e requisitos apresentados anteriormente, sabe-se que o sistema terá de possuir um servidor *web*, uma base de dados e utilizar os protocolos HTTP e TLS (HTTPS) para a troca de informação entre o servidor e o cliente *Android*.

Assim, abordam-se de seguida as tecnologias escolhidas para suportar a arquitetura definida na secção 3.4, começando-se pelo servidor *web*.

4.4.1. Servidor

Uma vez que será usado HTTPS para a comunicação com a aplicação *Android*, é necessário um servidor HTTP para dar resposta aos pedidos que serão feitos pelos utilizadores.

Um servidor HTTP é um programa que usa o protocolo HTTP para disponibilizar informação aos seus utilizadores, normalmente na forma de páginas *web*. Este servidor possibilita, normalmente, a utilização do protocolo de segurança HTTPS. O HTTPS consiste, basicamente, em utilizar um protocolo de segurança como o SSL ou TLS para proteger as comunicações entre duas entidades de ataques externos.

Existem diversos tipos de servidores HTTP como Apache, NGINX e IIS. Na implementação deste sistema será utilizado o servidor Apache, que é correntemente o mais utilizado para servir os mais variados *websites* a nível mundial. A escolha deste servidor baseia-se na experiência anterior que já se tem com este *software*, estando ele bem documentado e com uma vasta comunidade *online* sempre disposta a ajudar nos demais obstáculos que surgem diariamente. Como este é um *software* familiar, o tempo reservado ao seu manuseamento será largamente reduzido, um fator importante e decisivo em qualquer projeto.

O servidor necessita de ser devidamente programado para que se consiga enviar as respostas desejadas aos utilizadores. Para tal é necessário utilizar uma linguagem como Java ou PHP para manusear os dados e gerar as respostas a serem enviadas aos utilizadores. Tal assunto é abordado em mais detalhe no próximo ponto.

4.4.2. Framework

De modo a efetuar a programação do servidor para conseguir enviar as devidas respostas aos pedidos dos utilizadores, optou-se por utilizar uma micro-*framework web* PHP, a *Lumen*.

Uma *framework web* é uma implementação base para suportar o desenvolvimento de *websites*, oferecendo funcionalidades comuns que são normalmente necessárias na maioria dos projetos como o acesso à base de dados, gestão de sessão e validação de dados. É claro que a utilização de uma *framework* aumenta o *overhead* da aplicação (processamento extra para concluir uma determinada tarefa), devido às demais funcionalidades que a *framework* traz de raiz. Embora haja esta desvantagem, o tempo de

desenvolvimento das aplicações desenvolvidas com uma *framework* é drasticamente reduzido, uma vez que não se tem de implementar todas aquelas funcionalidades genéricas e medidas de segurança que a *framework* já traz consigo. As *frameworks* são programadas nas mais diversas linguagens de existentes, como Java, Ruby, Python e PHP. A *Lumen* é programada usando PHP, uma das linguagens mais utilizadas na *web*.

Uma das decisões a ter em conta na escolha desta *micro-framework* foi o seu *overhead*. A *Lumen* é basicamente uma versão muito simplificada da sua *framework* mãe, a Laravel, que possui um extenso leque de funcionalidades de raiz para responder às necessidades dos mais variados tipos de aplicações *web*. Esta simplificação permite que se diminua o *overhead* e que as respostas aos pedidos dos utilizadores sejam dadas de forma mais rápida e eficiente, aumentando o número de pedidos a que o servidor consegue responder num determinado período de tempo. Uma vez que não se necessita de funcionalidades muito complexas dado que a aplicação deverá fornecer serviços web, uma *micro-framework* com apenas as funcionalidades básicas é o ideal para este tipo de aplicação.

Um serviço *web* destina-se a fornecer informação não aos seus utilizadores humanos, mas sim aos programas que correm nos seus dispositivos. Um serviço *web* utiliza normalmente um formato de dados padrão como XML ou JSON para que os programas destinados a consumir os dados fornecidos pelo servidor possam extrair a informação de forma simples e eficiente.

Uma vez que a *Lumen* é baseada na Laravel, a sua sintaxe e utilização são muito semelhantes. Tal semelhança é também muito benéfica uma vez que já se tem alguma experiência no manuseamento da Laravel. Tal conhecimento irá eventualmente contribuir na redução do tempo de implementação das funcionalidades necessárias na componente do servidor.

Segundo a arquitetura apresentada no capítulo anterior, para que se consigam armazenar as informações dos utilizadores de forma persistente, é necessária uma base de dados. Aborda-se tal assunto no próximo ponto.

4.4.3. Base de dados

Uma vez que se vai utilizar a framework *Lumen*, esta suporta apenas 4 sistemas de bases de dados: MySQL, Postgres, SQLite e SQL Server.

Todas as soluções suportadas pela *framework Lumen* podem ser usadas gratuitamente, embora o SQL Server imponha algumas limitações como o tamanho máximo de 10GB à base de dados e 1GB de utilização de memória.

O SQLite é desenhado para aplicações pequenas e simples, normalmente usado em aplicações para *smartphones* quando é necessário armazenar uma maior quantidade de informação localmente. Restam-nos assim duas opções, o MySQL e o Postgres, uma vez que o SQL Server possui limitações que não estão presentes nestas duas soluções de código aberto.

Escolheu-se utilizar o MySQL uma vez que este é um sistema de base de dados de código aberto e do qual se tem experiência no seu manuseamento. Este está bem documentado e possui diversas ferramentas disponíveis como o MySQL Workbench e o phpMyAdmin que tornam o desenvolvimento de aplicações com esta tecnologia muito mais agradável. Este sistema de base de dados é correntemente o mais usado entre os sistemas de código aberto, embora outras tecnologias como MongoDB tenham vindo subir em popularidade nos últimos anos. A popularidade da solução escolhida é propícia a que se encontrem soluções através da comunidade de forma mais rápida para os problemas que possamos vir a encontrar ao longo da fase de desenvolvimento do sistema.

Definido o sistema de base de dados a utilizar, passa-se agora à escolha do formato de dados a adotar para efetuar a troca de informação entre as aplicações *Android* e *Lumen*, abordando-se tal assunto no próximo ponto.

4.4.4. Formato de comunicação

Para se conseguir comunicar de forma eficiente e rápida, é necessário utilizar um formato de dados que seja leve e simples de ler tanto do lado da aplicação *Android* como do lado do servidor. Quando se fala em leve e simples, quer-se dizer que o tamanho dos dados tem de ser o mais curto possível, ao nível da quantidade de dados a enviar, e a sua leitura tem de ser feita de forma eficiente, gastando o menor tempo de processamento possível.

Dois dos formatos que se enquadram neste perfil são o XML e o JSON. O XML é uma *markup language*, ou linguagem de marcação, que usa um conjunto de regras para a separação e representação de conteúdo num ficheiro de texto. A estruturação dos dados é feita através de etiquetas com nomes alusivos para separar devidamente os dados e facilitar a sua leitura, tanto pelos programas como pelas pessoas que o vão analisar. O JSON é baseado na notação de objetos oriunda do JavaScript, representando os dados através de pares de valores. Esta notação é fácil de manusear e de ler, tanto por parte de humanos como dos programas encarregues de analisar os dados.

Estes dois formatos são muito distintos, uma vez que o XML permite a representação de dados mais complexos e a inclusão de ficheiros, enquanto o JSON apenas permite a representação de dados básicos como números e texto.

Tal simplicidade torna-se benéfica em termos de segurança, uma vez que se põe de parte a possibilidade de se introduzir ficheiros maliciosos nas mensagens.

Em termos de desempenho, o JSON consegue atingir tamanhos mais reduzidos e velocidades superiores de criação e leitura de ficheiros, dependendo também da biblioteca utilizada [29].

Em termos de legibilidade, decidir qual a melhor pode ser muito discutível uma vez que cada um tem preferências distintas. Neste caso, tem-se preferência pelo formato JSON para representar os dados, uma vez que se acha que esta notação é muito mais intuitiva em relação ao XML.

Em termos de compatibilidade, ambas as plataformas *Android* e PHP têm mecanismos para efetuar a análise dos dados tanto no formato XML como em JSON.

Dados os factos acima mencionados, escolhe-se o JSON como formato padrão para a troca de dados entre a plataforma *Android* e a *framework Lumen*, uma vez que a seu desempenho e legibilidade são superiores ao XML sem trazer quaisquer desvantagens de compatibilidade.

Assim, termina-se a discussão acerca das tecnologias a utilizar do lado do servidor, passando-se no próximo ponto ao arranque da etapa de implementação do sistema.

4.5. Prototipagem da interface

Antes de começar a escrever código, é necessário fazer um esboço inicial (protótipo) da interface. Tal protótipo serve de auxílio não só na implementação da interface à medida que são implementadas as demais *fatias*, como também na identificação eventuais lacunas de usabilidade.

Inicialmente, elaborou-se o esboço utilizando um método convencional e flexível: desenho em quadro branco com marcador. Este método é muito útil para esboços iniciais uma vez que se conseguem fazer experiências e alterações de forma muito rápida. Quando se chegar a um ponto em que se esteja mais confiante com o resultado obtido, pode-se passar o esboço do quadro branco para um formato digital, aumentando um pouco mais a fidelidade do protótipo.

Um dos esboços iniciais apresenta-se como se segue:

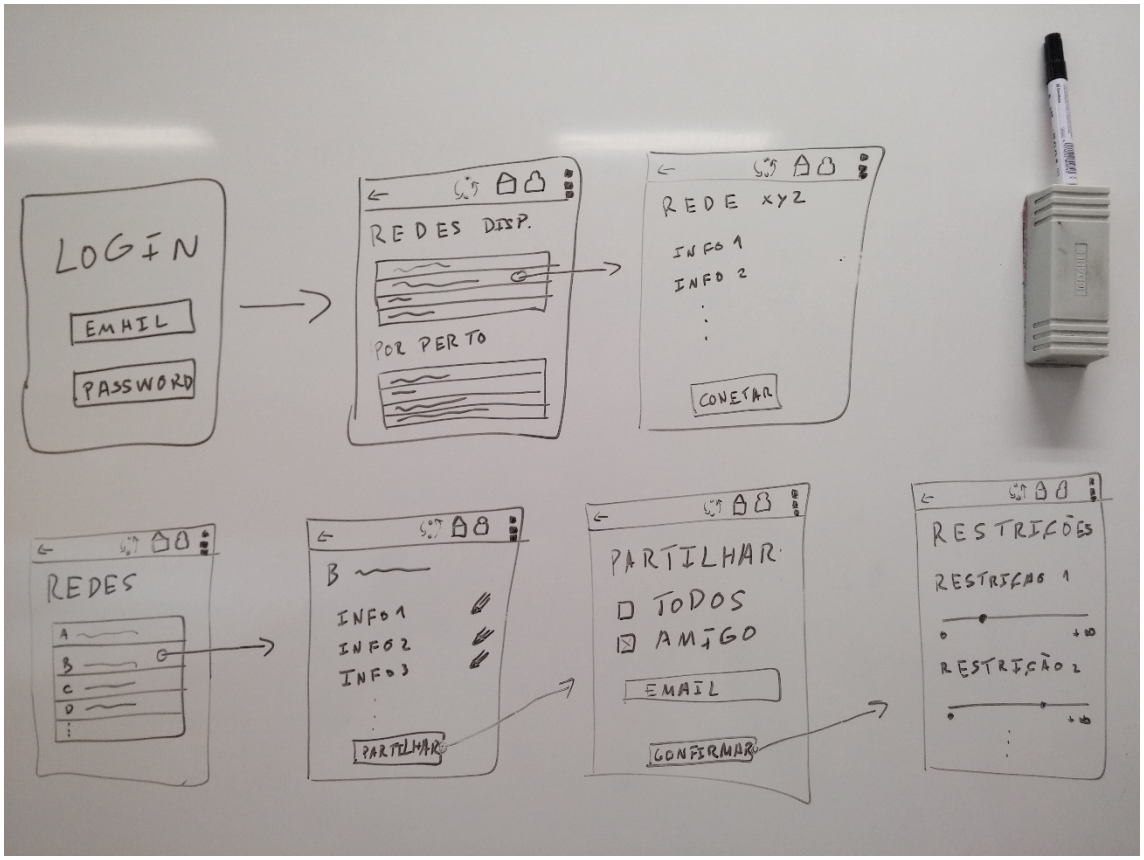

Figura 12 - Esboço inicial da interface

De seguida, fez-se mais uma iteração e passou-se para formato digital aumentando o nível de fidelidade do protótipo.

Figura 13 - Protótipo digital

Figura 14 - Protótipo digital

Com os protótipos da interface concluídos, passa-se à configuração do servidor antes de iniciar a escrita do sistema.

4.6. Configuração do servidor

Antes de começar com a implementação da aplicação *Android*, achou-se pertinente configurar previamente o servidor para que, quando for necessário implementar uma *fatia* que necessite de alguma implementação na componente do servidor, este já esteja devidamente operacional.

Para tal, foi instalado o servidor Apahe 2.4, o MySQL Server 5.6 e o PHP 5.6. O apache é o servidor *web* que vai servir os dados gerados através do PHP e armazenados na base de dados MySQL.

O próximo passo é a instalação de um certificado x.509 para ser possível utilizar o protocolo TLS nas comunicações. O certificado a ser utilizado é auto assinado de modo a reduzir os custos de implementação.

Os certificados x.509 podem custar desde algumas dezenas até alguns milhares de euros. Quanto mais caro o certificado, não só há mais rigor no processo de validação como também há um maior nível de confiança em tal certificado. Um certificado auto assinado é um certificado em que a entidade que certifica é a mesma que a entidade certificada, ou seja, quando uma entidade se certifica a si mesma. A desvantagem destes certificados é que não são confiados pelos *browsers web*, sendo apresentada uma mensagem aos utilizadores de que a ligação é insegura. Embora seja mostrada esta mensagem, as comunicações continuam a ser protegidas.

Para a emitir um certificado auto assinado é necessário, no nosso caso, aceder ao servidor Apache e executar os devidos comandos para a sua criação. Os ficheiros resultantes deste processo têm então de ser copiados para as devidas pastas, de modo a que o servidor possa aceder aos mesmos. Tal processo é descrito mais detalhe em [30].

Instalado o certificado, é necessário definir a prioridade dos protocolos a utilizar caso o cliente não consiga utilizar o protocolo mais seguro possível. Para tal, temos que definir uma propriedade denominada "CipherSuite" nas configurações do servidor Apache. Tal opção foi configurada para priorizar os protocolos da seguinte forma: TLS 1.2 > TLS 1.1 > TLS 1.0 > SSL 3 > SSL 2 [31].

Uma vez priorizados os protocolos mais seguros e eliminados da lista os considerados inseguros, torna-se necessário testar a segurança das configurações efetuadas com recurso à ferramenta de teste online oferecida pela Qualys SSL Labs [32]. Esta ferramenta analisa e identifica possíveis falhas nas configurações de um servidor *web* que utilize o protocolo HTTPS para proteger as suas comunicações. Tais falhas podem estar relacionadas com a inclusão de versões inseguras dos protocolos de segurança na lista de possíveis protocolos aceites para proteger as comunicações.

Passados os testes e com confiança de que as comunicações serão devidamente protegidas, procedeu-se à instalação e configuração da *framework Lumen*, seguindo-se os passos presentes na documentação oficial da mesma [33].

4.6.1. Base de dados

De modo a se conseguir armazenar as demais redes, é necessária uma base de dados que consiga dar resposta às necessidades do sistema. A base de dados tem de possibilitar a associação de uma rede a um utilizador assim como permitir a sua partilha com outros utilizadores do sistema. Esta deverá também possibilitar a associação das restrições de desempenho a cada partilha.

Para tal, desenhou-se a base de dados usando a ferramenta MySQL Workbench, resultando no diagrama entidade-relação apresentado na figura 15.

O diagrama elaborado permite que se associe um utilizador a várias redes através da relação 1-N entre as tabelas "users" e "connections". Para possibilitar a partilha de uma rede com outros utilizadores, utiliza-se uma relação 1-N entre as tabelas "connections" e "shares" e uma outra relação N-N entre as tabelas "shares" e "users", sendo as restrições de desempenho definidas nas entradas da tabela "shares". A tabela "connected_users" servirá para a implementação do controlo de utilizadores conectados que será abordado mais à frente.

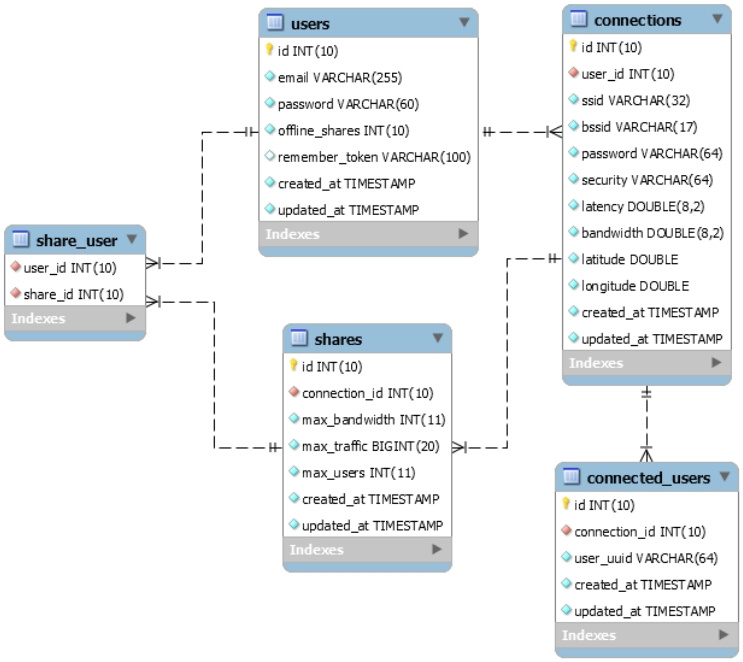

Figura 15 - Diagrama ER

Com o esquema da base de dados definido, foram usadas as funcionalidades fornecidas pela *framework Lumen* para trabalhar com a base de dados. Tais funcionalidades são as "migrations", que possibilitam o controlo de versões da base de dados, e o módulo Eloquent que trata do acesso à base de dados [34], [35].

Terminada a descrição da base de dados utilizada, abordam-se no próximo ponto os aspetos mais interessantes e cruciais que tornam possível o bom funcionamento do sistema.

4.7. Desenvolvimento

4.7.1. Ferramentas

No servidor, no desenvolvimento da aplicação *Lumen*, foi usada a ferramenta PhpStorm da JetBrains para a escrita de toda a sua componente lógica, uma vez que se tem preferência pelo seu vasto leque de funcionalidades. Para fazer o *debugging* dos serviços *web* a serem desenvolvidos, utilizou-se a ferramenta Postman, disponibilizada através de uma aplicação para o Chrome, e a ferramenta online Base64 Encode para codificar e descodificar informação manualmente.

No desenvolvimento da aplicação *Android* foi usado o SDK fornecido pela Google, o Android Studio. Esta ferramenta permite a criação tanto da interface gráfica como da lógica da aplicação. Para a criação da interface, a ferramenta

SDK recorre à notação XML, sendo que para a lógica recorre à linguagem de programação Java. Para fazer o *debugging* e teste da aplicação *Android* foram utilizados os dispositivos LG G2 e Huawei Y530 em conjunto com os mecanismos disponibilizados pela ferramenta Android Studio, nomeadamente Android Instrumentation e JUnit. Foram usados os dispositivos LG G2 e Huawei Y530 para ser possível testar o desempenho da aplicação em dispositivos com diferentes níveis de desempenho, representando o LG G2 os dispositivos de elevado desempenho e o Huawei Y530 os de baixo desempenho.

4.7.2. Decisões de desenho

Ao longo do desenvolvimento das várias *fatias*, uma das principais preocupações foi sempre o bom desempenho da interface com o utilizador. Para tal, usaram-se execuções assíncronas nas operações mais intensivas e propícias a bloquear as operações na interface, como pedidos ao servidor e mecanismos de conexão às redes Wi-Fi.

De modo a lidar com estas execuções assíncronas, optou-se por usar conceitos como a delegação de funcionalidades e *callbacks*.

A delegação consiste em delegar uma ou mais tarefas a um outro módulo, seguindo o princípio da inversão de responsabilidades. Tal princípio de desenho promove a modularidade e extensibilidade do sistema ao isolar novos processos num módulo em separado.

Uma vez que se promovem as execuções assíncronas, torna-se necessário introduzir o conceito de *callbacks* de modo a possibilitar a comunicação com a interface uma vez concluídas as operações em execução. Este conceito é usualmente utilizado na implementação do padrão de desenho Observer, que visa notificar determinadas entidades quando ocorrem mudanças no sistema. Neste caso, as notificações são feitas apenas para uma entidade, uma Activity. Cada objeto Activity é responsável por gerir as operações que se fazem num dado ecrã da aplicação, incluindo as operações da interface e pedidos ao servidor, por exemplo, resultando daqui a necessidade destes dois conceitos de delegação e *callbacks*.

Um outro exemplo é o processo de conexão, representado pelo seguinte diagrama de classes apresentado na figura 16.

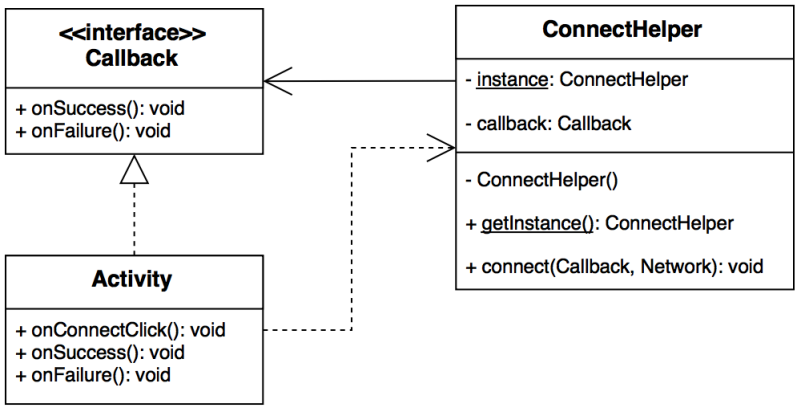

Figura 16 – Delegação e Callback

No diagrama anterior constata-se que a "Activity" delega ao "ConnectHelper" o processo de conexão a uma rede, fazendo uso do método "connect". A "Activity" implementa também a interface "Callback" para que o módulo "ConnectHelper" possa comunicar de volta o sucesso ou falha uma vez terminado o processo de conexão a uma rede.

O diagrama anterior ilustra também o uso do padrão de desenho Singleton, que visa criação e utilização de apenas um objeto do tipo ConnectHelper através de toda a aplicação. O uso deste padrão permite que se diminua o *overhead* associado à constante instanciação de objetos ConnectHelper, sempre que se pretenda aceder a uma rede em diferentes locais da aplicação, permitindo também a poupança de memória ao garantir que apenas uma instância é criada e mantida em memória.

Uma vez que todas as funcionalidades do sistema são centradas nas redes Wi-Fi, foi também usado o padrão de desenho Observer para notificar todos os módulos que necessitem de reagir prontamente à alteração das redes armazenadas no dispositivo. Tais mudanças podem influenciar tanto a atualização da interface como os serviços que correm assincronamente. A título de exemplo, temos o processo de sincronização, que pode alterar não só as redes a serem mostradas na interface como também as redes às quais a funcionalidade de conexão automática se poderá conectar.

A figura 17 representa a estrutura implementada para lidar com a notificação de alterações às redes armazenadas no dispositivo do utilizador. O PreferenceHelper fica responsável por guardar e ler as redes armazenadas nas SharedPreferences, um repositório privado de preferências que cada aplicação *Android* dispõe para armazenar dados. Sempre que é feita uma alteração numa rede, todos os PreferenceObserver são notificados. Correntemente, apenas existe um, a CustomApplication, existindo sempre flexibilidade de se adicionar mais no futuro. A CustomApplication, ao ser notificada de que foram efetuadas alterações, atualiza as redes em memória e notifica todos os NetworksObserver de tais alterações.

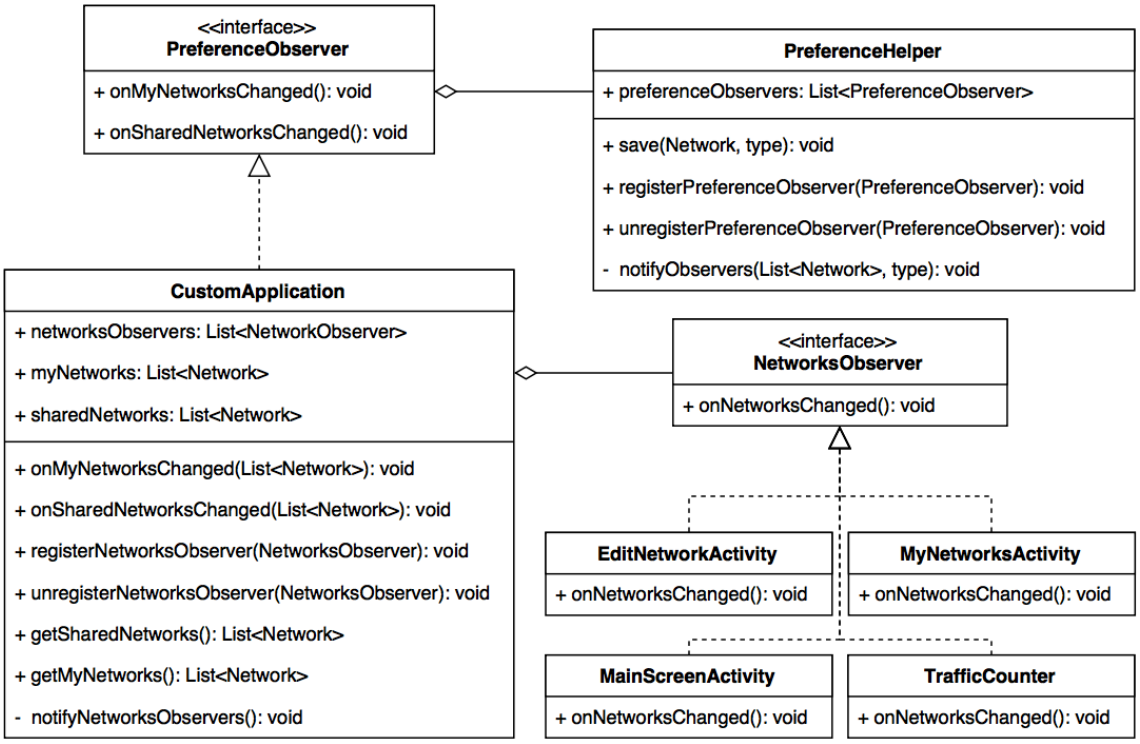

Figura 17 - Observers

Através desta estruturação com base em *observers* é possível aumentar largamente a eficiência da aplicação, ao garantir que as redes não são duplicadas na memória principal através dos diversos módulos e ao proporcionar que o acesso às mesmas seja feito eficientemente através da memória principal. Tal forma de acesso em memória principal evita que se aceda à memória secundária sempre que se necessite de informações de uma dada rede, uma vez que a memória secundária é muito mais lenta e requer mais tempo de processamento para efetuar a leitura da informação.

Um problema que surge é a concorrência no acesso às redes na memória principal. Tal problema é resolvido através da utilização de uma lista síncrona (CopyOnWriteArrayList) para armazenar as demais redes em memória, eliminando o problema de acessos concorrentes.

A operação mais intensiva que é feita em toda a aplicação é a sincronização, uma vez que esta envolve pedidos ao servidor e uma quantidade enorme de processamento e memória caso esteja presente um grande número de redes. Neste processo, faz-se um pedido ao servidor em que este retorna as redes do utilizador no formato JSON, isolando-se também este processo num módulo à parte, o Sync.

Para manusear todo o processo de fazer pedidos ao servidor e receber a informação retornada por este, foi criado o módulo API. Este módulo trabalha de forma assíncrona e faz também uso de *callbacks* para comunicar de volta a resposta do servidor ao módulo Sync.

Para ser mais cómodo manusear todas estas redes, foi criada uma classe Network, que uma vez instanciada, efetua a leitura de todas as informações de uma mensagem JSON retornada pelo servidor. Para lidar com a partilha foi criada uma classe Share, usada pela classe Network, que também dada uma mensagem em JSON com as informações de partilha extrai toda a informação de forma automatizada. Nestes dois módulos implementou-se também a sua conversão para JSON de modo a facilitar e padronizar tanto a troca de informação entre a aplicação *Android* e o servidor. Para armazenar as redes no dispositivo do utilizador e permitir o acesso *offiline*, padronizou-se também a sua conversão para formato de texto, permitindo o seu armazenamento no repositório SharedPreferences mencionado anteriormente. Com isto, isola-se toda a lógica e complexidade de troca e armazenamento das informações das redes em objetos do tipo Network e Share, permitindo mais flexibilidade a nível da escrita de código e promovendo o isolamento de um eventual ponto crítico de falha.

Um dos maiores problemas da sincronização é o elevadíssimo consumo de memória e tempo de processamento. Para tal, foi criado um *buffer* intermédio para armazenar as redes antes de serem gravadas na memória secundária, fazendo-se tal processo de uma só vez de modo a reduzir o tempo total de processamento. Para resolver o problema do elevado consumo de memória são dadas dicas ao *garbage collector* para eliminar variáveis desnecessárias em memória ao atribuí-las um valor nulo à medida que se tornam desnecessárias.

Depois do processo de sincronização, estas redes ficam agora disponíveis para serem partilhadas com outros utilizadores. A partilha pública e seletiva é feita através de um pedido ao servidor, enquanto a partilha *offline* requer um pouco mais de engenhosidade.

Para se efetuar a partilha sem acesso à Internet escolheu-se usar uma mensagem de texto (SMS) como método de transmissão. Uma das formas de fazer passar informação de um SMS para a aplicação *Android* é através da associação de um endereço *web*, com um determinado padrão, como forma de abertura da aplicação. Assim, quando o utilizador tentar abrir o endereço, será questionado se o deseja fazer com o Wi-Share. A título de exemplo temos o seguinte endereço: www.server.com/data/, onde "data" representa um objeto Network em formato de texto codificado em Base64, de forma a evitar erros de transmissão. Assim, e seguindo o exemplo anterior, sempre que se tente abrir o endereço com o formato www.server.com/.../, o utilizador será questionado se o quer fazer com o Wi-Share, possibilitando que se consigam ler as informações da rede partilhada por SMS e se consiga efetuar a conexão à mesma. Uma vez recebidos os dados pela aplicação, é usado o módulo ConnectHelper para fazer a conexão à rede.

À medida que o utilizador vai recebendo partilhas de várias redes, o módulo de conexão automática (AutoConnect) verifica se as redes nas

proximidades estão partilhadas, estabelecendo a conexão caso isso se verifique. Um dos problemas que surgem é: que rede escolher quando estão disponíveis várias redes nas proximidades?

Tal questão é discutida em [36], onde é utilizado o SNIR (*Signal to Interference and Noise Ratio*) como parâmetro de escolha da melhor rede, definido pela seguinte equação: $SNIR = \frac{RSSI_{current}}{\sum_{x \in S} L_x \times RSSI_x + N}$.

Na plataforma *Android* não é possível saber o número de dispositivos a comunicar num determinado canal (L) nem ter acesso ao nível de ruído do sinal proveniente de um ponto de acesso (N). A solução para o primeiro obstáculo é a utilização do número de pontos de acesso que operam no mesmo canal (ex. 2412MHz), enquanto para o segundo problema não há solução e tal parâmetro terá de ser excluído da equação do SNIR.

Como se conhece a largura de banda das ligações, achou-se pertinente introduzi-la na equação de modo a compensar as lacunas deixadas pela modificação anteriormente descrita. Assim, a equação final para o parâmetro de seleção é a seguinte: $T = \frac{RSSI_{current}}{\sum_{x \in S} C \times RSSI_x} \times B$.

Na equação apresentada, o RSSI representa a intensidade do sinal em mili-watts, x representa um ponto de acesso num conjunto S de pontos de acesso, C representa o número de pontos de acesso no canal de transmissão do ponto de acesso x e B representa a largura de banda em Mbps do sinal a ser testado ($RSSI_{current}$).

Para escolher a melhor rede procede-se ao cálculo do parâmetro T para todas as redes ao alcance e escolhe-se aquela que obtiver o valor mais elevado. Uma vez encontrada a melhor rede, procede-se à conexão a esta recorrendo ao módulo ConnectHelper.

Como se tratam de redes partilhadas com o utilizador, estas podem ter restrições de acesso impostas por quem as partilhou. Para tal, são necessários mecanismos que consigam controlar o tráfego, a largura de banda disponível e o número de utilizadores ligados à rede.

Para a contagem de tráfego, foram utilizados os mecanismos nativos da plataforma *Android* que permitem a leitura do tráfego consumido pelo *smartphone* do utilizador. Tal leitura é feita através do módulo TrafficCounter, de dois em dois minutos, de modo a conservar a bateria do dispositivo. O módulo TrafficCounter é executado de forma assíncrona para preservar o bom desempenho da interface do utilizador.

De modo a conseguir limitar com sucesso o número de utilizadores conectados em simultâneo a uma rede, foi utilizado um mecanismo de 'autenticação virtual' a estes. O termo virtual advém do facto de que se regista

que um utilizador se conectou a um determinado ponto de acesso através de um serviço *web* disponibilizado pela componente do servidor, embora não se saiba quem se conectou mas sim que dispositivo se conectou através da utilização do seu identificador único (IMEI).

De modo a atualizar este registo e permitir que outros utilizadores se conectem à medida que os utilizadores se vão desconectando, foi implementado um mecanismo de *heartbeat* com um intervalo de 2 minutos de modo a confirmar que o utilizador ainda se encontra a utilizar o ponto de acesso. A partir do momento que o utilizador se desconecte do ponto de acesso e deixe de responder ao *heartbeat*, o servidor deteta que o último sinal foi dado á mais de 2 minutos e elimina o registo desse utilizador, possibilitando que outro se possa conectar no seu lugar. O intervalo de dois minutos foi estipulado de forma a conservar a bateria do dispositivo e minimizar o número de pedidos ao servidor.

Ao fazer um levantamento das aplicações existentes, deparou-se com a aplicação de código aberto BradyBound, que disponibiliza a limitação da largura de banda tanto das redes móveis como de redes Wi-Fi. Com a devida autorização do seu criador, Mansour Behabadi, foi feita a integração do seu módulo que permite a limitação da largura de banda. Este módulo controla a largura de banda da ligação criando regras de tráfego através, da ferramenta *iptables*, para descartar pacotes TCP e obrigar este protocolo a adaptar a velocidade de transmissão de acordo com a quantidade de pacotes descartados. Tal não é possível para o protocolo UDP, uma vez que este não possui tais mecanismos. Aponta-se o facto de este módulo apenas funcionar em dispositivos com permissões especiais, uma vez que a plataforma *Android* não possui mecanismos nativos que permitam a limitação da largura de banda [37].

Uma adversidade ainda não mencionada foi a obtenção dos parâmetros de tempo de resposta e largura de banda de uma ligação aquando da sua adição ao sistema. Uma vez que a máquina que aloja o servidor *web* sofre de escassez de largura de banda, para se poder testar com alguma confiança o desempenho de uma ligação decidiu-se recorrer ao serviço OneDrive da Microsoft. O OneDrive é um serviço de armazenamento na cloud, funcionando como um disco rígido online onde se podem armazenar e aceder a ficheiros pessoais remotamente através da Internet. O método usado consiste em armazenar um ficheiro no OneDrive e efetuar a sua transferência para o dispositivo do utilizador quando for necessário medir o desempenho da sua ligação. Para tal, mede-se o tempo que o servidor demora a responder ao pedido de transferência assim como a duração da transferência do ficheiro, conseguindo-se obter assim os parâmetros necessários. A demora a responder ao pedido de transferência é usada como o tempo de resposta e a largura de banda é calculada dividindo o tamanho do ficheiro pela duração da transferência deste.

Por fim, criou-se o módulo DialogFactory para construir as mensagens a serem mostradas ao utilizador sempre que seja necessário notificar a ocorrência

de um erro ou conclusão do processo de sincronização, por exemplo. Embora esta não seja uma implementação à risca do padrão de desenho factory, a inspiração veio desse mesmo padrão, com o objetivo de centralizar e facilitar a criação de objetos do tipo AlertDialog ao tratar de toda a lógica de inicialização dos diversos tipos de mensagens. Este módulo suporta, não só a criação de mensagens à base de texto, como também a criação de um tutorial baseado em imagens. Este tutorial, a ser mostrado quando o utilizador corre a aplicação pela primeira vez, tenta introduzi-lo às demais funcionalidades ao seu dispor.

Assim, termina-se a descrição dos conceitos mais importantes utilizados no desenvolvimento do sistema, passando-se no próximo ponto à descrição da entrada do sistema em produção.

4.8. Fase de produção

Para a entrada na fase de produção, optou-se por colocar a aplicação no mercado de aplicações *Android*, a *Play Store*, para facilitar a distribuição aos utilizadores que se disponibilizem a testar a aplicação.

Foi utilizada a ferramenta Bugsnag para permitir a recolha de informação de contexto, como a *stack trace*, sobre eventuais erros que possam vir a ocorrer durante a fase de produção. Na ocorrência de um erro inesperado, tanto na aplicação *Android* como na *Lumen*, esta ferramenta recolhe informações acerca da origem desse mesmo erro e disponibiliza essas informações numa plataforma online para possibilitar uma fácil gestão dos diversos erros ocorridos. A ferramenta Bugsnag permite também adicionar mensagens, ou pistas, a certas partes do código, que são enviadas para esta plataforma online aquando da ocorrência de um erro, possibilitando uma identificação mais eficaz do problema. Assim, esta ferramenta facilitará a identificação e correção de eventuais erros que possam vir a ocorrer durante a fase de testes com utilizadores.

Uma vez integrado este serviço, publica-se a aplicação na *Play Store* e dá-se início à fase de testes com utilizadores, descrita no próximo capítulo.

4.9. Conclusões

Neste capítulo descreveu-se o processo de desenvolvimento do sistema, abordando-se a metodologia adotada, as tecnologias utilizadas, a prototipagem da interface, as configurações das ferramentas e finalmente a construção e escrita do sistema.

A metodologia de desenvolvimento foi baseada na Use Case 2.0 apresentada por Ivar Jacobson. Esta visa fazer uso das *fatias* definidas na fase

de especificação e usá-las, não só para planear os *sprints*, como também para validar a implementação das funcionalidades.

Depois, escolheram-se as tecnologias a usar. O servidor *web* utilizado foi o Apache, a *framework web* a *Lumen* e o formato de dados para a troca de informação entre a aplicação *Android* e o servidor foi o JSON.

Elaborou-se, primeiramente, um protótipo em quadro branco antes de se passar à escrita da aplicação, refinando-se este protótipo através de uma outra iteração em formato digital.

O próximo passo foi a configuração do servidor Apache e da *framework Lumen*, seguindo-se a modelação da base de dados.

Com as configurações feitas do lado do servidor, passou-se à construção do sistema dando início ao primeiro *sprint* e implementando as *fatias* consoante a ordem estipulada.

Concluídos todos os *sprints*, é publicada a aplicação, dando-se por concluída a fase de desenvolvimento e passando-se à fase de testes. Foram realizados testes unitários para garantir o cumprimento dos requisitos, testes de desempenho para assegurar o bom funcionamento da aplicação e testes com utilizadores para perceber que finalidade os utilizadores utilizaram a aplicação criada. Tais tópicos serão discutidos no próximo capítulo.

Capítulo 5

Testes e resultados

5.1. Introdução

No capítulo anterior foi concluída a implementação e posto o sistema em fase de produção, publicando-se a aplicação *Android* na loja de aplicações *Play Store*.

No presente capítulo abordam-se os testes unitários efetuados ao longo do desenvolvimento, os testes de desempenho antecedentes à entrada em fase de produção e, finalmente, os testes com utilizadores efetuados uma vez colocado o sistema em produção.

Começa-se, então, por abordar a temática dos testes unitários.

5.2. Testes unitários

Os testes unitários são importantes para que se assegure o bom e correto funcionamento do sistema, à medida que se implementam novas funcionalidades e se corrigem erros.

Os testes unitários foram realizados utilizando as ferramentas já embutidas no Android SDK (Android Instrumentation e JUnit), em conjunto com as *frameworks* Mockito e PowerMock.

A ferramenta Android Insrumentation corre os testes unitários num emulador ou num dispositivo real, permitindo que se consiga aceder ao comportamento das dependências da plataforma *Android* sem ter que simular o seu comportamento. Ao usar apenas a ferramenta JUnit não se tem acesso aos comportamentos das dependências da plataforma *Android*, mas como os testes correm localmente na JVM (Java Virtual Machine) a sua execução é muito mais rápida.

A *framework* Mockito possibilita que se consigam simular os comportamentos das dependências usadas na implementação das funcionalidades, de forma a produzir testes consistentes e reproduzir comportamentos específicos. Embora o Mockito seja uma ferramenta de simulação poderosa, esta não consegue simular métodos estáticos nem a criação de objetos dentro do código a ser testado. Para contornar tais adversidades, torna-se necessário utilizar a *framework* PowerMock, que corre sobre o Mockito e torna possível a simulação de métodos estáticos e a criação dinâmica de objetos dentro do código a ser testado. A desvantagem de usar esta *framework* é que não é possível usá-la em conjunto com a ferramenta Android Instrumentation, negando a possibilidade de usar a funcionalidade das bibliotecas da plataforma *Android*, tornando-se assim necessário simular o comportamento de todos os métodos das dependências usadas. O controlo extra que nos é dado tem assim o custo de introduzir mais complexidade aos testes produzidos, tornando este processo mais meticuloso e moroso.

Assim, usou-se a ferramenta Android Intrumentation, sempre que possível, de forma a simplificar o processo de teste, usando o JUnit em conjunto com o PowerMockito apenas quando necessário obter um maior controlo sobre o comportamento das dependências do componente a ser testado.

Como exemplo da primeira abordagem de teste usando Android Instrumentation, tem-se o teste dos objetos Network e Share, usados para representar uma rede e respetivos parâmetros de partilha. Estes objetos são responsáveis, não só por codificar as informações da rede e respetiva partilha para os formatos JSON e texto, como também pela sua descodificação. Estes são os objetos essenciais tanto para padronizar a troca de informação entre a aplicação *Android* e o servidor *web* como para o armazenamento da informação das redes no dispositivo. Uma vez que estes objetos apenas codificam e descodificam informação, o teste é feito através da ferramenta Android Instrumentation, possibilitando a utilização do comportamento das bibliotecas da plataforma *Android* e tornando o seu teste bastante simples.

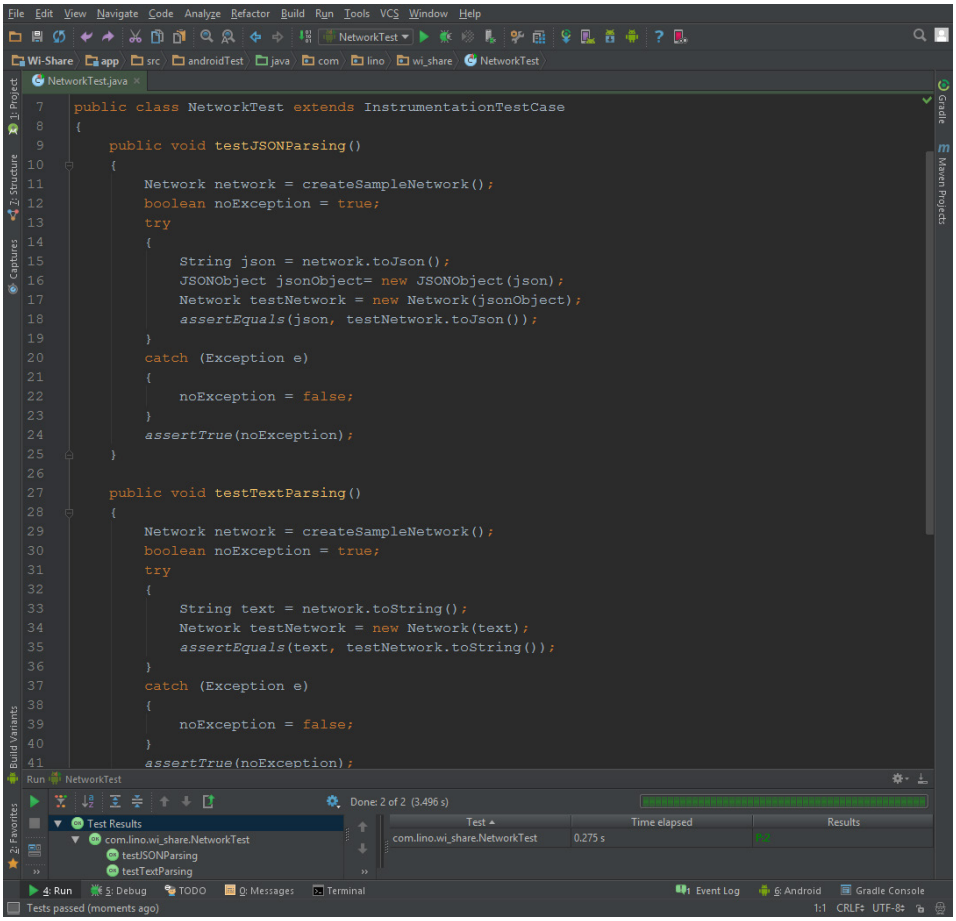

Figura 18 – Teste unitário de objetos Network

Como ilustrado na figura 18, no primeiro método de teste é criado um objeto Network, fazendo-se a sua conversão para JSON e instanciando-se outro objeto Network a partir do mesmo JSON produzido pelo primeiro objeto. Assim consegue-se testar a codificação e descodificação dos dados no formato JSON com sucesso caso os dados produzidos por estes dois objetos sejam iguais e sem a ocorrência de quaisquer exceções. Segue-se o mesmo processo para testar a codificação e descodificação do formato de texto no método testTextParsing.

Um exemplo da necessidade da segunda abordagem utilizando a *framework* PowerMock é o teste da conexão a redes Wi-Fi, ilustrado na figura 19. Para efetuar a conexão a redes Wi-Fi, é necessário trabalhar com as bibliotecas de gestão de Wi-Fi da plataforma *Android* e fazer pedidos ao servidor *web* para se confirmar que o limite de utilizadores não foi atingido. Tais operações são mais dificeis de testar com Android Instrumentation e, por isso, recorre-se ao JUnit e à *framework* PowerMock para simular o comportamento do módulo de gestão de Wi-Fi da plataforma *Android*, fazendo-o atuar como se a rede estivesse acessível e confirmar o estabelecimento da conexão. Foi, também, necessária a simulação do comportamento do módulo que faz as chamadas ao servidor para que se consiga simular a permissão da conexão à rede ou a sua negação por ser atingido o limite de utilizadores.

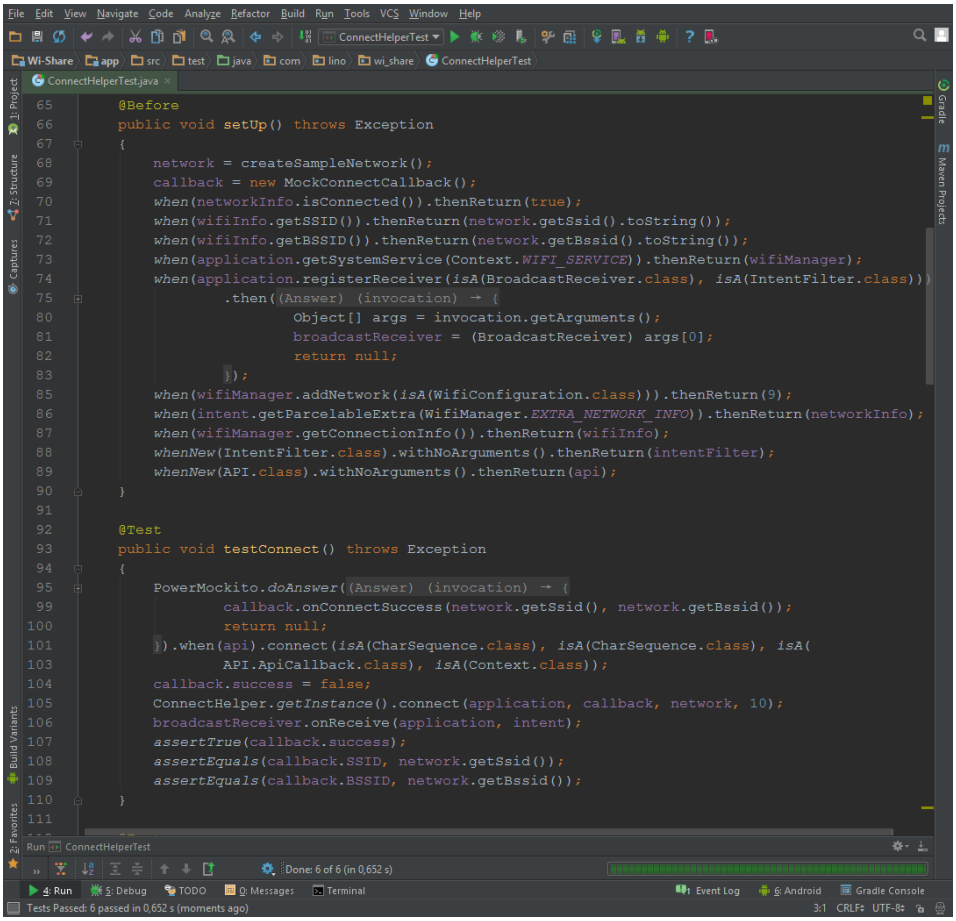

Figura 19 - Teste unitário do objeto ConnectHelper

Como ilustrado na figura 19, no primeiro método, definem-se os comportamentos dos objetos que fazem parte das dependências usadas no código a ser testado e, no segundo, testa-se o processo de conexão a uma rede. Como podemos constatar, a complexidade de testar uma funcionalidade em específico aumenta drasticamente em comparação aos testes feitos através da ferramenta Android Instrumentation, embora o tempo de execução aqui seja muito menor.

Todos os restantes testes foram efetuados à semelhança destes dois tipos de abordagem e consoante as necessidades de cada módulo a ser testado.

Uma vez escritos e validados os testes unitários, avança-se para os testes de desempenho feitos ao sistema antes de sua entrada em produção.

5.3. Testes de desempenho

De modo a garantir que o sistema está devidamente preparado para aguentar com cenários de utilização exigentes, foram efetuados testes de desempenho tanto ao servidor como à aplicação *Android*.

No servidor decidiu-se fazer pedidos em massa de modo a simular uma grande quantidade de utilizadores a aceder em simultâneo. Para fazer tal simulação de carga ao servidor foi usada a aplicação RESTful Stress, que usa a notação JSON para configurar os pedidos a efetuar ao servidor.

Antes de serem feitos pedidos em massa a todas as rotas do servidor, foram testadas as rotas individualmente de forma a identificar possíveis problemas de desempenho. Nesta fase, identificaram-se alguns problemas no processo de sincronização, em que o servidor começava a utilizar muita memória até ao ponto de bloquear quando mais de 4 utilizadores efetuavam a sincronização em simultâneo. A solução passou por utilizar o Eloquent, o módulo que trabalha com a base de dados, mais cuidadosamente, para que as operações fossem executadas maioritariamente através de SQL em vez de serem executadas pelo PHP em *runtime*. Através deste processo de otimização, conseguiu-se passar o teste de *stress* com mais de 20 pedidos de sincronização em simultâneo, um valor que se achou aceitável. Com os problemas de otimização resolvidos, procedeu-se aos pedidos em massa a todas as rotas do servidor em simultâneo, ilustrado na figura 20.

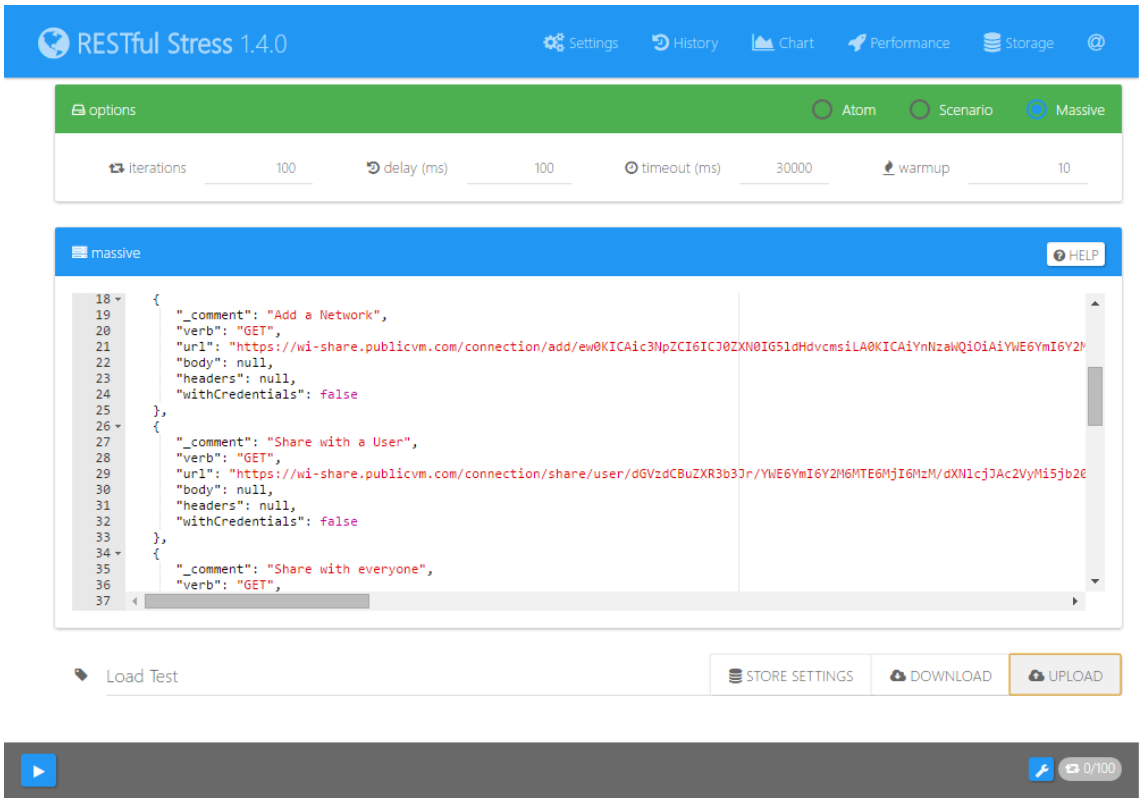

Figura 20 - Configuração dos pedidos em massa

Para fazer pedidos em massa foram configurados 10 pedidos, 1 para cada rota disponibilizada pelo servidor. Estes pedidos foram efetuados através de 4 máquinas de diferentes operadores de telecomunicações durante um ciclo de 10 iterações, efetuando-se 1000 pedidos provenientes de cada máquina, 4000 no total. Definiu-se um tempo máximo de resposta de 30 segundos a partir do qual

se dá o pedido como falhado. Os resultados foram satisfatórios, conseguindo o servidor responder a todos os pedidos com sucesso, situando-se o pior tempo de resposta nos 2,7 segundos e o melhor nos 67 milissegundos. Estes resultados são importantes para ser possível satisfazer os critérios de teste com parâmetros temporais definidos nos requisitos que rondam os 10 segundos.

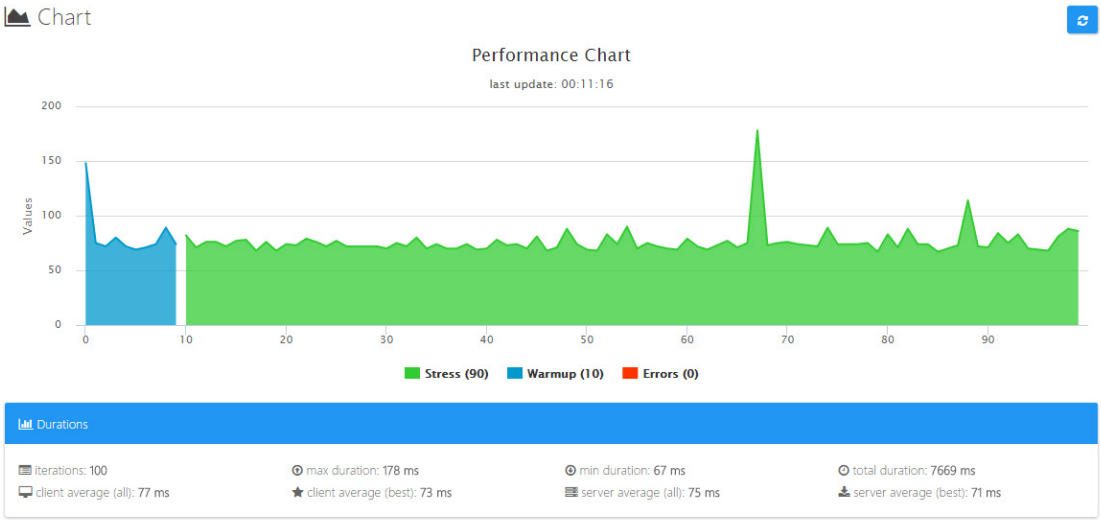

Figura 21 - Resultados dos testes em massa de uma das máquinas

Este teste serviu para constatar que o servidor consegue aguentar com uma carga razoável de pedidos em simultâneo. Ao observar os dados dos testes efetuados nas várias máquinas observou-se que os testes variam muito consoante o operador de rede e o *hardware* utilizado.

Na aplicação *Android* a principal preocupação foi com o processo de sincronização que, como mencionado na secção 4.7.2, é um processo bastante intensivo tanto a nível de tempo de processamento como de consumo de memória. Os *smartphones* usados para testar as métricas de desempenho foram o LG G2 e o Huawei Ascend Y530 de forma a representar diferentes categorias de desempenho.

Começa-se o teste com o G2, enviando-se 1000 redes do servidor para o dispositivo. Mesmo sendo um dispositivo com elevado desempenho, o G2 não conseguiu completar a operação devido ao elevado uso de memória. Procedeu-se à otimização do código, incentivando o *garbage collector* a libertar variáveis desnecessárias e criando também um *buffer* intermédio para proceder à gravação das redes de uma só vez. Estas otimizações deram fruto e conseguiu-se efetuar a sincronização das 100 redes em menos de 2 segundos, conseguindo-se chegar até às 1000 redes com uma duração de 8 segundos. Segundo estas métricas, o tempo de sincronização está dentro do requisito definido pela *fatia* 3.7, que é de 10 segundos.

Dada a diferença da capacidade de processamento do Y530, este mesmo processo com 1000 redes demora mais de 20 segundos a estar concluído, quebrando o requisito de 10 segundos para esta operação. Assim, fixou-se o número total de redes a sincronizar nos 500, obtendo-se um valor a rondar os 8 segundos e cumprindo-se assim com o requisito da *fatia* 3.7 ainda com alguma margem de manobra.

E, com este último, se concluem os testes de desempenho, passando-se no próximo ponto à descrição dos testes com utilizadores.

5.4. Testes com utilizadores

Verificado o bom funcionamento e desempenho do sistema, avança-se para os testes com utilizadores com o intuito de constatar a utilidade que os utilizadores deram à aplicação criada e perceber quais as funcionalidades mais relevantes e apelativas num cenário de utilização real.

Para testar a aplicação foram angariados 24 utilizadores, sendo-lhes pedido que instalassem a aplicação e a usassem durante um período de 1 mês, adicionando e partilhando redes conforme lhes fosse útil no seu dia-a-dia. Passados os 30 dias, foi-lhes realizado um inquérito para tentar perceber qual o uso dado à aplicação, quais as funcionalidades mais usadas e quais para eles foram mais importantes. Os inquiridos enquadram-se todos numa faixa etária entre os 20 e 30 anos de idade, dos quais a larga a maioria, 75%, são estudantes.

Questionados sobre o número de dispositivos utilizados, 50% dos inquiridos responderam que apenas utilizaram 1 dispositivo, 45.8% utilizaram dois dispositivos e apenas 4.2% utilizaram entre 2 a 5 dispositivos. Dos utilizadores que utilizaram mais que 1 dispositivo, todos deram uso à funcionalidade de sincronização, demonstrando assim a necessidade de se incluir tal funcionalidade na aplicação.

No que toca à partilha das redes, 75% dos inquiridos partilharam redes através da aplicação, alegando os restantes 25% que não tiveram necessidade de partilhar as suas redes com alguém. Destes 75%, metade partilharam apenas as suas redes, 38.9% partilharam apenas redes alheias e os restantes 11.1% partilharam tanto as suas como as redes alheias. O método de partilha preferido pelos utilizadores foi a partilha pública que obteve 55.6% das respostas, seguindo-se da partilha por SMS com os restantes 44.4%. Isto demonstra que os utilizadores ou partilham publicamente as suas redes ou partilham individualmente por SMS, não dando uso à funcionalidade de partilha através do email. Pensa-se que tal preferência advém da praticabilidade destas operações, uma vez que a partilha pública apenas requer um clique e a partilha por SMS requer a introdução do número de telemóvel. Estas operações tornam-

se mais práticas do que a introdução do endereço de email, que pode-se tornar mais confuso e moroso devido a questões de interpretação e utilização de carateres especiais.

Em termos de restrições, 44.4% dos inquiridos limitaram o número de utilizadores, 27.8% a largura de banda, 5,6% o tráfego diário e outros 5,6% fizeram uso de todos os limites anteriores. Os restantes 16.7% não impuseram quaisquer limitações nas suas partilhas. Estes dados revelam que, ao contrário do que se esperava, os utilizadores deram preferência à limitação do número de pessoas que se conectam à sua rede em vez da limitação da largura de banda. Este resultado pode estar relacionado com a limitação de a largura de banda apenas poder ser imposta nos dispositivos com permissões especiais (root), procurando assim os utilizadores a salvaguarda do desempenho das suas redes através da limitação do número de pessoas conectadas.

Questionados acerca da importância das funcionalidades disponibilizadas, os utilizadores classificaram como sendo as 3 funcionalidades mais importantes as seguintes: a não necessidade de introduzir password ao se conectar, a partilha de redes com conhecidos e a conexão automática.

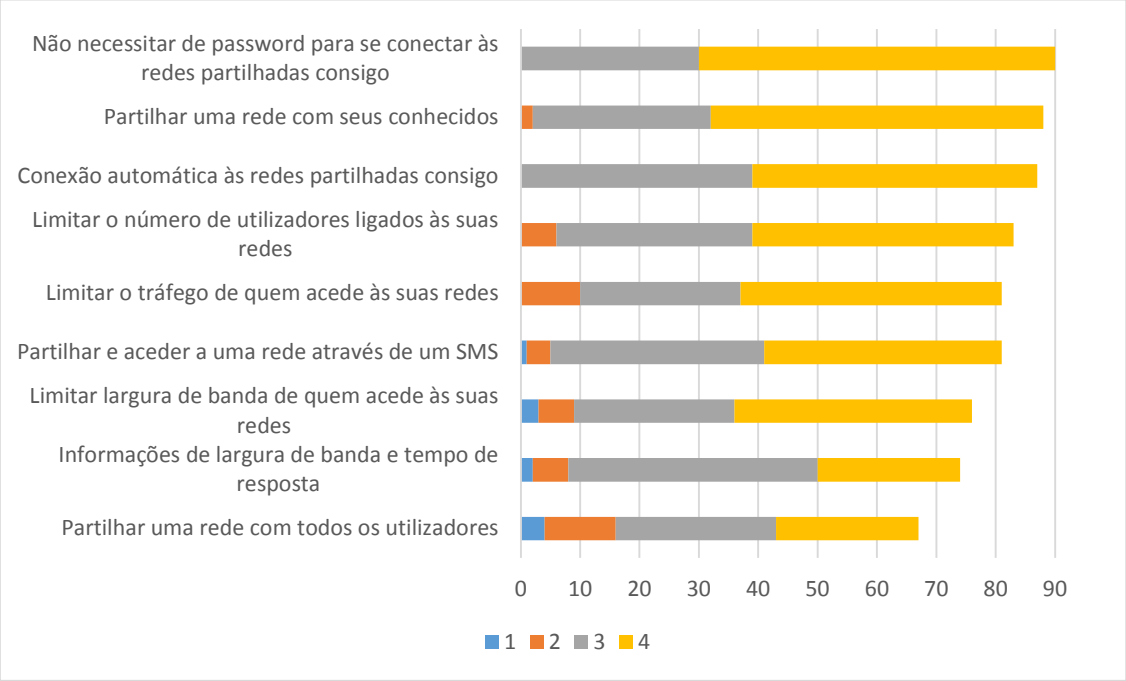

Figura 22 - Importância das funcionalidades

Em geral, os utilizadores deram uma classificação positiva a todas as funcionalidades, destacando-se como a menos popular a partilha com todos os utilizadores. Esta última corresponde aos resultados obtidos no inquérito inicial embora durante o período de testes os utilizadores tenham partilhado as suas redes maioritariamente com todos os utilizadores, que como mencionado anteriormente pode estar associado à simplicidade deste processo. Embora a não necessidade de introduzir a senha para se conectar às redes tenha sido a

segunda pior classificada inicialmente, depois do período de testes, esta foi a melhor classificada. Tal vem confirmar a apreciação feita na análise ao inquérito inicial, na secção 3.2.2.1, de que esta funcionalidade iria trazer vantagens de usabilidade reais e poderia ter passado despercebida.

5.5. Conclusões

Neste capítulo apresentaram-se todos os testes efetuados ao sistema. Abordaram-se os testes unitários feitos através das ferramentas Android Instrumentation e JUnit, os testes de desempenho feitos através da ferramenta RESTful Stress e os testes com utilizadores que foram realizados com a colaboração de todos os 24 participantes.

Os testes unitários foram efetuados para garantir que o sistema se comporta como esperado, tentando-se utilizar sempre que possível a ferramenta Android Intrumentation, de forma a tornar o processo de teste o mais simples possível e recorrendo apenas à *framework* PowerMock sempre que necessário obter um maior controlo sobre a simulação do comportamento das dependências do módulo a ser testado.

Antes de entrar com o sistema em produção, foram feitos testes de desempenho tanto ao servidor *web* como à aplicação *Android* para garantir que estes conseguiam aguentar com uma carga de utilização mais exigente. Foram encontrados problemas de otimização no processo de sincronização, tanto na aplicação *Android* como na aplicação *Lumen*, no servidor. Foram corrigidos tais problemas e entrou-se com o sistema em fase de produção.

Com o sistema em produção, foram feitos testes com utilizadores, servindo tais testes para apurar as funcionalidades mais usadas e importantes segundo a opinião dos utilizadores. Estes testes serviram para constatar que as funcionalidades de sincronização, partilha pública e limitação de utilizadores conectados são essenciais e foram as mais utilizadas durante o período de testes. Devido à discrepância entre a maioritária utilização da partilha pública e a classificação em termos de importância dada pelos utilizadores, pode-se também concluir que a partilha com outro utilizador em específico (através do email) pode necessitar de melhorias em termos de usabilidade. Os resultados confirmam também que a seleção das funcionalidades a implementar foi uma decisão razoável, uma vez que os utilizadores apenas descartaram as funcionalidades de limitação de tráfego diário e partilha através do email.

No próximo capítulo passam-se às conclusões finais, fazendo-se um balanço ao projeto e a todas as decisões tomadas. Com isso, apresenta-se também uma visão para uma futura evolução do sistema desenvolvido.

Capítulo 6

Conclusões e trabalho futuro

O presente trabalho foi realizado em torno da temática da partilha e conexão a redes Wi-Fi através de *smartphones* da plataforma *Android*. Estudaram-se as soluções existentes no mercado fazendo um levantamento das suas funcionalidades e identificando as suas limitações. Com base nesta análise, foi produzida uma especificação e desenvolvida uma aplicação para a plataforma *Android* para facilitar a partilha de configurações das redes Wi-Fi com terceiros e permitir a limitação dos recursos usados por quem acede às redes. Conclui-se então este projeto fazendo uma análise do trabalho realizado e apresentando possíveis ideias a explorar como trabalho futuro.

6.1. Análise do trabalho realizado

O trabalho apresentado procurou simplificar a partilha de configurações de redes Wi-Fi e possibilitar a limitação dos recursos usados por terceiros. As soluções proprietárias existentes, descritas na secção de trabalhos relacionados, embora resolvam o problema de simplificar o processo de partilha e acesso aos pontos Wi-Fi, têm a característica de serem sistemas proprietários e assim tornando impossível que utilizadores de diferentes plataformas consigam partilhar os seus pontos de acesso entre si. As soluções comunitárias, embora permitam que utilizadores de diferentes plataformas partilhem redes entre si, as condições de partilha são precárias, uma vez que as informações de segurança ficam acessíveis a todos e sem quaisquer mecanismos para controlar os recursos utilizados por quem acede às redes.

De forma a permitir uma partilha simples das configurações Wi-Fi, permitir que utilizadores de diferentes operadores partilhem os seus pontos de acesso entre si, à semelhança das soluções comunitárias, e manter a possibilidade de controlar os recursos usados por terceiros, à semelhança da solução da FON, surge uma série de desafios que se pretendeu ultrapassar com o desenvolvimento de uma aplicação *Android*. Tais desafios estão relacionados

com a necessidade de serem encontrados mecanismos simples, seguros e eficazes para se transmitirem as configurações das redes Wi-Fi entre utilizadores e conseguir limitar os recursos das redes sem ser necessário intervir/alterar os pontos de acesso.

Os desafios foram ultrapassados através do desenvolvimento de uma aplicação *Android* pela qual se faz não só a partilha e acesso às redes Wi-Fi, como também se limitam os recursos consumidos pelos utilizadores. Esta aplicação *Android* é apoiada por um servidor *web* para possibilitar o armazenamento das informações acerca do tráfego consumido pelos utilizadores numa determinada rede e também para conseguir controlar os utilizadores conectados a uma determinada rede através de um mecanismo de *heartbeat*. Para solucionar o problema da limitação da largura de banda foi necessário recorrer ao módulo de código aberto disponibilizado por Mansour Behabadi, embora este código apenas funciona com permissões especiais no equipamento (*root* do *smartphone*). Apareceram ainda outros desafios ao longo do desenvolvimento relacionados com o desempenho e maioritariamente na aplicação *Android*. Foi necessário utilizar execuções assíncronas e otimizações de código para garantir que a aplicação se mantém disponível e sem bloquear.

De forma a verificar o cumprimento de todos os requisitos para o sistema foram feitos testes unitários e de desempenho, ainda antes de colocar o sistema em produção. Os testes unitários permitiram que se identificassem alterações no comportamento da aplicação ao serem efetuadas mudanças e que se constatasse o correto comportamento das funcionalidades pretendidas. Os testes de desempenho permitiram identificar problemas de escassez de memória e de eficiência no processo de sincronização, tanto na aplicação *Android*, como no servidor *web*. Na aplicação *Android* foi necessário dar dicas ao *garbage collector* para limpar variáveis inutilizadas em memória e criar um *buffer* intermédio a guarda das redes em memória secundária. No servidor *web*, para se favorecer a execução das operações na base de dados em SQL em vez de *runtime*, foi necessário otimizar as chamadas às funções do módulo Eloquent, o módulo disponibilizado pela *framework Lumen* para fazer operações na base de dados.

Após a realização dos testes e a resolução dos problemas de desempenho, passou-se à fase de produção através da disponibilização da aplicação desenvolvida na *Play Store*. Neste ponto foi possível dar início aos testes com utilizadores, realizados com a colaboração de 24 indivíduos e com a duração de 30 dias. Após os 30 dias de testes, foram realizados inquéritos aos 24 participantes. Estes serviram para apurar que realmente existe a necessidade de partilhar configurações Wi-Fi, uma vez que a maioria dos utilizadores, 75%, partilharam redes através da aplicação e 83,3% desses utilizadores limitaram os recursos das suas redes. Conseguiu-se ainda apurar que o limite de utilizadores conectados a uma rede foi a restrição mais popular. Os testes com utilizadores revelam assim uma correta escolha inicial das funcionalidades a

implementar, uma vez que depois do sistema ser usado os utilizadores atribuíram elevados níveis de importância a todas elas.

Após todo o processo de desenvolvimento e dos testes com utilizadores conseguem-se dar todos os objetivos por cumpridos, uma vez que a aplicação desenvolvida simplifica o processo de partilha das configurações Wi-Fi, até mesmo estando *offline*, automatiza o processo de conexão às redes, oculta as credenciais de terceiros e consegue limitar a largura de banda, tráfego e utilizadores conectados a uma rede. Além destes, também se cumpriu com o objetivo de testar a aplicação com utilizadores de forma a obter *feedback*.

Embora tenham sido cumpridos todos os objetivos e se consiga juntar o melhor de todas as soluções estudadas nos trabalhos relacionados, existem certamente algumas limitações. Estas limitações começam com o módulo que controla a largura de banda, que requer permissões root para conseguir operar. Outra limitação deste módulo é que apenas consegue controlar efetivamente as comunicações sobre o protocolo TCP, permitindo que as comunicações sejam feitas sem restrições de velocidade através de outros protocolos, como o UDP.

A metodologia Use Case 2.0 que se utilizou para especificar os requisitos revelou-se clara, simples e eficaz na especificação dos requisitos do sistema. Embora a utilização desta metodologia tenha sido uma experiência positiva, existem otimizações que advêm da experiência para que se consiga utilizar esta metodologia com todo o seu potencial. É necessário encontrar um bom balanço de granularidade das *fatias* resultantes a análise dos casos de uso para que se consiga melhor estimar o esforço a despender nas interfaces com o utilizador e em cada componente distinto do sistema.

6.2. Trabalho futuro

Para permitir a partilha das configurações Wi-Fi com utilizadores que não tenham instalada a aplicação desenvolvida seu *smartphone*, sugere-se, como trabalho futuro, a introdução de uma funcionalidade que consiga transmitir o pacote de instalação, do dispositivo do utilizador que quer partilhar a sua rede, para o dispositivo do utilizador a receber a partilha. Esta funcionalidade permitiria que os utilizadores sem a aplicação instalada conseguissem proceder à sua instalação sem ter que se conectar à Internet.

Outro ponto a estudar seria a implementação de um mecanismo de encriptação das informações das redes armazenadas no dispositivo do utilizador, uma vez que, no decorrer da fase de implementação, se ficou a conhecer que, com acesso root, é possível aceder aos dados do repositório privado SharedPreferences. Embora os módulos Network e Share codifiquem os dados em Base64, estes ficam vulneráveis a ataques por parte de quem se interesse em fazer a engenharia reversa e faça a descodificação destes dados.

Referências

[1] G. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. Costa, e B. Walke, «The IEEE 802.11 universe», *IEEE Commun. Mag.*, vol. 48, n. 1, pp. 62–70, Jan. 2010.

[2] «802.11ac: The Fifth Generation of Wi-Fi Technical White Paper», *Cisco*. [Em linha]. Disponível em: http://cisco.com/c/en/us/products/collateral/wireless/aironet-3600-series/white_paper_c11-713103.html. [Acedido: 14-Jan-2016].

[3] X. J. J. Anitha, «4G Mobile Communications - Emerging Technologies», *Int. J. Multidiscip. Approach Stud.*, vol. 2, n. 6, pp. 109–114, Dez. 2015.

[4] «Fastest Mobile Networks and the Winners Are...», *PC Mag.*, pp. 126–168, Jul. 2014.

[5] The Government of the Hong Kong Special Administrative Region, «FON: A TECHNOLOGY BRIEF». Fev-2008.

[6] «Como Funciona | Fon». [Em linha]. Disponível em: https://corp.fon.com/pt/how-it-works. [Acedido: 27-Out-2015].

[7] S. Johnson, «Won't You Be My Neighbor?», *Discover*, vol. 27, n. 6, pp. 30–31, Jun. 2006.

[8] «Home | Fon». [Em linha]. Disponível em: https://corp.fon.com/pt. [Acedido: 05-Mar-2015].

[9] «Usufrua de WiFi gratuito em Portugal e em todo o mundo. | Fon». [Em linha]. Disponível em: https://corp.fon.com/pt/microsite/nos. [Acedido: 10-Fev-2015].

[10] «MEO WiFi». [Em linha]. Disponível em: https://www.meo.pt/suporte/ajuda-e-suporte/internet/meo-wifi. [Acedido: 28-Out-2015].

[11] «Parceiros - MEO WiFi | MEO». [Em linha]. Disponível em: https://www.meo.pt/internet/meo-wifi/parceiros. [Acedido: 28-Out-2015].

[12] «Tarifários - MEO WiFi | MEO». [Em linha]. Disponível em: https://www.meo.pt/internet/meo-wifi/tarifarios. [Acedido: 28-Out-2015].

[13] «Mandic magiC – Aplicações Android no Google Play». [Em linha]. Disponível em: https://play.google.com/store/apps/details?id=com.mandicmagic.android. [Acedido: 05-Mar-2015].

[14] «WiFi Free – Aplicações Android no Google Play». [Em linha]. Disponível em:

https://play.google.com/store/apps/details?id=com.yunshang.wifimap. [Acedido: 05-Mar-2016].

[15] «osmino Wi-Fi: WiFi gratuito - Aplicações Android no Google Play». [Em linha]. Disponível em: https://play.google.com/store/apps/details?id=com.osmino.wifil. [Acedido: 05-Mar-2015].

[16] «WiFi Map Pro - Senhas - Aplicações Android no Google Play». [Em linha]. Disponível em: https://play.google.com/store/apps/details?id=io.wifimap.wifimap. [Acedido: 05-Mar-2015].

[17] «Wifi Grátis Instabridge Wi-Fi - Aplicações Android no Google Play». [Em linha]. Disponível em: https://play.google.com/store/apps/details?id=com.instabridge.android. [Acedido: 05-Mar-2015].

[18] «WifiPass - Internet grátis - Aplicações Android no Google Play». [Em linha]. Disponível em: https://play.google.com/store/apps/details?id=com.movile.wp. [Acedido: 05-Mar-2015].

[19] «WiFi Finder - Aplicações Android no Google Play». [Em linha]. Disponível em: https://play.google.com/store/apps/details?id=com.jiwire.android.finder. [Acedido: 05-Mar-2015].

[20] «Friendly WiFi - Aplicações Android no Google Play». [Em linha]. Disponível em: https://play.google.com/store/apps/details?id=com.tam.logic. [Acedido: 05-Mar-2015].

[21] «Hotspotio – Aplicações Android no Google Play». [Em linha]. Disponível em: https://play.google.com/store/apps/details?id=com.hotspotio. [Acedido: 05-Mar-2016].

[22] «InstaWifi - Aplicações Android no Google Play». [Em linha]. Disponível em: https://play.google.com/store/apps/details?id=net.jessechen.instawifi. [Acedido: 05-Mar-2015].

[23] I. Jacobson, *USE-CASE 2.0 The Definitive Guide*. 2011.

[24] I. Sommerville, *Software engineering*, 9th ed. Boston: Pearson, 2011.

[25] «Dashboards | Android Developers». [Em linha]. Disponível em: http://developer.android.com/about/dashboards/index.html. [Acedido: 26-Nov-2015].

[26] «APK Expansion Files | Android Developers». [Em linha]. Disponível em: http://developer.android.com/google/play/expansion-files.html. [Acedido: 09-Fev-2015].

[27] «The Security Building Blocks of TLS/SSL», *Load Balancer*. [Em linha]. Disponível em: https://kemptechnologies.com/white-papers/security-building-blocks-tls/. [Acedido: 10-Dez-2015].

[28] M. L. Das e N. Samdaria, «On the security of SSL/TLS-enabled applications», *Appl. Comput. Inform.*, vol. 10, n. 1–2, pp. 68–81, Jan. 2014.

[29] K. Maeda, «Performance evaluation of object serialization libraries in XML, JSON and binary formats», em *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on*, 2012, pp. 177–182.

[30] H. Benoit, «Generating new certificate in XAMPP for Windows», *Benohead.* .

[31] «Strong SSL Security on Apache2 - Raymii.org». [Em linha]. Disponível em: https://raymii.org/s/tutorials/Strong_SSL_Security_On_Apache2.html. [Acedido: 17-Dez-2015].

[32] «SSL Server Test (Powered by Qualys SSL Labs)». [Em linha]. Disponível em: https://www.ssllabs.com/ssltest/index.html. [Acedido: 17-Dez-2015].

[33] «Lumen - PHP Micro-Framework By Laravel». [Em linha]. Disponível em: http://lumen.laravel.com/docs/installation. [Acedido: 17-Dez-2015].

[34] «Database: Migrations - Laravel - The PHP Framework For Web Artisans». [Em linha]. Disponível em: https://laravel.com/docs/5.2/migrations. [Acedido: 04-Jan-2016].

[35] «Eloquent: Getting Started - Laravel - The PHP Framework For Web Artisans». [Em linha]. Disponível em: https://laravel.com/docs/5.2/eloquent. [Acedido: 04-Jan-2016].

[36] H. S. Kim, E. Kim, e H. Kim, «QoE-driven Wi-Fi selection mechanism for next generation smartphones», em *2012 First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT)*, 2012, pp. 13–18.

[37] M. Behabadi, «BradyBound», *GitHub*. [Em linha]. Disponível em: https://github.com/oxplot/bradybound. [Acedido: 03-Jan-2016].

Anexo A

Tutorial da aplicação

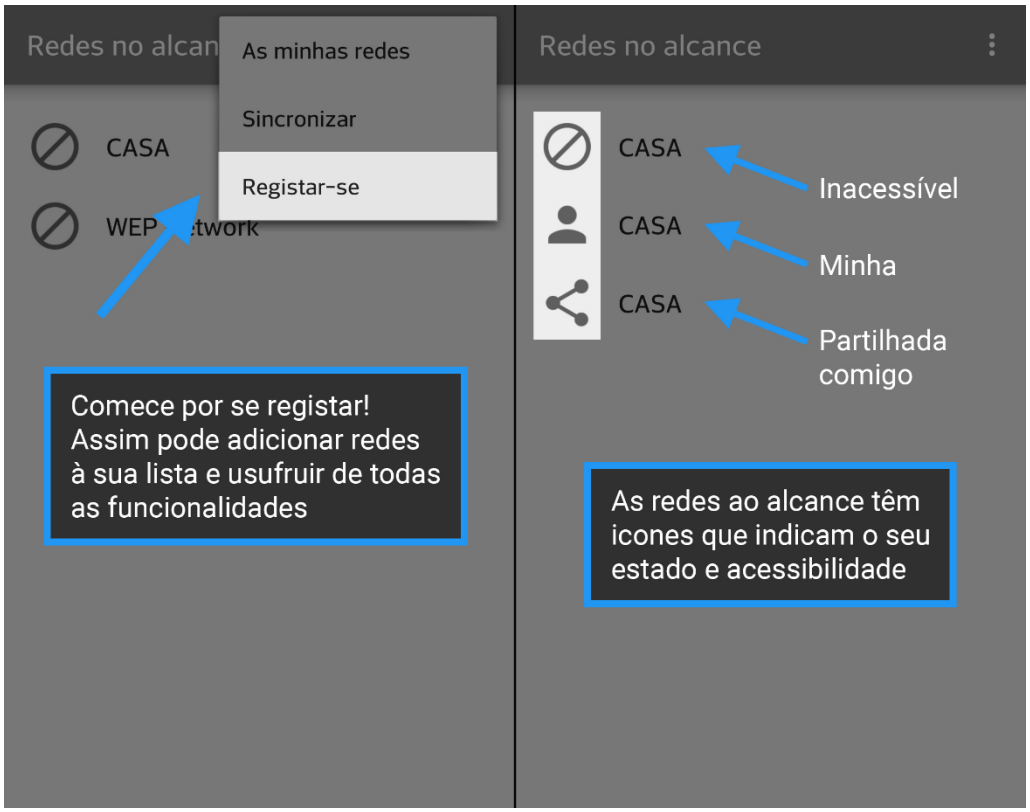

Figura 23 - Tutorial da aplicação

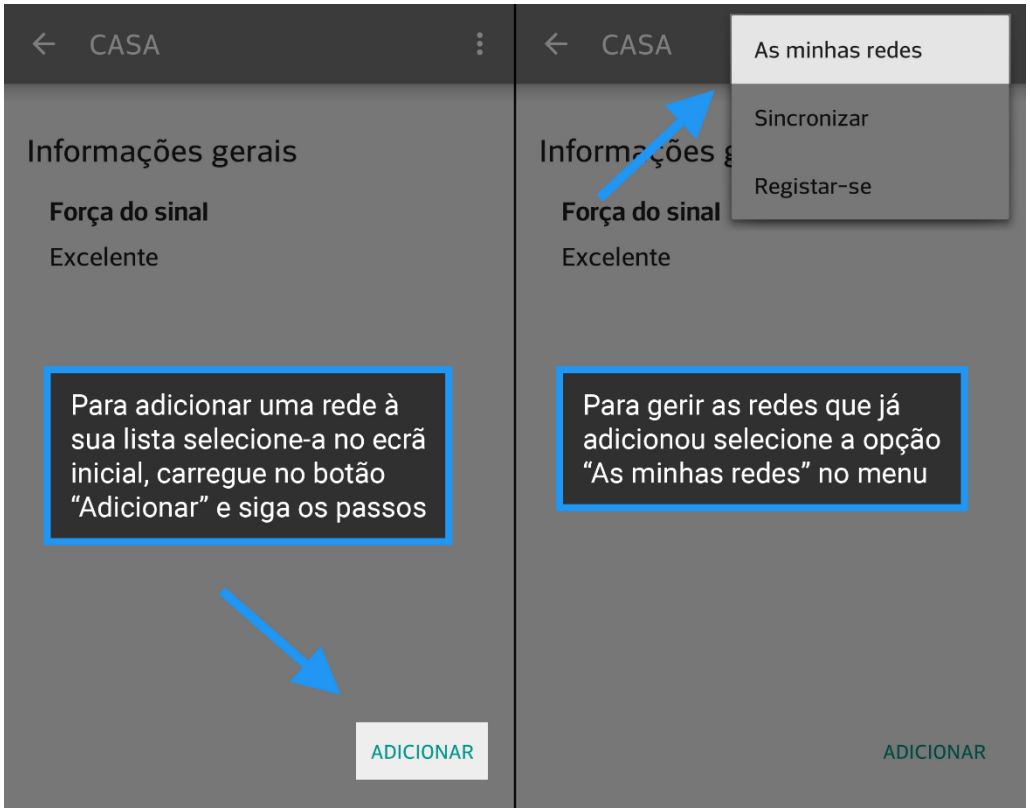

Figura 24 - Tutorial da aplicação

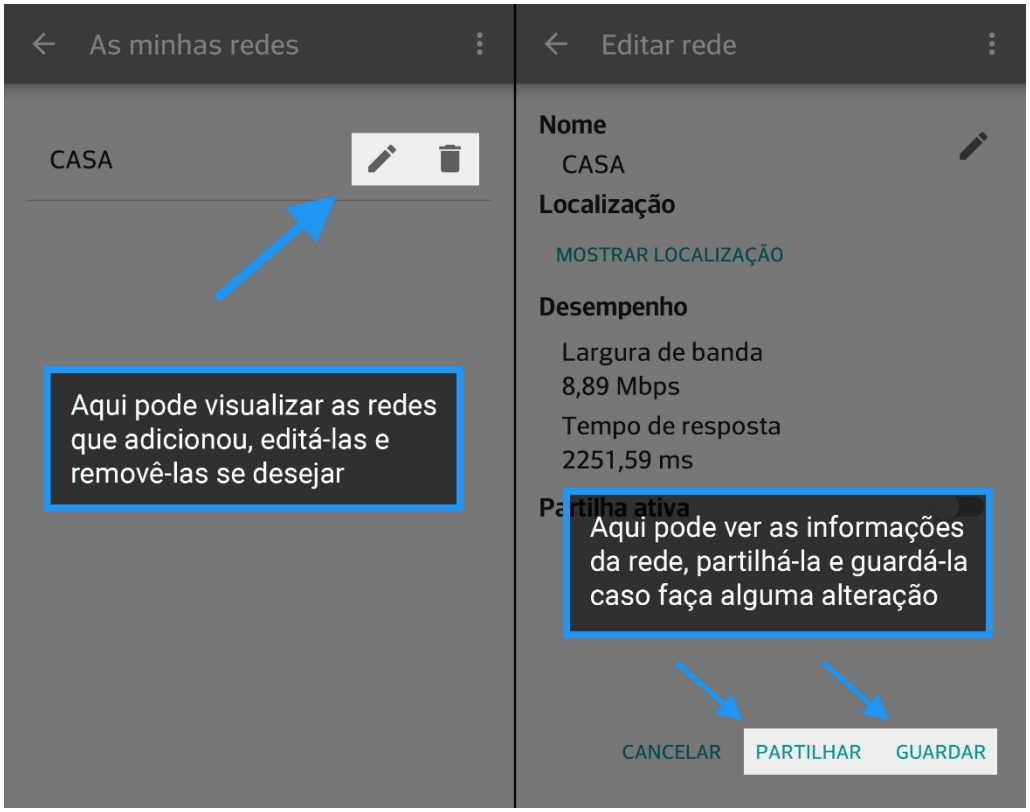

Figura 25 - Tutorial da aplicação

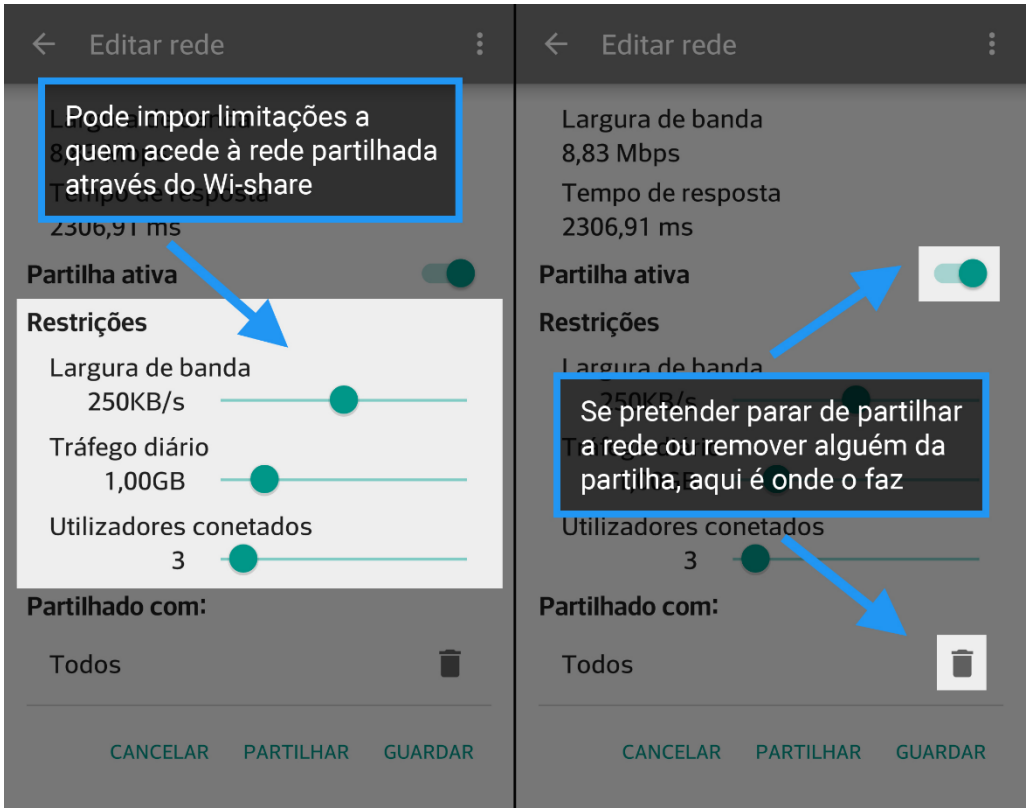

Figura 26 - Tutorial da aplicação

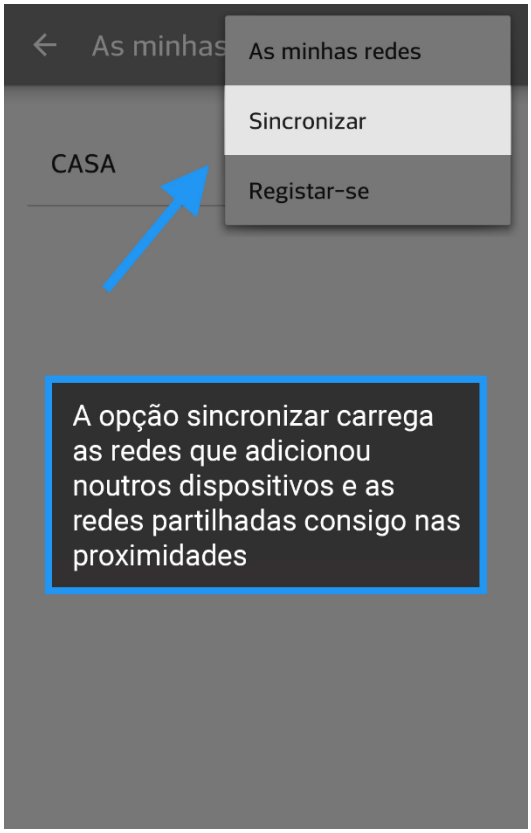

Figura 27 - Tutorial da aplicação