Universidade de Aveiro, Departamento de Electrónica, Telecomunicações e Informática, 2018

Geolocalização entre Conhecidos com Garantias de Privacidade

Aplicação Android

Autores

Daniel Gonçalves, Nº Mec. 80069

Miguel Simões, Nº Mec. 79957

Saveliy Ivanov, Nº Mec. 80298

Geolocalização entre Conhecidos com Garantias de Privacidade

Aplicação Android

Relatório de Projeto em Informática da Licenciatura de Engenharia Informática da Universidade de Aveiro, realizado por Daniel Gonçalves, Miguel Simões, e Saveliy Ivanov sob a orientação do professor André Zúquete, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática, e do professor João Paulo Barraca, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática.

Palavras Chave

Geolocalização
Privacidade
Partilha de localização
Segurança
Aplicação Android

Agradecimentos

Gostaríamos de agradecer aos nossos orientadores de projeto, André Zúquete e João Paulo Barraca pela disponibilidade em nos ajudar ao longo das diversas fases deste projeto.

Resumo

Atualmente existem muitas aplicações que permitem a partilha de dados relativos à localização, no entanto, nenhuma delas permite que os utilizadores tenham um grande controlo sobre os dados partilhados, sendo que na maior parte das vezes esses mesmos utilizadores não têm sequer conhecimento da informação que partilham a partir do momento em que instalam a aplicação nos seus dispositivos.

A proteção e privacidade dos dados pessoais é um dos temas mais discutidos nos dias que correm e é nessa perspetiva que este projeto surge. Pretende-se que os utilizadores tenham um grande controlo sobre todas as informações partilhadas relativamente à sua localização e que essas mesmas sejam totalmente privadas e protegidas de terceiros. Para além disso pretende-se ainda que os utilizadores possam personalizar os conteúdos que enviam, não só quanto à sua granularidade ou precisão, mas também quanto à sua modalidade ou formato.

Este relatório documenta a investigação de soluções já existentes, assim como a nossa solução face aos problemas encontrados e a alguns requisitos previamente estabelecidos para este projeto. Projeto esse que trata o desenvolvimento de uma aplicação que possa ser integrada por outras aplicações já existentes para conferir segurança e acrescentar um determinado nível de personalização a variadas funcionalidades que já são conhecidas e associadas a este tipo de aplicações.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Sistemas de geolocalização existem há já 40 anos, tendo sido o primeiro deles, Global Positioning System, mais vulgarmente conhecido por GPS, lançado em 1978 pelos Estados Unidos [1]. O serviço que este sistema proporciona é tão útil, não só para propósitos militares com que foi inicialmente concebido, mas também para propósitos civis, que vários governos desenvolveram os seus próprios sistemas de geolocalização de modo a não depender de tecnologia estrangeira, como a URSS/Rússia com o GLONASS, a União Europeia com o Galileo, ou a China com o BaiDou, entre outros.

O acesso público a sistemas de geolocalização como o GPS tornou a sua utilização nos mais diversos sectores do mercado possível. Um destes usos é a partilha de localização entre conhecidos, que permite seguir com precisão a localização de amigos, familiares, colegas de trabalho, ou outros, que partilhem a sua localização connosco. Find My Friends é um exemplo de uma aplicação móvel que nos permite seguir a localização dos nossos amigos e partilhar a nossa, mas cada vez vão surgindo mais e mais diversas aplicações para propósitos de partilha de localização. Porém, nenhuma destas aplicações permite aos seus utilizadores definir a granularidade da localização partilhada, nem envolvem os seus utilizadores num processo ativo de escolha e personalização da informação que partilham, comprometendo assim a privacidade das pessoas; aqui entra o nosso projecto, Geolocalização entre Conhecidos com Garantias de Privacidade, que visa criar uma aplicação móvel centrada na privacidade e segurança dos seus utilizadores.

1.1 Contexto

O nosso projecto está integrado no curso de Licenciatura em Engenharia Informática, leccionado na Universidade de Aveiro e foi desenvolvido ao longo do segundo semestre do ano letivo 2017/2018 no âmbito da unidade curricular Projeto em Informática, centrando-se no desenvolvimento de uma aplicação Android que funcione como um serviço para diversas outras aplicações que pretendam fazer partilha segura de informação personalizável acerca da geolocalização dos seus utilizadores.

1.2 Motivação

Já existem muitas soluções de partilha de localização entre conhecidos, mas nenhuma permite aos seus utilizadores total personalização da informação partilhada, nomeadamente no que toca à granularidade da mesma.

Assim, a motivação para este projecto passa por desenvolver algo único e actualmente inexistente no mercado, uma aplicação móvel construída a pensar especificamente na segurança e privacidade dos seus utilizadores, que os envolva ativamente num processo de gestão dos conteúdos que partilham.

1.3 Objetivos

O principal objectivo do nosso projecto é o desenho e desenvolvimento de uma aplicação móvel que suporte uma partilha de localização centrada na segurança e privacidade dos seus utilizadores. Esta servirá de base a outras aplicações que terão que providenciar a localização do utilizador e um canal de comunicação. No nosso projecto o exemplo desta integração é feito com a aplicação OwnTracks [2], capaz de obter a localização do utilizador a partir da Google Maps API [3] e que usa um broker MQTT [2][4] como canal de comunicação.

1.4 Estrutura do Relatório

Este documento está estruturado da seguinte forma:

- **Capítulo 1** - Introdução;
- **Capítulo 2** - Estado da arte e tecnologias utilizadas no projecto;
- **Capítulo 3** - Requisitos do sistema e arquitectura do projecto;
- **Capítulo 4** - Implementação das funcionalidades;
- **Capítulo 5** - Conclusão, resultados e trabalho futuro;

Capítulo 2

Estado da Arte

Actualmente não existem aplicações que permitam aos seus utilizadores um controlo total sobre a informação que é partilhada. Existem, no entanto, algumas soluções de partilha de localização que permitem alguma personalização, que vão ser discutidas neste capítulo.

2.1 Aplicações Existentes

2.1.1 OwnTracks

O OwnTracks é uma aplicação móvel open source que permite aos seus utilizadores acompanhar a sua própria localização e partilhá-la com família e amigos [2]. A informação pode ser partilhada através de um broker MQTT [2][4] ou por HTTP. O OwnTracks não faz uso de nenhum servidor central, ao contrário de muitas outras aplicações como o WhatsApp, aumentando a segurança dos dados partilhados [2].

Quanto à personalização na partilha da localização, o OwnTracks permite alterar a precisão, de um modo limitado, com que a localização é partilhada e a periodicidade dessa partilha. Podemos alterar a precisão da localização partilhada quando a aplicação está a correr em background ou em foreground, podemos alterar de quanto em quanto tempo a nossa localização é partilhada, de quantos em quantos metros percorridos, e é possível desligar completamente a partilha automática da localização, ficando o próprio utilizador responsável por fazê-lo manualmente [2].

A figura 2.1 mostra um exemplo de uma localização partilhada por um utilizador, com alguma imprecisão, o utilizador pode estar em qualquer ponto do círculo azul. Apesar de o OwnTracks permitir alguma personalização de informação, como se pode verificar, esta é bastante rudimentar, baseando-se em pontos e áreas circulares. Outro aspecto problemático do OwnTracks é a inabilidade de partilhar a localização com pessoas específicas, pois a aplicação utiliza um broker MQTT como canal de comunicação, o que implica que todas as pessoas conectadas ao mesmo broker conseguem ver as localizações uns dos outros [4]. Estas são algumas das deficiências que o nosso projecto procura combater.

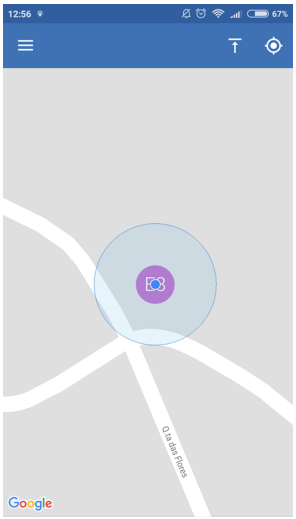

Figura 2.1: Posição do utilizador no OwnTracks

2.2 Tecnologias Utilizadas

Como já foi referido anteriormente, a aplicação desenvolvida será integrada com o OwnTracks, que irá "alimentar" a aplicação com a localização dos utilizadores, entre outros dados que serão usados para diversos fins. Para além da aplicação open source, as tecnologias utilizadas no projecto estão descritas abaixo.

2.2.1 Geohashing

De um modo simples, geohashing permite traduzir uma dada localização geográfica, em formato de coordenadas, num pequeno conjunto de letras e números, i.e. um hash [5]. Este hash é gerado a partir de divisões sucessivas do mapa do planeta em retângulos aos quais está associado um bit, 0 ou 1, como exemplificado na figura 2.2. Esta sequência de bits é depois codificada em base 32, dando origem a um hash.

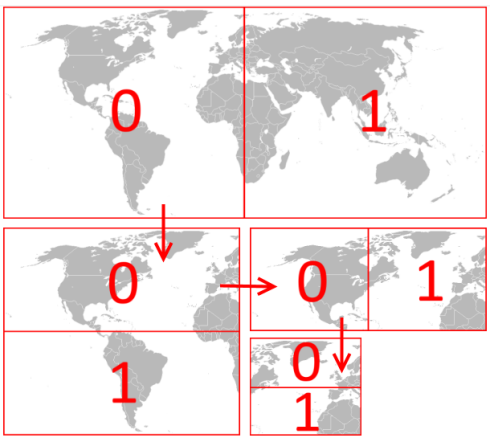

Figura 2.2: Algoritmo Geohash em funcionamento

Isto é extremamente útil porque permite partilhar a localização de um dado utilizador sem revelar as suas coordenadas exactas, salvaguardando assim a sua privacidade. Um exemplo

prático onde geohashing pode ser utilizado é em alertas de proximidade, em que dois amigos enviam periodicamente o hash da sua localização e verificam se a sua própria posição se encontra dentro do geohash recebido; é possível saber, assim, se os amigos estão próximos um do outro sem comprometer a privacidade de nenhum deles, ao contrário das soluções tradicionais em que se usam as duas posições para calcular a distância.

2.2.2 Geofencing

Um tipo de localização suportado pela aplicação desenvolvida são geofences. Uma geofence consiste num perímetro virtual referente a uma determinada área geográfica no mundo real [6]; neste caso de implementação, estas são geradas a partir de um ponto e um raio. Estas geofences são depois utilizadas para o envio de notificações quando um utilizador entra ou sai de uma determinada área. Por exemplo, um utilizador pode definir uma geofence para o Departamento de Electrónica, Telecomunicações e Informática da UA, indicando o ponto central e um raio que ajude a delimitá-la. Quando o utilizador entrar ou sair da área delimitada, será enviada uma notificação para as pessoas que esse utilizador seleccionou previamente a informar do acontecimento.

2.2.3 Protocolo MQTT

Para efectuar trocas de localização entre utilizadores é necessário um canal de comunicação entre os mesmos. O OwnTracks utiliza um broker MQTT como canal de comunicação, que faz uso de um sistema simples e leve de publish/subscribe; os utilizadores subscritos a um dado broker MQTT recebem as mensagens que lá são publicadas por outros utilizadores [4].

2.2.4 Google Maps API

A aplicação desenvolvida não fará qualquer uso de tecnologias de geolocalização para rastreamento da localização dos utilizadores; apenas irá processar localizações recebidas de outras aplicações que façam uso dela, no caso deste projecto, esta aplicação é o OwnTracks, que faz uso da Google Maps API para obter informações úteis sobre lugares e localizações partilhadas [2].

Com a Google Maps API, é possível visualizar a localização num mapa e obter informações sobre a mesma, como o nome da rua ou o código postal, o que facilita a partilha da localização ao providenciar ao utilizador uma camada de abstracção, isto é, o utilizador não precisa de saber as suas coordenadas para partilhar a sua localização, basta olhar para o mapa e sabe imediatamente onde se encontra [2]. Por exemplo, um dos casos de uso deste projecto consiste em que um utilizador partilhe apenas a cidade onde se encontra. Isto é possível devido aos dados extraídos da localização do utilizador pela Google Maps API, que são enviados do OwnTracks para a aplicação desenvolvida para processamento, neste caso a extracção do nome da cidade dos dados recebidos.

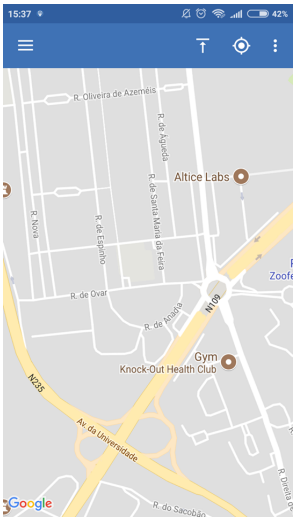

Figura 2.3: Google Maps API integrada no OwnTracks

2.2.5 E2EE - End-to-End Encryption

Um dos principais objectivos deste projecto é garantir a segurança e privacidade dos utilizadores e, para tal, optou-se por utilizar um protocolo de comunicação E2EE (end-to-end encryption) que previne qualquer entidade de ler mensagens trocadas entre utilizadores [7]. A implementação do protocolo E2EE usado no projecto utiliza o algoritmo Diffie-Hellman [8] para efeitos de troca de chaves, enquanto que a troca de chaves em si ocorre presencialmente (com recurso a QR Codes), garantindo assim a autenticidade dos intervenientes. Na Fig. 4 encontra-se ilustrado o envio de uma mensagem do utilizador Bob para o utilizador Alice. A mensagem que o Bob irá enviar, "Hello, Alice!", é cifrada com uma chave simétrica obtida durante o processo de troca de chaves com QR Codes mencionado anteriormente; tanto o Bob como a Alice possuem esta chave e usam-na para efectuar as suas comunicações de um modo seguro. Só quem possui a chave que os dois utilizadores concordaram em utilizar consegue ler as mensagens, em princípio a chave é exclusivamente conhecida pelos dois intervenientes.

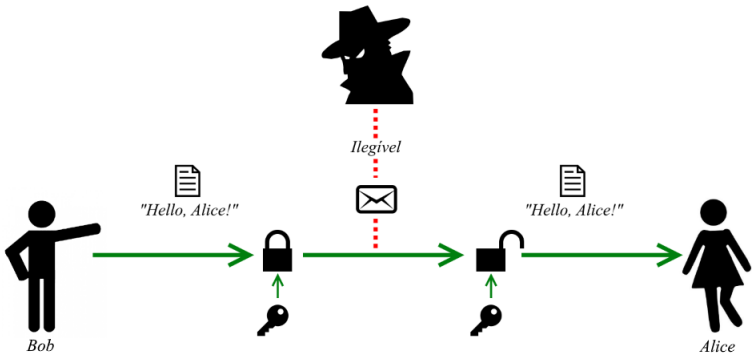

Figura 2.4: Ilustração prática do protocolo E2EE

Capítulo 3

Requisitos do Sistema e Arquitetura

Neste capítulo vão ser discutidos não só os requisitos de sistema acordados inicialmente com os orientadores do projecto, como as eventuais mudanças sofridas pelos mesmos ao longo de todas as iterações no desenvolvimento do projecto.

3.1 Requisitos Iniciais

Aquando da escolha do projecto, foi-nos disponibilizado um documento que descrevia quais os objectivos principais que a aplicação teria de cumprir, descritos abaixo:

- **Privacy by design** - privacidade será o foco central do projecto e terá que ser tida em conta desde a concepção até à implementação de todas e quaisquer funcionalidades. É necessário que seja possível o uso da aplicação sem expôr os dados do utilizador;
- **Contacto inicial presencial** - antes de os utilizadores começarem a comunicar (i.e. trocar localizações entre si), é necessário o estabelecimento de uma forma de comunicação segura entre os mesmos, efetuada na presença dos dois utilizadores de modo a garantir a autenticidade de cada um;
- **Suporte a vários tipos de informação** - utilizadores têm de ser capazes de trocar vários tipos de informação geocêntrica entre si, como posição actual, tempo até um determinado ponto ou morada. Terá que ser possível um utilizador personalizar a informação que envia para outros utilizadores;
- **Suporte a qualquer tipo de canal** - a nossa aplicação deve ser independente do canal de comunicação, funcionando em qualquer um, seja este SMS, Messenger, email ou outros, como um sistema de Publish/Subscribe de um broker MQTT usado no nosso exemplo de implementação.

Numa abordagem inicial, começou por se delinear uma série de tarefas de modo a ajudar a perceber qual a melhor maneira de realizar a implementação dos vários requisitos iniciais e eventuais funcionalidades futuras:

- Estudo de aplicações similares: Find My Friends, OwnTracks, entre outras;

- Rastreio de evidências de quebra de privacidade das aplicações estudadas;
- Definição de alto nível das funcionalidades a implementar;
    - Concepção dos protocolos de segurança, nomeadamente no que toca à distribuição de chaves usadas para uma comunicação segura.

3.1.1 Evidências de Quebra de Privacidade em Aplicações Existentes

Como foi referido acima, uma das tarefas iniciais consistia no estudo de aplicações actualmente existentes de modo a não só perceber as funcionalidades requeridas de uma aplicação de partilha de localização, como também encontrar falhas ao nível da privacidade.

Não foi necessária uma análise muito aprofundada para perceber que muitas destas aplicações possuíam características/funcionalidades bastante semelhantes, que incluíam a partilha da localização actual do utilizador, um mapa (obtido a partir da Google Maps API), gestão de contactos/amigos e alguns tipos de notificações (e.g. proximidade de um dado local ou pessoa). Foi igualmente fácil encontrar vulnerabilidades ao nível da privacidade e segurança dos dados de um utilizador, sendo estas as mais graves:

- Mensagens enviadas entre utilizadores não eram cifradas;
- Mensagens e dados dos utilizadores passavam por, ou eram guardadas num servidor central [10] (isto é um potencial problema, pois quem controla o servidor têm acesso aos conteúdos das mesmas);
- Os conteúdos das mensagens eram pouco ou nada personalizáveis quanto a granularidade, modalidade ou precisão;
- Mensagens continham dados sobre o utilizador que este não consentiu enviar explicitamente, como a sua posição absoluta no mapa;
- Houve ainda uma aplicação que guardava os trajetos diários e os partilhava com os amigos do utilizador, mais uma vez sem o seu consentimento explícito.

3.1.2 Funcionalidades a Implementar - Definição de Alto Nível

Tornou-se evidente que, para evitar todos os problemas referidos anteriormente, a aplicação desenvolvida teria que suportar um protocolo seguro de comunicação que garantisse a privacidade e segurança dos utilizadores e seus dados. Decidiu-se que a melhor maneira de o garantir passaria por fazer uso de um protocolo end-to-end encryption, que garante que nenhuma entidade tem acesso aos conteúdos das mensagens trocadas entre dois utilizadores, sem ser os próprios utilizadores que estabeleceram comunicação previamente [7].

Em adição a toda a parte de segurança e privacidade, a aplicação teria que suportar as funcionalidades básicas providenciadas por aplicações semelhantes, como a partilha da localização de um utilizador ou notificações de proximidade. No entanto, todas estas funcionalidades teriam

que ser personalizáveis: um utilizador tem que ser capaz de escolher a informação que quer enviar e para quem essa informação é enviada. Por exemplo, se a localização de um utilizador é "Universidade de Aveiro, 3810-193 Aveiro, Portugal" esse utilizador tem que ser capaz de personalizar a mesma de modo a conseguir partilhar que se encontra em "Aveiro" apenas, não comprometendo assim a sua localização exacta. No entanto, o mesmo utilizador deve ser capaz de enviar uma localização mais específica (e.g. a localização completa) para outra pessoa. Tendo isto em atenção, concluiram-se aos seguintes casos de uso que a aplicação teria que suportar:

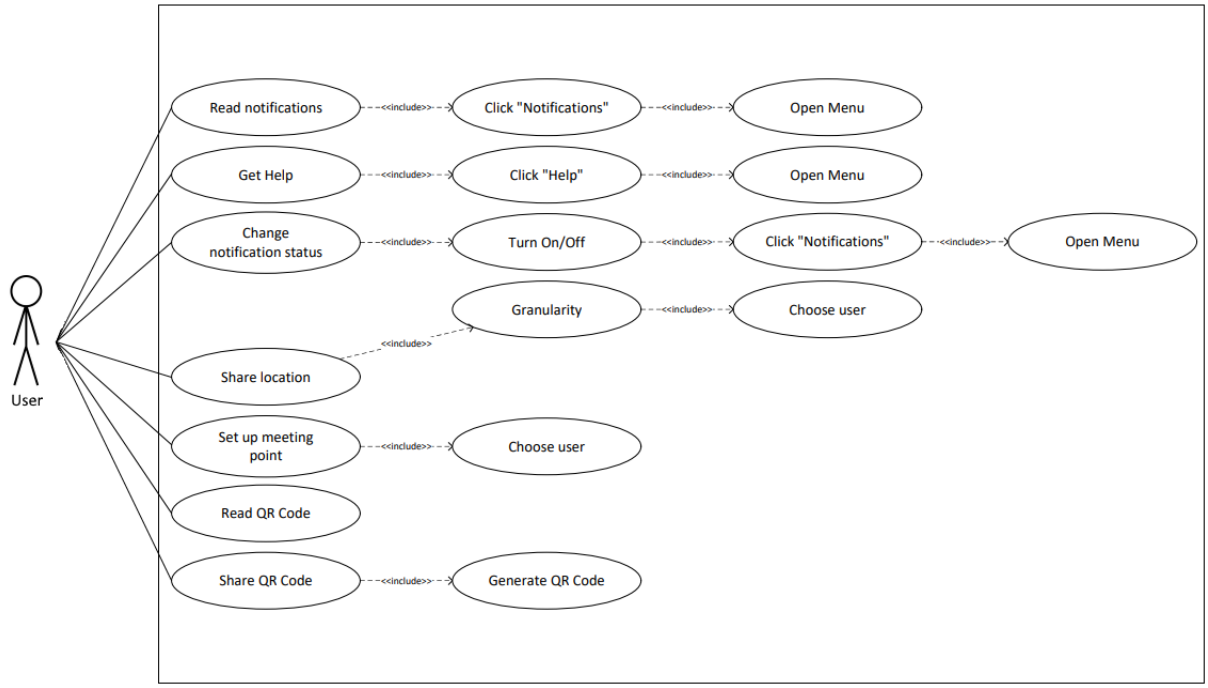

Figura 3.1: Casos de uso

- **Read notifications** - permite ao utilizador da aplicação ler as notificações que recebeu de outros utilizadores;
- **Get help** - permite ao utilizador da aplicação obter ajuda sobre a mesma;
- **Change notification status** - permite ao utilizador da aplicação ligar/desligar as notificações, as que envia e as que recebe;
- **Share location** - o foco da aplicação, que permite ao utilizador partilhar a sua localização com outros utilizadores com a granularidade desejada;
- **Set up meeting point** - permite ao utilizador estabelecer um ponto de encontro com outro utilizador.
- **Read QR Code** - permite ao utilizador ler um QR Code de modo a estabelecer um canal de comunicação seguro com outro utilizador;

- **Share QR Code** - permite ao utilizador gerar um QR Code de modo a estabelecer um canal de comunicação seguro com outro utilizador;

3.1.3 Requisitos Não-Funcionais

Estão descritos abaixo os requisitos não-funcionais que a nossa aplicação teria que cumprir:

- **Suporte a várias versões Android** - as pessoas atualizam o sistema operativo dos seus dispositivos com bastante frequência, por isso é necessário que a nossa aplicação seja compatível não só com versões Android anteriores, mas também versões mais recentes;
- **Performance** - dado que a aplicação irá ser integrada com aplicações já existentes e que a comunicação entre as duas irá ser frequente, torna-se necessário garantir que a aplicação não fique lenta durante todo o processo;
- **Fácil integração** - a integração da nossa aplicação deve ser fácil de concretizar, com modificações mínimas à aplicação que a irá integrar;
- **Usabilidade** - aplicação deve ser de fácil utilização, considerando a vasta quantidade de diferentes utilizadores existentes, uns mais tecnologicamente aptos do que outros;

3.1.4 Contexto de Utilização e Atores

A aplicação é destinada a todas as pessoas (i.e. developers) que queiram implementar nas suas aplicações um sistema de partilha de localização seguro e personalizável. Esta implementação pode ser concretizada com a integração da nossa aplicação, o que implica alterações mínimas à aplicação original.

3.2 Arquitectura do Sistema

3.2.1 Diagrama da Arquitetura

Abaixo, encontra-se ilustrado um diagrama da arquitectura do sistema na sua totalidade, ou seja, com as duas aplicações integradas e a comunicar entre si. A aplicação desenvolvida, e a que a irá integrar, que neste caso é o OwnTracks.

O sistema divide-se em duas partes: a aplicação desenvolvida, que inclui a base de dados, a API e o Serviço, e a aplicação que a integra (neste caso, o OwnTracks). Uma explicação do que cada uma das componentes faz encontra-se abaixo:

- **OwnTracks** - aplicação que integra e comunica com a nossa aplicação de modo a fornecer aos seus utilizadores maior segurança e privacidade na partilha de localização. Usa um Broker MQTT como canal de comunicação;
- **Aplicação** - aplicação desenvolvida ao longo deste projeto que é integrada pela componente descrita anteriormente. É responsável pelas 3 sub-componentes descritas abaixo e por providenciar um interface às suas funcionalidades:

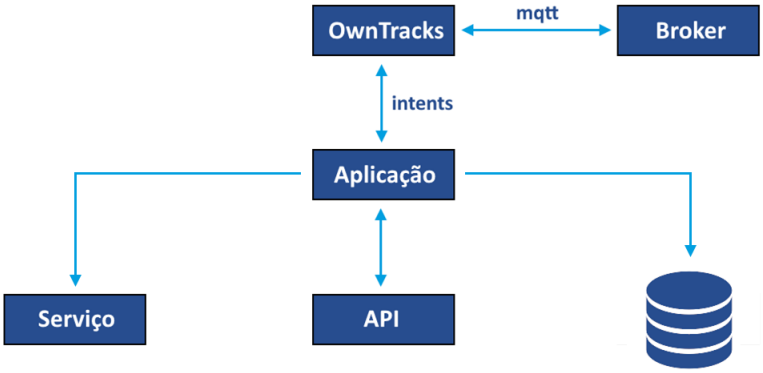

Figura 3.2: Diagrama da arquitectura

- **API** - biblioteca Android que inclui métodos usados para o estabelecimento de contacto inicial (geração/leitura de QR Codes) e para processar localizações fornecidas pelo OwnTracks (e.g. retorna tempo e distância de um dado ponto de encontro);
- **Serviço** - usado para "programar" a aplicação, ou seja, permite definir instruções sobre o que deve ser feito pela aplicação aquando de um determinado evento. Por exemplo, possibilita um utilizador definir a informação a enviar automaticamente de X em X tempo para outro utilizador;
- **Base de Dados** - armazenamento local ao dispositivo onde a aplicação se encontra instalada. Guarda todas as informações relevantes à aplicação, incluindo pontos de encontro, áreas definidas, contactos e chaves de comunicação.

3.2.2 Diagrama de Deployment

Abaixo encontra-se uma representação de alto-nível do modelo físico do projecto e a explicação das suas componentes:

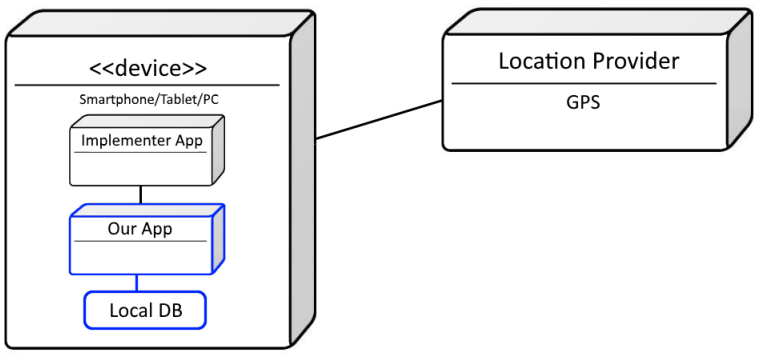

Figura 3.3: Diagrama de deployment

O modelo físico é bastante simples. Existe um dispositivo (i.e. smartphone com sistema operativo Android) que possui uma aplicação (Owntracks, "Implementer App" no diagrama) que integra a aplicação desenvolvida por nós ("Our App" no diagrama, correspondente à parte a azul); esta liga-se a uma base de dados local ao dispositivo em questão para armazenamento de todas as

informações relevantes à comunicação. O dispositivo obtém a localização do utilizador através de um "Location Provider", comumente o Global Positioning System. Esta localização irá ser alimentada à "Implementer App" que por sua vez a irá alimentar à aplicação desenvolvida para processamento.

Capítulo 4

Implementação

A aplicação consiste de 3 componentes já descritas anteriormente: uma API (biblioteca de processamento de dados), uma base de dados e um serviço. É responsável pela sua conjugação e por providenciar uma interface às suas funcionalidades, como por exemplo mostrar QR Codes gerados ou mostrar uma lista de localizações que o utilizador pode partilhar. Esta aplicação funciona como um prestador de serviços e gerador de conteúdos para outras aplicações que a podem implementar, com algumas alterações, como é exemplificado no capítulo 4.2.

4.1 Biblioteca de Processamento de Dados

A fase inicial do trabalho de implementação consistiu na criação de uma biblioteca Java para geração de informações personalizadas sobre uma dada localização. Para além dessa funcionalidade, a biblioteca também seria responsável por ler e escrever para uma base de dados, que contém todas as informações relevantes à partilha de localização de um utilizador, e por suportar a troca de chaves inicial necessárias a uma comunicação segura, usando o algoritmo Diffie-Hellman com recurso a QR Codes.

4.1.1 Métodos para Geração de Informações Personalizadas

De modo a providenciar aos utilizadores das aplicações de localização um maior controlo sobre a informação partilhada, foram desenvolvidos vários métodos de geração de localizações relativas a partir de localizações absolutas (i.e. coordenadas). Estes são:

- **Geração de geohash** - um geohash corresponde a uma String (i.e. hash) gerada a partir de coordenadas absolutas que representa uma área dentro da qual um utilizador se encontra. Os geohashes são utilizados em notificações de proximidade: um utilizador partilha o hash com outro e, se esse utilizador estiver dentro da área representada, significa que se encontra próximo do utilizador que lhe enviou o geohash, nunca sendo revelada a posição absoluta de nenhum dos utilizadores;
- **Criação de pontos de encontro** - utilizador pode criar um ponto de encontro que será armazenado pela biblioteca numa base de dados local. Este ponto pode depois ser partilhado com outros utilizadores;

- **Criação de geofences** - utilizador pode criar uma área à volta de um determinado ponto à sua escolha, como por exemplo a sua casa. Sempre que o utilizador entrar ou sair do geofence, a pessoa que o utilizador escolheu notificar irá receber uma notificação a dizer que o utilizador entrou/saiu do geofence;

- **Geração de tempo/distância de um ponto** - utilizador consegue partilhar tempo e/ou distância a que se encontra de um dado ponto de encontro ou geofence definidos previamente. A distância é calculada com recurso à fórmula de haversine [9];

- **Geração de uma lista de localizações relativas** - é apresentada uma lista ao utilizador de localizações próximas onde ele se pode encontrar. Em adição, é efectuado um parse da String de localização recebida da Google Maps API, devolvendo uma lista de localizações menos específicas de entre as quais o utilizador poderá escolher qual o grau de precisão com quer partilhar a sua localização. Por exemplo, a sua localização poderá ser "Universidade de Aveiro 3810-193 Aveiro Portugal" e o utilizador é capaz de escolher partilhar apenas "Aveiro" ou, caso deseje ser mais específico, poderá partilhar a String inteira.

4.1.2 Base de Dados

Tendo o projecto adotado um protocolo end-to-end, a comunicação é efectuada directamente entre dois dispositivos, não havendo nenhum servidor central para efeitos de armazenamento de informação, de modo a salvaguardar a segurança e privacidade dos utilizadores. Assim, todas as informações necessárias à partilha da localização de um dado utilizador são armazenadas localmente no seu dispositivo, numa base de dados SQLite. Existem apenas 6 tabelas, correspondentes a:

- **Pontos de encontro** - é guardado o nome do ponto de encontro definido e as suas coordenadas (latitude e longitude);

- **Geofences** - é guardado o nome do geofence definido, o seu raio e as suas coordenadas;

- **Contactos** - tabela usada para guardar um identificador (nº de telemóvel no nosso caso), e a chave simétrica a ser usada na cifra e decifra das comunicações com o utilizador em específico;

- **Nº de telefone do utilizador** - é necessário a aplicação ter acesso ao número de telefone do utilizador, pois este é usado como identificador único. Guardar o número na base de dados evita ter que perguntar sempre que o utilizador abre a aplicação;

- **Notificações (novas)** - nesta tabela são guardadas as notificações enviadas por outros utilizadores que o utilizador ainda não marcou como vistas;

- **Notificações (vistas)** - nesta tabela são guardadas as notificações enviadas por outros utilizadores que o utilizador já marcou como vistas;

Foram também criados vários métodos, dentro da biblioteca de processamento, para realizar operações CRUD (Create, Read, Update, Delete) na base de dados.

4.1.3 Troca de Chaves (Diffie-Hellman com Recurso a QR Codes)

Como foi mencionado anteriormente, a troca de chaves necessária para uma comunicação segura entre dois utilizadores é efectuada presencialmente através do algoritmo de Diffie-Hellman [8], com recurso a QR Codes.

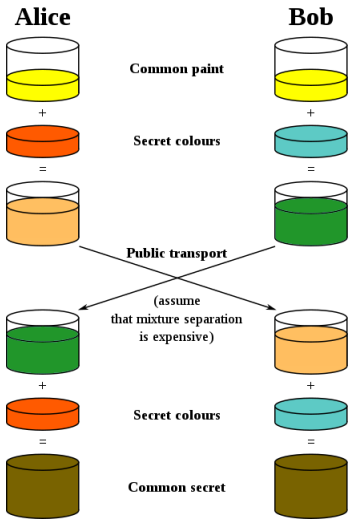

Figura 4.1: Diffie-Hellman

Antes de ser possível qualquer comunicação, é necessário que um dos utilizadores adicione o outro como "amigo" na aplicação desenvolvida. Adoptando o exemplo com cores na figura 4.1, um utilizador (Bob) que queira adicionar outro como amigo (Alice) irá gerar um QR Code contendo o seu número de telemóvel (lido automaticamente pela aplicação ou, caso isso não seja possível, pedido ao utilizador) e a cor verde, correspondente a um valor secreto apenas do conhecimento de Bob. A Alice lê o QR Code, que irá levar a aplicação a guardar o número de telemóvel do Bob e a gerar um QR Code contendo o valor secreto da Alice. O Bob terá então que ler o QR Code da Alice para concluir a troca de chaves, sendo estas armazenadas na base de dados local. No exemplo com cores acima, a cor amarela corresponde a algo comum que os dois utilizadores concordam em utilizar: um número primo muito grande (p) e um número inteiro (g)[1]. A cor verde corresponde a $B = g^b \bmod p$, em que b é um valor inteiro secreto gerado pelo Bob, correspondente à cor azul. De igual modo, a cor laranja-claro corresponde a $A = g^a \bmod p$, em que a é um valor inteiro secreto gerado pela Alice, correspondente à cor laranja. Os dois utilizadores irão, depois de partilhar os valores A e B um com o outro através de QR Codes, computar um valor s. No caso do Bob, $s = A^b \bmod p$ e no caso da Alice, $s = B^a \bmod p$. O valor de s é igual para ambos os utilizadores.

Assim, os utilizadores conseguiram gerar um segredo comum a partir da combinação dos seus segredos. O valor de s irá posteriormente ser usado para gerar uma chave simétrica AES de 128

[1]Na implementação, utilizaram-se valores gerados previamente e definidos no RFC 3526 (1536-bit MODP group) [10] de modo a evitar enormes tempos de espera para geração de QR Codes. Antes destes valores serem usados, a aplicação desenvolvida também era responsável por gerá-los, verificando-se uma espera entre 30 a 120 segundos, em média, para a geração de um QR Code, algo inaceitável para um utilizador de uma aplicação.

bits que será utilizada na cifra e decifra de mensagens trocadas entre o Bob e a Alice, garantindo a segurança das mesmas, pois apenas os utilizadores possuem a chave necessária à sua leitura.

4.2 Serviço

De forma a permitir aos utilizadores da aplicação um maior controlo sobre a partilha da sua localização, decidiu-se que seria relevante implementar a funcionalidade de envio de notificações de proximidade. Para isto, iria ser usado o algoritmo de Geohash, já explicado anteriormente, para converter a localização de um utilizador numa String que representa uma área onde o utilizador se encontra; esta é depois comparada com a localização do utilizador que recebe o geohash e, se ele se encontrar dentro do mesmo, irá ser notificado que se encontra próximo do utilizador que lho enviou.

Foi decidido que a melhor implementação da funcionalidade descrita acima seria uma partilha periódica do geohash, em que o utilizador escolhia a precisão do hash e o intervalo de tempo em que a sua localização era partilhada. De modo a implementar a periodicidade desejada, foi usada a classe JobScheduler do Android [11].

O utilizador começa por definir a precisão do geohash (1 a 12, este último sendo o mais preciso) e o intervalo de tempo com que quer partilhar a sua localização. Por fim, define para quem é que a sua localização irá ser enviada. Após concluir este processo, é definido um "alarme" que irá acordar a nossa aplicação quando o intervalo de tempo expirar, fazendo com que esta envie o geohash com a precisão definida anteriormente da localização actual do utilizador para o destinatário escolhido.

4.3 Integração com o OwnTracks

Para dar mais relevância a todo o trabalho desenvolvido até então, seria importante conseguir fazer a integração de uma aplicação de partilha de localização já existente no mercado com a app desenvolvida (CMIYC), de modo a demonstrar todas as funcionalidades que a nossa aplicação acrescentava. Surgiu então a oportunidade de fazermos a integração com a aplicação OwnTracks (referida na secção 2.1.1) uma vez que é open source e, por isso, seria posível alterar/adicionar as funcionalidades que fossem necessárias.

Como já foi referido anteriormente, o OwnTracks apenas fornece à aplicação o canal de comunicação, neste caso um broker MQTT, e as coordenadas GPS, que depois são tratadas pela aplicação desenvolvida para produzir conteúdo, como por exemplo várias moradas perto da localização atual do utilizador com diferentes graus de precisão, ou então o tempo e distância que o utilizador demora a chegar a um ponto de encontro definido anteriormente.

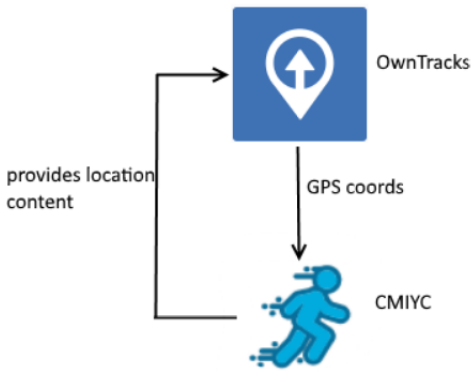

Figura 4.2: Diagrama de interação entre as duas aplicações

Para fazer a integração das duas apps foi necessário fazer algumas alterações à aplicação OwnTracks, nomeadamente na parte que envolve a escrita/leitura do broker MQTT. Uma vez que todos os dados trocados entre utilizadores têm que ser cifrados, toda a informação que é escrita para o broker tem de passar pela aplicação CMIYC (Catch Me If You Can) para poder ser cifrada com a respetiva chave do utilizador e só posteriormente é enviada para o canal de comunicação. Por esta razão, foram também necessárias alterações do lado da leitura de dados do broker: quando o destinatário recebe a mensagem, esta tem de ser primeiro redirecionada para a aplicação CMIYC com o intuito de ser decifrada e só depois é que o utilizador receberá uma notificação.

Ao nível da construção das mensagens foram também necessárias algumas alterações, como por exemplo, acrescentar um tag em todas as mensagens (JSON) que identificasse o destinatário para que todos os utilizadores que estivessem subscritos ao mesmo broker não lessem as mensagens que não lhes eram destinadas. Uma vez que o único tipo de mensagens que o OwnTracks implementava permitiam apenas a partilha da localização absoluta, foi ainda necessário criar 4 novos tipos de mensagem para que fosse possível integrar todas as novas funcionalidades: GeoHash, Address, MeetingPoint e Geofence.

4.3.1 Comunicação por Intents

Toda a comunicação feita entre as duas aplicações é efetuada através de `Intents` implícitos. Um `Intent` é uma descrição abstrata de uma operação a ser realizada e, sendo implícito, não especifica nenhum componente Android em específico como destinatário, podendo ser recebido por qualquer aplicação capaz de realizar a acção pedida [12]. Encontra-se ilustrado na figura 4.3 um `Intent` enviado do OwnTracks para a nossa aplicação:

```
public void hiddenIntentToSpecificAppProperly(String extra, String extraValue){
    Intent intent2 = new Intent(Intent.ACTION_SEND);
    intent2.setType("text/plain");

    PackageManager packageManager = getApplicationContext().getPackageManager();
    List<ResolveInfo> activities = packageManager.queryIntentActivities(intent2, 0);
    boolean isIntentSafe = activities.size() > 0;

    // Start an activity if it's safe
    if (isIntentSafe) {
        for (ResolveInfo activity : activities) {
            if (activity.activityInfo.packageName.contains("currentplacedetailsonmap")) {
                intent2.setData(Uri.parse("receiver_address://123"));
                intent2.setPackage(activity.activityInfo.packageName);
                intent2.putExtra(extra, extraValue);
                intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                getApplicationContext().sendBroadcast(intent2);
            }
        }
    }
}
```

Figura 4.3: Envio de um Intent

Como é percetível da figura, primeiro está a ser criado um `Intent` do tipo `text/plain`. Do lado do CMIYC foi especificado que a aplicação recebesse `Intents` deste tipo. Depois de o `Intent` ser criado, é necessário verificar, antes de o enviar, se existe alguma aplicação instalada no dispositivo do utilizador capaz de receber `Intents` deste tipo, e caso isso se verifique, é então necessário procurar pela aplicação certa, que neste caso chama-se "currentplacedetailsonmap", que identifica a nossa aplicação. Só depois de concluído este processo é que é enviado o `Intent`, com a chamada da função `startActivity(intent)`.

A função `startSpecificAppProperly()`, representada na figura 4.4, é responsável por acordar a aplicação CMIYC quando é necessária alguma interação do utilizador em que é preciso mostrar a interface dessa mesma aplicação. Por exemplo, quando o utilizador está a usar a app OwnTracks e carrega na opção de partilhar a sua localização, irá ser redirecionado para a nossa aplicação, onde vai pode escolher não só a informação que deseja partilhar, mas também com quem a deseja partilhar. Toda esta interação é feita entre o utilizador e a nossa aplicação. Quando o utilizador carrega na opção de enviar a informação para o destinatário escolhido, irá ser redirecionado para a aplicação inicial, o OwnTracks.

Porém, caso a interação entre a aplicação OwnTracks e CMIYC necessite de ser feita em background, ou seja, caso não seja necessário o utilizador interagir com a interface da nossa aplicação, não existe necessidade dessa interface ser mostrada, sendo os `Intents` enviados entre aplicações um pouco diferentes neste caso, como podemos verificar na figura 4.4:

```
//starts a specific app without user interaction
public void startSpecificAppProperly(String extra,String extraValue){

    Intent intent = new Intent();
    intent.setAction(Intent.ACTION_SEND);
    intent.setType("text/plain");

    PackageManager packageManager = getPackageManager();
    List<ResolveInfo> activities = packageManager.queryIntentActivities(intent, 0);
    boolean isIntentSafe = activities.size() > 0;
    System.out.println("isIntentSafe : "+isIntentSafe );
    System.out.println("activities list "+activities.toString());
    // Start an activity if it's safe
    if (isIntentSafe) {
        for (ResolveInfo activity : activities){
            if(activity.activityInfo.packageName.contains("currentplacedetailsonmap")){
                intent.setPackage(activity.activityInfo.packageName);
                intent.putExtra(extra,extraValue);
                startActivity(intent);
            }
        }
    }
}
```

Figura 4.4: Envio de um Intent por broadcast

A diferença destes `Intents` para os anteriores é que para além de termos que especificar o tipo dos mesmos, temos ainda que especificar o identificador associado ao tipo de informação enviado (URI). Do lado do CMIYC foi criada uma classe para lidar com os mesmos, `DecodeAddressReceiver`, que extende a classe `BroadcastReceiver`, que permite responder a `Intents` e/ou eventos de outras aplicações em background [13].

Na figura 4.5, podemos ver a definição da classe `DecodeAddressReceiver` no manifesto XML da aplicação CMIYC:

```
<receiver android:name=".DecodeAddressReceiver">
    <intent-filter>
        <action android:name="android.intent.action.SEND" />

        <data android:scheme="receiver_address" />
    </intent-filter>
</receiver>
```

Figura 4.5: Manifesto XML, definição de identificador dos Intents que a classe DecodeAddressReceiver irá receber

Esta classe vai estar constantemente à escuta de `Intents` cujo identificador se inicie por "receiver_address". Estes `Intents` vão ser recebidos pelo `BroadcastReceiver`, que depois irá executar uma ação conforme o `Intent` recebido. Por exemplo, se o `Intent` for um pedido ao CMIYC

para que retorne todos os pontos de encontro e/ou geofences existentes, o `BroadcastReceiver` vai, ao receber este `Intent`, fazer uma query à base de dados (através de métodos existentes na biblioteca criada) para retornar todos os pontos de encontro, que serão enviados de volta para o OwnTracks, através de outro `Intent`. Neste exemplo específico, estes `Intent`s são sempre enviados quando o OwnTracks é iniciado, de modo a poder mostrar ao utilizador todos os pontos de encontro (e/ou geofences) no mapa.

Este dois casos são exemplos de interações entre as duas aplicações, onde o utilizador não tem qualquer noção de que estão a acontecer, já que são efetuadas em background.

Capítulo 5

Conclusão

5.1 Sumário

No início deste relatório é apresentado o problema de segurança e privacidade existente em muitas das aplicações de partilha de localização actualmente existentes no mercado e é definido como objectivo do nosso projecto a criação de uma aplicação móvel que seja centrada nestes dois aspectos, em todas as fases de desenvolvimento.

Antes de começar a fase de desenvolvimento, foram analisadas aplicações existentes, entre as quais o OwnTracks [2], de modo a perceber tanto as funcionalidades como as deficiências que são transversais a este tipo de aplicações. Foram também analisadas várias tecnologias que seriam usadas ao longo do processo de desenvolvimento do nosso projecto, como o algoritmo Geohash [5], E2EE (end-to-end encryption) [7] e a Google Maps API [2].

Começou depois uma fase de pré-desenvolvimento, em que traçámos os requisitos que a nossa aplicação tinha que cumprir, como a partilha de localização personalizável, a troca presencial de chaves simétricas usadas na cifra e decifra de mensagens trocadas entre utilizadores, e a capacidade de ser integrada com aplicações já existentes no mercado. Nesta fase também foi concebida a arquitectura do projecto, que passaria a incluir o OwnTracks como aplicação que iria integrar a aplicação desenvolvida por nós.

Posteriormente, passámos à implementação do nosso projecto, que é discutida a fundo no Capítulo 4 deste relatório, com especial ênfase na integração da nossa aplicação com o OwnTracks, uma parte que consideramos de extrema importância.

Os resultados obtidos, conclusões e ideias para trabalho futuro são discutidos neste Capítulo. Em suma, consideramos que todos os requisitos iniciais propostos foram cumpridos, apesar de existir ainda margem para melhorias.

5.2 Principais Resultados Obtidos

Tendo em conta os requisitos iniciais deste projeto, pensamos que estes foram cumpridos. Há ainda algum trabalho de melhoramento a fazer a nível da solução de integração apresentada; nomeadamente, teria sido preferível manter o interface completamente do lado da aplicação que integra a nossa, deixando-a fora da visibilidade do utilizador para que pudesse funcionar única e exclusivamente como uma biblioteca ou um serviço em segundo plano.

Fora isso, a integração pode ser feita de uma forma relativamente fácil se a aplicação que oferece o canal de comunicação estiver minimamente preparada para apresentar a informação gerada pela nossa aplicação.

A solução é modular, temos a nossa aplicação que nos permite ter toda a componente de serviço com os controlos de periodicidade e depois temos a nossa API que é importada através de um ficheiro AAR (Android Archive). Similar a esta API, existe uma versão em formato JAR (Java Archive) que pode ser utilizada para aplicações desktop.

5.3 Trabalho Futuro

A título de trabalho futuro, reconhecemos que ainda pode haver bastante a fazer e que este projeto pode ser estendido a outras áreas que achamos interessantes.

No que toca a correções e melhoramentos achamos que ainda se poderiam implementar outras modalidades de partilha de informação e melhorar as existentes. No que toca a periodicidades, acrescentar outras opções como periodicidade com base na distância percorrida seria útil.

Com visão ao progresso tecnológico futuro e visto que agora já é possível fazer partilha segura da nossa localização podíamos aproveitar a nossa aplicação para controlo de smarthomes, por exemplo. Se quiséssemos ter um chá preparado quando chegássemos a casa no fim do dia podiamos utilizar os alertas de proximidade ou os geofences para comunicar, sempre de uma forma segura, com a nossa casa, ligando a nossa chaleira elétrica para que preparasse a nossa bebida. O mesmo poderia-se fazer para ligar o ar condicionado ou aquecimento. A nossa casa adaptar-se-ia aos nossos horários de uma forma bastante mais dinâmica. Sempre que estivéssemos a uma determinada distância ou tempo poderíamos ativar eletrodomésticos ou outros equipamentos.

Integração em aplicações para sistemas de infoentretenimento de automóveis também seria um possível objetivo futuro. Poderíamos partilhar a nossa localização sem precisar de interação contínua e ativa ou até enviar rápidas mensagens de localização já pré-definidas pela aplicação com base nos locais nas imediações extraídos da Google Maps API, mantendo a segurança na condução e ao mesmo tempo dos nossos dados.

Referências

[1] N. Council, D. Sciences, C. Systems, and A. Board, *The Global Positioning System: A Shared National Asset*. National Academies Press, 1995.

[2] "Owntracks booklet."

[3] "Google maps api documentation."

[4] "Faq - frequently asked questions | mqtt."

[5] C. Veness, "Movable type scripts."

[6] "What is geo-fencing (geofencing)? - definition from whatis.com."

[7] "End-to-end encryption."

[8] M. E. Hellman, B. W. Diffie, and R. C. Merkle, "Cryptographic apparatus and method," Apr. 29 1980. US Patent 4,200,770.

[9] Rick and Peterson, "Haversine formula," Apr 1999.

[10] T. Kivinen and M. Kojo, "More modular exponential (modp) diffie-hellman for internet key exchange (ike)," May 2003.

[11] "Jobscheduler | android developers."

[12] "Intent | android developers."

[13] "Broadcasts overview | android developers."