Relatório Técnico - iOS

Spoty - Geocaching Application

Data: Aveiro, 10 de Junho de 2017

Autores: 71902: Pedro Ferreira de Matos
71770: Tiago Alexandre Lucas de Bastos

Resumo: O Spoty é uma aplicação desenvolvida em Swift para a plataforma iOS. O objetivo da aplicação é simular uma aplicação de geocaching, em que em localizações específicas do mapa existem códigos QR que têm de ser lidos pelo utilizador. Além de descobrir e ler estes códigos, o utilizador tem a possibilidade de os partilhar com outros utilizadores através da tecnologia multipeer.

1 Introdução

Esta aplicação é feita em Swift 3.0 e tem como recursos as livrarias de iOS para utilização de mapas, leitura de QR codes e conectividade por multipeer.

Na aplicação a autenticação do utilizador está disponível através da base de dados do Firebase. Após a autenticação, o utilizador é transportado para uma vista com um mapa que indica a sua localização atual e que serve de “menu”, onde consegue visualizar as localizações de QR codes mais próximas. Ao ler um QR code, este depois será guardado juntamente com a localização do utilizador aquando da leitura; existe uma vista com todos os locais visitados e as mensagens dos QR codes lidos; e uma vista em que o utilizador tem acesso a outros utilizadores disponíveis para a troca de dados.

2 Âmbito e conceito da aplicação

O Spoty é uma aplicação de geocaching, que foi criada com âmbito de o utilizador ter de se deslocar para chegar às localizações/objetivos, e nas mesmas ler um QR code que contêm uma mensagem. Desta forma, o utilizador tem de “colecionar” os QR que estejam disponíveis, sendo que para tal é necessário que o utilizador se desloque, levando o a conhecer pontos turísticos ou históricos das redondezas e a promover a prática de exercício físico.

3 Detalhes técnicos

Design da Interface Visual

View Controllers

A programação móvel em iOS distingue-se bastante da programação em Android, e uma das diferenças mais importantes está no uso das View Controllers. Neste caso, não há a necessidade de escolher entre uma Actividade ou um Fragmento. Cada uma das nossas vistas é uma View Controller.

Mapa

Para emular um Mapa na aplicação foi usada a livraria MapKit que permite a criação de um MKMap na própria aplicação ou o uso da própria aplicação Mapas do iOS. No entanto, para o âmbito desta aplicação achamos apropriado o Spoty ter um mapa próprio.

O mapa vai indicar sempre a localização atual do utilizador, e pode mostrar também os pontos de leitura de QR codes que o utilizador tem de visitar. Estes pontos/localizações estão divididos em 2 grupos/sessões. Uma sessão tem 3 localizações e a outra tem 2. O utilizador pode optar qual sessão pretende ver no mapa através do botão “Change path”.

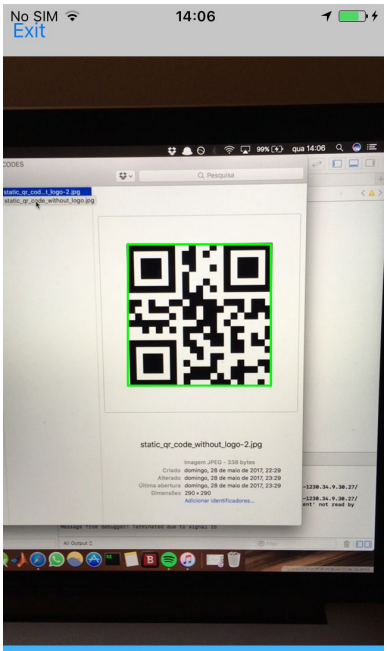
Ao haver uma leitura de QR code, o sistema vai guardar a localização em que o QR code foi lido e através de uma algoritmo de comparação vai decidir se a localização onde o QR code foi lido pertence a uma das localizações base. Se coincidir, essa localização vai deixar de ser mostrada ao utilizador no mapa. Desta forma, o utilizador sabe quais as localizações que faltam.

```
let coordinate_casa = CLLocation(latitude: (Double(lat_sess))!, longitude: (Double(long_sess))!)
let coordinate_array = CLLocation(latitude: Double(lat_array)!, longitude: Double(long_array)!)
let distanceInMeters = coordinate_casa.distance(from: coordinate_array)


if(distanceInMeters < 100){
    tmp = 1
}
```

QR code reader

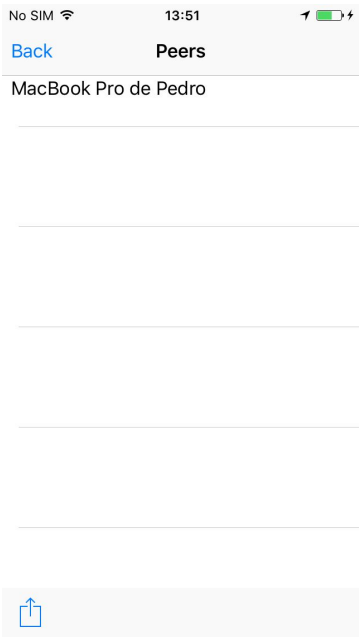
Um dos objetivos da aplicação é a leitura de QR codes de forma a conseguir armazenar na aplicação as mensagens que cada um contém. A mensagem que vem no QR code é armazenada numa lista que vai ter a mensagem e a localização onde a mensagem foi lida. Esta informação é guardada de forma persistente na aplicação, o que significa que mesmo após um novo login a informação vai continuar disponível.

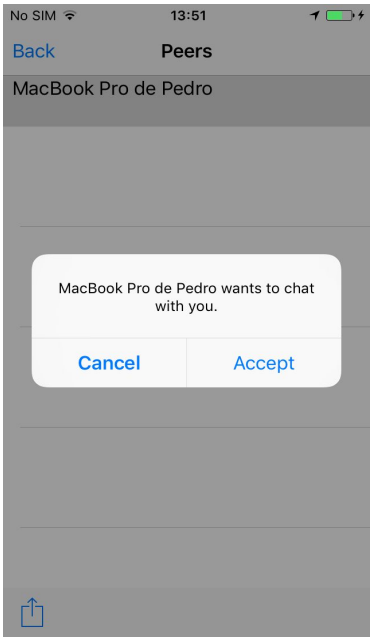


cm-apresentacao

Multipeer

A aplicação permite a troca de informação por multipeer entre utilizadores. A informação partilhada são as mensagens de QR code e as respectivas localizações. Para que haja esta troca, é necessário que ambos os utilizadores aceitem comunicar entre si.

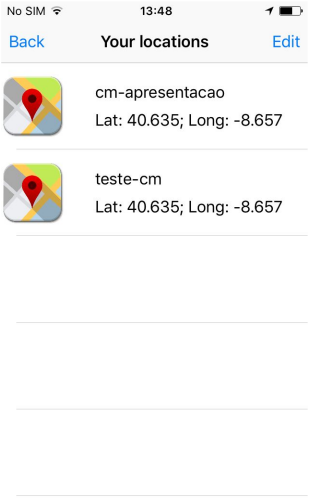




Como podemos ver nas figuras, cada utilizador tem acesso à lista de Peers existentes, e apenas é possível a troca de dados quando ambos aceitam. A troca de informação é automática, e após ter sido aceitei, a troca de dados é instantânea e a lista de mensagens dos QR codes é atualizada automaticamente, sendo que apenas as novas mensagens são adicionadas devido à verificação de mensagens repetidas.

Lista de mensagens e localizações

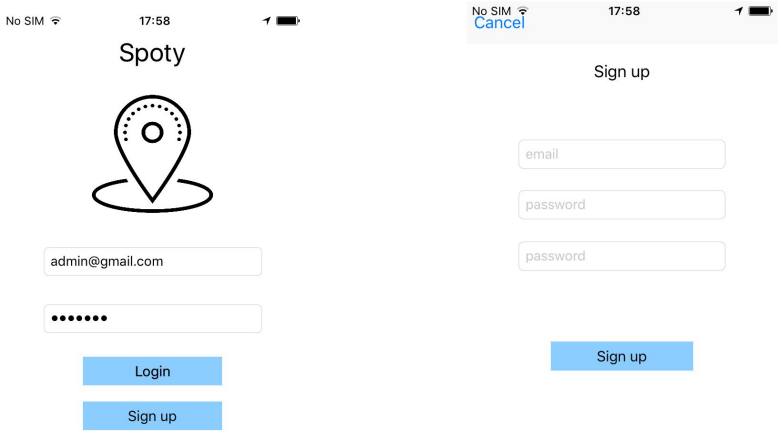
A aplicação tem uma lista de todas as mensagens lidas, juntamente com a localização do utilizador aquando da leitura do QR code. Essa lista é atualizada automática quando é lido um novo QR code, ou quando é recebida informação proveniente de outro peer.



Como é possível verificar na figura, estão presentes duas localizações, em que na primeira linha da célula está identificada a mensagem que estava escondida no QR code e na segunda linha da mesma célula está a latitude e longitude onde o QR code foi lido.

Login

O login é realizado com recurso à base de dados da Firebase, que permite que haja o registo de utilizadores e a verificação dos mesmos. Desta forma, qualquer utilizador se pode registar na aplicação e fazer login na mesma, desde que a aplicação tenha ligação à Internet para poder haver a verificação na base de dados da Firebase.



Persistência Local

A aplicação guarda de forma persistente e local toda a informação relacionada com as mensagens dos QR codes e com as coordenadas visitadas pelo utilizador. Desta forma, sempre que as view que precisam desta informação são carregadas, a aplicação encarrega-se de ir buscar as informações mais recentes que estejam guardadas.

Esta informação é guardada no sistema operativo de cada vez que é lido um QR code, ou quando o utilizador apaga a informação que tem.

Para realizar esta persistência, foi necessário criar classes de forma a guardar objetos próprios. Assim, é possível obter todos os objetos de uma determinada classe que estejam guardados no sistema operativo, como é possível verificar na imagem através do **as? [Locations]**.

```
//MARK: - PERSISTENCE
private func loadLocs() -> [Locations]? {
    return NSKeyedUnarchiver.unarchiveObject(withFile: Locations.ArchiveURL.path) as? [Locations]
}

private func loadCoords() -> [CoordsClass]? {
    return NSKeyedUnarchiver.unarchiveObject(withFile: CoordsClass.ArchiveURL.path) as? [CoordsClass]
}
```

4 Solução Implementada em iOS

Este trabalho não é idêntico ao realizado em Android, porque o Android Things é exclusivo da plataforma Android e não era possível realizar o mesmo projeto em iOS.

Desta forma, a nossa aplicação em iOS é totalmente diferente. No entanto, é possível estabelecer algumas conclusões devido aos conhecimentos obtidos em ambas as plataformas.

Em Android há uma grande vantagem, que é a da linguagem de programação ser uma linguagem que qualquer programador conhece e está familiarizado, enquanto que para aprender Objective-C ou Swift pela primeira vez é necessário perder mais tempo e procurar mais informação para criar a aplicação desejada. Outro fator importante, é o facto de o Xcode ser uma IDE mais leve que o Android Studio e o simulador de o iPhone funcionar muito bem no Macbook, enquanto que mesmo em Linux o simulador de um telefone Android deixa muito a desejar. Outros problemas irão ser considerados na conclusão.

5 Conclusão

O nosso grupo não pode desenvolver a mesma aplicação em Android e em iOS. No entanto, apesar de não termos tido essa possibilidade, conseguimos compreender algumas diferenças importantes entre a programação nas duas plataformas.

Como já foi dito no ponto 4, o facto de a programação em iOS ser um mundo à parte, é um mundo em que é muito fácil de se entrar. Existe inúmera documentação disponível e exemplos na Internet e o Xcode é um IDE muito agradável, ainda que a linguagem de programação seja diferente daquilo a que estamos habituados. Além disso, o facto de a programação em iOS ser muito fácil de executar com drag&drop ajuda a criar a interface gráfica de uma forma muito mais intuitiva e rápida do que com o Android. Desta forma, achamos que a programação em iOS é mais rápida e intuitiva.

Quanto ao projeto, não tivemos muitas dificuldades porque com alguma pesquisa e esforço foi possível encontrar documentação e exemplos que nos permitessem evoluir no projeto e criar as features que eram exigidas. Uma das features mais interessantes tratou-se do mapa e da comunicação multipeer. Como em Android estamos habituados a criar Mapas com o Google Maps e as aplicações em iOS usam a biblioteca MapKit o que criou alguma dificuldade iniciar porque era mais uma informação nova. No entanto, foi muito interessante trabalhar com a passagem de informação através de Multipeer. Foi um desafio muito interessante fazer com que fosse necessário os peers estabelecerem uma sessão entre eles para que pudessem trocar os dados. Isto foi muito importante porque antes de estabelecermos as sessões, se tentassemos trocar dados por Multipeer, era possível estabelecerem se demasiadas ligações a peers que não interessavam, o que provocava problemas na aplicação.

Concluímos que apesar de ambas as plataformas serem muito boas e termos tido facilidade em programas em ambas, a plataforma de iOS permite um desenvolvimento muito mais rápido e intuitivo da parte gráfica do que em Android. Além disso, como a programação em iOS está dividida por vistas, é mais fácil tornar as mesmas independendentes e dividir as funcionalidades. Podemos dizer que é uma programação mais “friendly” para o programador.

6 Referências e Recursos

http://www.appcoda.com/chat-app-swift-tutorial/
https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/CreateATableView.html
https://www.ralfebert.de/tutorials/ios-swift-multipeer-connectivity/
http://www.appcoda.com/barcode-reader-swift/
https://www.raywenderlich.com/136165/core-location-geofencing-tutorial
https://www.codeproject.com/Articles/869481/Using-MapKit-and-CoreLocation-Information-in-iOS
http://www.techotopia.com/index.php/An_Example_iOS_7_Location_Application
http://www.techotopia.com/index.php/IOS_7_App_Development_Essentials