FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Desenvolvimento de Apps Mobile Multi-OS com Requisitos de Informação Geográfica

Rúben Cristiano Campos Reis

Mestrado Integrado em Engenharia Informática e Computação

Orientador: António Miguel Pontes Pimenta Monteiro (Doutor)

Julho de 2015

Desenvolvimento de Apps Mobile Multi-OS com Requisitos de Informação Geográfica

Rúben Cristiano Campos Reis

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: José Manuel de Magalhães Cruz (Doutor)

Vogal Externo: Helena Rodrigues (Doutora)

Orientador: António Miguel Pontes Pimenta Monteiro (Doutor)

---

21 de Julho de 2015

Resumo

Nos últimos anos, os smartphones revolucionaram a forma como vivemos o nosso dia-a-dia. Passaram de simples aparelhos que apenas serviam para comunicar, para aparelhos poderosos e práticos que permitem realizar as mais variadas tarefas, de uma forma rápida e eficaz, em qualquer lugar.

Estas capacidades atraíram de imediato o mundo empresarial, que se apercebeu que existe uma grande vantagem em desenvolver aplicações móveis. Desta forma, qualquer empresa que queira ser inovadora nas soluções que apresenta e pretender liderar o mercado deve desenvolver as suas próprias aplicações. Mas existe uma grande variedade de dispositivos e de sistemas operativos e as empresas não têm capacidade, nem orçamento muitas vezes, para desenvolver para cada uma das plataformas. Surge então a necessidade de criar apenas uma aplicação que seja depois compatível com todos os sistemas operativos e que dê ao utilizador a experiência que esperam e a que estão habituados, de forma a abranger o máximo de utilizadores possíveis, com o menor custo e esforço possível.

Tal como muitas empresas, também a Gisgeo Information Systems, Lda sentiu essa necessidade de procurar soluções de custo reduzido e que permitissem desenvolver para os vários sistemas operativos, tendo apenas uma base comum. Dessa forma, o objetivo da dissertação é procurar e analisar as ferramentas que permitem criar essas aplicações, mas com um aspeto nativo, e que consigam aceder ao *hardware* do smartphone (câmara, GPS, acelerómetro,…). A Gisgeo é uma empresa especializada no desenvolvimento de *software* web e mobile com base geográfica e por isso é fundamental que as aplicações tenham acesso ao GPS do telemóvel, por exemplo.

Existem bastantes ferramentas que permitem resolver este problema, mas nem todas oferecem as funcionalidades requeridas para criar as aplicações da Gisgeo. Foi então efetuada uma análise cuidada para escolher a ferramenta mais adequada e depois desenvolvida (usando essa ferramenta) uma base tecnológica comum e uma aplicação para demonstrar os resultados.

Os pontos interessantes nesta dissertação foram a descoberta de uma *framework* que melhor suporta as necessidades apresentadas pela Gisgeo e que consigue aceder aos vários elementos do smartphone (até ao mais baixo nível) e a otimização da utilização de cartografia digital para mobile. Espera-se que, com essa base, seja possível à Gisgeo, criar todas as suas aplicações móveis para os vários sistemas operativos sem qualquer custo e com o menor esforço possível.

Abstract

In recent years, smartphones have revolutionized the way we live our life. They went from simple devices that only served to communicate to powerful and pratical equipment for carrying out the various tasks quickly and efficiently, anywhere.

These capabilities immediately attracted the business world, when they realized that there was a great advantage in developing mobile applications. Thus, any company that wants to be innovative in present solutions and want to lead the market should develop their own applications. But there is a wide variety of devices and operating systems, and some companies don't have the capacity or budget to develop for each platform. Then arises the need to create a single application that is compatible with all operating systems and give users the experience they want and are used to in order to cover the maximum possible users, with the lowest cost and effort possible.

Like many companies, also Gisgeo Information Systems, Lda felt the need to seek cost--effective solutions that allowed the development for multiple operating systems, having only one common core. Thus, the purpose of this dissertation was to look and analyse the tools that let us create these applications, but with native looking and with access to the smartphone hardware (camera, GPS, accelerometer,…). Gisgeo is a company specialized in developing web and mobile software with geographic information and so it's critical that applications have access to the phone's GPS, for example.

There are plenty of tools that allow to solve this problem, but not all offers the features required to create Gisgeo's applications. So, careful consideration was made to choose the most appropriate tool and then, using it, will be developed (using this tool) a common technology base was developed as well as an application to demonstrate the results.

The interesting points in this thesis where the finding of a framework that best supports the needs presented by Gisgeo and with access to the various smartphone elements (down to the lowest level) and the optimization of the use of digital maps for mobile. It is expected that, on that common core, it's possible to Gisgeo create all their mobile applications for multiple operating systems at reduced cost and with the least possible effort.

Agradecimentos

Este espaço é dedicado àqueles que contribuíram para que esta dissertação fosse realizada e que me ajudaram a concluir o Mestrado em Engenharia Informática e Computação. A todos eles, deixo apenas algumas palavras mas um profundo e sincero sentimento de reconhecido agradecimento.

Em primeiro lugar gostaria de agradecer ao meu orientador Miguel Pimenta Monteiro, por aceitar o meu convite para meu orientador, por me ter acompanhado ao longo de toda a dissertação e por ter partilhado alguma da sua experiência e sabedoria.

Em segundo lugar, à Gisgeo e aos meus colegas de trabalho, por todo o apoio, disponibilidade e amizade que demonstraram durante a realização da minha dissertação.

Por último, mas obviamente não menos importante, gostaria de agradecer às pessoas que sempre me apoiaram durante o meu Mestrado.

Aos meus pais, Adélia e Valentim, pela forma como me educaram, pelos esforços que fizeram e pela confiança e valores que me incutiram para que eu conseguisse realizar os meus sonhos e atingir os meus objetivos. Pelo apoio incondicional e por terem acreditado em mim. Um enorme obrigado!

Ao meu irmão, Nelson, por toda a compreensão, apoio, incentivo e amizade que sempre demonstrou durante o meu percurso. Obrigado irmão!

À minha namorada, Ana, um agradecimento especial pelo apoio e carinho diários, pela transmissão de confiança e palavras de força, em todos os momentos e por ter lutado a meu lado por termos uma boa vida juntos no futuro. Muito obrigado!

Aos meus tios e prima (Dulce, Amadeu e Rute) por todo o apoio e carinho que sempre me transmitiram. Um sincero obrigado!

À minha avó, por toda a educação e inspiração que sempre me transmitiu. Obrigado.

Um enorme obrigado a todos por acreditarem em mim e naquilo que faço. Espero um dia poder retribuir e compensar todo o carinho, apoio e dedicação que sempre me ofereceram. A eles, dedico este trabalho.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

base64	Técnica de codificação/descodificação de dados binários para texto (e vice-versa)
CSS	*Cascading Style Sheets* (linguagem utilizada nas folhas de estilo que definem a apresentação de documentos escritos em HTML ou XML)
Desktop	Termo usado para definir computadores que não são portáteis
IDE	*Integrated Development Environment* (ambiente integrado de desenvolvimento – ferramenta que facilita o desenvolvimento de software)
IETF	*Internet Engineering Task Force* (comunidade internacional que se preocupa com a evolução da Internet e o seu funcionamento)
IMAP	*Internet Message Access Protocol* (protocolo de gestão de *emails*)
IP	*Internet Protocol* (protocolo de comunicação usado entre as máquinas numa rede)
JSF	*Jabber Software Foundation* (antiga fundação responsável pela definição dos padrões das extensões do protocolo XMPP)
JSON	*JavaScript Object Notation* (formato de dados leve usado para a troca de informações e para a definição/inicialização de objetos em JavaScript)
MVC	*Model-View-Controller* (modelo de arquitetura utilizado no desenvolvimento de *software*)
MVVM	*Model View ViewModel* (modelo de arquitetura utilizado no desenvolvimento de *software)*
Nonce	Conjunto de caracteres gerado aleatoriamente usado para evitar *replay attacks*
POI	*Point of Interest* (ponto de interesse)
Push	A tecnologia *push* é um serviço de distribuição de conteúdo da Internet em que a informação sai de um servidor e é recebido por todos os clientes registados
QoS	*Quality of Service* (qualidade do serviço)
Replay attacks	Ataques maliciosos que consistem no atraso ou repetição das mensagens previamente enviadas com o objetivo de autenticação ou acesso a dados
Salt	Conjunto de caracteres usado pelo servidor para a cifragem de *passwords*
SASL	*Simple Authentication and Security Layer* (Camada Simples de Autenticação e Segurança)
SASS	*Syntactically Awesome Stylesheets* (forma de simplificar a criação de folhas de estilo CSS)

SCRAM	*Salted Challenge Response Authentication Mechanism* (protocolo de autenticação cliente-servidor baseado em desafios e respostas)
SDK	*Software Development Kit* (kit de desenvolvimento de software)
SIG	Sistemas de Informação Geográfica
SMS	*Short Message Service* (Serviço de Mensagens Curtas)
SMTP	*Simple Mail Transfer Protocol* (protocolo padrão para o envio de *emails*)
TCP	*Transmission Control Protocol* (protocolo de controlo de transmissões)
TLS	*Transport Layer Security* (protocolo de segurança que protege as comunicações via *internet*
XML	*eXtensible Markup Language* (linguagem que define um conjunto de regras para a codificação de documentos num formato legível para humanos e máquinas)
XMPP	*Extensible Messaging and Presence Protocol* (protocolo aberto e extensível baseado em XML usado frequentemente para a troca de mensagens instantâneas)

Capítulo 1

Introdução

A indústria dos telemóveis sofreu uma revolução com o aparecimento dos primeiros smartphones. Antes de surgir o primeiro smartphone, os telemóveis apenas eram usados para fazer chamadas telefónicas ou enviar mensagens escritas. Mas nos dias que correm essas são tarefas triviais, sendo o smartphone usado na maioria das tarefas diárias do utilizador, como aceder ao *e--mail* e às redes sociais, jogar, ouvir música e tirar fotografias. Estas capacidades fizeram com que os fabricantes de smartphones entrassem numa luta constante para conquistarem a liderança do mercado. Os novos dispositivos estão cada vez mais poderosos e com mais e melhores funcionalidades.

Outro fator que sempre atraiu os utilizadores foi a capacidade da instalação de aplicações de terceiros para que o smartphone possa assim aumentar, ainda mais, o leque de funcionalidades suportadas. Este fator atraiu também o mundo empresarial, que se apercebeu das vantagens de desenvolver aplicações móveis. Cada vez mais existem aplicações empresariais e se uma empresa pretende ser líder de mercado e ser reconhecida como inovadora e na vanguarda da tecnologia, deve desenvolver as suas aplicações móveis, quer seja para usufruir internamente, quer seja para distribuir pelos clientes.

Mas com o evoluir do mercado de smartphones, começaram a aparecer cada vez mais dispositivos e com características diferentes uns dos outros. Assim, é quase impossível desenvolver uma aplicação, para os vários sistemas operativos móveis, que corra em todos os dispositivos, desde o mais antigo até ao mais recente. Surgiu então a necessidade de criar aplicações multiplataforma e foi assim que surgiram as primeiras frameworks que permitiram o desenvolvimento simultâneo para vários sistemas operativos.

O número deste tipo de frameworks tem vindo a aumentar, mas nenhuma delas produz ainda aplicações com um nível de qualidade equiparável às aplicações desenvolvidas nativamente. A escolha da framework adequada dependerá então do contexto e dos objetivos da aplicação que se quer desenvolver.

1.1 Contexto e Motivação

Tal como outras empresas, também a Gisgeo Information Systems, Lda decidiu aproveitar as vantagens de desenvolver uma aplicação móvel para os seus clientes. Mas, visto que existe uma vasta gama de dispositivos e sistemas operativos móveis diferentes, decidiu então que deveria desenvolver uma aplicação multiplataforma, que suportasse a maioria dos dispositivos Android e iOS, mas minimizando ao máximo os custos do desenvolvimento.

Fundada em 2008, a Gisgeo Information Systems, Lda é uma empresa especializada no desenvolvimento de Sistemas de Informação Geográfica. Apresenta soluções de *software web* e *mobile* capazes de capturar, armazenar, analisar, gerir e apresentar dados georreferenciados, principalmente usando tecnologias *open-source*.

A Gisgeo é reconhecida principalmente pelas suas aplicações baseadas na posição geográfica e, na hora de escolher uma framework de desenvolvimento, isto não pode ser esquecido. Deve ser escolhida uma ferramenta que permita aceder aos recursos do dispositivo (GPS e câmara, por exemplo) e que, se possível, seja *open-source*. Para além disso, espera-se que permita otimizar a utilização de cartografia digital.

1.2 Objetivos e Resultados Esperados

O principal objetivo desta dissertação é a escolha de uma *framework* adequada para o desenvolvimento de aplicações multiplataforma, de baixo custo. Depois de selecionada a melhor, é também um objetivo importante o desenvolvimento de uma base comum tecnológica que possibilite à Gisgeo Information Systems, Lda desenvolver futuramente todas as suas aplicações móveis. Espera-se que com essa camada padronizada seja possível reutilizar as funcionalidades comuns a todas as aplicações relacionadas com a cartografia digital, a localização, a orientação e representação fotográfica.

Para além desse objetivo, e como prova de conceito, dever-se-iam ainda desenvolver aplicações para demonstração dos resultados tanto em Android como em iOS.

1.3 Estrutura do Documento

Para além da introdução, este documento apresenta mais seis capítulos. No capítulo 2, para além de ser apresentada, de forma breve, a história dos smartphones e uma descrição dos três sistemas operativos móveis com maior destaque, é também apresentada uma explicação dos métodos de desenvolvimento de aplicações móveis. Para além disso é apresentada uma análise das ferramentas mais relevantes que permitem o desenvolvimento de aplicações multiplataforma. É também apresentada uma breve descrição das aplicações que permitem georreferenciação, assim como as suas principais funcionalidades e requisitos.

Nos capítulos seguintes (3,4 e 5) o tema principal é a base tecnológica a desenvolver. No capítulo 3 são especificados os requisitos principais, assim como as funcionalidades que a base tecnológica suporta e a sua arquitetura. O capítulo 4 foca-se em todo o processo de desenvolvimento da base tecnológica, onde é exposta, mais detalhadamente, a ferramenta Appcelerator Titanium assim como os detalhes da implementação da base tecnológica. Os módulos genéricos intermédios, desenvolvidos na base tecnológica, estão descritos no capítulo 5. Neste capítulo é também exposta a forma de utilização dos módulos e são apresentados alguns exemplos.

Para testar as capacidades da base tecnológica foram então desenvolvidas duas aplicações para prova de conceito que são definida e apresentadas no capítulo 6.

Por último, o capítulo 7 é composto pela exposição das conclusões tiradas ao longo de toda a dissertação e são apresentados também alguns planos para o trabalho futuro.

Capítulo 2

Computação móvel multiplataforma

Com o aparecimento dos smartphones, e o rápido aumento da sua utilização nos últimos anos, surgiu então a necessidade de se desenvolverem aplicações móveis.

Mas existem várias abordagens para criar essas aplicações. Desde logo existe o desenvolvimento utilizando a *framework* nativa que acompanha o sistema operativo, até ao uso de frameworks multiplataforma.

De seguida é apresentada uma breve história dos smartphones assim como as várias abordagens para se criarem aplicações para os sistemas operativos móveis mais usados.

No fim, após a análise de todas as abordagens existentes de desenvolvimento, é escolhida apenas uma de forma a satiafazer os requisitos apresentados pela Gisgeo.

2.1 História do *Smartphone*

Em primeiro lugar deve-se fazer a pergunta "O que é um smartphone?". E por muito que se procure, a definição é bastante vaga. Muitos definem um smartphone como um telemóvel que inclui *software* para realizar tarefas adicionais (como e-mail ou acesso à internet). Mas também há quem considere que um smartphone é "um telemóvel que executa muitas das funções de um computador, normalmente com uma interface sensível ao toque, com acesso à internet, e um sistema operativo capaz de executar aplicações instaladas remotamente " [OD01].

De qualquer forma, há uma linha bastante ténue entre o que é um smartphone e o que não o é nos dias que correm. Mesmo os telemóveis mais "básicos" são capazes de executar muitas das mesmas funções.

Mas onde tudo isto começou? Qual foi o primeiro smartphone?

O primeiro conceito de um smartphone surgiu em meados da década de setenta. Mas essa ideia não se concretizou durante quase vinte anos. Só em 1992 é que surgiu o primeiro protótipo

do que, dois anos mais tarde se tornaria o "IBM's Simon Personal Communicator". O "Simon" foi desenvolvido pela IBM e pela BellSouth e consistia num ecrã sensível monocromático, com uma caneta e uma base de carregamento e tinha a capacidade de aceder ao *e-mail* e enviar faxes.

Mas nos anos de 2007/2008 houve uma grande revolução no setor dos telemóveis com o aparecimento dos primeiros smartphones com sistema operativo iOS e Android.

Os smartphones são então uma extensão dos telemóveis normais. Os telemóveis conseguem fazer chamadas, enviar mensagens e até gravar vídeos e tirar fotografias, mas não têm as capacidades que um smartphone tem, como GPS, acesso à internet através de Wi-Fi, acelerómetro e outros sensores e toda uma enorme gama de outras funcionalidades e aplicações.

2.2 Sistemas Operativos Móveis

Um sistema operativo móvel é todo o sistema operativo que opere um smartphone, tablet ou outro tipo de dispositivo móvel. A principal função é gerir os recursos do sistema, fornecendo simultaneamente uma interface entre o dispositivo e o utilizador. Para além disso, o sistema operativo também controla todo o *software* instalado no dispositivo.

Existem vários sistemas operativos móveis, mas apenas dois deles são os mais usados. O Android apareceu em 2008 e teve logo como objetivo poder ser usado numa grande variedade de dispositivos móveis. O iOS, criado pela Apple em 2007, foi desenvolvido para os seus dispositivos iPod, iPhone e mais tarde os *tablets* iPad. Nos últimos anos o Android ganhou bastante terreno ao iOS. No terceiro trimestre de 2014 o Android era claramente o líder dos sistemas operativos móveis, como mostra o gráfico da Figura 1 [IDC01].

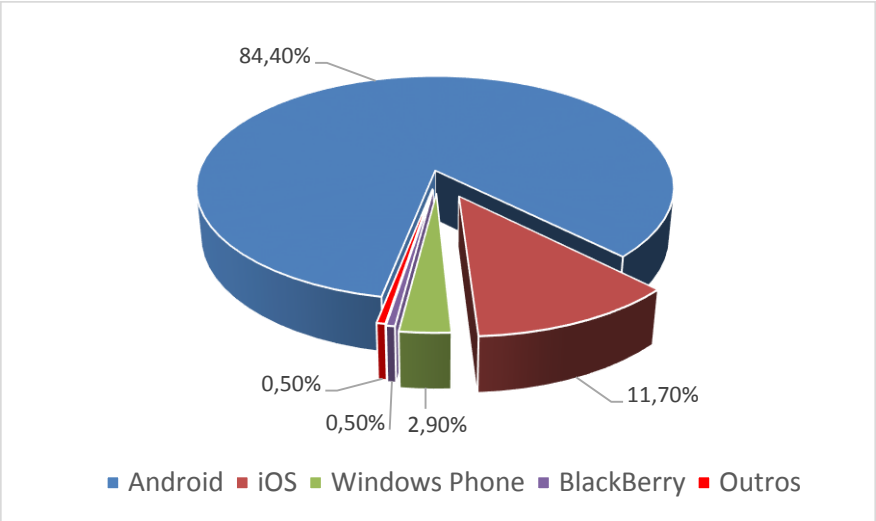


Figura 1: Comparação dos Sistemas Operativos Móveis

Houve uma grande variedade de sistemas operativos móveis, mas alguns deles, que tiveram grande sucesso nos primeiros anos, já quase desapareceram, como o Symbian e o BlackBerry.

2.2.1 Android

O Android é, atualmente, o líder de mercado dos sistemas operativos móveis e foi criado pela empresa Google. O primeiro smartphone com sistema operativo Android foi o HTC Dream e surgiu em 2008.

O Android foi desenhado para smartphones e tablets, mas com o decorrer dos anos foi também adaptado às televisões, automóveis, relógios, consolas, câmaras digitais e até computadores portáteis.

Este sistema operativo tem uma particularidade, é livre e de código aberto, o que encorajou um grande número de programadores a criarem projetos direcionados à comunidade. Estes projetos têm como objetivo adicionar novas funcionalidades ou simplesmente "trazer" versões mais recentes do Android para telemóveis mais antigos que, oficialmente, não receberiam o Android mais recente. Para além disso, o facto de ser livre e de código aberto permite aos utilizadores personalizarem os seus dispositivos.

2.2.2 iOS

O iOS é o segundo sistema operativo mais usado no mercado e foi desenvolvido pela Apple. O primeiro aparelho com sistema operativo iOS apareceu em 2007 (o sistema operativo ainda tinha o nome de iPhone OS).

Grande parte dos aparelhos da Apple (iPhone, iPod, iPad) têm o iOS como sistema operativo.

Ao contrário do Android, o iOS é desenvolvido em código proprietário, e não oferece o mesmo nível de personalização que o Android oferece.

O iOS é conhecido pelo estilo e fluidez que apresenta, uma vez que a Apple tem a interface de utilizador como a principal prioridade durante o seu desenvolvimento.

2.2.3 Windows Phone

A Microsoft também tem o seu sistema operativo móvel, o Windows Phone. Tal como o iOS, é de código proprietário e é o terceiro sistema operativo mais usado nos smartphones atuais.

O primeiro dispositivo com Windows Phone surgiu em 2010, com o Windows Phone 7 e ainda não conseguiu aumentar significativamente a sua quota de mercado, nunca tendo ultrapassado os 3% globais.

2.3 Desenvolvimento de Aplicações Móveis

A história das aplicações móveis é bastante interessante. Quando surgiram os primeiros smartphones, os seus *browsers* não eram muitos bons e apenas alguns sítios web funcionavam corretamente. Então, nessa altura, quando era preciso criar uma aplicação tinha que se desenvolver uma aplicação nativa para o sistema operativo do dispositivo.

Mas houve uma reviravolta com o aparecimento do primeiro iPhone. Agora os *browsers* são suficientemente bons para se criar uma versão móvel de um sítio web. Para além disso, a utilização de serviços web tem vindo a aumentar e surge então a oportunidade de se criarem aplicações que possam ser acedidas através de um *browser* normal. Posteriormente, serão analisadas algumas ferramentas que permitem o desenvolvimento de aplicações web.

Com o aumento do número de dispositivos e a grande variedade destes, tornou-se quase impossível criar uma aplicação (seja nativa ou web) que funcionasse corretamente em todos os dispositivos e respetivos sistemas operativos. Surge então a necessidade de se criarem aplicações com o auxílio de frameworks multiplataforma, ou seja, aplicações que, a partir do mesmo código fonte, funcionem nas várias plataformas. Com esta oportunidade, surgiram então várias frameworks de desenvolvimento multiplataforma, que permitem desenvolver aplicações para os vários sistemas operativos a partir de uma base comum de código e assim reduzir o custo e o tempo necessário para o desenvolvimento de uma aplicação. Existem bastantes frameworks disponíveis no mercado, onde as mais relevantes são analisadas mais à frente.

2.3.1 Desenvolvimento Nativo

O desenvolvimento nativo baseia-se, principalmente, no desenvolvimento de aplicações para determinado sistema operativo, usando ferramentas e bibliotecas fornecidas pelos próprios criadores do sistema.

As aplicações nativas têm algumas grandes vantagens em relação aos outros tipos de aplicações, como o acesso completo aos recursos do dispositivo e uma interface, interação e aspeto gráfico consistentes nas muitas aplicações existentes. Para além disso, as aplicações nativas podem aparecer nas lojas dos sistemas operativos correspondentes (Google Play, App Store,…). Mas existem alguns inconvenientes no desenvolvimento de aplicações nativas:

- Não se pode reutilizar código entre as várias plataformas;
- São necessários diferentes conhecimentos para cada plataforma;
- Geralmente são as de desenvolvimento mais oneroso, uma vez que é preciso contratar especialistas para cada plataforma.

2.3.2 Desenvolvimento *Web*

As aplicações móveis web são, essencialmente, as versões móveis de um sítio web mas com a aparência e comportamento de uma aplicação nativa. Assim, a principal diferença entre as aplicações nativas e aplicações web reside no facto de uma aplicação web correr num *browser* e não correr diretamente no dispositivo, havendo assim algumas limitações ao aceder às componentes de hardware ou software do dispositivo (e.g. GPS, acelerómetro, marcador de chamadas).

Tipicamente, o esforço e o tempo necessário para se desenvolver uma aplicação web é mais baixo, assim como o custo, pois apenas é necessário uma base de código comum. Para além disso, o desenvolvimento de aplicações web permite maior flexibilidade para se encontrarem e corrigirem erros que possam existir, assim como abranger um maior número de utilizadores possíveis (basta ter um *browser* instalado).

Em contrapartida, o desenvolvimento de aplicações web tem algumas desvantagens. Para além de não poder incorporar grande parte das componentes existentes num smartphone, não é totalmente universal (pode não correr em alguns dispositivos ou em certos *browsers* mais antigos) e não pode ser acedido de forma *offline* (sem ligação à internet). É sempre necessário uma ligação à internet e a velocidade da aplicação depende da velocidade dessa ligação assim como da eficiência do *browser*.

2.3.3 Desenvolvimento Multiplataforma

As metodologias de desenvolvimento multiplataforma vieram facilitar imenso o desenvolvimento de aplicações que sejam capazes de correr nas várias plataformas móveis. Normalmente, para se desenvolverem aplicações multiplataforma usam-se frameworks que permitam, através do mesmo código comum, compilar para as diferentes plataformas e assim abranger o maior número de utilizadores possíveis.

Desenvolver aplicações multiplataforma traz bastantes vantagens. Para além de reduzir os custos e o tempo de desenvolvimento, também permite abranger um maior número de utilizadores/dispositivos, sendo mais fácil fazer o marketing da aplicação uma vez que não é necessário criar mensagens para cativar um nicho ou um conjunto específico de pessoas. Outra das vantagens é haver uma aparência mais uniforme entre as várias plataformas (por vezes é complicado sincronizar duas equipas de desenvolvimento quando se estão a criar várias aplicações diferentes).

Mas o desenvolvimento multiplataforma também apresenta algumas desvantagens. Uma das maiores desvantagens é que não se pode agradar a todos os utilizadores. Uma boa aplicação multiplataforma é aquela que se assemelha com uma aplicação nativa, independentemente do sistema operativo em que está a correr e não aquela que é idêntica em todas as plataformas. E isso, por vezes, é difícil de alcançar porque, para além das várias plataformas terem aspetos da

interface gráfica diferentes, os seus utilizadores desejam uma interação e uma aparência a que já estão acostumados. Além disso, há também a perda de flexibilidade da aplicação que as várias plataformas oferecem, uma vez que ao desenvolver para os vários sistemas operativos procuram-se os pontos em comum entre estes. Apesar dessas desvantagens foram surgindo, ao londo dos anos, uma extensa variedade de ferramentas e *frameworks*.

2.4 Ferramentas e *Frameworks* de Desenvolvimento Multiplataforma

De forma a aumentar a velocidade de desenvolvimento de aplicações multiplataforma e facilitar as tarefas das equipas de desenvolvimento, que se podem tornar complexas, surgiram então as ferramentas de desenvolvimento multiplataforma acompanhadas de frameworks que incluem o modelo de desenvolvimento e as funcionalidades disponíveis.

2.4.1 RhoMobile

A RhoMobile Suite, baseada essencialmente na framework de código aberto Rhodes, contém um conjunto de ferramentas que permite criar aplicações nativas para a maioria dos smartphones usando tecnologias web como CSS3, HTML5, JavaScript e Ruby. As aplicações desenvolvidas através da RhoMobile Suite são aplicações puramente nativas e conseguem ter acesso a capacidades dos dispositivos como o GPS, câmara, *Bluetooth* e calendário. Os programadores podem controlar também como é que a aplicação se comporta nos diferentes dispositivos. As principais plataformas suportadas são o iPhone e iPad, Android, BlackBerry e Windows Mobile, tanto para smartphones como *tablets*.

A RhoMobile Suite consiste num conjunto de ferramentas para desenvolver, testar, depurar, integrar, implementar e gerir as aplicações para utilizadores ou para empresas. Dessas ferramentas destacam-se o Rhodes, o RhoElements, o RhoStudio e o RhoConnect. Para além disso, inclui um vasto conjunto de acesso às APIs dos dispositivos, um modelo incorporando MVC, entre outros serviços.

As principais vantagens de se usar o RhoMobile é o excelente suporte que a Motorola (proprietária da RhoMobile) e a equipa do RhoMobile oferecem. Sendo fácil de aprender a usar, o RhoMobile suporta grande parte dos recursos de *hardware* dos dispositivos e as ferramentas (RhoHub, RhoConnect, RhoGallery) trabalham muito bem e trazem bastante valor à aplicação.

Em contrapartida, desenvolver usando RhoMobile ainda apresenta alguns contras. De todos destacam-se o facto de o RhoStudio (IDE do RhoMobile) não ser de qualidade e apresentar muitas falhas comparativamente a outros IDEs, da documentação disponível estar algo desatualizada e de a aplicação resultante não se assemelhar a uma nativa. Além disso, o preço da licença empresarial é bastante alto.

2.4.2 PhoneGap

A framework PhoneGap é talvez a ferramenta de desenvolvimento de aplicações multiplataforma mais conhecida no mercado. O desenvolvimento de aplicações através do PhoneGap exige apenas o conhecimento de JavaScript, CSS3 e HTML5 e as suas aplicações consistem numa webview com 100% de altura e 100% de largura, com uma interface JavaScript que permite aceder aos recursos do sistema operativo. O resultado são aplicações híbridas, ou seja, nem são puramente nativas nem são puramente baseadas na web. O *software* subjacente ao PhoneGap e que lhe permite aceder aos recursos e APIs dos dispositivos tem o nome de Apache Cordova.

Atualmente, o PhoneGap suporta o desenvolvimento para uma grande variedade de plataformas como iOS, Android, BlackBerry, Windows Phone, LG webOS, Tizen, Bada, Firefox OS, Nokia Symbian OS e Ubuntu Touch. Para além disso, é possível aceder à maior parte dos recursos dos sistemas operativos e é possível também, através de módulos nativos customizados, aceder ainda a mais recursos do que, inicialmente, é oferecido no Apache Cordova. Ainda, o PhoneGap é também conhecido pela enorme comunidade de utilizadores que para além de partilharem os seus módulos, oferecem ajuda aos novos programadores e estão sempre em constante discussão sobre as características do PhoneGap e que melhorias lhe poderiam aplicar.

O PhoneGap tem como principais vantagens o facto de ser gratuito e de código aberto e de não necessitar de alta tecnologia, de ser fácil de aprender e de usar linguagens comuns (HTML5, CSS3 e JavaScript) das tecnologias da web. Além disso, permite utilizar o IDE que se pretender. Em contrapartida, o PhoneGap apresenta limitações no que toca à interface das aplicações e a integração de código nativo é complexa.

Um dos grandes problemas das aplicações resultantes do PhoneGap é a fraca performance que estas apresentam comparativamente às aplicações desenvolvidas nativamente. A Adobe, agora proprietária do PhoneGap, avisa mesmo que as aplicações podem mesmo ser rejeitadas pela Apple por serem demasiado lentas e não terem a aparência nativa necessária para estarem presentes na Apple AppStore.

Um dos IDEs mais utilizados é o Intel XDK que fornece as ferramentas necessárias para o desenvolvimento de aplicações Apache Cordova. Este IDE permite o acesso a várias APIs do Apache Cordova, assim como possui ferramentas de *design* e *debug*. Ainda tem a característica importante de conseguir criar aplicações aptas para integrarem as lojas de aplicações dos vários sistemas operativos.

Normalmente desenvolvem-se aplicações usando o PhoneGap quando os orçamentos para o desenvolvimento são baixos e as aplicações são pouco complexas ou quando se é principiante e se quer experimentar o desenvolvimento de aplicações móveis. O PhoneGap não é a ferramenta mais adequada quando as aplicações são maiores ou mais complicadas de desenvolver.

2.4.3 Xamarin

A framework Xamarin permite desenvolver aplicações nativas para várias plataformas a partir de código comum escrito em C#. Os programadores podem usar o Xamarin para criar aplicações nativas para iOS, Android e Windows com interfaces nativas mas com código partilhado entre as várias plataformas.

O Xamarin apresenta vários planos pagos que podem ser adquiridos para criar aplicações multiplataforma com algumas limitações (dependendo do plano adquirido). Para além desses planos, existe um outro, gratuito, que dá a possibilidade aos utilizadores de experimentarem a ferramenta ou então criarem as suas aplicações apesar de ser quase impossível desenvolver uma aplicação de qualidade, com o número de limitações que o plano gratuito apresenta.

Uma das grandes vantagens de usar Xamarin é a estabilidade e performance da aplicação resultante e de esta ser puramente nativa, pois o Xamarin compila o código para ficheiros app ou apk. Para além disso tem a vantagem de usar ferramentas nativas de *design* da interface.

Em contrapartida, o Xamarin pode ser bastante dispendioso (a licença mais usada ultrapassa os 800€ por ano, por programador). Também apresenta uma curva de aprendizagem, de elevado declive, associada ao uso do Xamarin. O programador/desenvolvedor deve ter conhecimento não só de C#/.Net mas também das plataformas móveis (iOS/Android) e do IDE do Xamarin. Outra das desvantagens é não suportar o uso de bibliotecas de código aberto populares no desenvolvimento iOS/Android. Visto que é um produto pago, o seu ecossistema ainda é relativamente pequeno e torna-se então difícil encontrar suporte para qualquer problema que surja.

Sumarizando, o Xamarin deve ser usado quando se quer uma aplicação o mais semelhante possível de uma nativa e quando se tem algum dinheiro para gastar.

2.4.4 Appcelerator Titanium

O Titanium é um produto de código aberto criado pela Appcelerator que permite o desenvolvimento de aplicações nativas para várias plataformas, a partir do mesmo código. Esta framework é baseada no desenvolvimento em JavaScript. O Titanium compila o seu código para aplicações nativas, tal como o Xamarin. Para além disso apresenta um grande número de acessos às APIs dos dispositivos o que a tornou bastante atrativa para os programadores nos últimos anos.

Neste momento, o Titanium oferece suporte para uma vasta gama de sistemas operativos, destacando-se o Android, o iOS, o Windows Phone e o BlackBerry, dependendo do sistema operativo onde está instalado o Titanium (devido às licenças da Apple só é possível criar aplicações para iOS num computador com OS X e para criar aplicações Windows Phone, só é possível num computador com o Windows instalado).

Uma das grandes vantagens do Titanium é o fácil acesso aos recursos nativos dos dispositivos. O Titanium suporta o acesso às ferramentas mais avançadas como a utilização direta do ecrã sensível, da câmara e GPS. Para além disso, é fácil aprender a usar a ferramenta e bastante

rápido a criar protótipos. Como principais vantagens, temos o facto de ser gratuita e de código aberto e das aplicações resultantes terem um alto desempenho (obviamente não é comparável com as aplicações desenvolvidas nativamente, mas as diferenças são poucas). A comunidade de utilizadores tem vindo a aumentar, havendo mesmo um Marketplace onde se pode comprar, vender ou partilhar módulos, elementos de *design* e extensões.

Em contrapartida temos a crescente complexidade de desenvolver com o Titanium. Quanto mais complexa a aplicação for, mais problemas técnicos irão surgindo. Além disso, apesar do desempenho das aplicações criadas com o Titanium ser bastante bom, por vezes ainda se pode observar alguns atrasos que poderão afetar a experiência do utilizador que se quer rápida, suave e confortável. Outra desvantagem do Titanium é o facto de o IDE (Titanium Studio) ainda apresentar alguns problemas. Este é uma versão alterada do Eclipse mas que apresenta alguns erros, é instável e chega a ser mesmo necessário reiniciar o Titanium Studio para se poder continuar a trabalhar.

Concluindo, o Titanium Studio deve ser usado quando se têm programadores web experientes e estes estão familiarizados com o Eclipse e quando se quer desenvolver uma aplicação mais complexa, com vários acessos às APIs dos dispositivos e com uma interface de alta qualidade semelhante à nativa.

2.4.5 Sencha Touch

Ao contrário das *frameworks* apresentadas anteriormente, o Sencha Touch tem como principal objetivo o desenvolvimento de aplicações web. Em vez das aplicações serem nativas, ou híbridas, e serem instalada nos dispositivos, são antes aplicações acedidas através do *browser* mas com uma aparência idêntica à nativa. O Sencha Touch suporta várias plataformas, entre elas o iOS, Android, Windows e o BlackBerry.

As aplicações são programadas em HTML5 com recurso a JavaScript, o que permite aos programadores criar as aplicações de forma rápida e fácil. Mas o Sencha Touch não oferece suporte aos principais recursos dos dispositivos diretamente, é necessário o Apache Cordova para que seja possível aceder a recursos como o acelerómetro, câmara e GPS por exemplo. Assim, é então possível compilar as aplicações originalmente web para nativas através de *software* externo.

Como principais vantagens do Sencha Touch é de destacar a arquitetura MVC, uma API extensível e uma vasta gama de temas para a interface. Para além disso, a empresa Sencha também produziu uma série de ferramentas interoperáveis (Sencha Architect ou Sencha Touch Charts) ou até mesmo uma extensão para integração com o IDE Eclipse, de forma a melhorar o processo de desenvolvimento de aplicações. Em contrapartida, as aplicações criadas com o Sencha Touch podem apresentar um nível de desempenho bastante baixo, comparável ao do PhoneGap e para se estender as aplicações a aplicações nativas e com mais acessos aos recursos do dispositivo, é necessário o uso do Apache Cordova e talvez a criação de extensões em linguagem nativa. Como uma das principais desvantagens está também o elevado custo para o uso da framework.

2.4.6 Qt

Qt é uma ferramenta para o desenvolvimento de aplicações multiplataforma, quer seja desktop ou móvel. As aplicações são desenvolvidas em QML (linguagem baseada em JavaScript e CSS) e os programadores têm também acesso a um extenso conjunto de bibliotecas escritas em C++, tal como os componentes gráficos também são escritos em C++.

Como principais vantagens destacam-se a facilidade de utilização da ferramenta, pois o programador tem acesso a APIs intuitivas, a uma documentação extensa e a exemplos de aplicações. Também oferece várias ferramentas integradas no seu IDE (Qt Creator IDE) para o design da aplicação e depuração por exemplo. Além disso a prototipagem é feita de uma forma rápida e simples.

Uma das grandes desvantagens de se usar Qt é o custo associado, que pode ultrapassar os 100€ mensais, e o facto de a abordagem de desenvolvimento se tornar um pouco restritiva. Ou seja, uma vez que o Qt oferece uma extensa cadeia de ferramentas e como se programa apenas em QML (linguagem específica do Qt), os programadores estão obrigados a aprender e usar QML e a usar todas as suas ferramentas, pois não há *software* externo que suporte esta linguagem. Atualmente as empresas tentam ao máximo reutilizar as habilidades dos seus funcionários e não as fragmentar ainda mais, o que poderia ser provocado pelo uso de Qt.

2.4.7 Ionic

A Ionic é uma das frameworks de código aberto de criação de aplicações móveis híbridas baseadas em HTML5 mais promissoras. Uma vez que foi construída com SASS, fornece muitos componentes para a criação da interface de forma a ajudar no desenvolvimento de aplicações ricas e interativas. Além disso, usa uma framework JavaScript MVVM e AngularJS como o "motor" da aplicação.

O AngularJS, com as duas vias de ligação de dados, interação com serviços back-end e APIs torna-se assim uma escolha comum para o desenvolvimento móvel. Com a última atualização do AngularJS para a versão 2.0, focada no desenvolvimento móvel, esta framework poderá vir a ganhar ainda mais popularidade. A Ionic também usa o Apache Cordova para aceder aos recursos dos dispositivos.

As principais vantagens da Ionic incluem o facto de ser de código aberto, de oferecer uma biblioteca HTML5, CSS e JavaScript otimizada para desenvolvimento móvel, a integração de ferramentas que permitem a criação de aplicações bastante interativas e de ser otimizada para AngularJS (que torna as aplicações mais robustas).

Como desvantagem, tal como o PhoneGap, está o desempenho das aplicações criadas. Apesar de os criadores da Ionic afirmarem que a ferramenta foi desenvolvida de forma a correr perfeitamente nos dispositivos mais recentes e de usarem técnicas adicionais para melhorarem a performance, está ainda está longe do desempenho das aplicações nativas. Para além disso, como

é uma framework relativamente recente e ainda em crescimento, não se encontra muita documentação nem uma comunidade extensa de utilizadores da Ionic.

2.4.8 Comparação entre as várias ferramentas de desenvolvimento

Existe um vasto número de ferramentas para o desenvolvimento de aplicações multiplataforma, mas foram analisadas apenas as mais relevantes para o desenvolvimento da dissertação. Além disso, existe um grande número de frameworks para a criação de aplicações web, mas só uma foi apresentada na análise (Sencha Touch) por ser uma das mais usadas no mundo empresarial. No entanto, como o foco da dissertação não é o desenvolvimento de uma aplicação web, as restantes frameworks de desenvolvimento de aplicações web não foram consideradas.

Para selecionar uma framework de desenvolvimento de aplicações multiplataforma consideraram-se sete fatores principais. Foram então:

- Suporte às plataformas desejadas;
- Velocidade de desenvolvimento;
- Acesso às APIs nativas;
- Aparência idêntica à nativa;
- Desempenho da aplicação;
- Curva de aprendizagem;
- Custos associados.

Para além disso, a framework deve suportar os requisitos necessários para o desenvolvimento da base e aplicação de prova de conceito que se pretendia implementarr. De entre esses requisitos está o acesso da aplicação à localização geográfica do dispositivo, o suporte a Sockets TCP e a criação de bases de dados SQLite, por exemplo.

Tendo todos esses fatores em consideração, na Tabela 1 é apresentada uma comparação entre as várias frameworks.

	RhoMobile	PhoneGap	Xamarin	Titanium	Sencha	Qt	Ionic
Plataformas suportadas	Bom	Bom	Bom	Bom	Bom	Bom	Bom
Velocidade de desenvolvimento	Médio	Bom	Bom	Bom	Médio	Mau	Bom
Acesso às APIs nativas	Médio	Médio	Bom	Bom	Mau	Mau	Médio
Desempenho da aplicação	Médio	Mau	Bom	Bom	Médio	Médio	Médio
Curva de aprendizagem	Médio	Bom	Médio	Médio	Médio	Mau	Bom
Custos Associados	Bom	Bom	Mau	Bom	Mau	Mau	Bom
Requisitos adicionais	Médio	Médio	Médio	Bom	Mau	Mau	Médio

Tabela 1: Comparação entre as várias frameworks

2.5 Aplicações com requisitos de informação geográfica

O principal requisito das aplicações móveis que trabalham com sistemas de informação geográfica é, para além da visualização de mapas que todos os dispositivos suportam, o acesso à sua localização através da leitura de coordenadas geográficas provenientes do sistema GPS. O sistema de navegação GPS (Global Positioning Service) é uma tecnologia de posicionamento global que, através da triangulação baseada em satélites, fornece a posição exata a qualquer aparelho recetor móvel.

O GPS ganhou popularidade ao aparecer pela primeira vez incorporado num smartphone. Desta forma, permite ao utilizador saber a posição exata onde se encontra e até mesmo calcular rotas e percursos para locais específicos, usando o acesso à internet ou *software* instalado no dispositivo.

O aparecimento deste serviço nos smartphones trouxe uma grande vantagem, de entre muitas: a sua integração com as aplicações. Dessa forma, torna-se então possível calcular ou partilhar a localização atual do utilizador a partir de qualquer aplicação que tenha acesso ao recetor GPS do telemóvel. Existem várias aplicações distintas que usam o GPS do dispositivo, mas todas elas têm algumas características em comum e certos requisitos que devem cumprir.

Uma aplicação que permita a georreferenciação calcula constantemente e em tempo real a posição exata do dispositivo e, por norma, oferece também uma visualização dessa posição num mapa. Para além disso, muitas aplicações possibilitam o cálculo de rotas/percursos desde a posição do utilizador até ao destino pretendido.

As aplicações que pretendam utilizar o GPS do dispositivo devem cumprir certos requisitos. O principal e mais importante requisito é o acesso ao GPS do smartphone. Outro dos principais requisitos é a utilização do método contínuo, ou seja, a aplicação deve estar sempre a calcular e

atualizar a posição do utilizador em tempo real, assim como deve minimizar ao máximo os problemas de exatidão dos valores das coordenas geográficas. A aplicação deve então apresentar sempre valores confiáveis, quer seja através da localização GPS, quer seja através das redes disponíveis (quando o dispositivo não recebe sinais GPS). Para além disso, deve também garantir a segurança e integridade da informação.

Durante o desenvolvimento de uma aplicação que permita a georreferenciação deve-se ter em consideração todos os requisitos e implementar as funcionalidades necessárias de forma a oferecer a melhor experiência de utilização possível.

2.6 Conclusões

As frameworks de desenvolvimento de aplicações multiplataforma estão em constante desenvolvimento e cada vez se aproximam mais das aplicações nativas. Mas, obviamente, ainda estão longe de atingir os níveis de desempenho e a interface idêntica à que as nativas apresentam. Por isso, na hora de escolher qual a melhor framework para se usar, é necessário ponderar todos os prós e contras de cada uma delas. Nenhuma delas é perfeita e a escolha da framework também dependerá do contexto e do propósito da aplicação.

Assim, depois de analisar e avaliar cada uma das frameworks e tendo em consideração as características que a aplicação desenvolvida durante a dissertação deve ter, a framework Appcelerator Titanium apresenta a maior adequação para os desenvolvimentos a efetuar na dissertação. A Xamarin também era adequada mas, uma vez que um dos requisitos para o desenvolvimento da dissertação é minimizar ao máximo os custos e a versão gratuita do Xamarin apresenta grandes limitações, então esta não foi a escolhida.

Capítulo 3

Base comum para desenvolvimento móvel multiplataforma

Os objetivos principais desta dissertação consistiam na escolha de uma *framework* de desenvolvimento de aplicações multiplataforma, principalmente iOS e Android, que reduzisse custos assim como facilitasse e agilizasse todo o processo de desenvolvimento.Após escolhida a *framework* mais adequada que cumprisse todos os requisitos necessários para o desenvolvimento da dissertação, a fase seguinte seria então desenvolver uma base tecnológica. A base tecnológica a desenvolver terá como principal objetivo a criação de aplicações móveis para Android e iOS da forma mais rápida e fácil possível. Tendo isso em consideração, juntamente com as necessidades da Gisgeo, há alguns requisitos e funcionalidades que a base tecnológica deve incluir.

3.1 Objetivos

A base tecnológica será uma tentativa de facilitar o desenvolvimento de aplicações móveis, tanto para Android como para iOS, e de reduzir o tempo gasto ao longo de todo o processo de desenvolvimento. Além disso, pretende também oferecer, no resultado final, uma aplicação com aspeto e características idênticas às aplicações nativas e oferecer a melhor experiência de utilização e o melhor desempenho possível.

Através da base tecnológica será possível aos programadores criar aplicações multiplataforma rapidamente e sem preocupações com as especificidades de cada plataforma. O único requisito para o desenvolvimento de aplicações através da base tecnológica é conhecer e ter alguma experiência em JavaScript, uma vez que todo o desenvolvimento é efetuado nesta linguagem. O Titanium permite também integrar módulos nativos e, para isso, é necessário ter conhecimento da liguagem nativa correspondente à plataforma pretendida.

3.2 Especificação de Requisitos

Uma vez que a base tecnológica será usada pela Gisgeo para desenvolver as suas aplicações móveis para Android e iOS, esta deve cumprir alguns requisitos para que a aplicação final resultante corresponda ao esperado e que garanta todas as necessidades da Gisgeo.

As futuras aplicações desenvolvidas através da base tecnológica devem ser de um alto nível de qualidade e oferecem uma experiência de utilização semelhante às aplicações nativas. Para isso, existem alguns requisitos a ter em consideração durante o processo de desenvolvimento da base tecnológica.

3.2.1 Requisitos Funcionais

3.2.1.1 Casos de Uso

A base tecnológica, de forma a garantir o máximo de qualidade e de adequação às necessidades da Gisgeo, deve ter em atenção alguns casos de uso.

Caso de Uso	Descrição
Acesso ao *hardware* do telemóvel	Aceder aos vários elementos físicos do telemóvel (acelerómetro por exemplo)
Comunicação com o servidor	Efetuar uma ligação a um servidor (por *socket* ou *webservice*) para intercâmbio de informação
Ligação a aplicações externas	Abrir aplicações externas como a aplicação de SMS, o telefone ou o *browser*
Configurações de sistema	Aceder às configurações do sistema como a linguagem ou o fuso horário
Armazenamento numa base de dados	Possibilidade de armazenar dados numa base de dados local
Autenticação/Identificação	Adicionar funcionalidade de autenticação ou identificação de utilizador a qualquer aplicação (*login*)
Suporte a Módulos	As várias funcionalidades devem ser desenvolvidas separadamente, sem dependências entre si, e deve ser possível adicionar facilmente módulos a uma aplicação

Tabela 2: Casos de uso da base tecnológica

3.2.1.2 *User Stories*

De forma a mostrar e especificar o pretendido para a base tecnológica são apresentadas de seguida algumas *user stories*. Com estas, é possível mostrar as maiores necessidades da Gisgeo e as principais funcionalidades que devem estar presentes na base tecnológica.

Descrição	Captação de sinal GPS através da antena integrada no telemóvel
Prioridade	Alta
Pré-condições	O telemóvel deve ter GPS e este tem que estar ativo.
Descrição	Acesso aos sensores do telemóvel (acelerómetro por exemplo) e efetuar cálculos para a deteção de quedas ou orientação do dispositivo.
Prioridade	Baixa
Pré-condições	O telemóvel deve possuir acelerómetro integrado.
Descrição	Comunicação com o servidor através de *sockets* ou *webservices*.
Prioridade	Alta
Pré-condições	O telemóvel deve ter acesso a uma ligação de dados (*WiFi*, GSM,…).
Descrição	Armazenamento de dados numa base de dados local (SQLite)
Prioridade	Alta
Pré-condições	-
Descrição	Ligação a aplicações base como chamada telefónica, envio de *SMS* e *browser*
Prioridade	Alta
Pré-condições	O telemóvel deve ter a aplicação correspondente instalada.
Descrição	Ligação a aplicação externa/integrada para leitura de *QRCodes/Barcodes*
Prioridade	Baixa
Pré-condições	O telemóvel deve possuir câmara fotográfica.
Descrição	Ligação à câmara do telemóvel
Prioridade	Baixa
Pré-condições	O telemóvel deve possuir câmara fotográfica.
Descrição	Deve ser possível alterar a linguagem da aplicação
Prioridade	Alta
Pré-condições	-
Descrição	A base tecnológica deve suportar vários módulos
Prioridade	Alta
Pré-condições	-

Descrição	A base tecnológica deve garantir a comunicação entre os diferentes módulos
Prioridade	Alta
Pré-condições	-
Descrição	Utilização da API de cartografia *mobile* da Here Maps
Prioridade	Alta
Pré-condições	-
Descrição	Utilização da câmara fotográfica para realidade aumentada com referenciação geográfica.
Prioridade	Alta
Pré-condições	O telemóvel deve ter GPS e câmara fotográfica.

Tabela 3: *User Stories* da base tecnológica

3.2.2 Requisitos Não-Funcionais

Para além dos requisitos apresentados anteriormente que definem as funções e componentes que a base tecnológica deve conter, existem alguns requisitos que não podem ser esquecidos e são também de extrema importância. De entre eles destacam-se:

- **Segurança:** as aplicações desenvolvidas através da base tecnológica devem garantir a segurança e confidencialidade dos dados;
- **Fiabilidade**
- **Robustez**
- **Estabilidade**
- **Usabilidade:** as aplicações resultantes devem ser fáceis de utilizar e devem permitir ao utilizador realizar uma tarefa de forma rápida e eficaz, sem complicações;
- ***Performance*/Tempo de resposta:** visto que foi usada uma *framework* de desenvolvimento multiplataforma, a *performance* das aplicações resultantes não atinge o nível das aplicações nativas, mas deve aproximar-se o máximo possível desse nível;
- **Compatibilidade com Android e iOS:** as aplicações resultantes devem ser compatíveis com as duas plataformas móveis.

3.3 Arquitetura

O projeto da base tecnológica é divido em duas partes principais: a aplicação, onde podem ser adicionados módulos e onde toda a interação entre estes é efetuada e os módulos, que são independentes (exceptuando casos raros) e trabalham autonomamente o máximo possível. Na hora da compilação para as plataformas móveis, a arquitetura é diferente pois é necessário incluir todos os elementos e bibliotecas do Titanium Studio.

3.3.1 Base tecnológica

A arquitetura da base tecnológica é bastante simples, sendo o modelo semelhante ao apresentado para a maioria das aplicações modulares. No topo encontra-se a base tecnológica ou camada padronizada, onde são efetuadas todas as interações entre módulos, assim como são criados e mostrados os elementos gráficos. Os módulos são elementos simples e independentes e que podem ser incluídos ou requisitados em qualquer momento do processo de desenvolvimento de uma aplicação. O modelo da arquitetura da base tecnológica é então o mostrado na Figura 2.

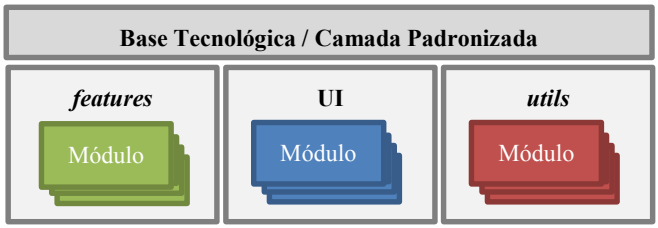


Figura 2: Arquitetura da base tecnológica

3.3.2 Aplicação compilada

Após todo o processo de desenvolvimento e na hora de compilação do código, as bibliotecas do Titanium Studio são compiladas juntamente com a base tecnológica criada, bem como os seus módulos. A aplicação resultante é caracterizada por uma arquitetura por camadas, onde no topo está a aplicação resultante, logo seguida da camada padronizada e os seus módulos. Logo abaixo encontram-se as bibliotecas do Titanium que acedem aos recursos tanto do Android como do iOS.

No fundo encontram-se os serviços externos que são acedidos via rede. Na Figura 3 pode ver-se o modelo da arquitetura da aplicação resultante.

Figura 3: Arquitetura da aplicação

Capítulo 4

Processo de Desenvolvimento

Um dos principais objetivos desta dissertação consistia em escolher uma *framework* de desenvolvimento de aplicações multiplataforma que permitisse a criação de aplicações para vários sistemas operativos, que fosse *open-source* e que reduzisse o tempo, dificuldade e custos ao longo de todo o processo de desenvolvimento.

Após uma cuidada análise das *frameworks* mais relevantes no mercado, chegou-se à conclusão que a que mais se adequa a este caso é a Appcelerator Titanium.

As tecnologias de desenvolvimento utilizadas ao longo de todo o processo resumem-se ao Titanium Studio, uma vez que é uma ferramenta bastante completa. De uma forma mais específica, apresenta-se a ferramenta escolhida assim como as metodologias e processos utilizados ao longo de todo o processo de desenvolvimento.

4.1 Appcelerator Titanium

4.1.1 Introdução

A *framework* Titanium, criada pela empresa Appcelerator Inc., surgiu em dezembro de 2008 oferecendo um vasto número de ferramentas para o desenvolvimento de aplicações multiplataforma. A sua rica documentação e os recursos oferecidos *online* também impulsionaram a Titanium na direção do sucesso. Inicialmene a *framework* era exclusivamente destinada ao desenvolvimento de aplicações *desktop* multiplataforma. O suporte a aplicações móveis apenas foi adicionado alguns anos mais tarde.

A abordagem da Titanium para a criação de aplicações multiplaforma consiste na técnica de *cross-compilation*, ou seja, a API JavaScript é independente da plataforma e através do mesmo código, compila para as várias plataformas suportadas. A Titanium utiliza JavaScript como linguagem principal de programação mas, ao contrário das *frameworks* HTML5 que incluem uma

vista HTML numa janela nativa, a Titanium converte todas as suas abstrações JavaScript para elementos puramente nativos. Dessa forma garante maior fluidez e uma melhor experiência de utilização comparativamente às outras *frameworks* de desenvolvimento de aplicações multiplataforma.

4.1.2 História

Em dezembro de 2008 surgiu a *framework* Titanium com o objetivo inicial do desenvolvimento de aplicações *desktop* multiplataforma. Só em meados de 2009 é que foi adicionado o suporte a plataformas móveis (iOS e Android).

No ano de 2010 a *framework* sofreu várias atualizações, de onde se destaca o suporte a iPad's, sendo também anunciado o suporte a BlackBerry. Mas só três anos depois é que esse suporte passou para uma versão Beta e assim tem continuado até hoje.

No ano seguinte, em 2011, a Appcelerator adquiriu a Aptana e a partir do Aptana Studio criaram o seu próprio IDE, com o nome de Titanium Studio.

Mais tarde, no ano de 2012, o desenvolvimento de aplicações *desktop* separou-se do desenvolvimento de aplicações móveis dando então origem ao TideSDK.

No final de 2014 a Appcelerator anunciou o Appcelerator Studio e sugeriu a migração do Titanium Studio para a nova plataforma. Mas, contrariamente ao Titanium Studio, para se desenvolver através do Appcelerator Studio é necessário adquirir uma licença (paga). Porém, a Appcelerator ofereceu, a quem utilizava o Titanium Studio, uma licença Indie vitalícia. Originalmente, esta licença teria o custo de trinta e nove dólares mensais e permite desenvolver aplicações para Android, iOS e Windows Phone utilizando a nova ferramente Appcelerator Studio. Para além disso, é possível aceder a recursos adicionais da Appcelerator, como os serviços da *cloud* e de *analytics* (análise de dados e estatísticas de utilização, por exemplo).

Para além disso, a ferramenta Titanium Studio continuará disponível para transferência dos repositórios da Appcelerator, sem qualquer custo e mantendo-se *open-source.*

4.1.3 Arquitetura

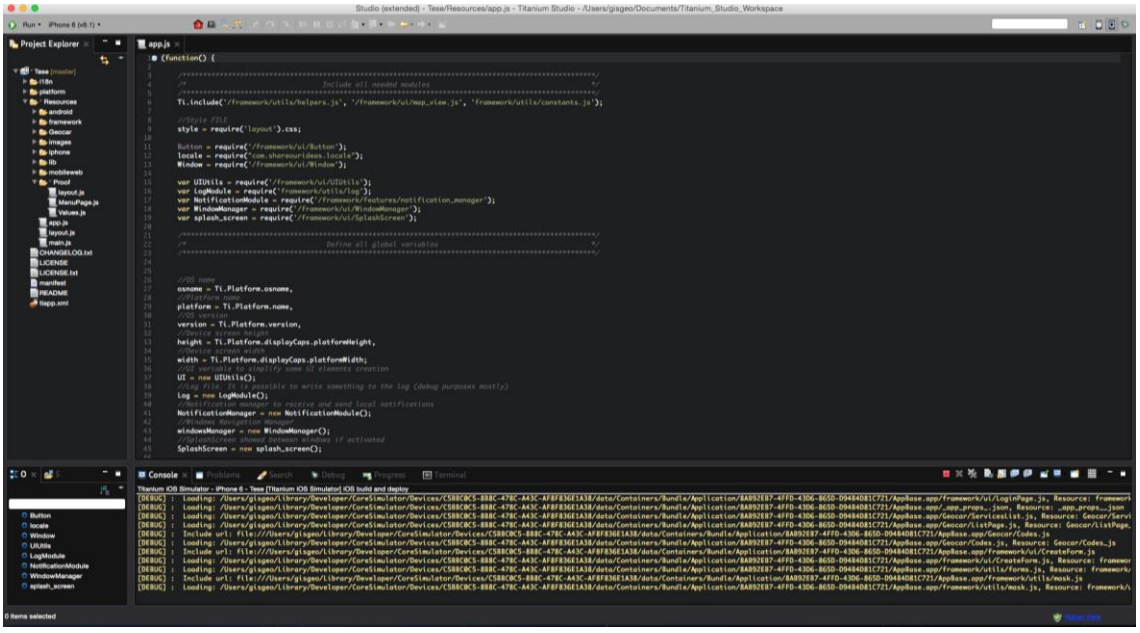
A arquitetura da Titanium é dividida em duas partes principais. A primeira consiste no agrupamento da aplicação com um interpretador autónomo de JavaScript para que o código da aplicação seja executado. A segunda é uma biblioteca Titanium que fornece o acesso a funcionalidades dos dispositivos como sensores, acesso a ficheiros e à interface nativa, entre outros. Esta biblioteca é acessível em qualquer parte do projeto facilitando bastante o processo de desenvolvimento.

4.1.4 Titanium Studio

A ferramenta Titanium Studio é o IDE desenvolvido pela Appcelerator, baseado no Aptana, que permite a criação de aplicações multiplataforma. Com um aspeto semelhante ao Eclipse e outros IDEs igualmente conhecidos, o Titanium Studio pode ser dividido em 5 secções principais:

- **Barra de ferramentas:** onde é possível executar a aplicação e escolher a plataforma de destino;
- **Explorador de projetos**: responsável por mostrar todos os projetos existentes na área de trabalho assim como os seus ficheiros;
- **Editor de código:** local onde se edita os ficheiros de código;
- ***Outline View*:** nesta janela é possível ver uma representação em árvore dos recursos que estão a ser editados;
- **Ferramentas úteis:** neste conjunto de separadores encontram-se várias ferramentas úteis como a consola, a lista de problemas encontrados e depuração, por exemplo.

Na Figura 4 é apresentada a interface do Titanium Studio que, para além do tema escuro, também permite alterar para o tema claro. É também possível alterar para a perspetiva de depuração, de forma a facilitar durante o processo.



4.2 Implementação

A base tecnológica, especificada no capítulo anterior, foi desenvolvida em JavaScript e consiste num conjunto de módulos, desenvolvidos separadamente, que podem (ou não) interligar-se entre si.

De forma a garantir uma maior facilidade e rapidez no processo de criação de aplicações móveis, a base tecnológica foi desenvolvida para que fosse de alto-nível e possibilitasse, através de poucas linhas de código, a criação (e representação) de elementos gráficos complexos, assim como o fácil acesso aos elementos dos dispositivos móveis.

A fácil reutilização de elementos comuns entre várias aplicações foi outro dos objetivos em mente durante todo o processo de desenvolvimento da base tecnológica.

Todo o processo de desenvolvimento baseou-se na metodologia de desenvolvimento ágil de *software* com *sprints* de duas semanas e o código produzido ao longo de todo o processo foi sempre mantido num repositório *git* para controlo de versões.

4.2.1 Implementação dos módulos

Os módulos desenvolvidos, assim como a base tecnológica, foram escritos em JavaScript. Mas nem todos os módulos foram criados da mesma forma. Existem dois tipos de módulos que variam, dependendo da necessidade e do uso que se vai dar a cada um.

Para o desenvolvimento dos módulos, existiam dois métodos possíveis: o método tradicional do JavaScript ou através da especificação CommonJS. A diferença entre os dois métodos reside na forma como estes são incluídos e no tipo de variáveis a que têm acesso. Se for usado o primeiro método é necessário utilizar a função do Titanium "*Ti.include*". Este tipo de especificação permite ter acesso a variáveis e funções globais e até mesmo fazer alterações nas mesmas.

Resumidamente utilizar o "*Ti.include*" é o mesmo que fazer uma cópia de todo o código que se encontra nesse ficheiro incluído. Isto por vezes é prático, mas em contrapartida traz problemas de *performance* e pode tornar a aplicação muito mais lenta e consequentemente piorar a experiência de utilização.

Já no segundo caso, é necessário utilizar a função *"*require". Esta especificação é mais rápida e mais eficaz e ao contrário da função "*Ti.include*" o código não é copiado para o novo ficheiro, mas é exportado através de um objeto.

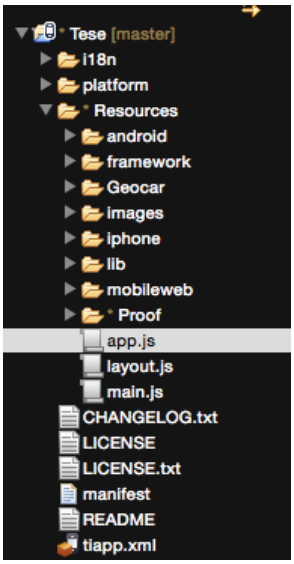
Ambos os métodos trazem vantagens e desvantagens, mas para a implementação dos módulos apenas foi utilizada a especificação CommonJS.

4.2.2 Detalhes da Implementação

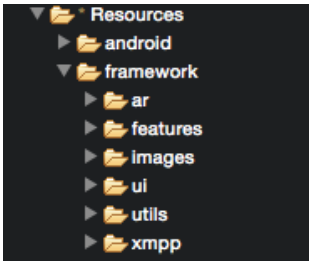
Para se desenvolver um projeto usando o Titanium Studio, todo o código necessário para a criação de uma aplicação deve estar dentro de uma pasta "*Resource*". Dessa forma, sempre que é necessário criar uma aplicação utilizando a base tecnológica tem que se manter todo o código necessário dentro dessa pasta.

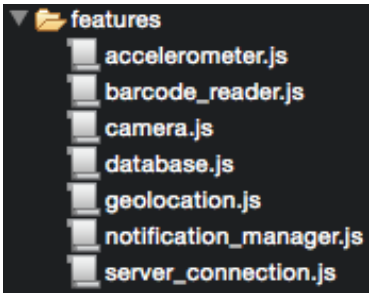
A base tecnológica, também ela obrigada a estar inserida na pasta "*Resources*", encontra-se sempre disponível para qualquer alteração ou melhoria na pasta *"framework"*.

Na hora de compilar todo o código, o primeiro ficheiro a ser aberto e processado é o ficheiro "app.js". É obrigatório ter este ficheiro sempre presente na raiz da pasta "*Resource*". É nele que são incluídos a maioria dos módulos fundamentais para a criação das aplicações móveis. Juntamente com ele, encontra-se um ficheiro "*main.js*" onde o programador deve inserir o seu código e incluir os módulos necessários ou outros ficheiros criados. Além destes, é possível encontrar também, na raiz da pasta "*Resources*" o ficheiro "*layout.js*" onde é possível alterar o aspeto da aplicação ao estilo CSS. Na Figura 5 pode ver-se a organização de pastas da base tecnológica.



Para se manter a organização no projeto, recomenda-se a criação de uma pasta onde serão inseridos todos os ficheiros criados pelo programador, evitando assim confusões e ambiguidades entre os ficheiros criados pelo utilizador e aqueles pertencentes à base tecnológica. Na Figura 7 percebe-se a forma como os módulos estão organizados por pastas. Dentro de cada uma das categorias (pastas) é possível verificar os ficheiros que definem cada módulo (ver Figura 6).





4.2.2.1 Ficheiro *tiapp.xml*

Dentro da pasta do projeto, mas fora da pasta "*Resources"* encontra-se um ficheiro chamado "*tiapp.xml"* que funciona como o ficheiro de configuração das aplicações desenvolvidas. Neste ficheiro é possível alterar o *id* da aplicação, a versão da aplicação, o ícone, a descrição e também escolher as plataformas de destino da aplicação. Para além disso, é neste ficheiro que se podem incluir módulos desenvolvidos nativamente, adicionar permissões para cada plataforma escolhida, definir as orientações permitidas, entre outras configurações.

4.2.2.2 Pasta i18n

A maioria das aplicações suporta uma variedade de idiomas e as aplicações desenvolvidas com a base tecnológica e o Titanium Studio também possuem essa característica.

Dentro desta pasta, encontram-se várias pastas com o código de cada idioma (pt, en, it,…) que contêm um ficheiro XML com a definição e tradução das várias palavras e frases usadas para a criação da aplicação. Automaticamente a aplicação deteta qual o idioma em que o dispositivo móvel está e irá definir o mais adequado.

4.3 Suporte ao protocolo XMPP

O protocolo aberto XMPP (Extensible Messaging and Presence Protocol), baseado em XML, foi desenvolvido originalmente para a troca de mensagens instantâneas e informação de presença e, inicialmente, tinha o nome de *jabber*.

O XMPP permite a troca de pequenos pacotes de dados estruturados (mas extensíveis), também chamados de "XML *stanzas*", através de uma conexão entre duas ou mais entidades. Normalmente, o XMPP é desenvolvido numa arquitetura cliente-servidor. Dessa forma, o cliente necessita de se ligar ao servidor (e autenticar, preferencialmente) para poderem trocar XML *stanzas*.

4.3.1 História

No ano de 1998, Jeremie Miller iniciou os trabalhos na tecnologia Jabber e, no ano seguinte, esta é anunciada como uma tecnologia de código aberto que permitia a troca de mensagens instantâneas e de informações de presença. Foi neste período de tempo que se desenvolveu também a sintaxe básica assim como a semântica do protocolo XMPP. Os mesmos protocolos são utilizados desde então apesar de terem sofrido algumas alterações e melhorias ao longo dos anos.

Em maio do ano 2000 surge o primeiro servidor Jabber juntamente com os principais protocolos (XML *streaming*, troca de mensagens, informações de presença, lista de contactos, entre outros).

A Jabber Software Foundation (JSF) foi criada em agosto de 2001 com o objetivo de coordenar o crescente número de projetos relacionados com a tecnologia Jabber.

Mais tarde, no ano de 2002, o IETF formou um grupo de especialistas na tecnologia Jabber com o objetivo de formalizar os seus protocolos básicos, aos quais deram o nome de XMPP.

O *software* baseado em XMPP começou a espalhar-se mundialmente através da internet e, segundo a XMPP Standards Foundation, atingiu os dez milhões de utilizadores em todo o mundo.

O ano de 2005 ficou marcado pelo lançamento de vários serviços em larga escala baseados em XMPP de onde se destacou o Google Talk.

Dois anos depois, a JSF altera o seu nome para XMPP Standards Foundation de forma a descrever melhor o seu foco em desenvolver extensões de código aberto para as especificações base de XMPP criadas pela IETF.

Nos últimos anos, empresas como o Facebook e a Microsoft têm desenvolvido aplicações baseadas em XMPP.

4.3.2 Pontos fortes

O protocolo XMPP é caracterizado principalmente por cinco pontos fortes.

Um dos maiores pontos fortes do XMPP é o facto de ser de código aberto, o que permite a sua utilização e alteração sem qualquer custo.

O segundo ponto forte do XMPP é a sua descentralização. A arquitetura de uma rede XMPP é similar à de uma rede de *email*, permitindo assim que qualquer utilizador possar ter o seu próprio servidor XMPP sem ser necessário um servidor central.

O facto de ter tantos anos de história (surgiu em 1999) é também uma grande vantagem, visto que é possível encontrar várias implementações do XMPP e assim escolher ou adaptar aquela que for mais conveniente.

Para além disso, o XMPP apresenta outro ponto forte, a sua flexibilidade. É possível acrescentar funcionalidades personalizadas ao XMPP. As extensões mais comuns são geridas pela XSF de forma a manter a interoperabilidade entre todos os elementos.

Por último, o XMPP apresenta um elevado nível de segurança uma vez que os servidores XMPP podem ser isolados da rede pública de XMPP e, para além disso, foram também desenvolvidos mecanismos de segurança (SASL e TLS por exemplo) de forma a aumentar a segurança na troca de mensagens.

4.3.3 Pontos fracos

Em contrapartida, o protocolo XMPP apresenta alguns pontos negativos, de onde se destacam quatro.

O primeiro está relacionado com as comunicações, que são baseadas em texto. Uma vez que o XML é baseado em texto, isto provocará uma maior sobrecarga da rede comparando com outro tipo de soluções (binárias por exemplo).

Além disso, o XMPP não suporta QoS. Isto relaciona-se com a probabilidade de sucesso em estabelecer uma ligação a um destino ou a entregar uma mensagem. No caso do XMPP, para garantir a entrega de mensagens, deve-se desenvolver sobre a camada de XMPP um mecanismo ou extensão que garanta essa entrega. Normalmente a solução consiste em atribuir números de identificação a cada mensagem enviada.

Outro dos pontos fracos do XMPP relaciona-se com o facto de a transferência de dados binários ser um pouco limitada. No protocolo XMPP todos os dados binários devem ser convertidos para base64 mas em casos como a transferência de ficheiros, em que a quantidade de dados é elevada, seria melhor enviar todos os dados *out-of-band* (numa sequência independente da sequência principal de comunicação), mas o XMPP não o permite. Por último, o XMPP ainda não suporta a cifragem *end-to-end* (proteção ininterrupta dos dados transferidos entre dois pontos que comunicam entre si sem serem interceptados ou lidos por terceiros). Recentemente, tem-se discutido a possibilidade de adicionar esta funcionalidade ao XMPP.

Comunicação entre cliente e servidor

Tipicamente, o processo de comunicação cliente-servidor baseia-se nos seguintes pontos:

- Determinação do endereço de IP e da porta a qual se deve ligar (tipicamente, esse endereço de IP é obtido através da resolução de um domínio);
- Criação e abertura de uma ligação TCP;
- Criação de uma *stream* XML nessa ligação TCP;
- Negociação TLS (Transport Layer Security) para a cifragem das comunicações (preferencialmente);
- Autenticação usando um mecanismo SASL;
- O cliente deve fornecer o recurso ao qual pretende aceder para que o servidor direcione o cliente (opcional);
- Realizar a troca de mensagens (XML *stanzas*);
- Fechar a sequência de dados XML e a ligação TCP.

4.3.4 Ligação TCP

Na comunicação entre o cliente e o servidor é definido que se deve criar e abrir uma ligação TCP entre ambos antes de qualquer troca de *streams* XML.

O TCP é um protocolo fiável, orientado à ligação, usado principalmente para a ligação *end--to-end*. Este protocolo encontra-se na camada 4 do Modelo OSI (camada de transporte – ver Figura 8) e é nele que assentam as aplicações fundamentais para o correto funcionamento da *web* (HTTP, FTP e SSH por exemplo).

O facto do protocolo TCP ser tão robusto e versátil tornou-o muito popular e útil para as redes globais uma vez que este protocolo verifica se os dados são enviados pela rede da forma e sequência correta, sem a existência de erros.

4.3.4.1 Características principais

Existem várias características que definem o protocolo TCP. A primeira tem a ver com o facto de ser orientado à ligação, ou seja, durante a comunicação é enviado um pedido de ligação para o destino e essa ligação é usada para a transferência de dados. Para além disso é também uma ligação ponto-a-ponto, ou seja, a conexão TCP é estabelecida entre dois pontos.

Outra das características do protocolo TCP é a sua confiabilidade. De forma a proporcionar uma entrega de pacotes de dados que seja confiável são usadas várias técnicas. Para além disso, o TCP permite também a recuperação de pacotes de dados perdidos, a recuperação de dados corrompidos, a eliminação de pacotes repetidos para além da possibilidade de recuperar a ligação caso ocorra algum problema com a rede ou com o sistema.

O facto de suportar a transferência simultânea em ambas as direções (bidirecional) durante toda a sessão é outra das características que também definem o protocolo TCP.

O TCP também é caracterizado pelo seu mecanismo de estabelecimento e finalização das ligações (a três e quatro tempos) para que seja possível a autenticação e encerramento de uma sessão completa. Para além disso, o TCP garante também que todos os pacotes foram bem enviados/recebidos.

4.3.4.2 SASL

O SASL (Simple Authentication and Security Layer) é um método que permite adicionar suporte a autenticação em protocolos baseados na ligação. Para usar esta especificação, o protocolo deve incluir um comando que permita identificar e autenticar um utilizador a um servidor e (opcionalmente) negociar numa fase posterior a camada de segurança a usar para as seguintes interações do protocolo.

O comando deve receber como argumento o nome do mecanismo SASL. Existem vários mecanismos SASL, de onde se destacam o DIGEST-MD5, SCRAM , PLAIN, ANONYMOUS, EXTERNAL entre outros. De todos os mecanismos disponíveis, foram implementados na base tecnológica três deles: PLAIN, DIGEST-MD5 e SCRAM-SHA-1.

O mecanismo PLAIN não codifica a *password* durante a autenticação e esta é enviada sem qualquer tipo de encriptação.

O método DIGEST-MD5 e o SCRAM-SHA-1 são definidos nas subsecções seguintes.

4.3.4.3 DIGEST-MD5

O DIGEST-MD5 é um método usado principalmente nos protocolos de comunicação e o seu principal objetivo é a negociação de credenciais (nome de utilizador e palavra-passe) e consequente autenticação. Este método é normalmente usado para confirmar a identidade de um utilizador antes de qualquer troca de informação, mantendo assim a segurança dos dados e diminuindo o risco de ataques por terceiros.

O método baseia-se na aplicação de uma função criptográfica (MD5) para codificar as mensagens enviadas e ao uso de *nonces*, para evitar *replay attacks*. Um *nonce* consiste num número ou conjunto de caracteres gerado aleatoriamente que é usado apenas uma vez para assim garantir que comunicações antigas não são reutilizadas para *replay attacks*.

Em 2011, este mecanismo foi considerado obsoleto mas, grande parte dos servidores que implementam o método SASL ainda suportam comunicações usando o DIGEST-MD5.

4.3.4.4 SCRAM-SHA-1

O SCRAM-SHA-1 é um mecanismo de autenticação usando principalmente em serviços como SMTP, IMAP ou XMPP. Este método fornece as mesmas funcionalidades que o DIGEST--MD5 mas com uma ligeira diferença. As funções de encriptação são mais modernas e garantem maior segurança.

Uma troca de mensagens durante o processo de autenticação é bastante complexa mas pode ser resumida em 4 fases.

A primeira consiste no envio de uma mensagem por parte do cliente em que fornece o nome de utilizador assim como um *nonce*.

O servidor irá responder à primeira mensagem do cliente, acrescentando à mensagem o seu próprio *nonce*, assim como o valor de *salt* e de iterações para serem usados posteriormente na função de encriptação da palavra-passe.

A terceira fase corresponde à resposta do cliente que vai enviar novamente o seu *nonce* e uma prova em como tem acesso ao nome de utilizador e à palavra-passe e assim tem permissões para se autenticar. Essa prova é fornecida através de uma sequência de cálculos. O primeiro consiste em concatenar a primeira mensagem do cliente, a primeira mensagem do servidor e a mensagem final do cliente ainda sem a prova. De seguida são então cifradas as palavras-passe. Esta cifragem é bastante complexa e são necessárias várias funções para garantir que foram bem cifradas e não há riscos de inverter essa codificação. A prova do cliente é então calculada através de um ou exclusivo entre a palavra-passe cifrada e o resultado de uma função criptográfica (HMAC, que recebe como argumento a palavra-passe cifrada novamente e a concatenação de mensagens calculada inicialmente). Após a obtenção da prova do cliente, esta é então adicionada à mensagem e enviada ao servidor.

Por último, o servidor responde com a sua prova, que consiste no resultado da função de encriptação HMAC que recebe a palavra-passe cifrada e a concatenação das mensagens iniciais. Após a receção, o cliente verifica se a prova do servidor está correta e é concluída então a autenticação.

Este é um dos mecanismos mais seguros devido às várias e complexas cifragens, mas nem todos os servidores XMPP têm este mecanismo implementado.

Capítulo 5

Módulos genéricos intermédios

Terminada a fase de desenvolvimento relativa à base tecnológica é então possível apresentar o resultado obtido. Resumidamente, todo o processo de desenvolvimento resultou numa base tecnológica que pode ser utilizada fácil e rapidamente por qualquer programador com um conhecimento básico de JavaScript. Nas subsecções seguintes são apresentados alguns dos módulos e exemplos de como utilizar esta camada padronizada.

5.1 Módulos de Acesso aos Recursos do Dispositivo

Existem alguns exemplos que ajudam a perceber melhor a arquitetura e o modo de utilização da base tecnológica. Como mencionado anteriormente, um dos principais objetivos da base tecnológica é facilitar o desenvolvimento de aplicações multiplataforma, reduzindo ao máximo a quantidade de código escrito. Para além disso, durante o desenvolvimento da base tecnológica todos os módulos foram separados de forma a eliminar qualquer dependência entre eles (excetuando alguns casos de funcionamento em conjunto). Os módulos desenvolvidos destinam-se então a ser integrados e utilizados no desenvolvimento de aplicações de mais alto nível a funcionar nas duas plataformas suportadas.

5.1.1 Geolocalização

Um dos principais e mais importantes módulos desenvolvidos é o acesso aos sensores de GPS para a obtenção da posição atual do dispositivo. Através deste módulo é possível aceder às coordenadas geográficas do *smartphone*.

Para além disso, o módulo de geolocalização oferece também a possibilidade de gravar as coordenadas geográficas no servidor (ou localmente numa base de dados SQLite, caso não seja

possível efetuar uma ligação ao servidor). Neste caso, existe uma ligação entre o módulo de geolocalização e os módulos de comunicação com o servidor e de armazenamento numa base de dados SQLite (apresentados mais à frente).

Para iniciar o processo de obtenção das coordenadas geográficas basta simplesmente adicionar o seguinte código à aplicação que o usa:

```
var Geolocation = require('/framework/features/geolocation');
var GPS = new Geolocation(arg);
```

Como se pode perceber, bastam duas linhas de código para ativar o módulo de geolocalização. Em primeiro lugar é necessário requisitar o módulo, ficando assim disponível para ser usado através da variável a ele associado. Para iniciar o processo de obtenção das coordenadas, só é necessário chamar a função "*new Geolocation(arg)*" em que "*arg*" pode ser verdadeiro ou falso, ou seja, se pretende ativar ou não a gravação automática das coordenadas GPS. Para além disso, este módulo oferece outras funções que permitem parar o processo de obtenção das coordenadas (*GPS.stopGeolocation()*), reiniciar o processo *(GPS.startGeolocation()*) ou guardar manualmente as coordenadas obtidas (*GPS.saveLocation()*).

Para além disso, tal como a maioria dos módulos de acesso aos recursos do dispositivo, é também possível aceder a outros valores associados ao módulo. Neste caso, é possível então aceder às coordenadas GPS, altitude, velocidade, entre outros através da função *GPS.getValues()*.

5.1.2 Acelerómetro

Outro dos módulos desenvolvidos na base tecnológica é o acesso ao acelerómetro do dispositivo. Ao criar uma instância do módulo, os valores a ele associados (acelerações) estão em constante atualização assim como o detetor de quedas (que se baseia nos valores do acelerómetro, explicado mais à frente).

De forma a criar o módulo de acesso aos dados do acelerómetro é necessário incluir o seguinte código, em qualquer lugar do projeto de desenvolvimento:

```
var Accelerometer = require('/framework/features/accelerometer');
var acc = new Accelerometer();
```

Como nos outros módulos, é necessário requisitar o módulo para que seja então possível utilizar o módulo e as funções a si associadas. Após o módulo estar disponível, já é possível então iniciar o processo de obtenção dos valores do acelerómetro e o processo de deteção de quedas. Para isso, basta então criar um novo objeto (neste caso um *Accelerometer*).

Também este módulo permite aceder aos valores associados ao acelerómetro através da função "*acc.getValues()*".

5.1.3 Câmara

O acesso à câmara do dispositivo é também importante para muitas aplicações, e dessa forma foi desenvolvido, na base tecnológica, um módulo que permite aceder à câmara do dispositivo possibilitando assim a captura de uma foto. O código necessário para abrir a câmara do dispositivo é semelhante ao exemplo a seguir apresentado:

```
var Camera = require('/framework/features/camera');
new Camera();
```

Inicialmente é então necessário requisitar o módulo para que fique disponível, bastando depois criar uma nova instância da câmara (objeto *Camera*) para que seja aberta a câmara instalada no dispositivo e permitir assim a captura de uma foto. Após a captura, é mostrado um ecrã de confirmação para aprovar (ou não) a foto.

5.1.4 Acesso a aplicações padrão instaladas

A base tecnológica desenvolvida permite, facilmente, aceder às aplicações padrão dos dispositivos, como a aplicação de SMS ou de telefone. Para isso, basta incluir o ficheiro *helpers*. Um exemplo de como aceder à aplicação de SMS, com o destinatário (*num_telefone)* previamente preenchido é simplesmente:

```
Ti.include('/framework/utils/helpers.js');
openSMSapp(num_telefone);
```

5.1.5 Notificações Locais

A possibilidade da criação de notificações locais é bastante útil em muitas aplicações. Dessa forma, foi implementado também na base tecnológica um gestor de notificações. Para a criação desse gestor, basta incluir o seguinte código:

```
var NM = require('/framework/features/notification_manager');
NotificationManager = new NM();
```

Desta forma, o gestor de notificações fica acessível através da variável *NotificationManager* e é então possível, a partir de agora, criar e apagar notificações. Para criar uma nova notificação é necessário chamar a função *newNotification*. Esta função recebe um objeto JSON como argumento onde são especificados o título e a mensagem a mostrar na notificação. Por exemplo:

```
NotificationManager.newNotification({
                                    title: 'Teste',
                                    message: 'Mensagem de teste'
                                    });
```

Este módulo permite também eliminar todas as notificações que estejam por ler ou que ainda se encontrem na barra de notificações. Basta então chamar a função *NotificationManager.cleanNotifications()* e todas as notificações desaparecem da barra de notificações.

5.1.6 Base de Dados SQLite

É importante que todas as aplicações guardem os seus dados numa base de dados local. Tendo isso em consideração, foi desenvolvido um módulo que permite criar uma base de dados SQLite para armazenar as informações mais importantes.

O módulo oferece principalmente 5 funções principais. As quatro primeiras são as tradicionais instruções da linguagem SQL utilizadas numa base de dados (INSERT, SELECT, UPDATE, DELETE). A quinta função permite executar qualquer tipo de instrução na base de dados, por exemplo, quando a instrução SQL é complexa de mais e as funções básicas do módulo não permitem executar esse tipo de tarefa.

INSERT

Uma das mais importantes e mais utilizadas numa base de dados é a função de inserção de novos dados. Esta permite adicionar novas entradas à base de dados. Dessa forma, foi então criada a função de INSERT que recebe um objeto JSON como parâmetro e através dos valores desse objeto, adiciona as entradas à base de dados. Para utilizar esta função, basta adicionar o código seguinte:

```
var db = new Database();
db.insert(args);
```

O objeto *args* corresponde a uma especificação JSON semelhante ao exemplo apresentado:

```
var args  = {
    table_name: 'USERS',
    parameters: [
        { type: 'INTEGER',name: 'age'},
        { type: 'TEXT', name: 'name'} ],
    values : [
        { age: 20, name: 'John'},
        { age: 22, name: 'Smith'} ]};
```

O objeto JSON tem uma estrutura particular, no caso desta função INSERT. Tem dois campos (parâmetros e valores) onde o primeiro define o nome e o tipo de parâmetros que se vai usar para, posteriormente, se utilizar no campo *values*. Assim, através do mesmo objeto JSON, os campos são definidos, assim como o valor com que devem ser adicionadas as entradas à base de dados.

SELECT

A função de SELECT, que permite pesquisar numa base de dados, é outra das mais utilizadas. Então, foi criada no módulo uma função que permite facilmente realizar essa tarefa. Similarmente à função de INSERT, também esta recebe como argumento um objeto JSON onde são especificados tanto o nome da tabela, como os argumentos para a pesquisa. É também possível incluir um campo adicional que permite definir condições complementares (como GROUP BY por exemplo). Para utilizar este módulo, basta então inserir o seguinte código no projeto da aplicação:

```
var db = new Database();
db.select(args);
```

O objeto JSON passado como argumento (*args*) tem uma estrutura semelhante ao apresentado em baixo.

```
var args = {
    table_name: 'USERS',
    parameters: [age, name],
    condition: 'GROUP BY AGE' };
```

Como se pode perceber, o objeto especifica a tabela onde se quer fazer a pesquisa, os parâmetros que se pretende pesquisar e ainda se define que os resultados da pesquisa devem ser agrupados por idade.

UPDATE

Muitas vezes é necessário atualizar certas entradas que estão guardadas numa base de dados. Para concluir essa tarefa é então utilizada a função de UPDATE. No módulo desenvolvido para a base tecnológica, que permite a criação, acesso e manipulação de base de dados, também é possível fazer essas atualizações. Desse modo, sempre que seja necessário alterar uma das entradas existentes na base de dados, o código seguinte tem que ser adicionado ao projeto:

```
var db = new Database();
db.update(args);
```

Tal como nos exemplos anteriores, o código necessário para manipular a base de dados resume-se a duas linhas de código. O objeto passado como parâmetro (*args*) tem uma estrutura idêntica à do exemplo abaixo. A título de exemplo, este código mostra a especificação de uma alteração à idade do utilizador que se chama “John”.

```
var args = {
    table_name: 'USERS',
    parameters: [{id: age, value: 23}],
    condition: "WHERE NAME = 'John'" };
```

DELETE

É também possível, através da função DELETE, eliminar entradas da base de dados. Neste caso, o código necessário para eliminar as entradas pretendidas, também se resume a poucas linhas de código. Por exemplo, utilizando o código abaixo, será eliminado da base de dados o utilizador que se chama "Smith":

```
var db = new Database();
var args = {
    table_name: 'USERS',
    condition: "WHERE NAME = 'Smith'"
};

//Não pode ser DELETE porque essa função
//já pertence às bibliotecas do Titanium
db.remove(args);
```

EXECUTAR COMANDO

Apesar das funções já descritas permitirem realizar a maioria das tarefas relacionadas com bases de dados, por vezes é necessário executar comandos mais complexos de modo a executar operações mais evoluídas, como por exemplo um JOIN. Tendo isso em consideração, foi desenvolvida a função "*executeCmd"* que permite ao programador escrever a totalidade do comando a realizar. Usando um dos exemplos anteriores, se o programador pretender eliminar o utilizador "Smith" da tabela "Users" pode também concluir essa tarefa usando o seguinte código:

```
var db = new Database();
db.executeCmd("DELETE FROM USERS WHERE NAME = 'Smith'");
```

5.2 Módulos de Criação de Elementos Gráficos

De forma a facilitar e agilizar o processo de criação das interfaces gráficas da aplicação, existem alguns elementos previamente definidos que podem ser criados de uma forma muito mais rápida e eficaz comparativamente ao método originalmente oferecido pelo Titanium Studio. Todos os elementos gráficos têm um aspeto e *layout* predefinido que pode ser alterado através do ficheiro *layout.js* (localizado na pasta '*framework/ui/'*) que tem o mesmo princípio de funcionamento de um ficheiro CSS. Neste ficheiro é possível alterar a cor dos botões, a disposição dos botões num menu, a cor de fundo da janela, entre outros. Nas subsecções seguintes são apresentados os elementos gráficos cuja criação foi facilitada pela base tecnológica, juntamente com alguns exemplos.

5.2.1 Mapa (Google Maps e Apple Maps)

Visto que a Gisgeo é uma empresa que desenvolve SIG, é de extrema relevância permitir a criação de páginas com representação geográfica o mais fácil e rápido possível. Com isso em mente, foi criado um módulo que permite então a criação de uma página onde é mostrado um mapa (Google Maps e Apple Maps, dependendo da plataforma) com a possibilidade de adicionar marcadores

Para criar uma página com um mapa basta adicionar o código seguinte em qualquer parte do projeto:

```
var MapView = require('/framework/ui/map_view');
var maps= new MapView();
```

No exemplo acima é também possível passar uma lista com marcadores a mostrar inicialmente no mapa.

Este módulo permite também adicionar uma nova rota ao mapa criado e para isso basta chamar a função '*maps.addNewRoute(route)*' que recebe como argumento uma lista com os pontos que constituem a rota.

Os exemplos da Figura 9 mostram uma simples janela, com um mapa e representação da posição atual do dispositivo.

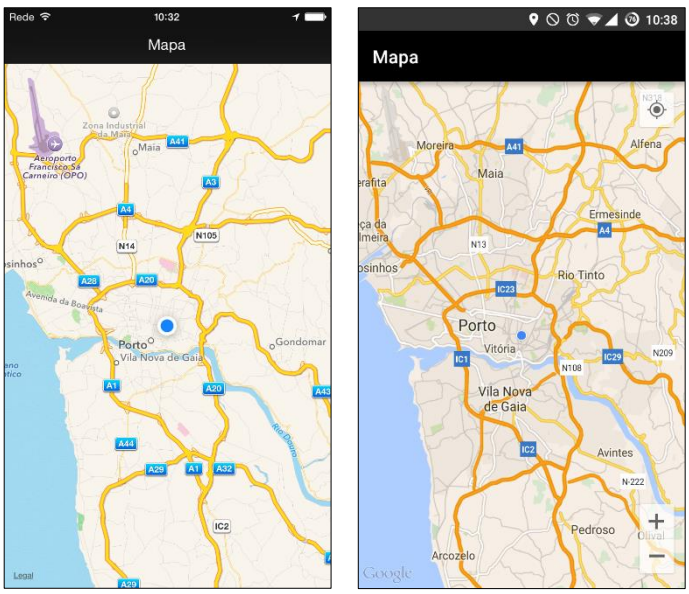


Figura 9: Mapa (iOS/Android)

5.2.2 Botão

Um dos principais e mais usados elementos gráficos é o botão. O tradicional método de criação de botões oferecido pelo Titanium Studio foi substituído por uma versão muito mais simples e eficaz. Para criar um botão usando a base tecnológica, só é necessário chamar a função '*new Button*(*title*,*func*)' passando-lhe como argumento o título do botão a apresentar, assim como a função a executar assim que o botão é clicado. Por exemplo, para criar um botão que irá abrir a câmara assim que é clicado, basta inserir o seguinte código, em qualquer parte do projeto:

```
var button_CAMERA = new Button('Câmara', function() {
    var Camera = require('/framework/features/camera');
    new Camera();
});
```

O resultado da inserção desta pequena fracção de código permitiria criar um botão semelhante ao da Figura 10:

5.2.3 Menu

A maioria das aplicações móveis permite realizar várias tarefas e, normalmente, encontra-se na página principal da aplicação um menu com vários botões. De forma a facilitar a criação desse elemento várias vezes utilizado, foi criado um módulo que permite adicionar facilmente um menu a uma página.

A criação de um menu consiste na construção de um objeto JSON onde são especificados os botões para cada linha do menu e este adaptar-se-à automaticamente ao número de botões adicionados.

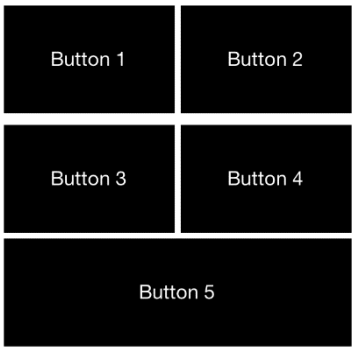
Por exemplo, para criar um menu composto por três linhas, em que na primeira e segunda linha se encontram dois botões e na última linha um único botão, só é necessário adicionar o seguinte código:

```
var buttons = { lines: [[b1,b2],[b3,b4],[b5]] };
```

A disposição dos botões já foi criada, resta agora criar o menu e adicionar os botões pretendidos. A criação do menu é bastante fácil, bastando para isso inserir o código abaixo:

```
win.add(new Menu(button));
```

Assim, o menu é criado e é imediatamente adicionado à janela (*win*) criando um resultado semelhante ao da Figura 11.



5.2.4 Página de *Login*

Um dos elementos gráficos mais usados nas aplicações com autenticação é a página de login. Então para simplificar o processo de criação, e reduzir o tempo necessário para criar uma página de *login,* foi desenvolvido um módulo que reduz significativamente o código necessário para a sua criação. Assim, para criar uma página de *login* apenas é necessário o seguinte código:

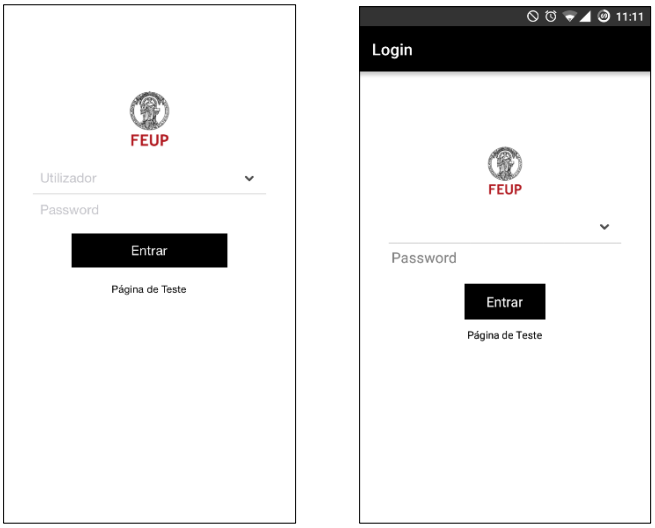
```
new LoginPage(args);
```

Onde *args* é um objeto JSON semelhante ao seguinte:

```
var args={
    //URL para o ícone que aparece no topo da página (opcional)
    icon:url_icon,
    //Mostrar um spinner em vez de uma caixa de texto (opcional)
    spinner:true,
    //valores do spinner(se este for ativado)
    values:users,
    //informação adicional a mostrar na pagina de login (opcional)
    info:info_adicional,
    //função a executar se o login for efetuado com sucesso
    success: successCallback
}
```

O módulo de criação de uma página de *login* permite criar vários tipos de páginas. No objeto enviado como argumento, é possível escolher um ícone para inserir antes dos campos de texto usados para o *login*, permite trocar a caixa de texto usada para inserir o nome de utilizador por um *spinner* onde é possível escolher entre vários utilizadores já registados/inseridos. É de extrema importância adicionar o campo *success* ao objeto enviado como parâmetro, caso contrário, mesmo que o *login* seja efetuado com sucesso, não acontecerá nada.

Dois ecrãs das páginas de *login* criadas tanto para iOS como para Android podem ser observados na Figura 12.



5.2.5 Formulário

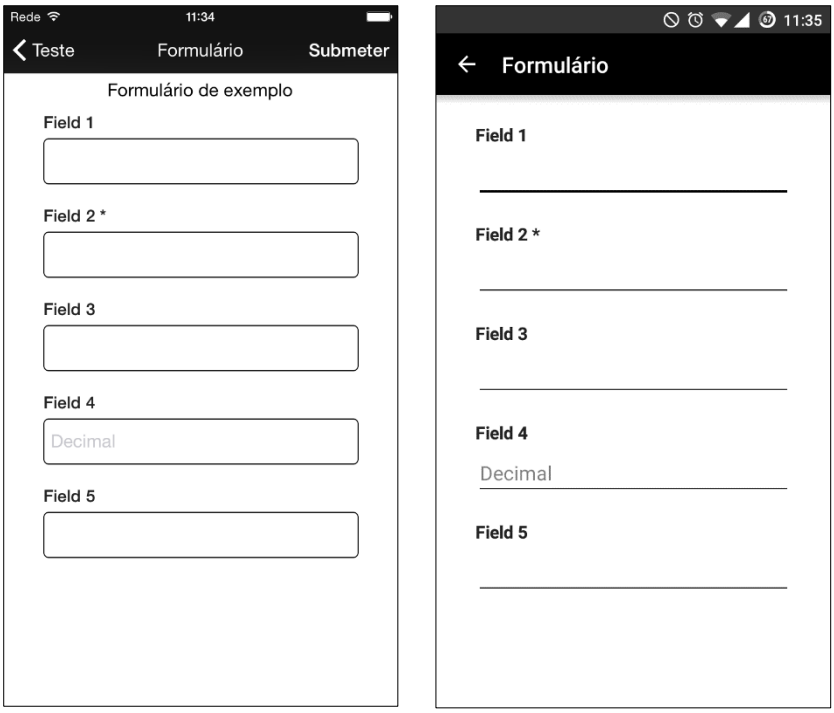
A maioria das aplicações móveis utilizam formulários e, normalmente, a criação de todos os campos necessários é bastante morosa e pode tornar-se complexa. Para agilizar todo esse processo, foi criado um módulo que simplifica a criação de formulários. Então, para criar um formulário só é necessário o seguinte código:

```
var Form = require('/framework/ui/CreateForm');
new Form(args);
```

O objeto *args* é um objeto escrito no formato JSON semelhante ao apresentado em baixo. Por definição existem vários tipos de campos possíveis para adicionar ao formulário (*date, time, phone, password,…*). É também possível adicionar uma variável *required* (que define o preenchimento do campo como obrigatório) assim como a definição de uma *hint* para ajudar no preenchimento do formulário.

```
var args={
  title:'Abastecimento',//titulo do formulário
  fields:[//campos para preencher o formulário
         {title:'Data',type:'date',id:'date'',required:true},
         {title:'Observações',type:'text',id:'observations'},
         (...)
};
```

Nos exemplos da Figura 13, é possível observar ecrãs resultantes da criação de formulários, para iOS e Android.



5.3 Funcionalidades Adicionais

Para além de todos os módulos fundamentais para o bom funcionamento das aplicações resultantes da base tecnológica, foram também inseridos/desenvolvidos alguns módulos que enriquecem a base tecnológica. Nas subsecções seguintes são apresentadas algumas dessas funcionalidades adicionais presentes na base tecnológica.

5.3.1 Leitor de Código de Barras/QRCode

O módulo do leitor de código de barras e QRCodes é bastante importante em certas aplicações. Com ele é possível ler códigos de barras tradicionais ou os recentes QRCodes. Este módulo integra o módulo desenvolvido pela Appcelerator, baseado na biblioteca ZXing[1].

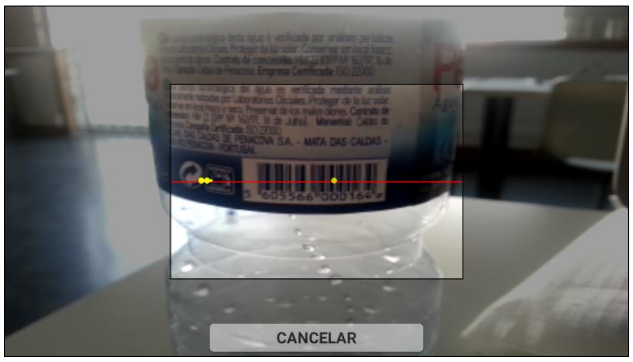
Através deste módulo é possível descodificar (e codificar) qualquer *barcode/qrcode* através da câmara fotográfica integrada no dispositivo.

[1] Biblioteca *open-source* que permite a codificação e descodificação de códigos de barras e QRCodes

Tal como outros módulos desenvolvidos na base tecnológica, também este pode ser acedido facilmente, reduzindo significativamente o código necessário para iniciar a secção que permite a leitura e descodificação de *Barcodes/QrCodes*. Para iniciar a descodificação, só é necessário o seguinte código:

```
var BarcodeScanner = require('/framework/features/barcode_reader');
var scanner = new BarcodeScanner();
```

Através destas duas únicas linhas de código, é iniciado o processo de descodificação e assim que é encontrado algum *Barcode/QrCode* é guardado o resultado num objeto JSON onde é especificado o tipo de código de barras e o resultado da descodificação. Para aceder a esse resultado, basta chamar a função "*scanner.getResult()"* e assim ter acesso aos dados obtidos na leitura. Na Figura 14, é apresentada a interface da aplicação que permite a descodificação dos *Barcodes/QrCodes*.



5.3.2 *Framework* de Realidade Aumentada Georreferenciada

Nos últimos anos, a realidade aumentada transportou-se para as aplicações móveis e tem vindo a ganhar bastante popularidade. A incorporação de realidade aumentada numa aplicação móvel pode fornecer ajuda em determinados contextos e até mesmo auxiliar na realização de tarefas.

Com o avançar da tecnologia de realidade aumentada, surgiram então os primeiros sistemas de realidade aumentada com georreferenciação. Assim, tornou-se possível apresentar informações úteis ao utilizador, dependendo do local onde se encontra e para onde está a olhar. Uma das aplicações mais usadas deste tipo de realidade aumentada é a integração em aplicações de turismo. Normalmente esse tipo de aplicações permite ao utilizador/turista ver os pontos de interesse mais importantes, assim como alguma informação adicional como a história do local, a distância a que se encontra, e outra informação útil.

Tendo isto em consideração e conjugando com as necessidades e características inerentes da Gisgeo, foi adicionado à base tecnológica um módulo de realidade aumentada georreferenciada.

5.3.2.1 Implementação

O módulo foi desenvolvido tendo como base o projeto *augmentTi* (https://bitbucket.org/softlywired/augmentedti). Este projeto é *open-source* e como tal foi possível efetuar (bastantes) mudanças para que se adaptasse à base tecnológica. Inicialmente este projeto consistia numa aplicação que verificava os sensores do dispositivo e usava a API do Google Places para mostrar os pontos de interesse mais importantes perto do utilizador.

Para mostrar os locais mais importantes perto do utilizador, é necessário calcular a distância, a orientação (*bearing*) e o ângulo entre o local onde se encontra o dispositivo e o ponto de interesse. Para calcular esses três principais elementos, são necessários alguns cálculos.

O cálculo da distância poderia ser efetuado com uma de três fórmulas possíveis: Haversine[2], Lei dos Cossenos[3] (na trigonometria esférica) e o Teorema de Pitágoras. Com qualquer uma das três fórmulas é possível calcular a distância, variando apenas a exatidão dos cálculos e o poder de processamento. A função de Haversine é a fórmula mais exata das três, mas em contrapartida necessita de muito poder de processamento, podendo reduzir a *performance* da aplicação. Dessa forma, foi então escolhida a Lei dos Cossenos para calcular a distância por ser, das três, a que se torna mais adequada para a base tecnológica e que equilibra mais a exatidão e a *performance.* A distância entre o ponto 1 e o ponto 2 é então calculada através da seguinte fórmula:

$$d = \cos^{-1}(\sin(lat_1) * \sin(lat_2) + \cos(lat_1) * \cos(lat_2) * \cos(lng_2 - lng_1)) * r$$

Sendo o "*r*" o raio da Terra, ou seja, 6371 (km) e "lat" e "lng" a latitude e longitude dos pontos, respetivamente.

Outro dos valores a calcular é o *bearing*, ou seja, a orientação de um ponto em relação à posição do dispositivo tendo como referência a Terra. Para se calcular essa relação entre os dois pontos, é necessário uma sequência de cálculos:

$$latRad_1 = lat_1 * \frac{\pi}{180}$$

$$latRad_2 = lat_2 * \frac{\pi}{180}$$

$$\Delta lng = (lng_2 - lng_1) * \frac{\pi}{180}$$

$$y = \sin(\Delta lng) * \cos(latRad_2)$$

$$x = \cos(latRad_1) * \sin(latRad_2) - \sin(latRad_1) * \cos(latRad_2) * \cos(\Delta lng)$$

$$bearing = atan2(y, x)$$

[2] Equação usada em navegação que fornece as distâncias entre dois pontos de uma esfera a partir das suas coordenadas.
[3] Generalização do Teorema de Pitágoras que pode ser aplicada a qualquer triângulo

Com o valor do *bearing*, é agora possível calculcar o valor do ângulo entre os dois pontos. Para isso, basta converter o *bearing* calculado anteriormente para graus:

$$degree = \left(\left(bearing * \left(\frac{180}{\pi}\right)\right) + 360\right) \% 360$$

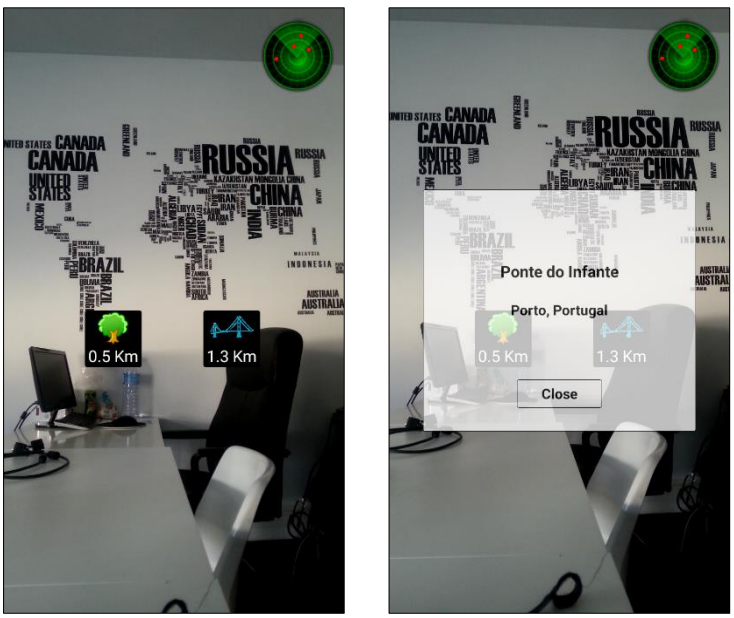
5.3.2.2 Interface

Após terem sido calculados todos os valores necessários, passa-se à próxima fase: a construção da interface. De uma forma mais global, a vista principal é composta por duas camadas principais.

A primeira camada tem como principal função a representação da imagem adquirida pela câmara fotográfica.

A segunda camada, que é sobreposta à primeira, tem como objetivo exibir os pontos de interesse nos locais apropriados, do ponto de vista da câmara. Esta segunda camada tem uma particularidade, está dividida em 4 quadrantes diferentes, onde cada um é responsável por 90 graus da bússola, e estando um deles centrado na orientação atual do dispositivo (direção para onde a câmara aponta). Para cada ponto de interesse são é então calculada a sua direção usando as equações acima mencionadas e, combinando-os com a orientação do dispositivo (direção para onde a câmara aponta), é possível obter a sua posição relativa ao dispositivo. Calculando o ângulo entre a orientação do dispositivo e a direção de cada ponto de interesse, cada um destes pode ser desenhado no quadrante correspondente. Desta forma, o processamento e o desenho dos vários POIs são efetuados de uma forma mais rápida e consequentemente tornam a movimentação (quando o dispositivo é rodado ou movimentado) dos POIs mais suave.

Na Figura 15 podem ver-se alguns exemplos da interface desenvolvida.



5.3.2.3 Exemplos de utilização

A *framework* de realidade aumentada georreferenciada está sempre acessível na base tecnológica. Para incluir e utilizar este módulo é necessário inserir o seguinte código:

```
var AR = require('/framework/ar/ar_controller');
AR.startAR(win);
```

É necessário chamar a função *startAR* para que seja iniciado todo o processo de representação da câmara e a sobreposição dos POIs, assim como todos os cálculos necessários para o bom funcionamento deste módulo.

5.3.3 Ligação a Servidores XMPP

Um dos requisitos que surgiu durante o processo de desenvolvimento e que não estava nos requisitos iniciais foi a ligação a servidores XMPP.

O desenvolvimento deste módulo teve como base o projeto *titanium-xmpp* (https://github.com/jmartin82/titanium-xmpp). Apesar de ser *open-source* e do código original ser específico para o Titanium Studio, o projeto já tinha sido descontinuado há alguns anos e existiam partes do código incompatíveis com as novas versões do Titanium Studio. O código foi então alterado para que fosse compatível com a ferramenta e que fosse de encontro às necessidades da Gisgeo e dos requisitos da base tecnológica.

Um exemplo de uma comunicação entre um cliente e um servidor XMPP é apresentado no Anexo A.

5.3.3.1 Modo de utilização

O módulo de ligação a servidores XMPP, assim como os restantes módulos desenvolvidos na base tecnológica, podem ser acedidos em qualquer parte do projeto, bastando para isso incluir, em primeiro lugar, o módulo e de seguida criar uma nova instância do objeto. Para a criação do objeto são necessários 3 argumentos: URL do servidor, o nome de utilizador a usar no momento de *login,* assim como a *password*. O código seguinte permite uma melhor compreensão da utilização deste módulo:

```
var Server = require('/framework/features/server_connection');
var connection = new Server("server_url", "username", "password");
```

A partir deste momento a aplicação fica apta a receber qualquer mensagem enviada através dos servidores XMPP e permite também enviar mensagens para qualquer outro utilizador que utilize o protocolo, basta inserir o código seguinte:

```
connection.sendMsg('user_destino', 'mensagem');
```

5.3.4 Deteção de Quedas

As aplicações baseadas em SIG trazem bastantes benefícios e podem ser utilizadas em vários cenários possíveis. Estes podem ir desde a monitorização de pessoas de idade até ao montanhismo. Em ambos os casos é útil e muito utilizado detetar possíveis quedas dos utilizadores da aplicação. Uma das principais razões pela perda de capacidades e mobilidade nas pessoas de idade, o desaparecimento ou morte dos montanhistas, entre outros é a falta de assistência quando ocorre uma queda. Para prevenir esses casos, foi então desenvolvido um módulo que permite detetar quedas, através dos movimentos efetuados pelo *smartphone*.

A deteção de quedas utiliza essencialmente o sensor do acelerómetro do *smartphone*. Através da diferença de acelerações do dispositivo, é então possível detetar possíveis quedas. Ou seja, nas atividades diárias que qualquer ser humano realiza, a aceleração ronda o valor da aceleração da gravidade terrestre (1G) e uma queda é normalmente precedida de um período de queda livre. Dessa forma, quando o valor da aceleração do dispositivo baixa drasticamente do valor 1G, iniciar-se-à então o processo de deteção que irá calcular constantemente a aceleração do dispositivo e verificar se esta sobe repentinamente para um valor acima dos 3G (normalmente, o mínimo utilizado é de 3G, que acontece quando do impacto com o solo ou outro objeto rígido). Se a aceleração atingir o limite inferior (inferior a 1G) e o limite superior (3G ou mais) num curto período de tempo, pode tratar-se de uma queda. No entanto, utilizando apenas este método, o número de falsos alarmes pode ainda ser elevado. Para reduzir esse número, pode ter-se também em consideração que após uma queda mais violenta, normalmente, o utilizador permance imóvel ou deitado durante um certo período de tempo. Para além disso, é também utilizado a orientação do dispositivo, ou seja, na maioria das quedas há uma diferença de 90º entre a orientação inicial e a orientação final de um dispositivo que permaneça fixo em relação ao seu utilizador. Utilizando estes métodos auxiliares, é possível reduzir significativamente o número de falsos alarmes e tornar o processo de deteção de quedas mais fiável.

Este módulo de deteção de quedas está integrado com o módulo do acelerómetro, bastando para isso importar o módulo do acelerómetro e criar uma nova instância.

```
var Accelerometer = require('/framework/features/accelerometer');
var acc = new Accelerometer();
```

O detetor de quedas é ativado automaticamente, assim que um objeto *Accelerometer* é criado. Uma vez que o módulo ainda se encontra numa fase bastante experimental, apenas é mostrado um alerta a informar que foi detetada uma queda. É possível também ao programador definir a função a executar assim que uma queda é detetada.

5.4 Ti.include ou *require*

Existem duas formas de incluir os ficheiros desenvolvidos na base tecnológica: a abordagem tradicional de JavaScript ou a abordagem CommonJS.

Os ficheiros que sejam desenvolvidos pelo método tradicional têm que ser incluídos através da função *"Ti.include"*. Apesar destes ficheiros terem acesso a variáveis e funções globais, não é recomendável a utilização desta abordagem, uma vez que consiste em copiar todo o conteúdo do ficheiro para um novo ficheiro, causando assim problemas de *performance*.

Em contrapartida, a abordagem CommonJS é mais rápida e eficaz e não efetua nenhuma cópia do conteúdo do ficheiro incluído.

Dessa forma, apenas se recomenda a utilização do "*Ti.include*" para aceder a dados estáticos, em que não seja necessário a criação de novos objetos. Assim, todos os módulos foram desenvolvidos utilizando a especificação CommonJS. Já os ficheiros onde estão definidas algumas funções auxiliares e variáveis estáticas (como os códigos de cores, por exemplo), utilizaram a especificação tradicional de JavaScript.

Capítulo 6

Provas de Conceito

Após a conclusão do processo de desenvolvimento da base tecnológica e de forma a testar as capacidades desta, assim como a sua *performance* e comportamento numa aplicação do mundo real, foram então desenvolvidas duas aplicações de demonstração como prova de conceito.

A primeira aplicação, de carácter mais sério e complexo, visava testar as capacidades da base tecnológica em criar aplicações de maior qualidade, enquadradas com o cenário da Gisgeo e os seus clientes.

De forma a testar a rapidez e facilidade de criação de aplicações usando base tecnológica, foi ainda desenvolvida uma segunda aplicação. Com esta, tencionava-se verificar se seria possível, num curto espaço de tempo e com pouco código, desenvolver uma aplicação funcional, facilmente adaptável ao mundo real.

Nas subsecções seguintes são apresentados alguns detalhes e requisitos das aplicações desenvolvidas, juntamente com a arquitetura e os resultados obtidos.

6.1 Aplicação de Recolha de Amostras

A primeira aplicação tem como principal objetivo fornecer às equipas de uma empresa de recolha de amostras de água um método mais rápido e fácil de aceder às tarefas diárias que lhes são atribuídas assim como submeter os resultados e os detalhes de cada recolha, logo que a tarefa esteja terminada. Com a esta aplicação espera-se que todo o processo de recolha de amostras de águas seja melhorado, aumentado a eficácia e eficiência de cada equipa.

6.1.1 Requisitos/Funcionalidades

A aplicação desenvolvida deve cumprir alguns requisitos e deve conter algumas funcionalidades específicas, mesmo sendo ela apenas de carácter demonstrativo. Alguns dos requisitos são comuns à maioria das aplicações móveis. Para além da autenticação, deve ser possível também preencher, guardar e enviar os valores de um formulário, receber tarefas e submeter tarefas.

6.1.1.1 Requisitos Funcionais

Casos de uso

Para garantir a maior qualidade possível e garantir o bom funcionamento de toda a aplicação e as suas funcionalidades, há alguns casos de uso a ter em consideração.

Caso de Uso	Descrição
Comunicação com o servidor	A aplicação deve conseguir comunicar com o servidor para a troca de informações
Configurações de sistema	A aplicação deve aceder às configurações do sistema como o idioma ou o fuso horário
Armazenamento numa base de dados	A aplicação deve armazenar os dados temporários (antes da sincronização com o servidor) numa base de dados local
Autenticação/Identificação	A aplicação deve permitir ao utilizador ou equipa autenticar-se (*login*)

Tabela 4: Casos de uso da prova de conceito A

User Stories

Para uma melhor compreensão dos objetivos e funcionalidades que a aplicação deve ter, são apresentadas de seguida algumas *user stories*.

Descrição	Deve ser possível receber as tarefas ou serviços a realizar.
Prioridade	Alta
Pré-condições	O telemóvel deve ter acesso a uma ligação de dados (*WiFi,* GSM,…).
Descrição	Deve ser possível enviar as tarefas ou serviços já realizados.
Prioridade	Alta
Pré-condições	O telemóvel deve ter acesso a uma ligação de dados (*WiFi,* GSM,…).

Descrição	A comunicação com o servidor deve ser efetuada através de *webservices*.
Prioridade	Alta
Pré-condições	O telemóvel deve ter acesso a uma ligação de dados (*WiFi*, GSM,…).
Descrição	Os dados temporários devem ser armazenados numa base de dados local.
Prioridade	Alta
Pré-condições	-
Descrição	Deve ser possível autenticar-se através do nome de utilizador e *password.*
Prioridade	Alta
Pré-condições	O telemóvel deve ter acesso a uma ligação de dados (*WiFi,* GSM,…).
Descrição	Deve ser possível associar uma viatura a um utilizador assim que este se autentica
Prioridade	Média
Pré-condições	O telemóvel deve ter acesso a uma ligação de dados (*WiFi,* GSM,…).
Descrição	A aplicação deve apresentar formulários diferentes para cada tipo de amostras
Prioridade	Alta
Pré-condições	-

Tabela 5: *User Stories* da prova de conceito A

6.1.1.2 Requisitos Não-Funcionais

Para além dos requisitos funcionais, todas as aplicações devem cumprir alguns requisitos não-funcionais que garantem uma maior qualidade e eficiência. Estes requisitos são também de extrema importância e também eles não podem se esquecidos. De entre eles, destacam-se:

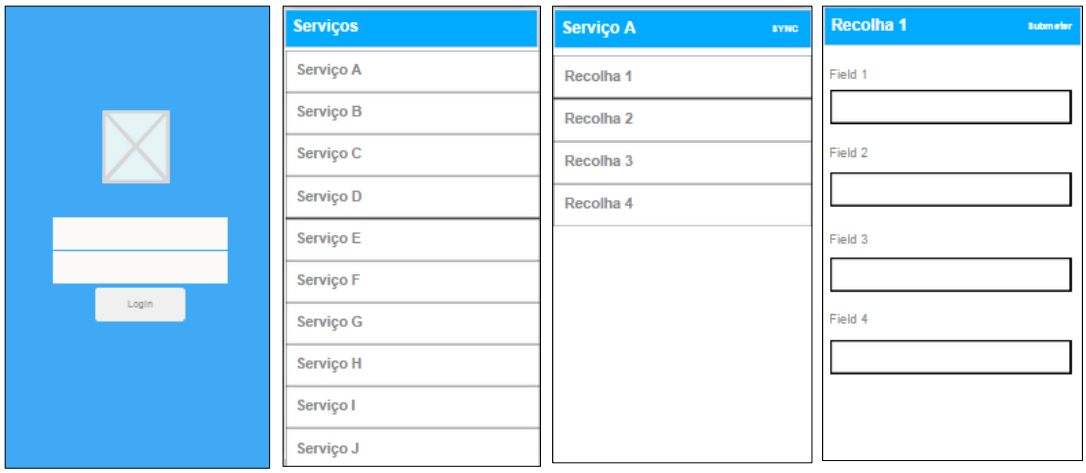
- **Segurança:** a aplicação deve guardar e transmitir os dados em segurança, sem perda de informações
- **Fiabilidade**
- **Robustez**
- **Estabilidade**
- **Usabilidade**
- ***Performance*/Tempo de resposta**
- **Compatibilidade com Android e iOS**

6.1.1.3 Cumprimento dos Requisitos

A maioria dos requisitos e funcionalidades da aplicação de recolha de amostras foram garantidos pela base tecnológica previamente desenvolvida, reduzindo assim, significativamente, o esforço necessário para garantir o cumprimento dos restantes requisitos.

6.1.2 Interface

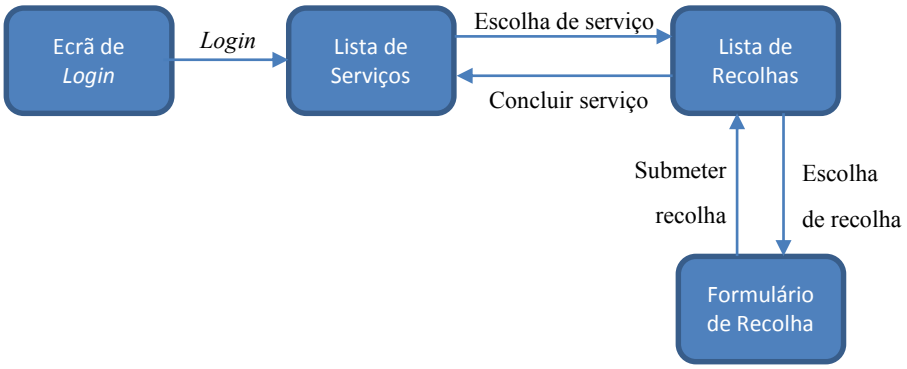
A aplicação para prova de conceito é bastante simples, consistindo, basicamente, em quatro ecrãs diferentes. O primeiro (à esquerda) corresponde ao ecrã de *login*. Nele o utilizador escolhe o seu nome de utilizador de uma lista já preenchida e, seguidamente, insere a sua *password*. Se o *login* for efetuado com sucesso, será apresentado um ecrã com todos os serviços pendentes atribuídos ao utilizador. Escolhendo um dos serviços da lista, é então apresentada uma lista de tarefas que compõem o serviço. Existem vários tipos de recolha e, tendo isso em consideração, sempre que é escolhida uma das tarefas, será apresentado um formulário relativo ao tipo de amostra que define a tarefa.



Da esquerda para a direita, na Figura 16, pode observar-se o *mockup* do ecrã de *login*, a lista de serviços pendentes, a lista de tarefas que compõem um serviço e o formulário relativo a cada tipo de amostra.

A aplicação deve ter um aspeto e comportamento idêntico ao de uma aplicação nativa (Figura 17), tendo sempre em consideração os padrões de desenho de *interfaces,* tanto do Android como do iOS.

A apresentação de ecrãs na aplicação segue uma sequência semelhante à apresentada no fluxograma da Figura 17.



6.1.2.1 Ecrã de *Login*

O ecrã inicial que surge quando é iniciada a aplicação e não existe nenhuma sessão ativa é o ecrã de *login*. Este ecrã é composto por dois elementos de entrada, um botão e o logo da empresa. O primeiro dos campos de entrada é uma lista de utilizadores pré-preenchida com os utilizadores registados no servidor e o segundo é um campo para a inserção da *password* correspondente ao utilizador. Após os dois campos estarem preenchidos, o utilizador deve premir o botão de *login* e verificar se conseguiu iniciar uma sessão com sucesso.

Se ocorrer algum problema durante a autenticação, será apresentada uma mensagem ao utilizador com a descrição do erro. Se a autenticação for efetuada com sucesso, a aplicação irá verificar se o utilizador tem uma viatura associada e se esta não existir, aparecerá a opção de escolher a viatura (normalmente identificada pela matrícula). Para a aplicação avançar para o ecrã seguinte, o *login* tem que ser efetuado com sucesso e deve ser associada uma viatura ao utilizador.

6.1.2.2 Lista de Serviços

O ecrã que se segue consiste numa lista de serviços atribuídos ao utilizador que ainda não foram concluídos. Estes serviços são obtidos, inicialmente, de um servidor remoto e cada um deles é composto por várias tarefas (mais especificamente, recolhas).

Nesta lista, apenas são mostrados os serviços que ainda não foram concluídos (ainda faltam realizar algumas tarefas/recolhas). Os que já foram concluídos ou foram já enviados para o servidor remoto ou encontram-se na base de dados local à espera de uma sincronização.

Assim que é escolhido um dos serviços da lista, a aplicação irá avançar para um novo ecrã.

6.1.2.3 Lista de Recolhas

Após o utilizador ter escolhido o serviço que pretende realizar, será mostrada uma lista com todas as tarefas que compões o serviço. As tarefas podem ter dois estados possíveis: concluídas ou pendentes. Assim que todas as tarefas foram concluídas, é então possível submeter o serviço

e enviar para o servidor (caso o dispositivo tenha acesso a uma ligação de dados) ou guardar as tarefas na base de dados local.

As recolhas podem ser de vários tipos, variando assim o tipo de formulário a preencher na hora da realização da tarefa.

Assim que uma das tarefas é escolhida, a aplicação avançará para um novo ecrã.

6.1.2.4 Formulário de Recolha

Este ecrã irá depender do tipo de recolha escolhida, variando assim o número e tipo de campos de entrada a preencher para finalizar a recolha. O ecrã é composto por um título (a identificar o tipo de amostra) seguido de uma lista de campos a preencher. No topo do ecrã encontra-se um botão de submissão que permite ao utilizador concluir a tarefa.

Na hora da submissão, se ocorrer algum erro (deixar um campo obrigatório por preencher, por exemplo) é mostrada uma mensagem com a descrição do erro. Se o formulário estiver correto, não houverem problemas e a tarefa for finalizada com sucesso, a aplicação volta ao ecrã das listas de recolhas.

6.1.3 Arquitetura

A aplicação móvel desenvolvida divide-se em duas partes: a aplicação desenvolvida usando a base tecnológica e o servidor de *back-end*. Para a prova de conceito, apenas foi desenvolvida a aplicação móvel através da camada padronizada desenvolvida anteriormente. O servidor de *back--end* já existia e foi disponibilizado pela Gisgeo.

6.1.3.1 Aplicação Móvel

O desenvolvimento da aplicação móvel foi bastante rápido e fácil, uma vez que foi utilizada a base tecnológica já desenvolvida. Para a criação da aplicação, e de forma a separar bem todos os ecrãs, foram criados três ficheiros responsáveis pelas chamadas à base tecnológica para a criação dos elementos gráficos e outros dois ficheiros, onde são obtidos os dados do servidor e os códigos dos vários tipos de recolhas. Em média, os ficheiros desenvolvidos não ultrapassam as 50 linhas de código (comentários incluídos) o que comprova a eficácia da base tecnológica desenvolvida.

Dos três ficheiros criados, responsáveis pela interação com a camada padronizada, um deles é responsável pela criação da página de *login*, outro pela lista de serviços e o último pela lista de recolhas. A criação dos formulários é uma tarefa da base tecnológica e por isso não é necessário construir manualmente todo o formulário. De seguida, são expostos alguns detalhes destes três ficheiros que permitem criar facilmente a aplicação de prova de conceito.

Página de *Login*

O ficheiro que permite a criação da página de *login* baseia-se apenas na definição de três funções, sendo a primeira a que procede à autenticação, a segunda que executa se ocorrer algum erro, e a terceira a que corre se o *login* for finalizado com sucesso. Após a definição das três funções principais, resta apenas criar a página de *login* através da função "*new LoginPage(args)"* (como exposto na subsecção 5.2.4).

Lista de Serviços

O ecrã responsável por apresentar a lista de serviços resume-se à definição da função a executar quando um elemento da lista é selecionado e à criação da tabela. A tabela é preenchida com os dados obtidos do servidor, que são processados previamente e só depois adicionados à tabela. Nesta tabela são apresentados unicamente os serviços que ainda não foram concluídos. Assim que um serviço é concluído, este irá desaparecer da lista e será enviado para o servidor (ou guardado na base de dados local se não houver uma ligação de dados ativa).

Lista de Recolhas

Tal como a lista de serviços, também a criação da lista de recolhas que compõem um serviço se resume à definição da função a executar assim que um elemento é selecionado e à criação da tabela. Neste caso, a tabela será preenchida pelas várias recolhas que definem o serviço e todas elas são mostradas, assim como o seu estado, na lista apresentada.

Assim que o utilizador realizar todas as recolhas, o botão de submissão ficará ativo e poderá então concluir o serviço e enviá-lo automaticamente para o servidor (ou guardar localmente).

6.1.3.2 *Webservices*

De forma a obter todos os dados necessários para o bom funcionamento da aplicação, esta acede a uma API REST fornecida pela Gisgeo. Através dela, é possível obter os utilizadores, viaturas, serviços e recolhas atribuídas assim como enviar os serviços e recolhas já concluídas. A API tem a seguinte definição:

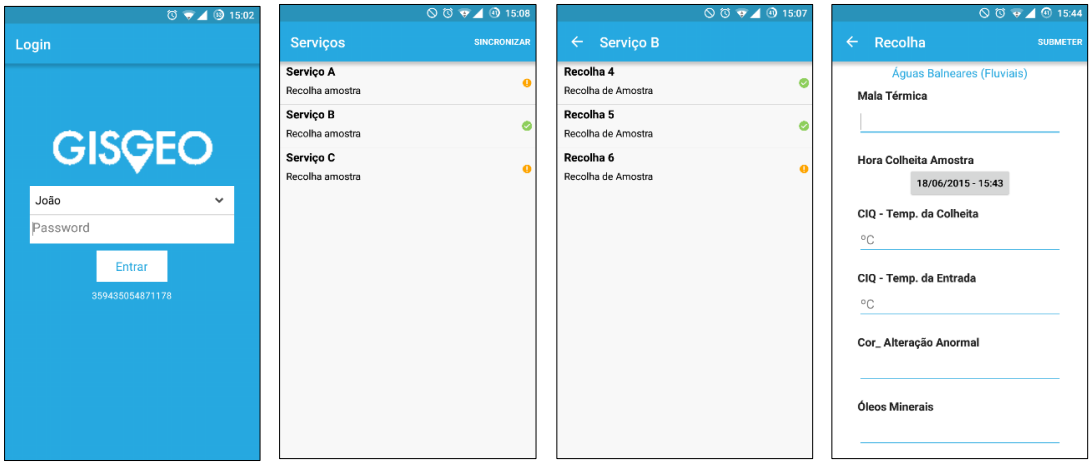
Método	Localização	Descrição
GET	/users	Obter os nomes de utilizadores.
GET	/cars	Obter as viaturas (identificadas pela matrícula)
GET	/services	Obter os serviços e recolhas atribuídos
POST	/users/login	Enviar os dados para autenticação
POST	/services/submit	Enviar os serviços concluídos

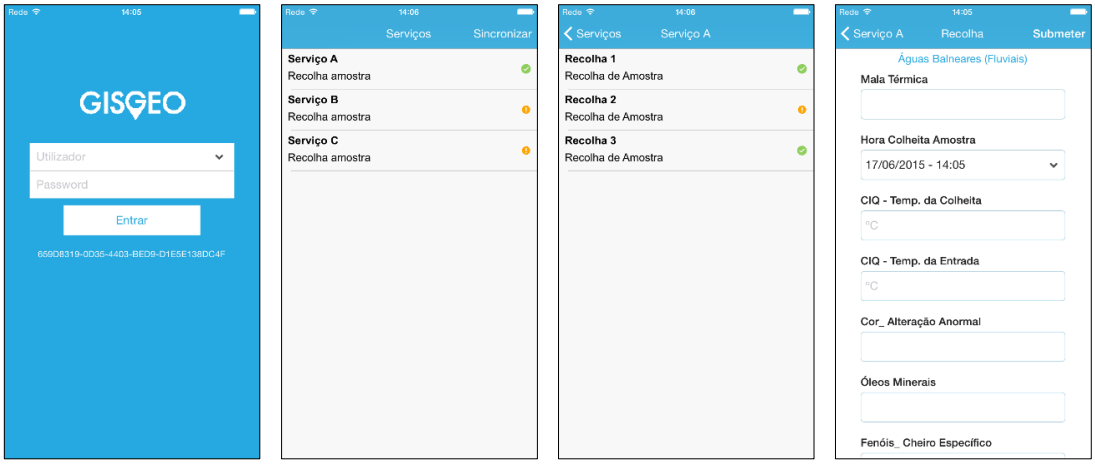
Tabela 6: Descrição da API REST da prova de conceito A

Existem cinco tipos de pedidos diferentes, três deles pelo método de GET e dois deles por POST. Os métodos de GET são utilizados para obter a lista dos utilizadores (juntamente com o seu número de identificação), a lista de viaturas e a lista de serviços (incluindo as recolhas que compõem o serviço). Os pedidos por POST são utilizados para realizar a autenticação (enviando as credenciais) e para submeter os serviços já concluídos, assim como as tarefas a eles associadas.

6.1.4 Produto Final

Todo este processo de desenvolvimento levou à criação de uma aplicação funcional, estável, adequada ao contexto em que se insere e com níveis de qualidade e desempenho elevados. Para além disso, foi possível manter a aplicação fiel aos *mockups* inicialmente criados. Na Figura 18 e na Figura 19, são apresentados alguns *screenshots* da aplicação, primeiro em Android e depois em iOS. Apesar da aplicação não exigir uma grande distinção entre as duas plataformas, é possível verificar que foram utilizados os elementos nativos de cada plataforma, assim como foram seguidos os padrões de desenho de ambos os sistemas operativos.





6.1.4.1 Testes

A aplicação final foi testada em ambas as plataformas (Android/iOS). Mas, visto que é necessário uma licença de desenvolvimento para testar em dispositivos iOS físicos, só foi possível testar no simulador iOS fornecido pelo Xcode. Para Android, foram usados alguns dispositivos físicos para realizar os testes finais à aplicação, de onde se destacam:

- Telemóvel Samsung Galaxy SIII, com sistema operativo Android 5.1
  - CPU: Quad-core 1.4 GHz Cortex-A9;
  - Ecrã: Super AMOLED 4.8”, 1280x720;
  - Memória Interna: 16GB;
  - RAM: 1GB;

- Tablet Insys MID77D1, com sistema operativo Android 4.2.2
  - CPU: Dual-Core ARMv7 1.2 GHz NEON;
  - Ecrã: Capacitivo 7” IPS, 1024x600;
  - Memória Interna: 8GB;
  - RAM: 1GB;

- Telemóvel Sony Xperia M2, com sistema operativo Android 4.4.4
  - CPU: Quad-core 1.2 GHz Cortex-A;
  - Ecrã: TFT Capacitivo 4.8”, 960x540;
  - Memória Interna: 8GB;
  - RAM: 1GB;

Todos os dispositivos conseguiram instalar e executar a aplicação corretamente, sem falhas nem grandes problemas de desempenho. Apesar da diferença entre os sistemas operativos e as versões dos vários dispositivos, a aparência e o comportamento manteve-se sempre idêntico.

6.1.5 Conclusões

A base tecnológica desenvolvida permitiu, facilmente, criar uma aplicação de alta qualidade, fiável e robusta com apenas algumas linhas de código. Obviamente, a *performance* da aplicação não é comparável às aplicações nativas, notando-se pequenos atrasos no carregamento das páginas ou na transição entre elas. O atraso é bastante pequeno, quase imperceptível para o utilizador, mas que pode trazer alguns problemas ao nível da interação com o utilizador e a usabilidade da aplicação.

A base tecnológica permitiu garantir, à partida, a maioria dos requisitos e funcionalidades que deveriam estar presentes na aplicação e reduzir bastante o tempo e o custo de desenvolvimento, comparativamente ao desenvolvimento nativo.

6.2 Aplicação para representação de POIs

A segunda aplicação desenvolvida visava testar a rapidez e facilidade de criar aplicações para Android e iOS utilizando a base tecnológica.

O principal objetivo não era criar uma aplicação complexa, de alta qualidade e fiável, mas sim verificar se, usando a base tecnológica, era possível desenvolver uma aplicação mais básica, mas interessante, facilmente adaptável ao mercado de aplicações móveis e do contexto da Gisgeo.

A principal missão do desenvolvimento consistia em criar uma aplicação, em menos de duas horas, com o menor código possível.

6.2.1 Requisitos/Funcionalidades

Apesar de se tratar de uma aplicação básica e fácil de desenvolver, existem alguns requisitos e funcionalidades que devem ser garantidos no produto final.

6.2.1.1 Requisitos Funcionais

Casos de uso

Para garantir a maior qualidade possível e garantir o bom funcionamento de toda a aplicação e as suas funcionalidades, há alguns casos de uso a ter em consideração.

Caso de Uso	Descrição
Comunicação com o servidor	A aplicação deve conseguir comunicar com o servidor para a troca de informações
Representação de Dados	A aplicação deve permitir visualizar no mapa os pontos de interesse obtidos do servidor remoto
Realidade Aumentada Georreferenciada	A aplicação deve incluir o módulo de realidade aumentada georreferenciada de forma a visualizar os POIs mais perto assim como a direção para chegar até eles

Tabela 7: Casos de uso da prova de conceito B

User Stories

Para uma melhor compreensão dos objetivos e funcionalidades que a aplicação deve ter, são apresentadas de seguida algumas *user stories*.

Descrição	Deve ser possível receber os POIS a partir do servidor remoto.
Prioridade	Alta
Pré-condições	O telemóvel deve ter acesso a uma ligação de dados (*WiFi,* GSM,…).
Descrição	Deve ser possível ver os POIs num mapa.
Prioridade	Alta
Pré-condições	O telemóvel deve ter acesso a uma ligação de dados (*WiFi,* GSM,…).
Descrição	Deve ser possível ver os POIs e a sua direção, utilizando o módulo de realidade aumentada georreferenciada.
Prioridade	Alta
Pré-condições	O telemóvel deve ter acesso a uma ligação de dados (*WiFi*, GSM,…) e deve possuir câmara fotográfica.

Tabela 8: *User Stories* da prova de conceito B

6.2.1.2 Requisitos Não-Funcionais

Para além dos requisitos funcionais, todas as aplicações devem cumprir alguns requisitos não-funcionais que garantem uma maior qualidade e eficiência. Estes requisitos são também de extrema importância e também eles não podem se esquecidos. De entre eles, destacam-se:

- **Segurança:** a aplicação deve guardar e transmitir os dados em seguraça, sem perda de informações
- **Estabilidade**
- **Usabilidade**
- ***Performance*/Tempo de resposta**
- **Compatibilidade com Android e iOS**

6.2.1.3 Cumprimento dos Requisitos

Tal como a primeira aplicação de prova de conceito, também a maioria dos requisitos e funcionalidades principais da segunda aplicação foram garantidos pela base tecnológica previamente desenvolvida.

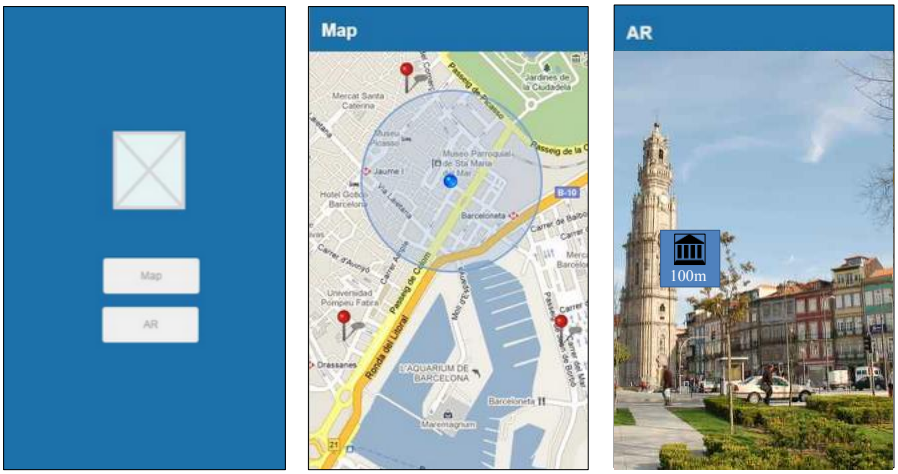
6.2.2 Interface

A segunda aplicação para prova de conceito é bastante básica e simples. Consiste em três ecrãs principais, sendo o primeiro o menu principal onde são apresentados dois botões (o primeiro abre o mapa com a localização dos POIs e o segundo abre o módulo de realidade aumentada georreferenciada).

No segundo ecrã (mapa) apenas são mostrados os marcadores que representam os POIs obtidos do servidor e a localização atual do dispositivo.

Por último, no terceiro ecrã (realidade aumentada georreferenciada,) é apresentada uma vista da câmara onde são sobrepostos os elementos gráficos correspondentes à localização de cada POI.

Na Figura 20 são apresentados alguns *mockups* de baixa fidelidade da segunda aplicação de prova de conceito.



6.2.3 Arquitetura

A aplicação móvel, tal como na primeira prova de conceito, pode ser dividida em duas partes: a aplicação móvel desenvolvida através da base tecnológica e o servidor de *back-end*.

Para prova de conceito apenas foi desenvolvida a aplicação móvel para Android e iOS através da base tecnológica previamente criada, uma vez que o servidor de *back-end* já existia e foi fornecido pela Gisgeo.

6.2.3.1 Aplicação Móvel

O processo de desenvolvimento da segunda aplicação móvel para prova de conceito foi ainda mais fácil e rápido do que o desenvolvimento da primeira aplicação, uma vez que também foi utilizada a base tecnológica criada.

Toda a aplicação resume-se a dois ficheiros JavaScript. O primeiro ficheiro é responsável pela criação dos elementos gráficos e chamadas à base tecnológica e o segundo pela obtenção dos dados do servidor.

Novamente se comprova a eficácia da base tecnológica visto que o código necessário para a criação da aplicação não ultrapassa as 80 linhas (excetuando as alterações efetuados ao ficheiro de *layout* para modificações ao nível da aparência).

Menu principal

O ficheiro principal da aplicação baseia-se apenas na criação do menu e dos botões que o constituem. Inicialmente estes são criados definindo as funções a executar no momento em que são clicados (neste caso, a criação de uma página com um mapa e a chamada ao módulo de realidade aumentada georreferenciada).

Após a criação dos botões, apenas é necessário criar o menu, usando os botões previamente criados e a localização do ícone a apresentar.

6.2.3.2 *Webservices*

Para a obtenção dos dados relativos aos vários pontos de interesse (POIs), foi utilizada uma API bastante simples fornecida pela Gisgeo. A API tem a definição mostrada na Tabela 9:

Método	Localização	Descrição
GET	/pois	Obter os POIs (localização e detalhes)

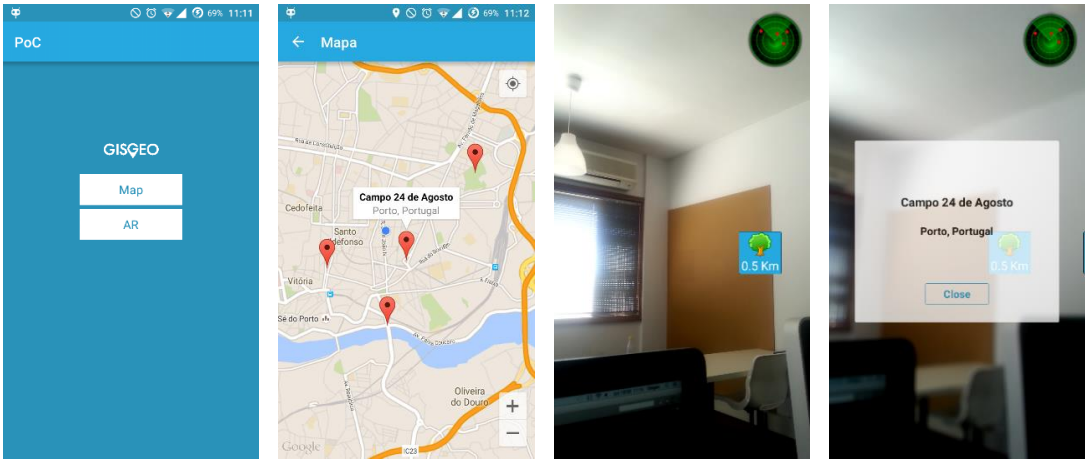
Tabela 9: Descrição da API REST da prova de conceito B

6.2.4 Produto Final

O processo de desenvolvimento levou à criação de uma aplicação bastante simples, mas funcional e com capacidade para ser melhorada e posteriormente integrada em contextos reais.

Apesar da curta duração do processo de desenvolvimento, foi possível criar uma aplicação fiel aos *mockups* criados e com boa *performance*. Para além disso, foi possível testar completamente as capacidades da base tecnológica.

Na Figura 21 são mostrados alguns ecrãs do produto final da segunda prova de conceito. Nos ecrãs apresentados, é utilizado o Campo 24 de Agosto como exemplo.



6.2.4.1 Testes

A aplicação final apenas foi testada na plataforma Android, visto que não foi possível testar em dispositivos físicos iOS por falta de licença de desenvolvimento (o simulador iOS fornecido pelo Xcode não permite aceder à câmara, de forma a testar o módulo de realidade aumentada). De forma a testar a aplicação em Android, foram utilizados alguns dispositivos físicos de onde se destacam:

- Telemóvel Samsung Galaxy SIII, com sistema operativo Android 5.1
  - CPU: Quad-core 1.4 GHz Cortex-A9;
  - Ecrã: Super AMOLED 4.8”, 1280x720;
  - Memória Interna: 16GB;
  - RAM: 1GB;

- Tablet Samsuns Galaxy Tab3, com sistema operativo Android 4.4.2
  - CPU: Dual-core 1.2 GHz Cortex-A9;
  - Ecrã: TFT Capacitivo 7”, 1024x600;
  - Memória Interna: 8GB;
  - RAM: 1GB;

- Telemóvel Samsung Galaxy SIII mini, com sistema operativo Android 5.1
  - CPU: Dual-Core 1GHz Cortex-A9;
  - Ecrã: Super AMOLED Capacitivo 4”, 800x480;
  - Memória Interna: 8GB;
  - RAM: 1GB;

A aplicação foi instalada e executada corretamente em todos os dispositivos de teste, sem falhas nem problemas de desempenho.

6.2.5 Conclusões

Através da base tecnológica desenvolvida foi possível desenvolver uma aplicação funcional e facilmente adaptável a contextos reais em pouco menos de duas horas, através de poucas linhas de código. É óbvio que a *performance* e a robustez da aplicação não são as melhores possíveis, mas podem ser facilmente melhoradas.

O nível de complexidade da aplicação é bastante baixo, mas permitiu testar ao máximo as capacidades da camada padronizada já desenvolvida.

Capítulo 7

Conclusões e Trabalho Futuro

A constante evolução dos *smartphones* e o consequente surgimento de novos e melhorados dispositivos levará a uma extensa variedade de características, que irão tornar o desenvolvimento nativo muito especializado e demorado, se se quiserem aproveitar todas elas.

Diferentes resoluções ou diferentes versões do sistema operativo são algumas das razões que tornam o desenvolvimento multiplataforma a melhor solução para quem pretende atingir um público-alvo mais amplo, com um custo reduzido.

Apesar de ainda não existir uma ferramenta que permita o desenvolvimento de aplicações em simultâneo para os sistemas operativos móveis mais relevantes no mercado, existem já muitas ferramentas que permitem para os 3 principais (mas não em simultâneo), como o Titanium. Atualmente é impossível desenvolver em simultâneo para as três principais plataformas móveis, uma vez que para desenvolver para iOS é necessário programar num ambiente Apple e para programar para Windows Phone tem que se usar um ambiente Windows. Excluindo essa particularidade, já existem no mercado bastantes ferramentas que permitem desenvolver aplicações móveis multiplataforma, com um alto nível de qualidade e com um vasto número de ferramentas para auxiliarem no processo de desenvolvimento.

Uma das melhores ferramentas gratuitas para a criação de aplicações móveis multiplataforma, é o Titanium Studio. Com ela, foi possível desenvolver uma base tecnológica bastante completa, que satisfaz a maioria dos requisitos comuns às aplicações móveis e em particular com alguns requisitos próprios dos sistemas de informação geográficos.

A ferramenta Titanium, tal como a maioria das *frameworks* de desenvolvimento multiplataforma, foca-se na agilização do processo de desenvolvimento para as várias plataformas móveis. Mas, mesmo assim, o processo não é o mais eficaz e rápido possível. Dessa forma, a base tecnológica desenvolvida melhora bastante todo o processo de desenvolvimento, tornando-o ainda mais rápido e eficaz. Com algum conhecimento de JavaScript e poucas linhas de código, é possível criar aplicações para Android e iOS de uma forma mais fácil, rápida e eficaz.

7.1 Satisfação dos Objectivos

A base tecnológica tinha como principal objetivos permitir a criação de aplicações multiplataforma de um modo mais rápido e eficaz, assim como melhorar o método de obtenção, processamento e representação de informação geográfica. Para além disso, esperava-se que com a base tecnológica fosse possível criar aplicações (Android e iOS) de qualidade, eficazes, robustas e com altos níveis de desempenho e de usabilidade. Para além disso, foram adicionados alguns requisitos pela Gisgeo de forma a satisfazer as suas principais necessidades.

Todos os objetivos e requisitos necessários para o correto funcionamento da base tecnológica e das aplicações de si provenientes foram satisfatoriamente cumpridos. A base tecnológica garante assim, à partida, todos os requisitos necessários para a criação de aplicações inseridas no contexto da Gisgeo, tanto para Android como para iOS.

7.2 Trabalho Futuro

A base tecnológica oferece a possibilidade de criar aplicações Android e iOS de forma rápida e eficaz, mas obviamente, ainda há certos pormenores que poderiam ser melhorados. A base tecnológica, apesar de bastante rica e completa, ainda tem lugar para muitos aperfeiçoamentos e correções.

Um dos principais aspetos a corrigir é o desempenho das aplicações resultantes da base tecnológica. Apesar das aplicações apresentarem um alto nível de desempenho, ainda se podia aproximar mais do nível das aplicações desenvolvidas nativamente.

Outro dos aspetos a corrigir seriam os módulos de deteção de quedas e de realidade aumentada georreferenciada que, apesar de funcionais e disponíveis para inclusão nas aplicações desenvolvidas, ainda se encontram em fases bastante primitivas e com algumas deficiências.

Para além de todas as correções, deveria ser incorporada na base tecnológica a visualização de mapas usando a API da Here. Isto não foi possível durante todo o processo de desenvolvimento da base tecnológica visto que a Gisgeo, apesar de cliente da Here Maps, não tem acesso às APIs e SDKs que permitiriam a incorporação de mapas Here na base tecnológica.

Seria também importante incluir, no futuro, a possibilidade de incluir nas aplicações as notificações *push*, que não foi possível desenvolver na base tecnológica devido à inexistência de servidores desse tipo na Gisgeo.

O processo de desenvolvimento de aplicações móvies foi nitidamente melhorado com a base tecnológica, mas ainda há lugar para melhoramentos. A base tecnológica pode então ser melhorada no modo de facilitar todo o processo de desenvolvimento de aplicações móveis, assim como diminuir o código necessário para a criação de uma boa aplicação.

Por último, espera-se também que seja possível melhorar a documentação da base tecnológica de forma a facilitar o seu uso aos programadores que a irão usar no futuro.

Referências

[AC01] Wargo, John M, and Erica Sadun. 2013. Apache Cordova 3 Programming. Addison-Wesley Professional.

[CP01] Costa, Diogo. 2013. "Cross-Platform Mobile Development Using Web Technologies."

[CP02] Hartmann, Gustavo, Geoff Stead, and a DeGani. 2011. "Cross-Platform Mobile Development." Tribal, Lincoln House, The …, no. March: 1–18. https://wss.apan.org/1539/JKO/mole/Shared Documents/Cross-Platform Mobile Development.pdf.

[CP03] Sommer, Andreas, and Stephan Krusche. 2013. "Evaluation of Cross-Platform Frameworks for Mobile Applications." Proceedings of the 1st European Workshop on Mobile Engineering, 363–76. http://subs.emis.de/LNI/Proceedings/Proceedings215/363.pdf.

[IDC01] IDC. Smartphone OS Market Share, Q3 2014. http://www.idc.com/prodserv/smartphone-os-market-share.jsp, 2014. [Online; Acedido em 5 de janeiro de 2015].

[IF01] Sposaro, Frank, and Gary Tyson. 2009. "iFall: An Android Application for Fall Monitoring and Response." Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society: Engineering the Future of Biomedicine, EMBC 2009, 6119–22. doi:10.1109/IEMBS.2009.5334912.

[OD01] Oxford University Press. Oxford Dictionaries. http://www.oxforddictionaries.com/definition/english/smartphone, 2015. [Online; Acedido em 9 de janeiro de 2015].

[PG01] Natili, Giorgio. 2013. PhoneGap 3 Beginner ' S Guide. Packt Publishing.

[PG02] Shotts, Kerri. 2014. PhoneGap 3.x Mobile Application Development. Packt Publishing. [PG03] Wargo, John M. 2012. PhoneGap Essentials: Building Cross-Platform Mobile Apps.

[PNI] Cerf, V., and R. Kahn. 1974. "A Protocol for Packet Network Intercommunication." IEEE Transactions on Communications 22 (5). doi:10.1109/TCOM.1974.1092259.

[RH01] Nalwaya, Abhishek. 2011. Rhomobile Beginner's Guide. Packt Publishing.

[SASL] IETF. "Simple Authentication and Security Layer (SASL)". http://tools.ietf.org/html/rfc2222, outubro de 1997. [Online; Acedido em abril de 2015]

[SCRAM] IETF. "Salted Challenge Response Authentication Mechanism (SCRAM)". http://www.networksorcery.com/enp/rfc/rfc5802.txt, julho de 2010. [Online; Acedido em abril de 2015]

[SHA1] IETF. "US Secure Hash Algorithm 1 (SHA1)". http://tools.ietf.org/html/rfc3174, setembro de 2001. [Online; Acedido em abril de 2015]

[TCP] IETF. "TRANSMISSION CONTROL PROTOCOL". http://tools.ietf.org/html/rfc793, setembro de 1981. [Online; Acedido em abril de 2015].

[TI01] Anderson, John. 2013. Appcelerator Titanium: Up & Running. O'Reilly Media.

[TI02] Brousseau, Christian. 2013. Creating Mobile Apps with Appcelerator Titanium. Packt Publishing.

[TI03] Appcelerator. "Titanium SDK and Studio Documentation". http://docs.appcelerator.com/titanium/3.0/, 2015. [Online; Acedido em fevereiro de 2015]

[XA01] Liberty, Jesse. "Learning to Master Cross-Platform Mobile Development with Xamarin About the Author."

[XA02] Peppers, Jonathan. 2014. Xamarin Cross-Platform Application Development. Packt Publishing

[XMPP01] IBM. "Meet the Extensible Messaging and Presence Protocol (XMPP)". http://www.ibm.com/developerworks/library/x-xmppintro/, setembro de 2009. [Online; Acedido em março de 2015

[XMPP02] XMPP Wiki. "XMPP". http://wiki.xmpp.org/web/Main_Page, fevereiro de 2015. [Online; Acedido em março de 2015]

[XMPP03] IETF. “Extensible Messaging and Presence Protocol (XMPP): Core”. http://tools.ietf.org/html/rfc6120, março de 2011. [Online; Acedido em março de 2015]

[XMPP04] Saint-Andre, Peter. “XMPP is even more wonderful with WebSocket”. http://blog.andyet.com/2014/10/30/websocket, 30 de outubro de 2014. [Online; Acedido em março de 2015]

[XMPP05] XMPP Standards Foundation. “XEP-0184: Message Delivery Receipts”. http://xmpp.org/extensions/xep-0184.html,1 de março de 2011. [Online; Acedido em abril de 2015].

Anexo A: Comunicação XMPP

Exemplo de comunicação XMPP:

```
<!--SEND-->
<stream:stream to='im.apinc.org' xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams' version='1.0'>

<!--RECEIVED-->
<?xml version='1.0'?>
<stream:stream xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams' id='4031981648'
     from='im.apinc.org' version='1.0' xml:lang='fr'>
<stream:features>
    <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
    <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
        <mechanism>PLAIN</mechanism>
        <mechanism>DIGEST-MD5</mechanism>
        <mechanism>SCRAM-SHA-1</mechanism>
    </mechanisms>
    <register xmlns='http://jabber.org/features/iq-register'/>
</stream:features>

<!--SEND-->
<auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl' mechanism='DIGEST-
     MD5'/>

<!--RECEIVED-->
<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     bm9uY2U9IjI2MTIyNDk2NzYiLHFvcD0iYXV0aCIsY2hhcnNldD11dGYtOCxhbGdvcm
     l0aG09bWQ1LXNlc3M=
</challenge>

<!--SEND-->
<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     dXNlcm5hbWU9InJ1YmVuY2NyZWlzIixyZWFsbT0iaW0uYXBpbmMub3JnIixub25jZT
     0iMjYxMjI0OTY3NiIsY25vbmNlPSJZbk5XMnJuQWx5SHZ0TiIsbmM9IjAwMDAwMDAx
     Iixxb3A9YXV0aCxkaWdlc3QtdXJpPSJ4bXBwL2ltLmFwaW5jLm9yZyIscmVzcG9uc2
     U9ImY5ZDk2OTYzM2FkM2VjYTRjYjMwZjEzODhiMGZjMWQ2IixjaGFyc2V0PSJ1dGYt
     OCI=
</response>

<!--RECEIVED-->
<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
    cnNwYXV0aD1iODE1OTBlZTVkYjBlOTIyM2UzM2I0NWI3OTVkN2M3Mg==
</challenge>

<!--SEND-->
<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

<!--RECEIVED-->
<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

```
<!--SEND-->
<stream:stream xmlns:stream='http://etherx.jabber.org/streams'
     xmlns='jabber:client' to='im.apinc.org' version='1.0'>

<!--RECEIVED-->
<?xml version='1.0'?>
<stream:stream xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams' id='1357729393'
     from='im.apinc.org' version='1.0' xml:lang='fr'>
<stream:features>
    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>
    <session xmlns='urn:ietf:params:xml:ns:xmpp-session'/>
    <register xmlns='http://jabber.org/features/iq-register'/>
</stream:features>

<!--SEND-->
<iq xmlns="" type="set" id="bind_1">
    <bind xmlns="urn:ietf:params:xml:ns:xmpp-bind">
        <resource/>
    </bind>
</iq>

<!--RECEIVED-->
<iq id='bind_1' type='result'>
    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
        <jid>rubenccreis@im.apinc.org/68298798414301296583621O7</jid>
    </bind>
</iq>

<!--SEND-->
<iq xmlns="" type="set" id="bind_1">
    <bind xmlns="urn:ietf:params:xml:ns:xmpp-bind">
        <resource/>
    </bind>
</iq>
<iq xmlns="" type="set" id="sess_1">
    <session xmlns="urn:ietf:params:xml:ns:xmpp-session"/>
</iq>

<!--RECEIVED-->
<iq type='result' id='sess_1'>
    <session xmlns='urn:ietf:params:xml:ns:xmpp-session'/>
</iq>

<!--SEND-->
<presence xmlns=""/>
<message xmlns="" to="gisgeo@pandion.im">
    <body>Ola!</body>
</message>

<!--RECEIVED-->
<presence from='rubenccreis@im.apinc.org/68298798414301296583621O7'
     to='rubenccreis@im.apinc.org/68298798414301296583621O7'/>
<presence from='rubenccreis@im.apinc.org/Madrugadas-iMac'
     to='rubenccreis@im.apinc.org/68298798414301296583621O7'><c
     xmlns='http://jabber.org/protocol/caps' node='http://pidgin.im/'
     hash='sha-1' ver='VUFD6HcFmUT2NxJkBGCiKlZnS3M='/><x xmlns='vcard-
     temp:x:update'><photo>30200ffa81a58e252867022902b13e64a81c7da7</ph
```

```
      oto></x><delay xmlns='urn:xmpp:delay'
      from='rubenccreis@im.apinc.org/Madrugadas-iMac' stamp='2015-04-
      27T10:09:45Z'></delay><x xmlns='jabber:x:delay'
      stamp='20150427T10:09:45'/></presence>
<message from='gisgeo@pandion.im/Madrugadas-iMac'
      to='rubenccreis@im.apinc.org/68298798414301296583621O7'
      type='chat' id='purpleecf8e3d9'><active
      xmlns='http://jabber.org/protocol/chatstates'/><body>Ola!</body></
      message>

<!--SEND-->
</stream:stream>

<!--RECEIVED-->
</stream:stream>
```

Anexo B: Suporte Android L

Suporte ao Android L:

Barra de navegação colorida

Date Picker colorido