Mestrado em Engenharia Informática
Estágio
Relatório Final

iPhone – OOBIAN Living Knowledge App Development

Filipe Torres Alves da Mota
fmota@student.dei.uc.pt

Orientadores:
Miguel Grade
Paulo Rupino Cunha

Data: 14 de Novembro de 2012

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Agradecimentos

O estágio curricular foi uma experiencia bastante valiosa que me obrigou a contactar com uma nova realidade e que me fez aprender bastante, será uma experiência que marcará o meu futuro profissional. Assim, não posso deixar de agradecer às entidades/pessoas que me ajudaram e continuam a ajudar a passar esta importante e marcante fase da minha vida.

A minha primeira palavra é dirigida à empresa Maisis – Sistemas de Informação, Lda e a todos os seus colaborados que me acolheram e me ajudaram na integração.

Gostaria de expressar também o meu reconhecimento ao meu orientador, o Engenheiro Miguel Grade, que me acompanhou e ajudou desde que iniciei o trabalho na Maisis. Agradeço também ao Engenheiro Filipe Augusto que sempre se mostrou disponível para me assistir em algum problema ou dúvida que pudesse ter. O meu muito obrigado a todos os colegas da equipa OOBIAN que sempre me fizeram sentir como parte da equipa.

Quero também deixar um agradecimento ao meu orientador do departamento, o Professor Doutor Paulo Rupino da Cunha pela disponibilidade demonstrada, pelo acompanhamento e preciosas indicações.

Por fim quero agradecer à minha família e amigos, que sempre me apoiaram e motivaram ao longo destes anos do meu percurso académico. Sem eles nunca teria sido possível chegar a este ponto.

Resumo

Este documento descreve o trabalho realizado e respetivas metodologias utilizadas pelo autor na empresa Maisis – Sistemas de Informação, Lda., no contexto da disciplina Dissertação/Estágio do Mestrado em Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra. O estagiário foi integrado no projeto OOBIAN com o objetivo de desenvolver uma aplicação cliente para dispositivos iOS.

O OOBIAN é uma ferramenta de gestão de informação e conhecimento, desenvolvida pela Maisis, que tem como objetivo organizar a informação dispersa de uma empresa, presente em diferentes repositórios e bases de dados. Com esta informação organizada e estruturada, o OOBIAN permite efetuar pesquisas eficazes e realizar uma navegação na estrutura de conhecimento, tornando a informação numa fonte valiosa de conhecimento de negócio.

A plataforma OOBIAN apresenta uma arquitetura cliente-servidor e conta atualmente com um cliente web, o OOBIAN Insight. O projeto proposto pela Maisis (cliente OOBIAN iOS) surgiu devido à necessidade de permitir a ubiquidade da plataforma e da consulta da informação estruturada, criando assim uma plataforma mais rica e com mais potencial comercial.

Como será explicado neste documento, a aplicação desenvolvida será um produto *trimmed-down* de funcionalidades existentes no cliente web. Haverá aspetos do cliente web que não estarão presentes no cliente iOS, devido às restrições típicas deste tipo de dispositivos, mas serão também aproveitadas as características dos mesmos para desenvolver novas funcionalidades.

Palavras-Chave

iOS, iPhone, OOBIAN, Protobuf, REST, Smartphone.

Índice

Indice de Figuras

Capítulo 1: Introdução

1.1. Âmbito do estágio

O presente estágio foi desenvolvido no âmbito da disciplina de Dissertação/Estágio do Mestrado em Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, sob a orientação Professor Paulo Rupino da Cunha. O estágio teve lugar nas instalações da empresa MAISIS – Sistemas de Informação, Lda., sedeada em Aveiro. Na empresa, a orientação do estágio ficou a cargo do Eng. Miguel Grade.

O estágio teve início no dia 1 de Setembro de 2011 e terminou no dia 30 de Junho de 2012. Decorreu num período laboral de 40 horas semanais.

O trabalho desenvolvido e apresentado neste documento insere-se no contexto de um sistema de gestão de informação desenvolvido na Maisis, denominado OOBIAN, e que constitui um dos seus produtos.

1.2. Motivação

Nesta era de informação que atravessamos, damos uma importância fulcral ao conhecimento, a informação que se possui sobre um determinado domínio pode ser vital para uma dada organização. No entanto, o aumento do volume de informação existente traz novos desafios relacionados com a gestão e organização da mesma. Se dentro de elevados volumes de informação não for relativamente fácil encontrar, por exemplo, um determinado artigo, esta deixa de ser um benefício e passa a ser um desperdício de espaço.

É neste contexto que surge o OOBIAN, um produto empresarial desenvolvido pela Maisis, que visa a organização e estruturação de informação, essencial para o bom funcionamento de qualquer empresa ou organização. O OOBIAN recolhe toda a informação dispersa da empresa, armazenada em diversos repositórios e bases de dados e organiza-a no seu servidor. A informação da organização é então interpretada e relacionada com base numa análise contextual/semântica, recorrendo a ontologias, tornando assim a consulta e pesquisa de conteúdos simples e intuitiva [1].

Com a rápida e impressionante evolução dos dispositivos móveis que temos vivido nos últimos anos, a ubiquidade da informação passou a desempenhar um papel fundamental, o *desktop* e o *notebook* estão a ser rapidamente substituídos pelo *smartphone* e o *tablet* na consulta de informação [2].

À semelhança do que se tem vindo a passar com diversos serviços e aplicações surge a necessidade de adaptar uma ferramenta como o OOBIAN aos dispositivos móveis. Sendo o OOBIAN um produto empresarial, é imprescindível facultar a sua utilização em qualquer lugar, bastando apenas para o efeito uma ligação à internet.

É nesta conjuntura que surge a ideia de desenvolver aplicações cliente do OOBIAN para dispositivos móveis. A aplicação a desenvolver durante o estágio será um cliente OOBIAN para dispositivos móveis iOS.

De seguida, e para uma contextualização de todo o ambiente, será apresentada a Maisis – Sistemas de Informação, Lda. e o OOBIAN.

1.3. Maisis – Sistemas de Informação, Lda.

A empresa Maisis – Sistemas de Informação, Lda. [3], foi constituída em 28 de Dezembro de 1994, tendo, originalmente, como principal atividade, a prestação de serviços à PT Inovação. Situada em Aveiro, é também um dos associados fundadores da Inova-Ria – um pólo emergente de empresas tecnológicas, e tem já uma forte tradição em desenvolvimento de *software* de qualidade.

Atualmente a atividade da Maisis centra-se em atividades como:

- Desenvolvimento e integração de sistemas para gestão de redes de telecomunicações;
- Personalização de plataformas integradas de gestão de redes;
- Desenvolvimento de soluções de bases de dados baseadas em Tecnologia Web;

- Desenvolvimento de sistemas de informação complexos;
- Prestação de serviços de engenharia de software;

A Maisis conta com uma equipa de consultores que acompanha o projeto desde a conceção até à sua exploração. Possui uma vasta experiência em desenvolvimento de software, abrangendo diferentes ambientes, arquiteturas, técnicas, sistemas operativos, bem como diversas linguagens de programação (Java, C/C++, C#, .NET, JSP, ASP, EJB, Struts, Hibernate, J2EE).

O *software* desenvolvido pela Maisis pode ser encontrado em produtos da PTInovação [4], em diversas operadoras de telecomunicações nacionais e estrangeiras, e em empresas de áreas tão distintas como comércio, indústria, serviços, administração pública ou tráfego rodoviário.

1.4. A plataforma de gestão de informação OOBIAN

O OOBIAN [1] é uma ferramenta de gestão de informação e conhecimento, desenvolvida pela Maisis, que permite efetuar pesquisas e realizar uma navegação pelos conteúdos de uma organização. O seu objetivo principal é organizar a informação dispersa de uma organização, presente em diferentes repositórios e bases de dados. Efetuando pesquisas eficazes por todos os conteúdos dispersos, dentro e fora da organização, o OOBIAN permite tirar o máximo proveito da informação não estruturada, transformando-a numa fonte valiosa de conhecimento de negócio [1]. O OOBIAN serve-se de ontologias para interpretar e relacionar um conjunto disperso de informação com base em análises contextuais e semânticas.

O produto atualmente comercializado apresenta uma arquitetura Cliente/Servidor, com um cliente WEB, o OOBIAN Insight. O OOBIAN tem uma fácil integração com todos os tipos de bases de dados e sistemas empresariais e apresenta uma interface de *Web Service* para permitir esta fácil integração. Disponibiliza igualmente diversos conectores e ADD-ONs para produtos Microsoft e aplicações *open-source*, assim como suporte para todas as recomendações W3C Semantic WEB [3], como OWL e RDFS.

O OOBIAN Insight, o cliente atualmente usado, permite o acesso ao servidor OOBIAN através de qualquer *browser* em qualquer máquina. O cliente consiste em cinco módulos base: o

OOBIAN Drill, o Rich Knowledge Reader, o Xplorer, o OOBIAN Maps e o OOBIAN Document Viewer.

No módulo OOBIAN Drill, Figura 1, é possível realizar uma navegação no repositório do servidor OOBIAN usando uma representação gráfica dos conteúdos e instâncias sobre a forma de nós e das suas relações sobre a forma de ligações entre os nós.

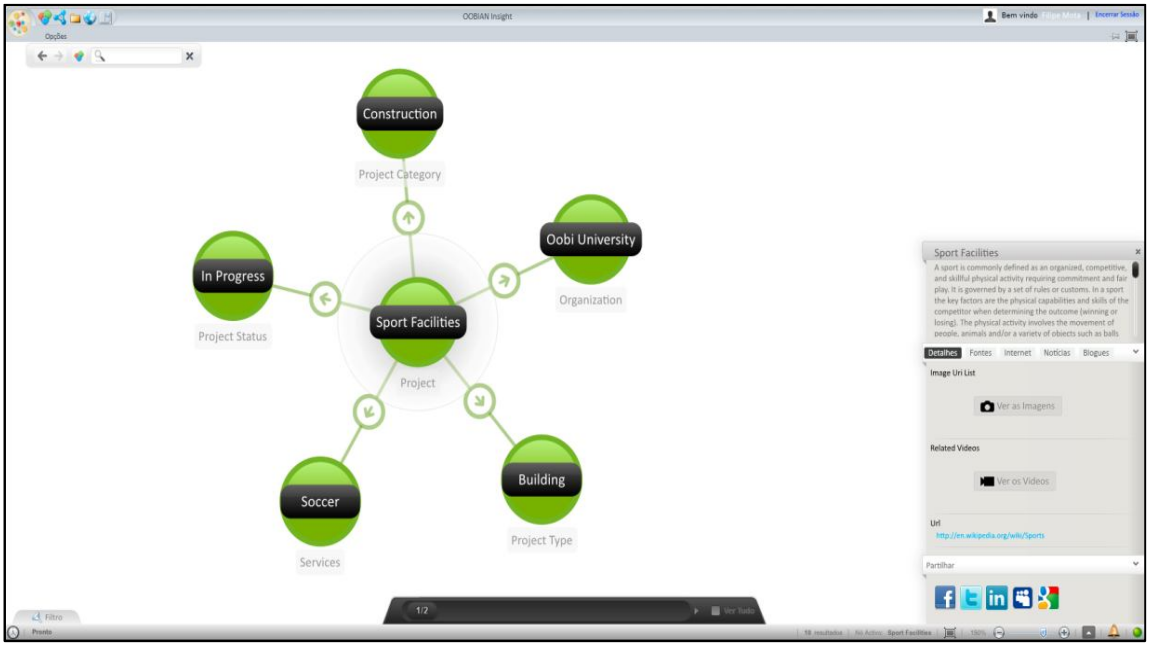


Figura 1 - OOBIAN Drill

O Rich Knowledge Reader, Figura 2, tem como objetivo proporcionar ao utilizador uma navegação no repositório do servidor OOBIAN num esquema de árvore usando mecanismos de *drill up* e *drill down* para explorar os conteúdos presentes nos diferentes níveis de forma simples e intuitiva.

No módulo Xplorer é feita a gestão de ficheiros no repositório. É possível pesquisar ficheiros e filtrar os resultados consoante o tipo de ficheiro desejado. No Xplorer é igualmente possível adicionar ficheiros ao repositório. No caso de ser necessário visualizar o conteúdo dos ficheiros, é usado o OOBIAN Document Viewer, que permite a visualização de ficheiros texto, apresentações e folhas de cálculo sem o uso de outra aplicação. Por fim, no módulo OOBIAN Maps é possível consultar os conteúdos que tenham características geolocalizáveis num mapa.

Outras funcionalidades do OOBIAN Insight são a pesquisa avançada de conteúdos, com possibilidade de desambiguação e filtragem; a partilha de conteúdos em redes sociais; a subscrição de notificações de conteúdos; a possibilidade de escolha de conteúdos favoritos; e a visualização de conteúdos multimédia sem recurso a outras aplicações.

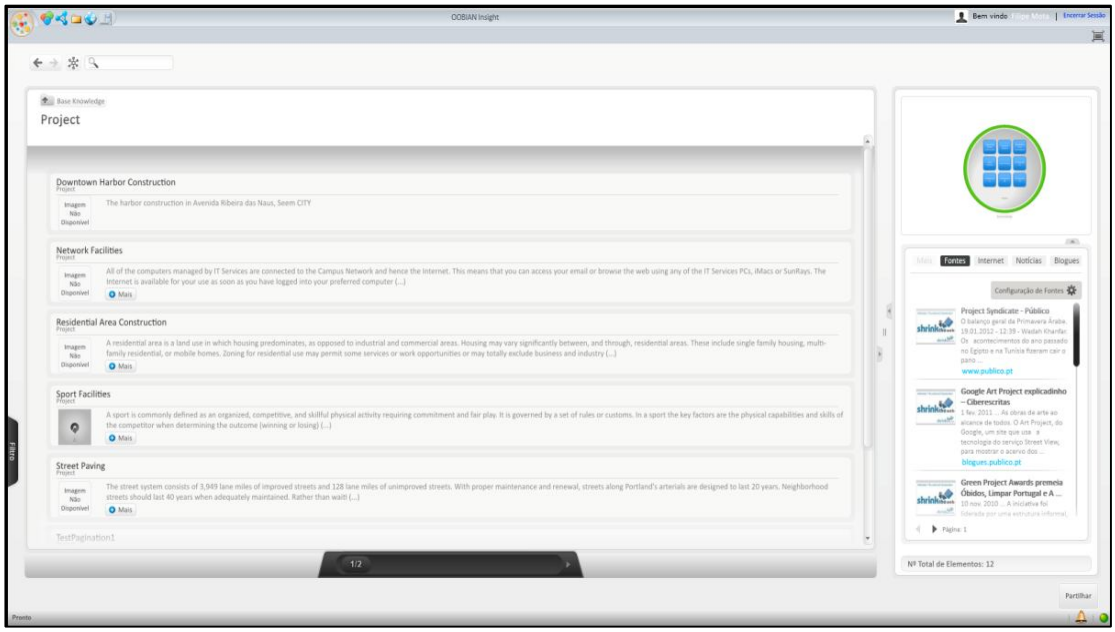


Figura 2 - Rich Knowledge Reader

1.5. Estrutura do relatório

O presente relatório é composto por dois volumes. No primeiro volume é apresentado o relatório propriamente dito e no segundo volume são apresentados os Anexos que complementam o relatório e que considerámos relevantes para uma melhor compreensão do trabalho desenvolvido.

Este documento, o primeiro volume do relatório, divide-se em sete diferentes capítulos:

Capitulo 1 – Introdução: Neste capítulo é feita a introdução ao documento. É apresentado o âmbito do estágio e a motivação do mesmo. É apresentada a empresa acolhedora (Maisis) e o seu produto OOBIAN, de modo a melhor entender o âmbito da aplicação que será desenvolvida. É ainda apresentada a estrutura do documento e os anexos presentes.

Capitulo 2 – Conceitos e tecnologias: Neste capítulo são apresentados os conceitos e tecnologias usadas durante o estágio. É ainda descrita a comunicação entre o servidor e o cliente móvel.

Capitulo 3 – A proposta: Neste capítulo é apresentada a proposta de estágio. É apresentada a necessidade da aplicação e as principais funcionalidades a serem implementadas.

Capitulo 4 – Implementação: Neste capítulo é apresentado todo o processo de implementação da aplicação. É descrito o processo de desenvolvimento e de escolha da ferramenta de desenvolvimento. De seguida são apresentados os Casos de Uso, a prototipagem realizada e descritos os requisitos, a arquitetura e a especificação funcional e lógica da aplicação.

Capitulo 5 – Testes: Neste capítulo é apresentada a metodologia de testes e descritos os testes efetuados.

Capitulo 6 – Análise ao Planeamento: Neste capítulo será apresentado o planeamento elaborado inicialmente e as diferenças para a calendarização real das tarefas.

Capitulo 7 – Conclusão: Neste capítulo serão apresentadas as conclusões. São analisados os principais desafios do projeto, descrito o produto final e o possível trabalho futuro, e por fim apresentadas as considerações finais.

O relatório inclui ainda os seguintes anexos, presentes no segundo volume:

Anexo A – Estudo de ferramentas de desenvolvimento iOS: Documento onde é feito o estudo da ferramenta de desenvolvimento a usar no projeto.

Anexo B – Prototipagem de papel: Documento onde são apresentados scans da primeira fase de prototipagem realizada, a prototipagem em papel.

Anexo C – Documento de Requisitos: Documento de requisitos do projeto. Além de apresentados e descritos os requisitos, é feito um estudo da interação entre o utilizador e o dispositivo, descrita a prototipagem feita e ainda apresentados os Casos de Uso.

Anexo D – Documento de Especificação: Documento de especificação do projeto. São listados os requisitos do projeto e apresentada a arquitetura do sistema.

Anexo E – Protótipo Funcional: Documento onde são apresentados *screenshots* do protótipo funcional.

Anexo F – Plano de Testes: Documento onde é descrito o plano de testes do projeto. É feita uma descrição do sistema e descritos os casos de teste.

1.6. Sumário

Neste primeiro capítulo foi feita uma apresentação de todo o documento. Foi descrito inicialmente o âmbito do estágio e a motivação do mesmo. De seguida, foi apresentada a empresa Maisis – Sistemas de Informação, Lda. e o seu produto OOBIAN, para melhor compreender a motivação do projeto. Por fim, foi descrita a estrutura do relatório, composto por este documento e os seus anexos.

No capítulo seguinte serão apresentadas as tecnologias estudadas e usadas durante este estágio.

Capítulo 2:
Conceitos e tecnologias

2.1. O sistema operativo iOS

iOS [5], anteriormente conhecido como iPhone OS, é um sistema operativo desenvolvido pela Apple [6] para dispositivos móveis. Foi, originalmente, desenvolvido exclusivamente para o iPhone mas, atualmente, é utilizado também no iPod Touch, iPad e AppleTV.

O iOS deriva do Mac OS X [7], também desenvolvido pela Apple, com o qual partilha a base do sistema operativo Darwin [8] e é, assim, também baseado em Unix. No iOS existem quatro camadas de abstração, são elas a camada Core OS, a camada Core Services, a camada Média e a camada Cocoa Touch. A versão atual do sistema operativo é a 6.0.1, lançada na mesma data do *iPhone 5* [9].

A interface do iOS é baseada no conceito de manipulação direta, usando gestos multi-toque. A interação com o sistema operativo inclui gestos como *swipe*, *tap*, *pinch*, entre outros, que possuem definições específicas no contexto do iOS e na sua interface multi-toque. Os acelerómetros internos dos dispositivos iOS são usados por algumas aplicações para responder aos movimentos e rotações. As aplicações iOS, tal como as aplicações para sistemas Mac OS X, são escritas na linguagem de programação *Objective-C*.

2.2. Ferramenta de desenvolvimento Xcode

O Xcode [10] é um conjunto de ferramentas destinado ao desenvolvimento de aplicações para iOS e Mac OS. Sendo a forma nativa de desenvolvimento para iOS, suporta todas as APIs nativas e apenas corre em sistemas Mac OS. A linguagem usada no desenvolvimento é o Objective-C. Neste conjunto de ferramentas estão incluídos um ambiente

de programação, um simulador de iOS, um Interface Builder, o iOS SDK e outros frameworks como Cocoa Touch e Foundation [11].

Além das já incluídas, possibilita a fácil integração de outros *frameworks*, algumas delas estudadas na elaboração deste relatório, como a PhoneGap [12], a Hockey [13] (ferramenta que torna possível o suporte de redes *ad hoc*), entre outras. Outra das vantagens é a grande comunidade de utilizadores desta ferramenta, o que permite a rápida descoberta e correção de bugs e problemas, assim como o bom suporte e documentação disponíveis *online*.

Uma das desvantagens é o facto de a linguagem de programação utilizada ser uma linguagem menos comum, Objective-C. Esta linguagem é usada essencialmente para sistemas operativos iOS e Mac OS. Se o programador nunca tiver tido contacto com esta linguagem, terá sempre que ter o esforço de a aprender, o que pode reduzir a produtividade.

2.3. Comunicação com o servidor OOBIAN

2.3.1. WebServices - REST

As várias aplicações cliente OOBIAN (Web, iOS e Android) necessitam de estar em frequente comunicação com o servidor. Esta comunicação é feita usando WebServices, existindo nesta área duas opções, REST e SOAP. O servidor OOBIAN utiliza para efetuar as suas comunicações com as aplicações cliente o método REST.

O termo Representational State Transfer (REST), foi usado pela primeira vez por Roy T. Fielding em 2000, quando o mesmo definiu REST como um estilo arquitetural para sistemas distribuídos hipermédia [14]. Às arquiteturas que seguem este estilo foi dado o nome de arquiteturas RESTful.

O REST fornece um conjunto de restrições de arquitetura que, quando aplicado como um todo, enfatiza a escalabilidade de interações entre componentes, a generalidade de interfaces, a implementação independente de componentes e o uso de componentes intermediários de modo a reduzir a latência da interação e o reforço da segurança.

O maior exemplo de um sistema com o estilo REST é a World Wide Web. O REST exemplifica como a arquitetura Web surgiu, caracterizando e restringindo as interações de quatro componentes chave da Web, servidores, *gateways*, *proxies* e clientes, sem impor

limitações sobre os participantes individuais. Assim, o que o REST faz essencialmente é a gestão do comportamento dos participantes [15].

O REST consiste basicamente em clientes e servidores, os clientes fazem pedidos aos servidores, os servidores processam os pedidos e devolvem as respostas adequadas. Os pedidos e respostas são construídos tendo em conta a transferência de representações de recursos. Um recurso é essencialmente qualquer conceito coerente com significado que possa ser acedido, enquanto a representação de um recurso é tipicamente um documento que engloba o estado atual de um recurso. Um cliente inicia o envio de pedidos quando se encontra pronto para realizar a transição para um novo estado. Enquanto um ou mais pedidos estão pendentes, considera-se que o cliente se encontra em transição. A representação de cada estado da aplicação contem atalhos que podem ser usados da próxima vez que o cliente escolher iniciar uma nova transição de estados [16].

Apesar de deixar a implementação dos componentes individuais à escolha de cada um, o estilo REST indica seis restrições arquiteturais [17]. A primeira está relacionada com o aspeto cliente-servidor, onde uma interface uniforme deve separar os clientes de servidores. Isto cria uma separação de interesses onde, por exemplo, os clientes não se preocupam com o armazenamento de dados que fica a cargo dos servidores e que assim permite melhorar a portabilidade do código do cliente. Por sua vez, os servidores não estão preocupados com a interface ou estado do cliente, o que proporciona servidores mais simples e escaláveis. Esta restrição permite que servidores e clientes sejam substituídos ou desenvolvidos de forma independente, desde que a interface que os liga não seja alterada. A comunicação cliente-servidor é igualmente limitada pelo facto de nenhum contexto do cliente ser armazenado no servidor entre pedidos. Cada pedido feito por um cliente contem toda a informação necessária para o mesmo ser tratado pelo servidor e todos os estados da sessão são tratados no cliente. O servidor pode até ter a capacidade de tratar os estados, o que a segunda restrição indica é que o estado do servidor seja endereçável por um URL como outro qualquer recurso. Isto não só torna os servidores mais visíveis para monitoramento, mas também os torna mais confiáveis face a falhas parciais de rede, e ainda melhora a sua escalabilidade. Tal como acontece na World Wide Web, os clientes podem armazenar em cache as respostas aos seus pedidos. Assim, diz a terceira restrição que as respostas devem definir, implícita ou explicitamente, se podem ou não ser armazenadas na cache dos clientes. Esta restrição visa impedir que clientes reutilizem dados obsoletos ou inadequados para resposta a pedidos adicionais. Uma cache bem gerida elimina parcialmente ou mesmo completamente algumas interações cliente-servidor, melhorando assim a escalabilidade e o desempenho. Um cliente não consegue normalmente perceber se está ligado diretamente ao servidor ou a algum intermediário. A quarta restrição do REST aponta para um sistema por camadas. Servidores intermédios podem melhorar a escalabilidade do sistema, permitindo um balanceamento de carga dos pedidos recebidos e fornecendo caches compartilhadas. Os servidores deverão ser capazes de prolongar temporariamente ou personalizar a funcionalidade de um cliente, transferindo para ele a lógica da aplicação para que possa executar. Esta é a quinta restrição e a única opcional. A última

restrição é a existência de uma interface uniforme. Como já foi referido, a existência de uma interface uniforme entre clientes e servidores é necessária. Esta interface simplifica a arquitetura e permite que tanto cliente como servidor evoluam de forma independente. Para um serviço ser considerado estritamente RESTful terá que cumprir todas as restrições não opcionais.

O REST foi originalmente descrito no contexto HTTP mas não se encontra limitado a este protocolo. As arquiteturas RESTful podem ser baseadas em qualquer protocolo da camada de aplicação, desde que este tenha um vocabulário rico e uniforme para aplicações baseadas na transição de estados.

2.3.2. Serialização de dados

2.3.2.1. Caracterização do cenário

Tratando-se este um projeto para o desenvolvimento de uma aplicação para um dispositivo móvel, interessa tomar atenção às características e restrições deste tipo de dispositivos.

Durante o uso da aplicação haverá uma constante troca de dados com o servidor, razão pela qual a aplicação necessita de uma ligação à internet constante, quer seja via rede celular ou via rede wireless. Este aspecto torna necessário garantir que os dados transmitidos pela e para a aplicação tenham o tamanho mais reduzido possível.

Outra restrição deste tipo de dispositivos é a capacidade de bateria e processamento, logo importa garantir que os processos de serialização e deserialização consomem o menor número de recursos possível.

Com base nas restrições apresentadas, é importante proceder a um estudo sobre os diferentes métodos de serialização de dados com o objetivo de encontrar o método mais vantajoso para a transmissão de dados da aplicação cliente para o servidor.

2.3.2.2. Métodos analisados

As primeiras opções analisadas foram os métodos de serialização nativos na linguagem de programação Objective-C, são eles JSON e Property Lists. Foram analisadas também outras implementações de JSON disponíveis na linguagem de programação usada, entre elas JSONKit [18], TouchJSON [19], NextiveJSON [20] e SBJSON [21].

Entre as várias implementações de JSON, alguma pesquisa permitiu perceber que a implementação JSONKit é a que apresenta uma melhor performance. Podemos observar isto no gráfico presente na seguinte figura:

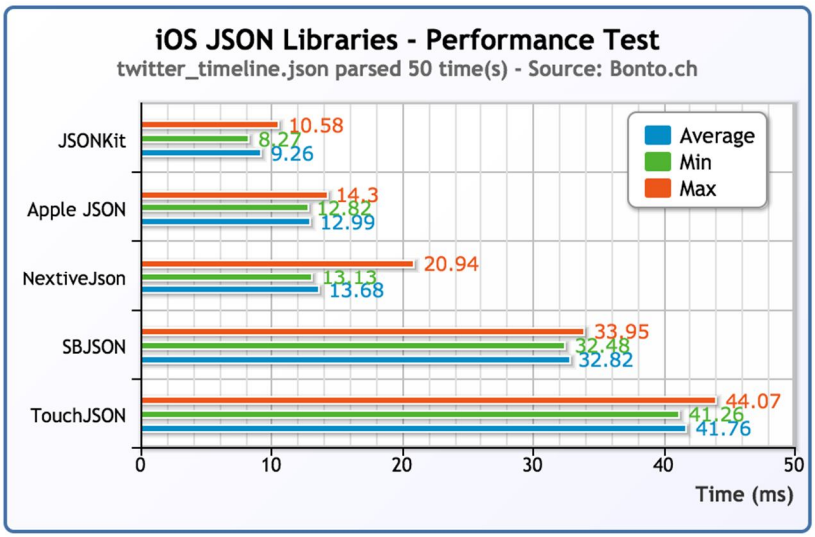


Figura 3 – Teste de performance de várias implementações JSON num iPhone 4S com o iOS 5.0.1 [22]

Além destes métodos foram ainda considerados e analisados, os métodos BSON [23], Protocol Buffers [24] e XML [25]. Numa aplicação para dispositivos móveis, um aspeto mais crítico que as velocidades de serialização e deserialização, é o tamanho dos dados serializados. Podendo existir restrições de largura de banda e de trafego nas ligações, será importante garantir que os dados serializados transmitidos pelo servidor à aplicação cliente têm o tamanho mais reduzido possível.

O servidor OOBIAN encontra-se implementado em JAVA. Podemos concluir, observando o gráfico da Figura 4, que para esta linguagem o tamanho dos dados serializados usando a tecnologia Protocol Buffers é consideravelmente mais pequeno que o tamanho da

maioria das outras tecnologias. Este gráfico está inserido num estudo que compara os diferentes métodos de serialização, tendo em conta as velocidades de serialização e deserialização, assim como o tamanho dos dados serializados [26].

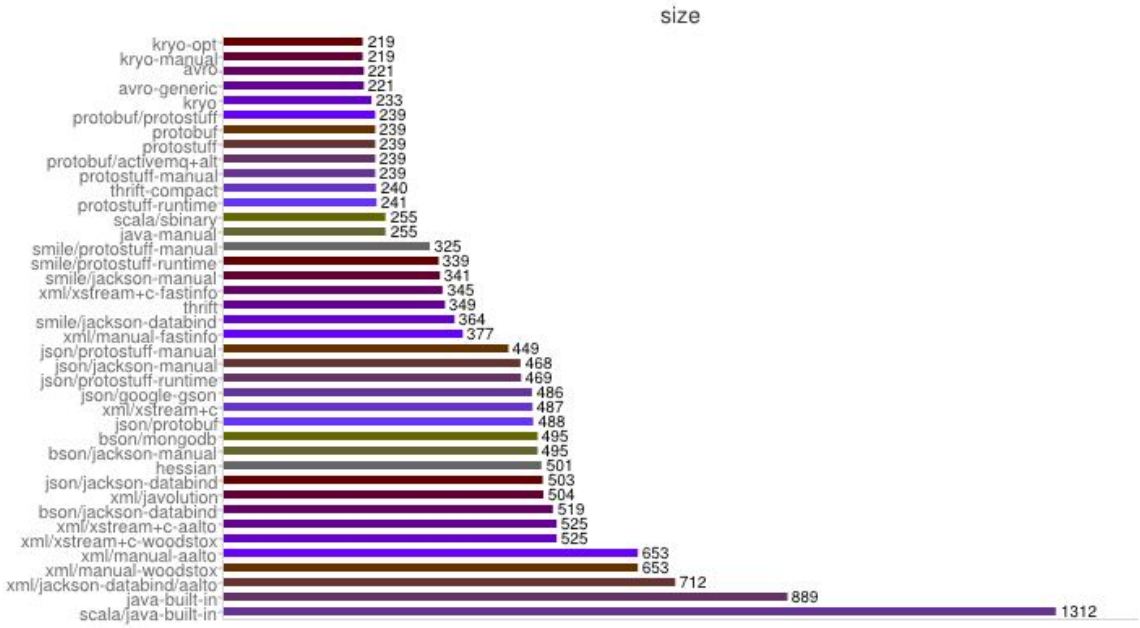


Figura 4 – Tamanho dos dados serializados por diferentes métodos de serialização

Na escolha do formato de serialização utilizado nos dados transmitidos entre servidor e cliente teve influência o conjunto de formatos já usados nos serviços do servidor OOBIAN, JSON e Protocol Buffers. Analisando os aspectos de cada um dos formatos, foi dada pela empresa preferência ao formato Protocol Buffers, já que é método utilizado pelos clientes Web e Android.

2.3.2.3. Método escolhido - Protobuf

Protocol Buffers, também conhecido como Protobuf, é um formato de serialização desenvolvido pela Google, extensível, independente da linguagem de programação e plataforma. A implementação original foi feita nas linguagens C++, Java e Python, mas atualmente implementações em variadas linguagens estão disponíveis ou em desenvolvimento. Os principais objetivos do Protobuf são a simplicidade e a performance e são usados amplamente pela Google para armazenamento e transmissão de informação estruturada.

Nos Protocol Buffers as estruturas de dados (chamadas de “mensagens”) e serviços são definidos nos ficheiros Proto Definition (.proto), os quais são compilados usando o compilador protoc. Esta compilação gera código que corresponde com as definições das mensagens. Nos

ficheiros gerados (.pb.h e .pb.m no caso do Objective-C) serão definidas as classes, em Objective-C neste caso, onde são definidos as mensagens e serviços presentes no ficheiro Proto Definition correspondente.

2.4. Sumário

Neste capítulo foram apresentadas as tecnologias usadas no desenvolvimento deste projecto. Foi apresentado o sistema operativo iOS, sistema operativo usado nos dispositivos móveis a que se destina a aplicação. De seguida foi apresentado o ambiente de desenvolvimento Xcode. O processo de escolha deste ambiente de desenvolvimento será descrito mais à frente, no capítulo 4. Foram também apresentadas as tecnologias usadas na comunicação do servidor com a aplicação.

No capítulo seguinte será apresentada a proposta, onde será descrita a necessidade da aplicação a desenvolver e as principais funcionalidades esperadas.

Capítulo 3:
A proposta

3.1. A necessidade do cliente iOS

Atualmente o OOBIAN, ferramenta de gestão de informação e conhecimento já apresentada, conta com um cliente Web, o OOBIAN Insight. Este cliente pode ser utilizado em qualquer máquina e em qualquer *browser*, desde que estes tenham suporte para a tecnologia Siverlight. O OOBIAN conta igualmente com diversos plug-ins para, entre outros, Microsoft Office, Sharepoint e Alfresco.

Surge no entanto a necessidade de expandir o seu alcance aos dispositivos móveis. Estes dispositivos têm-se tornado cada vez mais uma ferramenta empresarial e rapidamente *smartphones* e *tablets* estão a substituir desktops e *notebooks*. Torna-se fulcral proporcionar a ubiquidade dos dados presentes nas estruturas de conhecimento do OOBIAN e possibilitar a utilização da ferramenta em dispositivos móveis iOS, iPhone, iPad e iPod Touch.

Assim, a aplicação a desenvolver será uma aplicação completamente nova, adaptada às necessidades e restrições dos dispositivos móveis iOS. A sua interface será baseada no já existente cliente Web para facilitar a aprendizagem por parte de utilizadores já familiarizados com o OOBIAN, havendo no entanto algumas diferenças nas funcionalidades dos diferentes clientes. O cliente Web, sendo desenvolvido com o objetivo de ser utilizado em máquinas desktop, aponta não só para a consulta de informação, mas também para o fornecimento de informação para o servidor OOBIAN. No caso dos dispositivos móveis, o seu desenvolvimento será mais virado para a consulta de informação, logo a aplicação estará assente em duas funcionalidades chave: pesquisa de informação e navegação na estrutura de conhecimento do repositório ao qual se encontra ligada.

Apesar de algumas restrições, em termos de limitações de bateria, capacidade de processamento ou tamanho do ecrã, os dispositivos móveis apresentam também algumas características que permitirão implementar funcionalidades que não se encontram presentes no cliente Web. A geolocalização permitirá ao utilizador ver os conteúdos geográficos no mapa enquanto compara com a sua posição atual, assim como permitirá pesquisar os conteúdos perto de si. Poderá também, ao consultar os detalhes de uma conteúdo na estrutura de conhecimento, aceder aos contactos para realizar uma chamada telefónica (se o dispositivo assim o permitir), enviar um e-mail ou ver a localização no mapa.

Esta aplicação será usada em conjunto com o cliente Web e as outras ferramentas OOBIAN e constituirá uma mais valia para o pacote já comercializado com a plataforma OOBIAN.

3.2. O cliente OOBIAN iOS

No desenvolvimento de aplicações para dispositivos móveis há que ter em consideração outros fatores não considerados aquando um desenvolvimento para máquinas desktop. Será importante ter em conta o processamento reduzido destes dispositivos e as suas limitações de bateria, ecrã e periféricos.

Como já foi referido, ao contrário do cliente Web, o cliente para dispositivos móveis focar-se-á essencialmente na consulta de informação, sendo as suas principais funcionalidades a pesquisa de conteúdos e a navegação na estrutura de conhecimento.

A pesquisa de informação deverá estar acessível em qualquer ponto da aplicação. Aquando a inserção de um termo a pesquisar, serão apresentadas ao utilizador sugestões de termos idênticos e, caso o utilizador não aceite nenhuma das sugestões, e se for necessário, será apresentado um ecrã de desambiguação de um termo pesquisado.

A navegação no repositório será feita usando mecanismos de *drill-down* e *drill-up*, permitindo assim uma fácil e intuitiva consulta de informação. Estarão ainda inseridas na navegação, e sempre que se justifique na aplicação, opções de filtragem de conteúdos. Esta filtragem poderá feita com base numa palavra-chave, num tipo de conteúdo ou usando nós próprios. Ao escolher um nó, que poderá ser qualquer instância do repositório, apenas serão apresentados conteúdos ligados direta ou indiretamente a esse nó.

A informação consultada, seja por via de pesquisa ou navegação, poderá estar em vários formatos, texto, mapas, imagens, vídeo, documentos etc.. A aplicação permitirá a visualização de todos os formatos possíveis de informação presente no repositório, incluindo a possibilidade descarregar documentos e visualizá-los no dispositivo na correspondente aplicação. Tirando partido da característica de geolocalização presente nos dispositivos móveis iOS, a aplicação permitirá também visualizar conteúdos geolocalizáveis em mapas e relacioná-los com a localização atual do utilizador. Será importante igualmente permitir a partilha da visualização consultada via correio eletrónico.

A aplicação deve permitir também ao utilizador adicionar conteúdos a uma lista de favoritos para um acesso mais célere aos mesmos. Igualmente deve ser possível subscrever

notificações de conteúdos para que o utilizador seja notificado assim que ocorrer alguma alteração no repositório.

Outro aspeto importante, será manter a consistência entre a interface da aplicação e a interface da aplicação OOBIAN Insight, o cliente WEB, não deixando no entanto de cumprir os padrões de *design* de aplicações para o sistema operativo iOS. Como em qualquer aplicação cliente, será fulcral assegurar a segurança das comunicações entre o cliente e o servidor e minimizar os tempos de acesso à estrutura de conhecimento presente no servidor.

3.3. Sumário

Neste capítulo foi apresentada a proposta do trabalho a desenvolver. Foi inicialmente justificada a necessidade de uma aplicação cliente para dispositivos móveis para a plataforma OOBIAN. De seguida foram descritas as principais funcionalidades a implementar na aplicação.

No próximo capítulo será apresentado e descrito o processo de desenvolvimento.

Capítulo 4: Implementação

4.1. Processo de desenvolvimento

Partindo da necessidade de dotar os dispositivos móveis de uma aplicação cliente OOBIAN, iniciou-se o desenvolvimento. Este iniciou-se em Setembro de 2011 com os primeiros estudos sobre a plataforma OOBIAN e sobre o desenvolvimento para dispositivos móveis iOS e terminou em Junho de 2012 com a conclusão dos testes da aplicação.

O primeiro ponto do desenvolvimento foi o estudo e experimentação da plataforma OOBIAN, em especial do cliente Web OOBIAN Insight, de modo a ganhar algum conhecimento do funcionamento da mesma e das suas funcionalidades.

De seguida foi feita uma análise sobre as ferramentas de desenvolvimento para dispositivos móveis iOS disponíveis no mercado. Foram analisadas várias ferramentas e avaliadas as que mais se inseriam no âmbito do projeto, às várias ferramentas de desenvolvimento de jogos para estes dispositivos, por exemplo, não foi dado grande relevo, como se pode verificar no subcapítulo *4.2. – Escolha da ferramenta de desenvolvimento*.

Escolhida a ferramenta de desenvolvimento, foi feita mais uma análise ao cliente Web de modo a fazer um levantamento de funcionalidades a implementar onde ainda foram incluídas novas funcionalidades próprias de uma aplicação para dispositivos móveis. A partir deste levantamento, foram construídos os Casos de Uso da aplicação, que serão apresentados e descritos mais à frente.

Depois de concluídos os casos de uso, foi elaborada uma versão inicial da lista de requisitos da aplicação. Iniciou-se neste momento a fase de prototipagem que permitiu simular o comportamento da aplicação e elaborar uma versão final de requisitos da aplicação. A fase de prototipagem, assim como os vários protótipos elaborados serão descritos mais à frente.

O passo seguinte foi a elaboração do documento de requisitos (ver anexo D) onde, além da listagem e descrição dos requisitos da aplicação, foram incluídos os casos de uso e a prototipagem gráfica para facilitar um melhor entendimento do funcionamento da aplicação. Foi também incluído neste documento um estudo sobre os principais gestos utilizados na interação entre o utilizador e o dispositivo.

Terminado o documento de requisitos, foi elaborado o documento de especificação (ver anexo E). Neste documento é descrita a arquitetura do sistema a ser implementando segundo uma perspetiva funcional e lógica.

Terminada a especificação e arquitetura da aplicação, foi desenvolvido um protótipo funcional, ainda sem a comunicação com o servidor implementada. Este protótipo permitiu corrigir alguns aspetos de usabilidade e *design* que tinham sido descurados.

Neste ponto, a meio do ano letivo, foi elaborado o relatório intermédio de estágio e foi feita a apresentação e avaliação do mesmo. Esta avaliação permitiu corrigir algumas práticas erradas e garantir a correta continuidade do projeto.

Foi iniciada de seguida a implementação da aplicação, que quando concluída foi submetida a uma fase de testes, descrita em detalhe no Capitulo 5.

4.2. Escolha da ferramenta de desenvolvimento

Com o aumento do uso de dispositivos móveis, também aumentou o número de programadores e de ferramentas de desenvolvimento. Apesar de a Apple disponibilizar a sua própria ferramenta para o desenvolvimento de aplicações para dispositivos móveis iOS, diversas outras ferramentas têm vindo a aparecer no mercado. Foi necessário portanto proceder a um estudo intensivo a fim de perceber qual a ferramenta de desenvolvimento que melhor se adaptaria ao projeto em causa.

Com os resultados deste estudo foi elaborado um relatório, que pode ser consultado no Anexo B. Desde estudo foi concluído que a ferramenta nativa, o Xcode, seria a melhor opção para a realização deste projeto.

As vantagens presentes no uso do Xcode são a sua interoperabilidade, já que é possível adicionar ao ambiente de desenvolvimento um número elevado de *frameworks*; a comunidade grande de utilizadores, que além de desenvolverem novos *frameworks* que adicionam novas funcionalidades, ajudam na correção de problemas e bugs, e providenciam uma grande quantidade de documentação *online*; um Interface Builder potente e eficaz, mas também a possibilidade de programar a interface; ser a forma nativa de desenvolvimento, estando sempre a par das novidades e últimos *updates* do sistema operativo dos dispositivos iOS; e o preço, reduzido ou grátis para quem possuir máquina Mac OS com o ultimo sistema operativo instalado. Como desvantagens na utilização do Xcode temos: a necessidade de ter uma máquina com sistema operativo Mac OS; o uso de um novo ambiente de desenvolvimento e de uma linguagem não muito comum, o Objective-C; e o facto de apenas ser desenvolvida uma aplicação para um dispositivo e não multi-plataforma.

A realização deste estudo foi um facto determinante para o sucesso do projeto. A empresa não possui muita experiência no desenvolvimento de aplicações para dispositivos móveis e sendo o interesse da mesma desenvolver aplicações para várias plataformas, iOS e Android, foi importante realizar um estudo detalhado para analisar não só qual seria a ferramenta mais apropriada a nível técnico mas também a nível financeiro.

4.3. Casos de Uso

Neste subcapítulo serão apresentados os casos de uso elaborados. Cada caso de uso descreve um cenário de interação entre o sistema e o utilizador.

Os casos de uso devem ser o mais claros possível para que todos os eventuais leitores possam entendê-los de igual modo. Para melhor entender o funcionamento da aplicação para cada cenário, cada caso de uso foi descrito utilizando quatro aspetos: o contexto real, os mockups gráficos, o diagrama de caso de uso e a descrição pormenorizada. No contexto real, é apresentada uma situação fictícia que poderia constituir um cenário real de utilização da aplicação. De seguida, são apresentados os mockups gráficos dos ecrãs responsáveis pelo caso de uso em questão, para melhor entender o funcionamento da aplicação e a sua interação com o utilizador. Para finalizar foi elaborado um diagrama de caso de uso e uma tabela que o descreve ao pormenor.

De seguida serão apresentados os diagramas dos casos de uso elaborados. A descrição completa de todos os casos de uso pode ser consultada no segundo capítulo do documento de requisitos (ver anexo D). Neste documento será apresentado, a título de exemplo a descrição completa do caso de uso Mapas.

O primeiro diagrama de casos de uso apresentado é o Login, Figura 5, responsável pela autenticação do utilizador no sistema. Fazem parte dele dois casos de uso, Login e Logout.

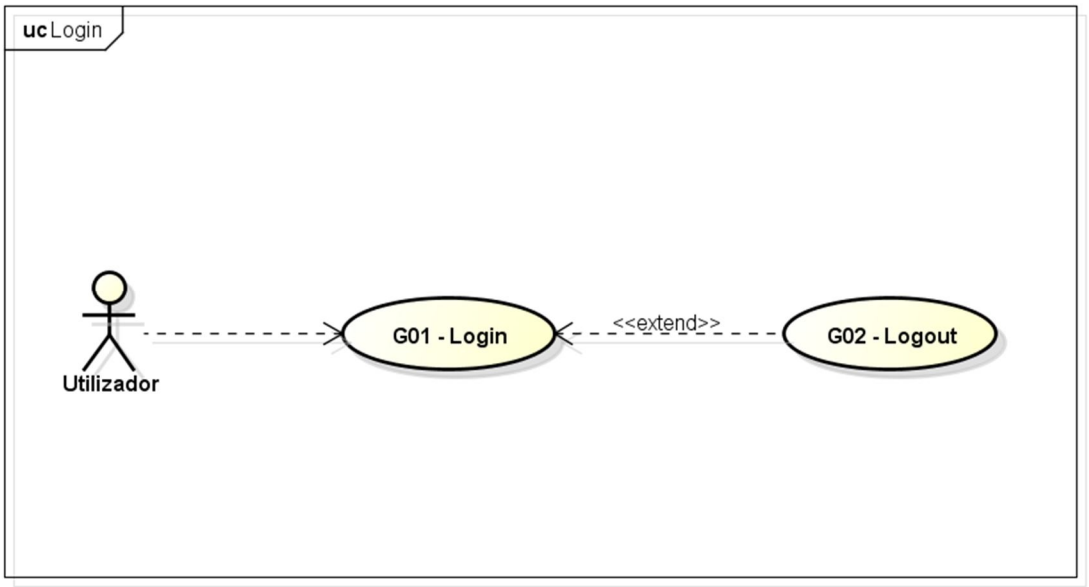


Figura 5 – Diagrama de casos de uso Login

De seguida temos o diagrama de casos de uso do módulo Rich Knowlegde Reader, Figura 6, constituídos por diversos casos de uso, entre os quais, Pesquisar e Navegar, duas das funcionalidades criticas da aplicação.

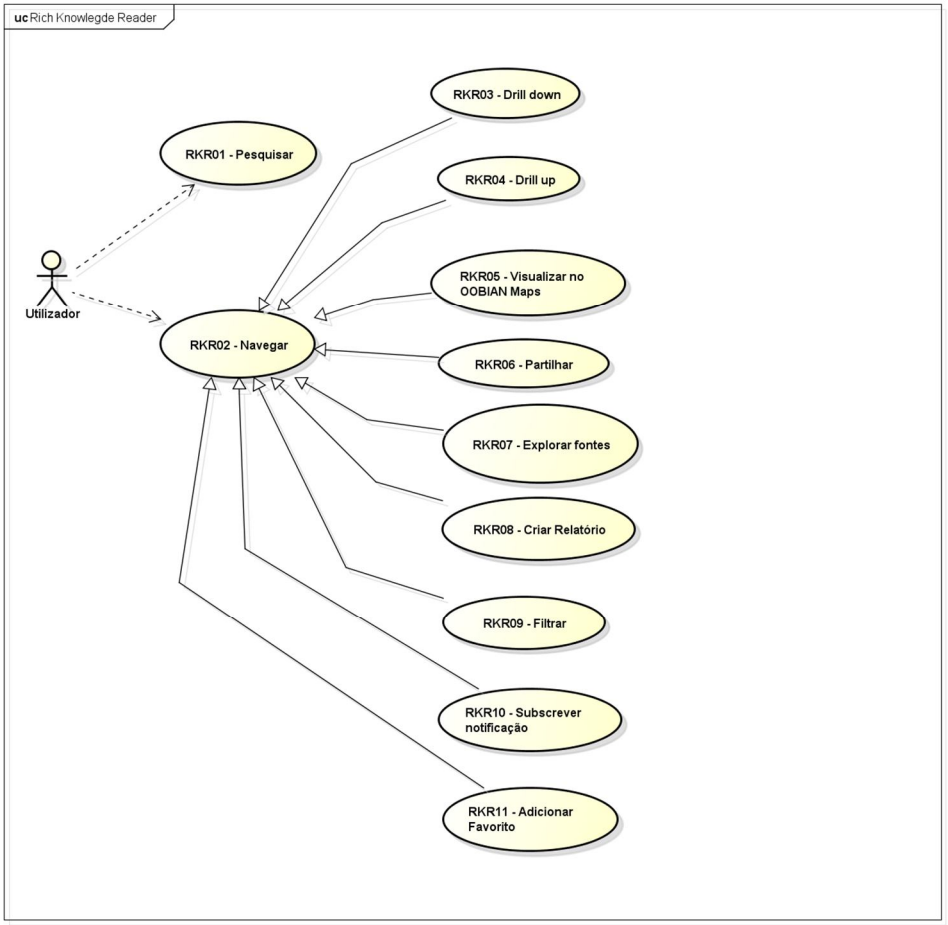


Figura 6 – Diagrama de casos de uso Rich Knowlegde Reader

Para o módulo Mapas apresenta-se uma descrição completa.

Contexto real:

O módulo Mapas é de vital importância já que falamos de uma aplicação para um dispositivo móvel, onde a geolocalização é um aspeto importante.

Podemos imaginar por exemplo que o vendedor José Saraiva desloca-se de comboio a outra cidade para várias reuniões com clientes. Durante a viagem deseja organizar o seu dia e apresenta no mapa, filtrando o conteúdo que deseja, os clientes com que vai reunir nesse dia. Assim estudará um percurso que lhe permita aproveitar ao máximo o tempo disponível.

Protótipos:

Figura 7 – Ecrã Mapas

Casos de uso:

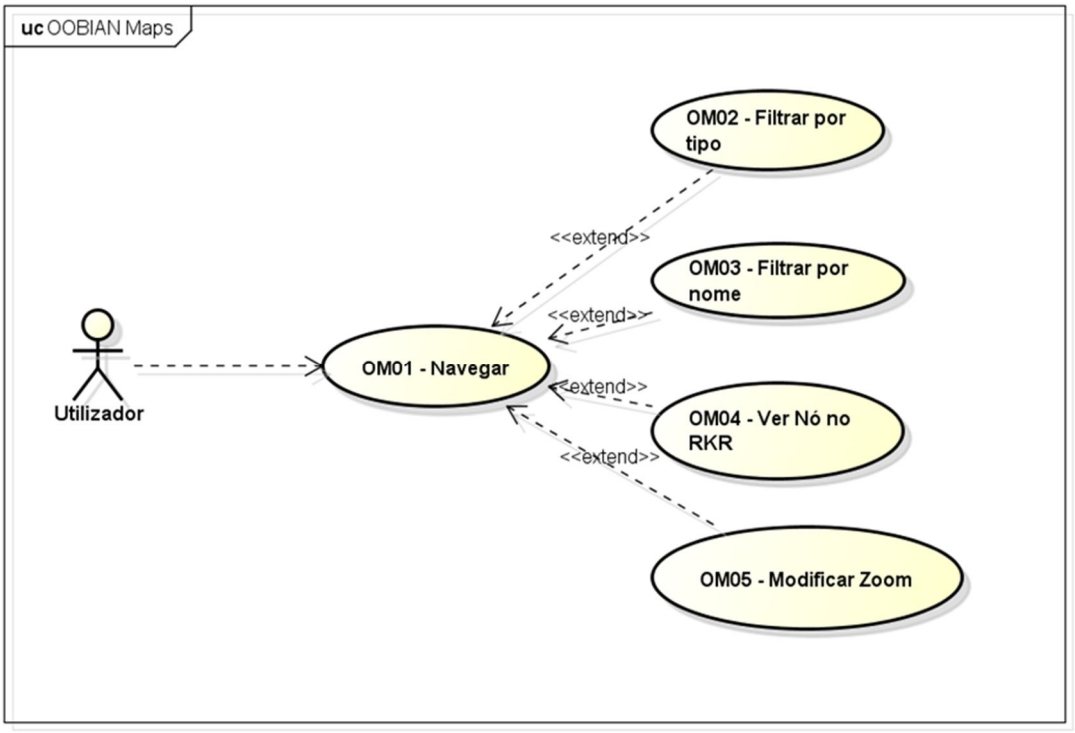


Figura 8 – Diagrama de casos de Uso do módulo Mapas

Nome	**OM01**	**Navegar**
Descrição	O utilizador navega no mapa onde estão localizados os nós georreferenciados.	
Necessidade	Torna-se necessário o utilizador poder navegar livremente pelo mapa.	
Utilizadores	Todos.	
Pré condições	O utilizador encontra-se autenticado na aplicação e encontra-se no módulo Mapas.	
Passos na relação entre o Software e o Utilizador	1. O utilizador, utilizando o evento de toque *Drag*, move o mapa e explora-o como desejar;	
Pós condições	O mapa mostra a área escolhida pelo utilizador.	

Figura 9 – Caso de Uso OM01 do módulo Mapas

Nome	**OM02**	**Filtrar por tipo**
Descrição	O utilizador efetua uma filtragem de nós presentes no mapa	
Necessidade	Existe a necessidade de fazer uma filtragem direta no mapa para que o utilizador possa encontrar o que pretende mais rapidamente.	
Utilizadores	Todos.	
Pré condições	O utilizador encontra-se autenticado na aplicação e encontra-se no módulo Mapas.	
Passos na relação entre o Software e o Utilizador	1. O utilizador abre o menu de aplicação; 2. O utilizador seleciona a opção “Filtragem” e escolhe o tipo dos nós que deseja visualizar; 3. Aplica a filtragem.	
Pós condições	A filtragem é aplicada e o mapa de resultados passa apenas a conter os elementos que obedecem aos parâmetros da filtragem.	

Figura 10 – Caso de Uso OM02 do módulo Mapas

Nome	**OM03**	**Filtrar por nome**
Descrição	O utilizador efetua uma filtragem de nós presentes no mapa	
Necessidade	Existe a necessidade de fazer uma filtragem direta no mapa para que o utilizador possa encontrar o que pretende mais rapidamente.	
Utilizadores	Todos.	
Pré condições	O utilizador encontra-se autenticado na aplicação e encontra-se no módulo Mapas.	
Passos na relação entre o Software e o Utilizador	1. O utilizador abre o menu de aplicação; 2. O utilizador seleciona a opção “Filtragem” e escolhe a palavra-chave ou termo que deseja utilizar para proceder à filtragem; 3. Aplica a filtragem.	
Pós condições	A filtragem é aplicada e o mapa de resultados passa apenas a conter os elementos que obedecem aos parâmetros da filtragem.	

Figura 11 – Caso de Uso OM03 do módulo Mapas

Nome	**OM04**	**Ver Nó no RKR**
Descrição	O utilizador vê uma entidade georreferenciada no módulo Rich Knowlegde Reader.	
Necessidade	O utilizador tem necessidade de ver uma entidade georreferenciada no módulo Rich Knowlegde Reader para conseguir ver os seus detalhes e relacionamentos com outras entidades.	
Utilizadores	Todos.	
Pré condições	O utilizador encontra-se no módulo Mapas e existem entidades georreferenciadas.	
Passos na relação entre o Software e o Utilizador	1. O utilizador pressiona o marcador da entidade georreferenciada no mapa; 2. Seleciona a opção Ver no Rich Knowlegde Reader;	
Pós condições	Aplicação navega para a entidade escolhida dentro do módulo selecionado.	

Figura 12 – Caso de Uso OM04 do módulo Mapas

Nome	**OM05**	**Modificar zoom**
Descrição	O utilizador modifica o nível de *zoom* do mapa que se encontra a visualizar.	
Necessidade	O utilizador tem necessidade modificar os níveis de *zoom* do mapa.	
Utilizadores	Todos.	
Pré condições	O utilizador encontra-se no módulo Mapas.	
Passos na relação entre o Software e o Utilizador	1. O utilizador muda os valores de *zoom* na barra de estado ou utiliza os eventos de toque *Pinch* e *Spread* para modificar o zoom;	
Pós condições	O *zoom* do documento é modificado de acordo com a selecção do utilizador.	

Figura 13 –Caso de Uso OM05 do módulo Mapas

Por fim, é apresentado o diagrama de casos de uso do módulo de Perfil, Figura 14. Além da gestão dos dados de utilizador, ainda são definidos os casos de uso de gestão de favoritos, notificações e fontes externas.

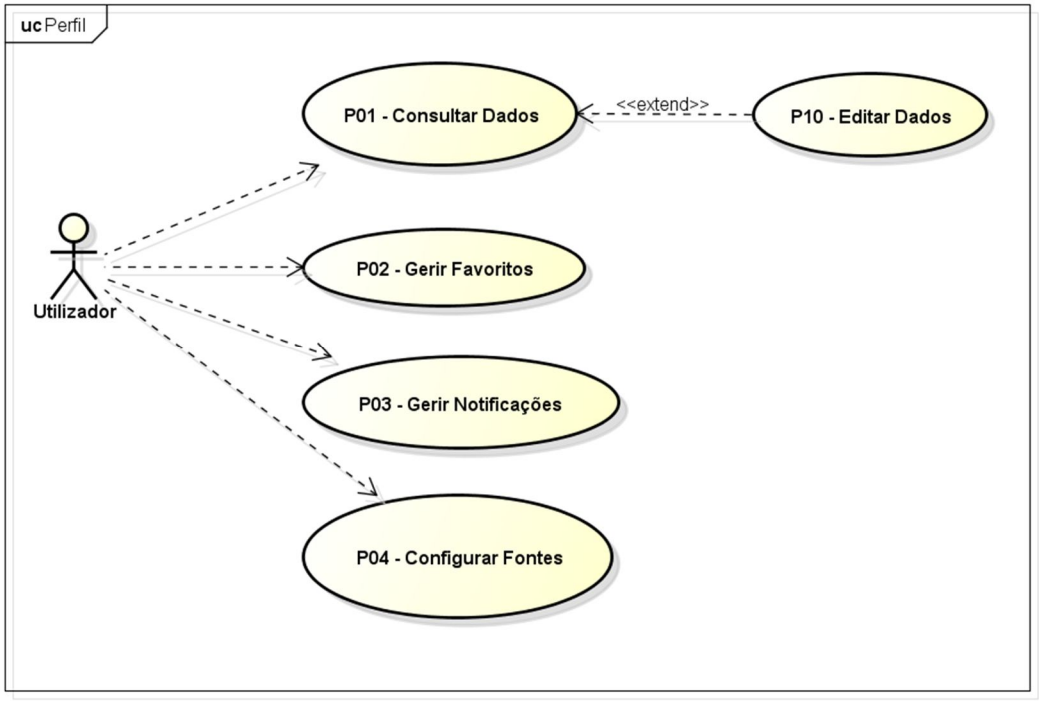


Figura 14 – Diagrama de casos de uso Perfil

4.4. Prototipagem

A fase de prototipagem revelou-se fulcral para aprimorar a interface e a usabilidade da aplicação. A primeira abordagem de prototipagem foi feita em papel, e tendo em conta os padrões de *design* do sistema operativo iOS, fornecidos pela Apple [28]. Em papel foi feita uma prototipagem de baixa fidelidade, ainda muito experimental, como podemos observar na **Figura 15**. Os protótipos desenvolvidos nesta fase podem ser consultados no anexo C.

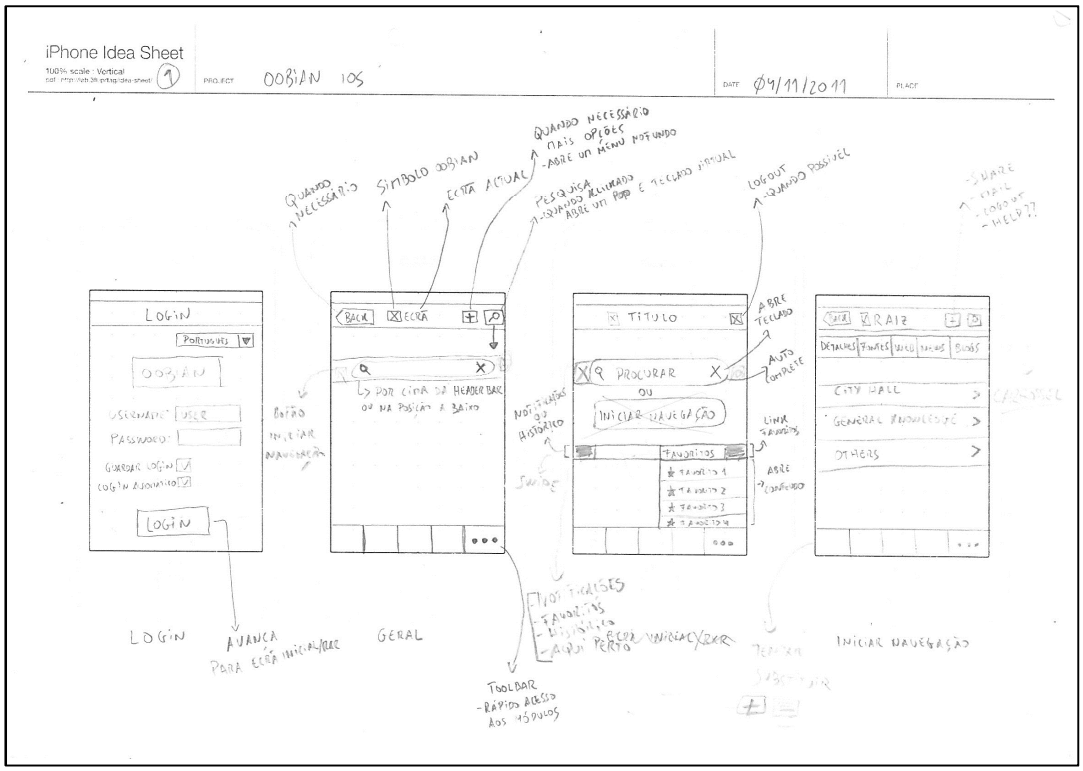


Figura 15 – Protótipo em papel

Do papel, avançou-se para *mockups* gráficos. Usando o *software* Balsamiq Mockups [29] foram desenvolvidos vários *mockups* mais pormenorizados que os protótipos de papel, como podemos ver na Figura 16. Estes *mockups* foram incorporados no documento de requisitos para proporcionar uma melhor compreensão das funcionalidades esperadas na aplicação. Usando estes *mockups* e o Microsoft PowerPoint foram feitos mapas de navegação para os diferentes casos de uso, este artefacto permitiu uma análise mais pormenorizada sobre a navegação na aplicação.

Figura 16 – Protótipo gráfico

No seguimento da fase de prototipagem foi desenvolvido um protótipo funcional da aplicação, como podemos ver na Figura 17 que se revelou fundamental para o sucesso do projeto. Depois de testado no simulador e num dispositivo iOS, o protótipo permitiu corrigir algumas práticas de *design* menos corretas, melhorando assim a usabilidade da aplicação. Alguns screenshots do protótipo desenvolvido podem ser consultados no anexo F.

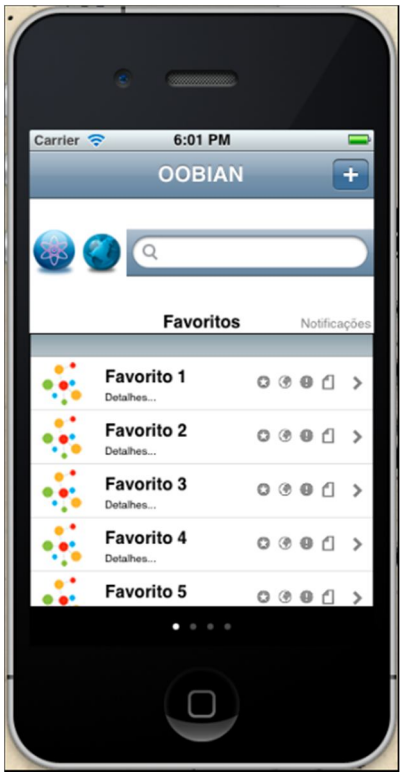


Figura 17 – Protótipo funcional

4.5. Requistos

Depois de feito o levantamento dos requisitos, estes foram organizados segundo o modelo FURPS+ [30]. O FURSP+ é um modelo desenvolvido pela Hewlett-Packard, que define os requisitos de um projeto de *software* segundo os seguintes atributos: Funcionalidade, Usabilidade, Confiabilidade, Desempenho e Suportabilidade. O "+" em FURPS+ tem como objetivo relembrar a inclusão dos seguintes requisitos: restrições de Design, requisitos de implementação, requisitos de interface e requisitos físicos. É também proposta uma organização destes requisitos por componentes do sistema, que em conjunção como o modelo FURPS+ permite capturar os requisitos de *software* com uma maior facilidade e organização.

Após uma análise cuidada, foram identificados os seguintes componentes de sistema:

1. Infra-estrutura/ Geral
2. Navegação na estrutura de conhecimento
3. Pesquisa de conteúdos
4. Perfil do utilizador
5. Filtragem de conteúdos
6. Detalhes de instância
7. Mapas
8. Filtragem de conteúdos geográficos

Os requisitos listados de seguida foram separados em requisitos funcionais e requisitos não funcionais, de acordo com o modelo FURPS+. Os requisitos funcionais foram ainda separados tendo em conta os componentes de sistema identificados.

Neste documento apenas é feita uma listagem dos requisitos. Para uma informação mais detalhada relativa aos mesmos, consultar o **Documento de Requisitos** (ver anexo D) e o **Documento de Especificação** (ver anexo E).

A listagem de requisitos obedece à seguinte nomenclatura:

ID do requisito	Nome do requisito	Prioridade

4.5.1. Requisitos Funcionais

Os requisitos funcionais considerados para o desenvolvimento da primeira versão do OOBIAN iOS foram os seguintes:

Infra-estrutura/ Geral

R.OOB-IOS.IG01v1	Requisitos mínimos para a execução da aplicação no dispositivo móvel	Muito Alta
R.OOB-IOS.IG02v1	Alargamento das possibilidades da aplicação consoante as dimensões e as características do dispositivo anfitrião	Média
R.OOB-IOS.IG05v1	Disponibilizar histórico de utilização	Alta

Navegação na Estrutura de Conhecimento

R.OOB-IOS.NV01v1	Navegação na estrutura de conhecimento	Muito Alta
R.OOB-IOS.NV02v1	Apresentação dos detalhes do nó em consulta	Muito Alta

Pesquisa de Conteúdos

R.OOB-IOS.PQ01v1	Pesquisa com janela de desambiguação de resultados em tempo real	Muito Alta
R.OOB-IOS.PQ02v1	Retorno de resultados em forma de nós da estrutura de conhecimento	Média

Perfil

R.OOB-IOS.P01v1	Apresentação dos dados do utilizador	Alta
R.OOB-IOS.P02v1	Edição dos dados do utilizador	Média
R.OOB-IOS.P03v1	Listagem e gestão de conteúdos favoritos	Média
R.OOB-IOS.P06v1	Gestão de fontes de informação externa	Média

Filtragem de conteúdos

R.OOB-IOS.FC01v1	Filtragem por *keyword*	Alta
R.OOB-IOS.FC02v1	Filtragem por tipo	Alta
R.OOB-IOS.FC03v1	Filtragem por nó de filtragem	Alta

Detalhes de instância

R.OOB-IOS.DT02v1	Partilha de conteúdos por e-mail	Baixa
R.OOB-IOS.DT03v1	Interligação com o OOBIAN Maps para nó georreferenciado	Alta
R.OOB-IOS.DT04v1	Apresentação de conteúdos multimédia	Alta
R.OOB-IOS.DT05v1	Relacionamento entre o nó em consulta e a estrutura de conhecimento	Alta
R.OOB-IOS.DT06v1	Visualização de documentos	Alta

Mapas

R.OOB-IOS.MP01v1	Visualização de dados geográficos da estrutura de conhecimento	Alta
R.OOB-IOS.MP02v1	Navegação no mapa	Alta
R.OOB-IOS.MP03v1	Apresentação de detalhes de objeto selecionado no mapa	Alta
R.OOB-IOS.MP04v1	Interligação com o Rich Knowledge Reader para nó georreferenciado	Alta
R.OOB-IOS.MP05v1	Deslocação do mapa para a georreferencia do utilizador	Alta

Filtragem de conteúdos geográficos

R.OOB-IOS.FCG01v1	Filtragem de conteúdos geográficos por *keyword*	Alta
R.OOB-IOS.FCG02v1	Filtragem de conteúdos geográficos por tipo	Alta

Como indicado no documento de especificação (ver anexo E) alguns requisitos funcionais não foram considerados para o desenvolvimento da primeira versão do OOBIAN iOS.

Os seguintes requisitos estão relacionados com notificações e por essa razão não foram considerados para esta versão da aplicação. Quando foi feito o primeiro levantamento de requisitos, foi pensado que as notificações funcionariam através de um serviço que acedesse ao servidor e recebesse os respetivos dados relativos a novas notificações. Em Outubro de 2011, a Apple lançou o iOS 5 e a nova Central de Notificações. Esta nova funcionalidade impede que aplicações tenham serviços próprios de notificações e remete todas as notificações para um serviço da Apple que depois as encaminha para o respetivo dispositivo. Foi decidido pela gerência não considerar os requisitos relativos a notificações e proceder a um posterior estudo sobre este novo serviço da Apple e possivelmente, implementar estes requisitos numa próxima versão da aplicação.

Perfil

R.OOB-IOS.P04v1	Receção de notificações	Média
R.OOB-IOS.P05v1	Listagem e gestão de subscrições de notificações	Média

4.5.2. Requisitos Não Funcionais

Os requisitos não funcionais levantados foram os seguintes:

Requisitos de Usabilidade

R.OOB-IOS.NF01v1	Redução do número de interações com o dispositivo móvel para operações básicas	Alta

Requisitos de Interface

R.OOB-IOS.NF02v1	Consistência entre o interface da aplicação e os *standards* do sistema operativo iOS	Alta
R.OOB-IOS.NF03v1	Consistência entre o ambiente da aplicação e a aplicação OOBIAN Insight	Alta

Requisitos de Confiabilidade

R.OOB-IOS.NF04v1	Transparência de exceções do lado do utilizador	Alta

Requisitos de Desempenho

R.OOB-IOS.NF05v1	Minimização de tempos de acesso à estrutura de conhecimento	Alta

Requisitos de Segurança

R.OOB-IOS.NF06v1	Garantia de segurança nas comunicações entre a aplicação e o servidor	Alta

4.6. Arquitetura da aplicação

Arquitetura interna da aplicação

Na Figura 18 podemos observar um diagrama com a arquitetura interna da aplicação. O componente REST API é o responsável pela comunicação com o servidor OOBIAN. Todos os outros componentes usam serviços por si implementados para proceder à troca de dados com o servidor.

O componente Mapas usa ainda a API do dispositivo responsável pelos Mapas. Com o lançamento da versão 6.0 do sistema operativo iOS, esta API deixou de ser da responsabilidade da Google (Google Maps) para ser uma API da Apple.

O componente Navegação acede igualmente à API Google Search de modo a fazer pesquisas com a instância ou classe que estão a ser navegadas. Realiza pesquisas normais, pesquisas com fontes escolhidas pelo utilizador e pesquisas por notícias relacionadas.

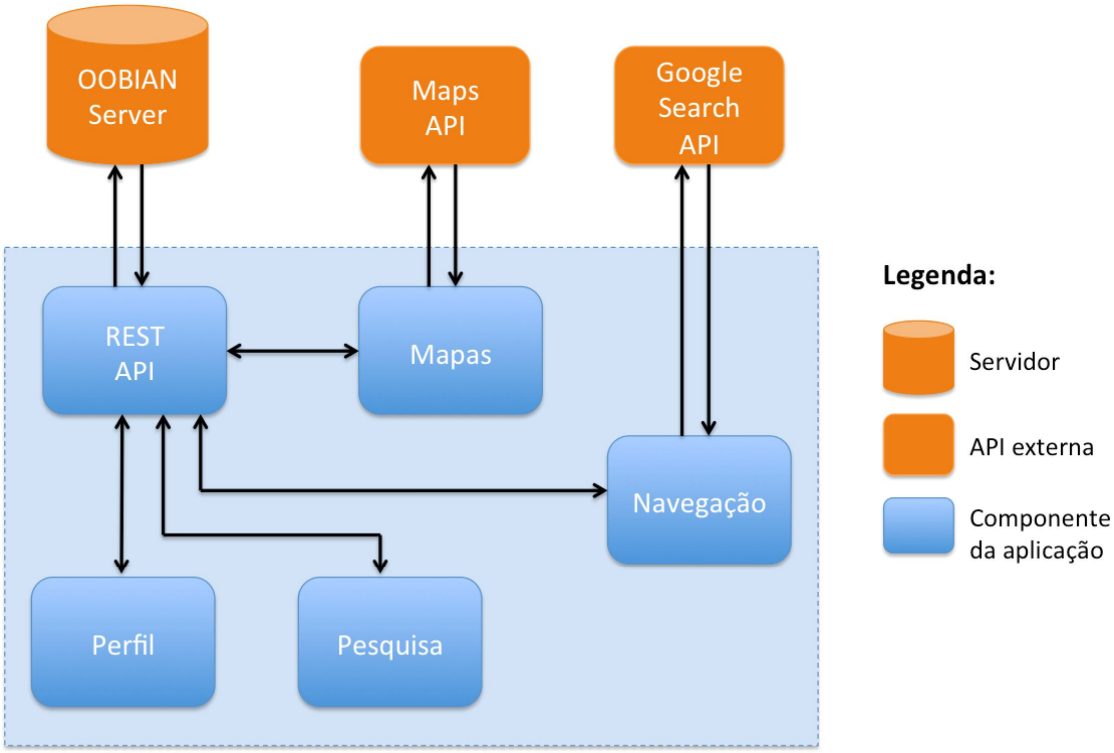


Figura 18 – Diagrama da arquitetura interna da aplicação

Módulos da aplicação

A aplicação irá ser constituída por diferentes módulos com diferentes funções. De seguida serão descritos os módulos que irão constituir a aplicação, assim como os serviços consumidos por cada um. Estes serviços já se encontram implementados no servidor OOBIAN.

Autenticação

Este módulo trata da autenticação no servidor por parte do utilizador, será o primeiro a ser carregado na aplicação. Só depois de uma autenticação realizada com sucesso podem os outros módulos da aplicação ser carregados e outros serviços do servidor chamados. Este módulo consome os seguintes serviços:

- ValidateLogin – serviço que procede à autenticação do utilizador no servidor. O pedido envia o *username* e *password* e recebe, em caso de sucesso, o *token* identificador da sessão. Esse *token* será utilizado em todos os pedidos para identificar o utilizador e a sessão aberta.
- logoff – serviço que termina a sessão no servidor. O pedido envia o *token* identificador da sessão e recebe a confirmação do sucesso da operação.

Pesquisa

Este módulo efetua a pesquisa de conteúdos na estrutura de conhecimento. Consome o seguinte serviço:

- search – este serviço é responsável pela procura de conteúdos. Faz o pedido onde envia o *token* identificador da sessão e a(s) keyword(s) de procura e recebe uma lista com os resultados da procura.

Navegação

Este é o módulo responsável pela navegação na estrutura de conhecimento. Esta navegação é feita para classes ou instâncias. Poderá ser iniciada no ecrã inicial ou nos módulos de Pesquisa e Mapas. Consome os seguintes serviços no servidor:

- getRootClassPaginated – serviço que navega para a classe raiz da estrutura de conhecimento. É enviado o *token* identificador da sessão e são recebidos os dados relativos à classe raiz.
- getOClassFromKBPaginated – serviço que navega para uma classe. É enviado, além do *token* identificador da sessão, o ID da classe assim como o seu *namespace* e são recebidos os dados relativos à classe pedida. Estes dados estão paginados e, por uma questão de otimização, apenas a primeira página é recebida, havendo depois a possibilidade do utilizador pedir mais dados, se assim for desejado. No caso de ser pedida uma filtragem, são também enviados os dados relativos a filtros ativos.
- getOInstanceFromKBPaginated – serviço que navega para uma instância. É enviado no pedido, além do *token* identificador da sessão, o ID e *namespace* da instância. Os dados recebidos relativos à instância pedida, por uma questão de otimização, vêm paginados e apenas a primeira página é recebida, havendo depois a possibilidade do utilizador pedir mais dados ao servidor. No caso de ser pedida uma filtragem, são também enviados os dados relativos a filtros ativos.
- getClusterPage – serviço que pede outras páginas de dados relativos a classes ou instâncias. No pedido, além do *token* identificador da sessão, é declarado de que classe ou instância se pretendem mais dados e o tamanho e página dos dados pretendidos.
- navigateToSimilarInstances – serviço que faz o pedido ao servidor por instâncias semelhantes a uma dada instância. Além do *token* identificador de sessão, é transmitido ao servidor a respetiva instância de que se pretendem instâncias semelhantes. É recebida uma lista de instâncias, caso elas existam.
- getInstanceData – serviço que faz o pedido ao servidor pelos ficheiros associados a uma determinada instância. Além do *token* identificador de sessão, é transmitido ao servidor a respetiva instância de que se pretendem os ficheiros associados. É recebida uma lista de documentos associados a essa instância, caso ela os tenha.

Neste módulo serão ainda usados os serviços da Google *web* e *news* para fazer pedidos por pesquisas na internet (com ou sem fontes) e pesquisas em notícias.

Perfil

O módulo Perfil é responsável, para além da apresentação e alteração dos dados do utilizador, a gestão de favoritos e fontes externas. Os serviços consumidos neste módulo são os seguintes:

- getUserInfo – serviço que faz o pedido ao servidor pelos dados do utilizador. Envia o *token* identificador da sessão e o *username* do utilizador e recebe os dados do utilizador.
- updateUser – serviço que faz o pedido de alteração de dados do utilizador. Envia o *token* identificador da sessão e os dados alterados do utilizador.
- getFavorites – serviço que pede ao servidor a lista de favoritos associada ao utilizador. Envia o *token* identificador da sessão e o *username* do utilizador e recebe uma lista de favoritos.
- saveFavorite – serviço que adiciona um favorito à lista de favoritos associada ao utilizador. Envia, além do *token* identificador da sessão e do *username*, os dados referentes à instância que se pretende adicionar como conteúdo favorito.
- deleteFavorite – serviço que remove um favorito da lista de favoritos associada ao utilizador. Envia, além do *token* identificador da sessão e do *username*, o nome do conteúdo favorito.
- getNewsSources – serviço que pede ao servidor a lista de fontes externas associada ao utilizador. Envia apenas o *token* identificador da sessão e recebe a lista de fontes.

Mapas

O módulo Mapas é o responsável pela visualização dos conteúdos geolocalizáveis presentes na estrutura de conhecimento. Os serviços consumidos por este módulo são os seguintes:

- getGeoInstancesTypes – serviço que faz o pedido ao servidor pelos tipos de instâncias geolocalizáveis presentes na estrutura de conhecimento. É enviado o

token identificador da sessão e é recebida uma lista de tipos de instâncias. Esta lista é usada, não só para a filtragem, mas para a construção dos diferentes pins para cada tipo de conteúdo.

- getGeoArea – serviço que faz o pedido ao servidor pela informação geográfica, isto é, os conteúdos geolocalizáveis presentes, de uma determinada área do mapa. Além do *token* identificador da sessão, é transmitida ao servidor a área geográfica pretendida. A resposta é uma lista de instâncias e de clusters de várias instâncias presentes na respetiva área.
- getCluster – serviço que faz o pedido pelas instâncias, e os seus dados, presentes num determinado cluster recebido aquando o pedido getGeoArea. No pedido é identificado o *cluster* e é recebida uma lista das instâncias que o compõem.

4.7. Especificação Funcional/ Lógica

Apresenta-se de seguida uma descrição da especificação do projeto, quer de uma perspetiva funcional quer de uma perspetiva lógica. Esta especificação encontra-se detalhada no documento de especificação do projeto (ver anexo E).

4.7.1. Perspetiva funcional

De uma perspetiva funcional, a aplicação OOBIAN iOS seguirá o modelo Cliente-Servidor. A comunicação entre o dispositivo móvel e o servidor OOBIAN é feita através da aplicação cliente OOBIAN iOS, que se serve de Web Services, mais concretamente serviços REST, para comunicar com o servidor OOBIAN. Na figura seguinte podemos observar o funcionamento do sistema.

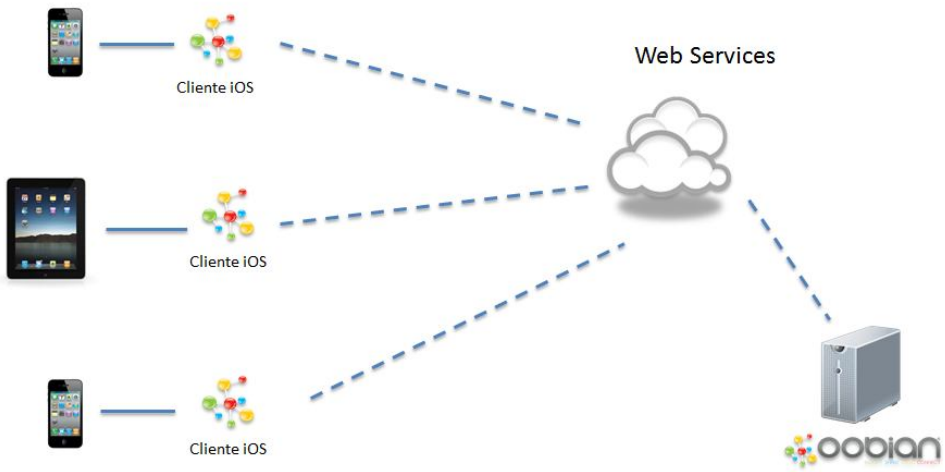


Figura 19 – Arquitetura geral do cliente OOBIAN iOS

A aplicação permitirá ao utilizador efetuar pesquisas e navegar na estrutura de conhecimento, bem como gerir o seu perfil e todas as opções associadas. As principais funcionalidades da aplicação foram esquematizadas na figura que se segue.

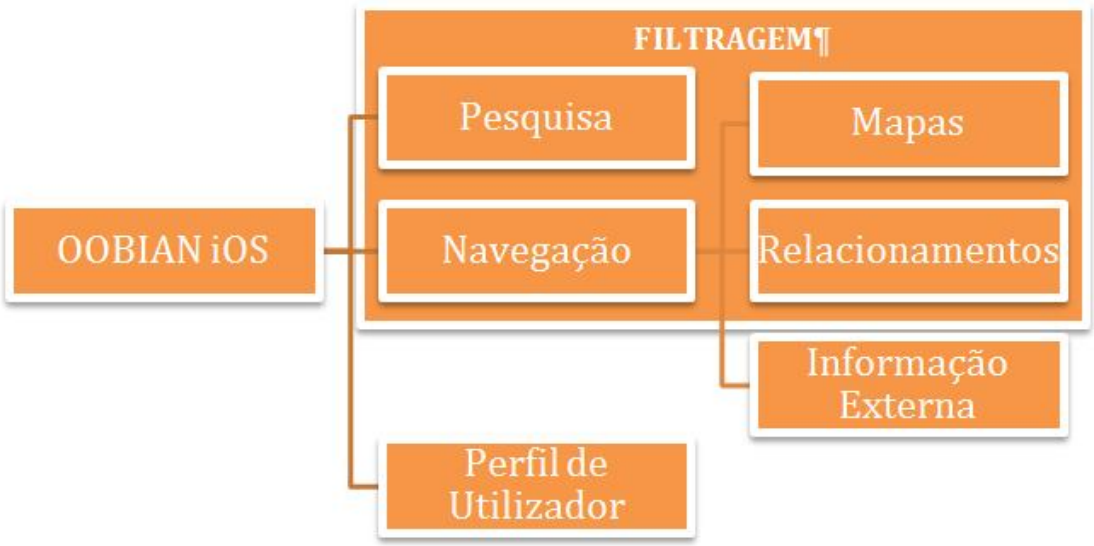


Figura 20 – Funcionalidades do cliente OOBIAN iOS

Ao entrar na aplicação, o utilizador terá que fazer o login no sistema. Feito o login com sucesso, o utilizador é remetido para o ecrã inicial, pertencente ao módulo Rich Knowlegde Reader. Neste ecrã inicial, o utilizador tem várias opções: pode efetuar uma pesquisa de conteúdos, pode iniciar uma navegação da estrutura de conhecimento, pode aceder ao módulo de Perfil ou ao módulo de Mapas. No módulo Perfil, além do acesso aos dados do utilizador, será também possível consultar e configurar os Favoritos, Notificações e Fontes.

4.7.2. Perspetiva Lógica

Estrutura da aplicação

A aplicação cliente OOBIAN iOS terá como base o modelo MVC. Preliminarmente prevê-se a utilização de uma classe do tipo UIViewController (controlador) para cada módulo (Rich Knowledge Reader, Perfil e Mapas) e ainda para o login, filtragem e histórico.

A camada View será implementada através do Interface Builder (usando o ficheiro .xib de cada controlador) ou através de código-fonte na classe do controlador. A camada Controller será implementada nas respetivas classes, enquanto a camada Model será implementada em classes próprias. As classes das camadas Controller e Model serão descritas mais à frente.

A informação circulará entre a aplicação cliente e o servidor através de dados serializados em Protobuf (Protocol Buffers). Como já foi descrito no capítulo 2, esta tecnologia é a que apresenta melhores resultados no tamanho dos dados serializados. Tendo este sistema um dispositivo móvel como cliente, é necessário nunca esquecer as restrições do mesmo, como capacidade de processamento e duração de bateria. O processamento no lado do dispositivo móvel terá que ser o mais reduzido possível assim como o tamanho dos dados transmitidos, precavendo a possibilidade do dispositivo de encontrar ligado a uma rede celular onde o valor a pagar dependa do tamanho do tráfego.

A comunicação será feita através de serviços REST disponibilizados pelo servidor OOBIAN. Esta comunicação poderá ser representada pela seguinte figura:

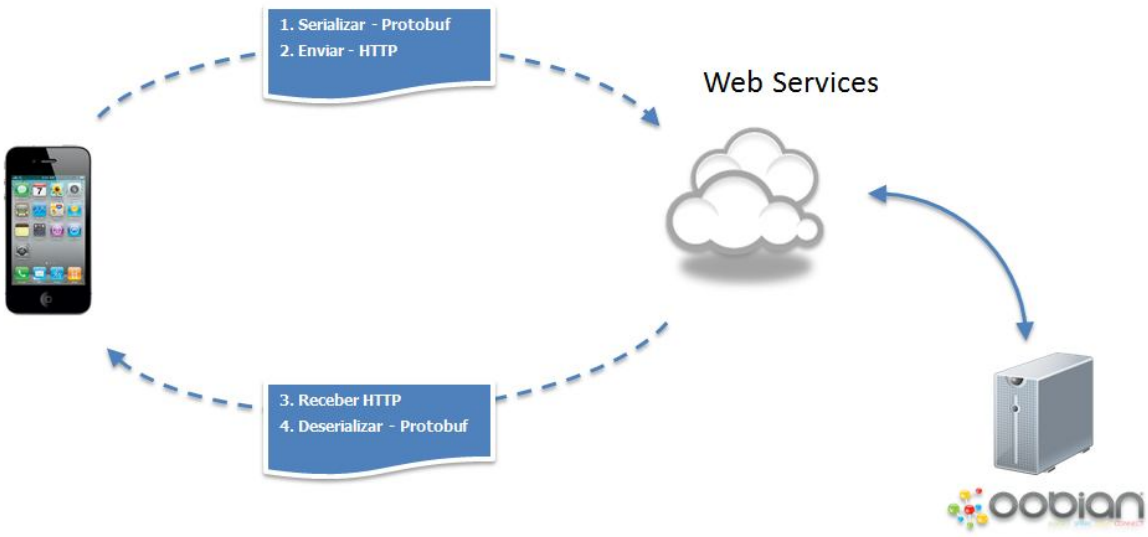


Figura 21 – Comunicação com o servidor OOBIAN

Estrutura da interface da aplicação

Na Figura 22 podemos observar os componentes de um ecrã típico da aplicação, componentes presentes em grande parte dos ecrãs da aplicação.

O ecrã da aplicação encontra-se dividido em duas regiões. A primeira, aqui chamada de Header Bar, é responsável por exibir o título e os botões de Menu e de Back. O botão de Menu exibirá o menu de aplicação enquanto que o botão Back fará com que o utilizador volte ao ecrã anterior (comportamento habitual em aplicações iOS).

Na segunda região, denominada de ecrãs da aplicação, será exibida toda a informação ao longo da utilização da aplicação. Esta região mudará consoante o módulo a ser utilizado ou os conteúdos a ser exibidos.

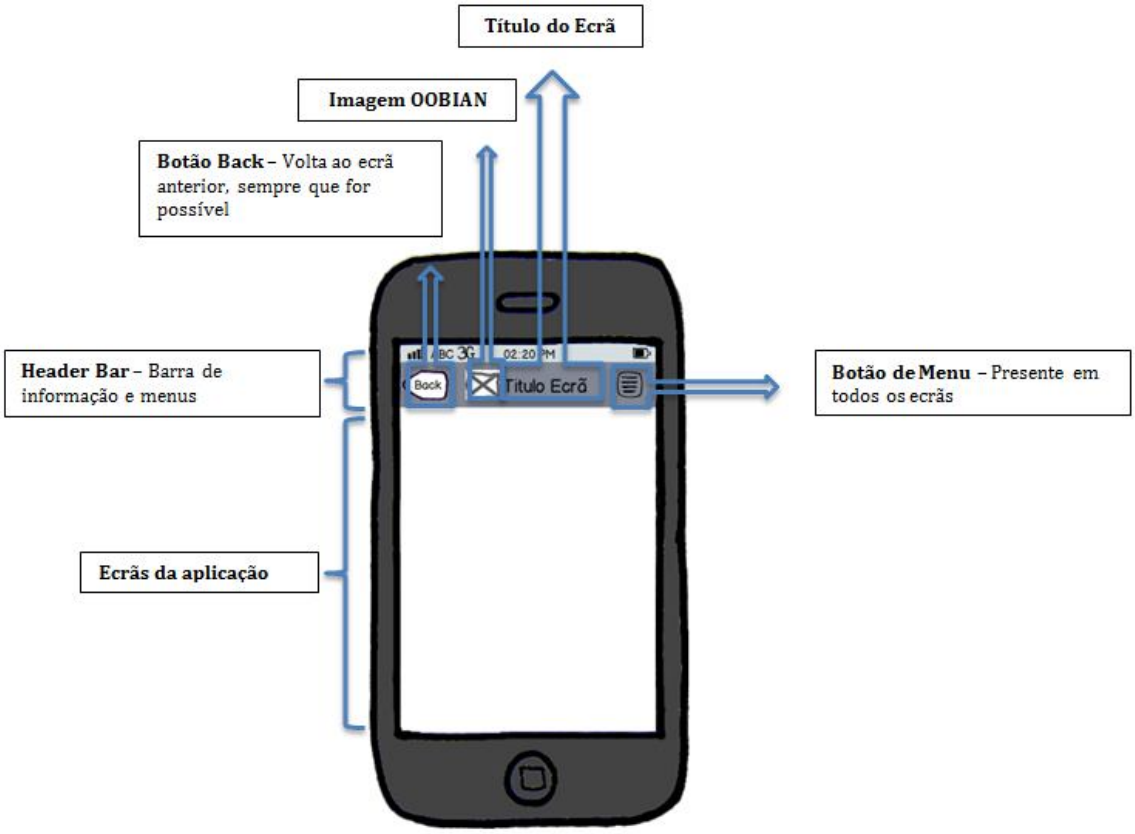


Figura 22 – Ecrã típico da aplicação

4.8. Sumário

Neste capítulo foi descrito todo o processo que levou à implementação da aplicação.

Este processo foi iniciado com os primeiros estudos sobre o OOBIAN e sobre as ferramentas de desenvolvimento para dispositivos móveis iOS. De seguida foi feito um levantamento de requisitos e elaborados os casos de uso. Seguiu-se a fase de prototipagem, onde foram concebidos protótipos em papel, *mockups* gráficos e *storyboards*. Foi elaborado o documento de requisitos e o documento de especificação. Terminados e aprovados estes documentos, foi desenvolvido um protótipo funcional que precedeu a implementação da aplicação.

No próximo capítulo, será abordada a fase de testes, fase que completou o processo de desenvolvimento.

Capítulo 5:
Testes

5.1. Metodologia de teste

A última fase do processo de desenvolvimento de qualquer aplicação é uma das mais importantes, a fase de testes. Myers, 1979, define testes como “O teste consiste em executar o programa com a intenção de encontrar erros (bugs)”[31] e Dijkstra define como “O teste de programas pode ser usado para mostrar a presença de defeitos, mas nunca para mostrar a sua ausência” [32]. Atualmente, grande parte dos custos no ciclo de vida de *software* derivam da manutenção dos mesmos. Será impossível reduzir a zero os erros de qualquer aplicação, mas podendo corrigir muitos deles ainda antes do lançamento, permitirá à empresa poupar em transtornos e gastos assim como evitar o descontentamento do utilizador final.

Neste projeto a fase de testes adquiriu ainda um papel mais importante porque, além da normal fase de testes, foi ainda preciso testar a aplicação tendo em conta as *guidelines* da Apple, já que a aplicação será distribuída através da AppStore. Mais à frente serão descritos os testes realizados com vista a garantir a aprovação da aplicação na AppStore.

Durante o desenvolvimento da aplicação foram realizados testes unitários para cada funcionalidade implementada, incindido principalmente na comunicação com o servidor e no tratamento dos dados recebidos. Com a implementação de novas funcionalidades, todos os testes unitários foram repetidos. Paralelamente, foram elaborados casos de teste que permitiram elaborar um plano de testes, disponível no anexo F.

Concluído o desenvolvimento da aplicação, o plano de testes elaborado foi usado pela equipa de testes para testar a aplicação. Terminada a primeira fase de testes, foi elaborado um relatório inicial onde foram descritos os bugs encontrados. Depois da correção dos mesmos, foi feita uma segunda fase de testes, e o relatório de testes foi atualizado.

De seguida será apresentado o plano de testes seguido e mais à frente descritos os cuidados especiais tidos nos testes para garantir que a aplicação fosse aprovada na AppStore.

5.2. Plano de testes

O plano de testes foi elaborado paralelamente à implementação dos requisitos e funcionalidades descritos no documento de requisitos e documento de especificação. Todos os casos de testes foram testados nos dispositivos iPhone e iPad e nas orientações vertical e horizontal. Para garantir o teste eficaz da internacionalização, todos os casos de testes foram repetidos com as duas línguas implementadas na aplicação, língua portuguesa e inglesa.

De seguida é apresentada uma listagem dos casos de teste. A descrição detalhada destes pode ser consultada no plano de testes em anexo (ver anexo G).

Infra-estrutura/Geral

O seguinte conjunto de casos de testes destina-se ao teste de aspetos gerais da aplicação.

Instalação da aplicação

Neste caso de teste foi testada a instalação da aplicação nos dispositivos. Como a aplicação ainda não se encontra na AppStore, esta instalação foi feita através de uma máquina com Mac OS X e Xcode.

Utilização da aplicação

Aqui foi testado o carregamento da aplicação, depois de instalada no dispositivo.

Internacionalização

Este caso de teste serviu para testar o bom funcionamento da internacionalização da aplicação, disponível atualmente nas línguas portuguesa e inglesa.

Autenticação no servidor

Neste caso de teste foi testada a autenticação no servidor. Foi testada a autenticação depois de introduzir as credenciais e a autenticação feita automaticamente com as credenciais já gravadas.

Terminar sessão

Aqui foi testado o fim de sessão. O logout é feito no servidor e na aplicação.

Navegação na estrutura de conhecimento

Os seguintes casos de teste destinam-se a testar e garantir o correto funcionamento da navegação na estrutura de conhecimento, uma funcionalidade crítica da aplicação.

Iniciar navegação na classe raiz do repositório

Aqui foi testado o início de navegação na estrutura de conhecimento a partir da classe raiz.

Navegar para uma subclasse

Neste caso de teste foi testada a navegação para uma subclasse.

Navegar para uma instância

Neste caso de teste foi testada a navegação para uma instância.

Navegar para a classe *parent* de uma subclasse

Neste caso de teste foi testada a navegação para de uma subclasse para a sua classe *parent*, utilizando um mecanismo de *drill up*.

Navegar para a classe *parent* de uma instância

Neste caso de teste foi testada a navegação para de uma instância para a sua classe *parent*, utilizando um mecanismo de *drill up*.

Explorar os detalhes de uma classe

Este caso de teste serve para testar a opção de explorar os detalhes de uma classe quando nos encontramos numa navegação pela estrutura de conhecimento.

Pesquisa de conteúdos

Os seguintes casos de teste destinam-se a testar a pesquisa de conteúdos na estrutura de conhecimento.

Pesquisa com tabela de desambiguação

Neste caso de teste é testada a pesquisa com desambiguação. À medida que o utilizador vai escrevendo o que deseja pesquisar são sendo apresentadas sugestões.

Pesquisa normal

Aqui será testada a pesquisa normal, isto é, a pesquisa efetuada quando o utilizador escreve o texto pelo qual deseja pesquisar e prime o botão Pesquisar.

Perfil do utilizador

Os casos de testes apresentados de seguida tratam do perfil e dados do utilizador.

Apresentação dos dados do utilizador

Neste caso de teste é testada a apresentação correta dos dados do utilizador.

Edição dos dados do utilizador

Neste caso de teste é testada a edição correta dos dados do utilizador.

Favoritos

Os seguintes casos de testes estão relacionados com a apresentação e gestão dos favoritos.

Consulta de conteúdos favoritos no ecrã inicial

Este caso de teste destina-se a testar a correta visualização dos favoritos no ecrã inicial.

Consulta em ecrã próprio

Neste caso de teste testa-se a correta visualização dos favoritos no ecrã próprio.

Adicionar um conteúdo à lista de favoritos

Aqui testa-se a adição de um conteúdo à lista de favoritos.

Remover um conteúdo favorito no ecrã do conteúdo

Neste caso de teste é testada a remoção de um favorito da lista de favoritos, estando a navegar no conteúdo que se deseja remover.

Remover um conteúdo favorito no ecrã próprio

Aqui é igualmente testada a remoção de um favorito da lista de favoritos, mas no ecrã próprio de apresentação dos favoritos.

Histórico

Os casos de teste descritos de seguida prendem-se com a apresentação do histórico de utilização da aplicação.

Consulta do histórico no ecrã inicial

Neste caso de teste é testada a consulta do histórico no ecrã inicial da aplicação.

Consulta do histórico em ecrã próprio

Aqui é testada a consulta do histórico de utilização em ecrã próprio.

Navegação para item de histórico

Este caso de teste serve para testar a navegação, a partir da lista do histórico da utilização quer seja no ecrã inicial quer seja em ecrã próprio, para um item de histórico.

Lista de proximidade (Aqui Perto)

O seguinte conjunto de casos de teste, destina-se a testar a lista de proximidade (também referida como Aqui Perto) onde são apresentadas as instâncias que se encontram na zona da localização atual do utilizador.

Consulta da lista de proximidade no ecrã inicial

Neste caso de teste é testada a consulta da lista de proximidade no ecrã inicial da aplicação

Consulta da lista de proximidade em ecrã próprio

Aqui é testada a consulta da lista de proximidade em ecrã próprio.

Navegação para os detalhes de uma instância na lista de proximidade

Neste caso de teste é testada a navegação para os detalhes de uma instância presenta na lista de proximidade.

Navegação para a localização de uma instância na lista de proximidade

Aqui é testada a navegação para a localização de uma instância presente na lista de proximidade, usando para isso o módulo Mapas.

Fontes

Os seguintes casos de teste estão relacionados com as fontes de informação externa, utilizadas para efetuar pesquisas na internet sobre os conteúdos presentes na estrutura de conhecimento.

Consulta das fontes de informação externa

Neste caso de teste é testada a consulta da lista de fontes de informação externa.

Consulta das entradas relativas a informação externa

Aqui é testada a consulta de um item presente na lista de fontes de informação externa.

Filtragem de conteúdos

Os casos de teste descritos de seguida testam a filtragem de conteúdos na navegação de classes e instâncias presentes na estrutura de conhecimento.

Filtragem de conteúdos por *keyword*

Este caso de teste testa a filtragem com base numa (ou mais) *keyword*.

Filtragem de conteúdos por tipo

Aqui é testada a filtragem com base no tipo de instância.

Filtragem de conteúdos por nó de filtragem

Neste caso de teste é testada a filtragem usando uma instância como nó de filtragem. Ativada esta filtragem, apenas serão exibidos conteúdos que se relacionem com a instância escolhida como nó de filtragem.

Limpar filtros ativos

Este caso de teste destina-se a testar a opção de limpar os filtros ativos, opção que desativa todos os filtros.

Navegação numa instância

Os seguintes casos de teste servem para testar a navegação numa instância e as diferentes opções disponíveis.

Visualizar descrição

Neste caso de teste é testada a visualização correta da descrição de uma instância.

Visualizar detalhes

Neste caso de teste é testada a visualização correta dos detalhes de uma instância e a interação com os mesmos (e-mail, telefone, URL, etc.), se assim for possível.

Interligação com o OOBIAN Maps para nó georreferenciado

Aqui é testada a interligação com o módulo Mapas para um nó georreferenciado. Se um nó for georreferenciado será possível abrir a sua localização no módulo Mapas.

Navegar a partir de relações de instância

Neste caso de teste é testada a navegação para instâncias que estão relacionadas com a instância que está a ser consultada.

Paginação em relações de instância

Aqui é testada a paginação usada para mostrar as relações da instância que está a ser consulta com outras instâncias. Para relações com muitas instâncias há a necessidade de otimizar os pedidos ao servidor e a interface, mostrado apenas cinco instâncias mas havendo a possibilidade de mostrar mais.

Apresentação de conteúdos multimédia

Neste caso de teste é testada a visualização dos conteúdos multimédia (imagens ou vídeos) associados a uma instância.

Visualização de documentos associados a uma instância

Este caso de teste serve para testar a visualização da lista de documentos associados a uma instância.

Visualização de documentos

Aqui é testada a visualização de um documento presente da lista de documentos associados a uma instância.

Visualização dos detalhes do documento

Neste caso de teste é testada a visualização dos detalhes de um documento.

Partilha de conteúdos por e-mail

Este caso de teste serve para testar a partilha de conteúdos por e-mail. É enviado um e-mail pelo utilizador com a descrição da instância que está a consultar e um link para a mesma.

Mapas

O seguinte conjunto de casos de teste dizem respeito ao módulo Mapas e às suas funcionalidades.

Iniciar Mapa

Neste caso de teste apenas é testado o carregamento do módulo Mapas na aplicação.

Navegação no mapa

Aqui é testada a navegação no mapa, usando os gestos *swipe* e *pinch*.

Apresentação de detalhes de objeto selecionado no mapa

Neste caso de teste é testada a visualização dos detalhes de um objeto presente no mapa, quando selecionado.

Interligação com o Rich Knowledge Reader para nó georreferenciado

Este caso de teste serve para testar a interligação do módulo Mapas com o Rich Knowledge Reader, abrindo os detalhes da instância selecionada no mapa.

Mudar estilo de mapa

Neste caso de teste é testada a mudança de estilos de mapa, entre o estilo standard e o estilo satélite.

Filtragem de conteúdos geográficos

O seguinte conjunto de casos de teste destina-se a testar a filtragem de conteúdos geográficos no módulo Mapas.

Filtragem de conteúdos geográficos por *keyword*

Neste caso de teste é testada a filtragem de conteúdos geográficos usando uma (ou mais) *keyword*.

Filtragem de conteúdos geográficos por tipo

Este caso de teste testa a filtragem de conteúdos geográficos com base no tipo dos mesmos.

5.3. Aceitação na AppStore

A AppStore é a loja *online* de aplicações para produtos Apple, tanto para computadores como para dispositivos moveis. No que trata a dispositivos móveis iOS, é a maior fonte de distribuição de aplicações e como tal, a aplicação cliente OOBIAN iOS utilizará os seus serviços para chegar ao utilizador final. Como é conhecido, a Apple tem uma política de aceitação de aplicações na AppStore muito rígida, o que leva à necessidade de uma atenção especial no desenvolvimento e testes de aplicações que se pretenda serem distribuídas via AppStore.

Durante o desenvolvimento da aplicação as restrições e imposições da Apple para a aceitação de aplicações na sua loja *online* foram tidas em conta, foram seguidas as *guidelines* da Apple no que toca à interface das aplicações [33] e no que toca à aceitação de aplicações na AppStore [34].

Sendo esta uma aplicação cliente para a plataforma OOBIAN, certos parâmetros de aceitação encontram-se automaticamente cobertos como a necessidade de a aplicação em causa não imitar o comportamento de aplicações desenvolvidas pela Apple e não replicar as funcionalidades de outras aplicações já presentes em número considerável na AppStore. A necessidade que não ter conteúdos com direitos de autor (marcas, imagens, vídeos ou musicas) e de não ter materiais de caracter pornográfico, racista, politica, obsceno ou ofensivo está igualmente coberta.

Durante o desenvolvimento, já com o conhecimento deste tipo de restrições por parte da Apple, foram tidos em atenção os cuidados de não usar livrarias privadas não definidas no iOS SDK e funcionalidades do mesmo SDK não documentadas. Como em qualquer aplicação bem implementada, esta não poderá “crashar” e apresentar avisos ou erros durante a compilação. Outros aspetos tidos em conta durante o desenvolvimento, foi o fato da aplicação ter que pedir autorização ao utilizador para aceder aos serviços de localização do dispositivo assim como notificar o utilizador se a ligação à internet não estiver ativa, já que a aplicação vai necessitar de uma ligação ativa. Em relação à interface, foram tidas em consideração, como já foi referido, as *guidelines* fornecidas pela Apple e o cuidado de não usar interfaces semelhantes a aplicações da própria Apple.

Outro aspeto tido em conta na fase de testes foi a existência de possíveis *memory leaks* na aplicação. Foram realizados testes usando a aplicação Instruments (disponível com o ambiente de desenvolvimento Xcode) para identificar e corrigir possíveis *memory leaks*. Seria ainda desejável utilizar a aplicação Instruments para realizar outros testes, mas devido à falta de tempo isso não foi possível.

Por ultimo, na submissão da aplicação na AppStore também foram tidos alguns aspetos em conta. A descrição e *keywords* têm que estar de acordo com a aplicação e têm que ser fornecidos os ícones da aplicação nos tamanhos pedidos pela Apple.

5.4. Sumário

Neste capítulo foi abordada a fase de testes. Inicialmente, foi explicada a metodologia usada. De seguida foi apresentado o plano de testes e listados os casos de testes elaborados. O plano de testes foi seguido e todos os testes foram aprovados. Outro aspeto importante, tratando-se de uma aplicação iOS, é a aceitação na AppStore, aceitação esta que a Apple pode dificultar. Por fim foram descritos todos os cuidados tidos no desenvolvimento e na fase de testes de modo a preparar a aplicação para a submissão na AppStore.

O próximo capítulo é de conclusões, onde será feita uma análise ao planeamento, descritos os desafios do projeto e o trabalho futuro.

Capítulo 6:
Análise ao Planeamento

O estágio realizado teve a duração de um ano letivo, tendo sido iniciado em Setembro de 2011 e concluído em Julho de 2012. Para o efeito, foi elaborado um planeamento inicial, mas, como acontece em todos os projetos, acabaram por ocorrer alguns imprevistos que, naturalmente, ocasionaram pequenos desvios em relação à previsão inicial. Estando o projeto inserido na disciplina de Dissertação/ Estágio do Mestrado em Engenharia Informática, da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, existiam à partida duas grandes *milestones*:

- A entrega e apresentação do relatório intermédio de estágio, a 24 de Janeiro de 2012;
- A entrega e apresentação do relatório final de estágio, a 12 de Julho de 2012.

Todo o planeamento teve de ser feito tendo em conta estas duas datas (que foram naturalmente cumpridas), tendo, contudo, havido necessidade de serem planeadas outras milestones, consubstanciadas com a entrega de artefactos do projeto.

O trabalho inicialmente previsto realizar foi divido em sete tarefas, que serão seguidamente elencadas (tal como se encontravam na proposta de estágio):

T1. Estudo e formação sobre o desenvolvimento para dispositivos móveis e algoritmos de visualização de informação

Estudo sobre os conceitos relevantes para a implementação do sistema:

- Ambientação às ferramentas e metodologias do projeto
- Caracterização da plataforma OOBIAN
- Desenvolvimento de aplicações para dispositivos móveis
- Estudo das melhores práticas de usabilidade de software

T2. Levantamento de requisitos

Refere-se ao estudo e levantamento dos requisitos do sistema a implementar, com a elaboração de um documento onde deverão ser definidos, com rigor, os requisitos funcionais e não funcionais, devendo conter, preferencialmente, “casos de uso” e alguns Wireframes.

T3. Prototipagem

O objetivo desta tarefa será estruturar, planear e desenhar a interface de utilização com vista à navegação na estrutura de conhecimento. Existem diferentes formas de realizar esta tarefa. Para o efeito propusemo-nos à construção dos seguintes outputs:

- Mapas de Navegação
- *Storyboards*
- Protótipos de UI
- Mockups gráficos

Convém referir que, sendo o *objetivo deste estágio, o desenvolvimento de uma aplicação para dispositivos móveis que apresenta uma determinada estrutura de conhecimento*, é crítico dotar o sistema de uma interface intuitiva, de alta usabilidade e escalável, que permita visualizar "conhecimento" de uma forma rica. Esta fase irá ter um papel importante na experiência de utilização da aplicação a criar. Consequentemente, este ponto é de extrema relevância para o resultado final do estágio.

T4. Especificação

Nesta parte deverá ser criado um ou mais artífices que definam de uma forma detalhada a especificação do sistema. Uma boa especificação, para além de tornar o projeto mais claro e tangível, auxilia e evita também possíveis dissabores nas fases posteriores.

Para além deste trabalho, propõe-se neste ponto, que acontecerá sensivelmente a meio do estágio, que a *defesa intermédia do estágio* seja estruturada e apresentada.

T5. Implementação

Nesta tarefa será desenvolvida a implementação do sistema. Para isso, pede-se que para além da implementação, sejam também criados testes de unidade e testes de automação.

T6. Testes de Aceitação e *Bugfixing*

Nesta fase, pretende-se que seja feita a validação do sistema implementado. Para isto, pretende-se que seja emulado o comportamento real do uso da aplicação. O objetivo pretendido, é garantir um *software* de melhor qualidade, à custa da identificação, análise e correção de possíveis problemas na aplicação desenvolvida.

T7. Auditoria, Revisão e Escrita do Relatório de Estágio

Neste ponto, terá de ser assegurado que os artifícios criados estão consistentes e corretos, de acordo com as políticas definidas para a organização e projeto. A aprovação do trabalho realizado, ficará dependente dos resultados dos processos de verificação e validação realizados.

Por fim, será também assegurada a escrita do "Relatório final do estágio" e feita a apresentação do mesmo.

A primeira entrega foi a de um relatório sobre o estudo de ferramentas de desenvolvimento. Este documento foi elaborado ao longo de um mês e teve a sua entrega na data prevista, ou seja, no dia 13 de Outubro de 2011.

Para a segunda entrega, pedia-se o documento de requisitos. Apesar do levantamento de requisitos e da elaboração de "casos de uso", tivessem sido realizados na data prevista, a fase de prototipagem exigiu mais tempo que o inicialmente previsto. De facto, constatou-se ser necessário proceder a um estudo mais cuidado da interação humano-dispositivo, bem como dos padrões de *design* do sistema operativo iOS. Como foi tomada a decisão de incluir protótipos gráficos (*mockups*) no documento de requisitos, tendo em mente possibilitar uma melhor compreensão do funcionamento da aplicação, a entrega deste documento acabou por ter que ser ligeiramente retardada.

De seguida foi elaborado o documento de especificação, que apesar de ter sido concluído no tempo previsto, acabou por ter que ser entregue um pouco mais tarde do que o inicialmente planeado, pelo arrastamento verificado na elaboração do documento de requisitos, anteriormente explicado.

Entregues e aprovados estes artefactos, iniciou-se o desenvolvimento de um protótipo funcional. Esta *milestone* já estava a ser preparada com o estudo da linguagem de programação utilizada no desenvolvimento de aplicações iOS na ferramenta Xcode, Objective-C. Apesar do atraso citado na elaboração do documento de requisitos e do documento de especificação, foi possível terminar o protótipo funcional no tempo inicialmente previsto.

Terminado o protótipo funcional e simultaneamente elaborado o relatório intermédio de estágio, foi desenvolvido um estudo e implementação da comunicação da aplicação com o servidor OOBIAN. Este estudo estava previsto ficar concluído no início do mês Fevereiro de 2012. Contudo, ocorreu a necessidade de um trabalho suplementar na elaboração da sua escrita e preparação da apresentação deste relatório, pelo que houve um deslizamento temporal de cerca de 15 dias.

Simultaneamente, em meados do mês de Fevereiro de 2012, foi apresentado o *relatório intermédio de estágio*. De seguida, foram utilizados alguns dias para corrigir alguns aspetos apontados da apresentação.

Para a implementação final da aplicação, que estava planeada terminar em final de Abril de 2012, face às razões atrás expostas, esta acabou por se ter de estender até ao final do mês de Maio de 2012.

Paralelamente foram elaborados os *casos de teste e elaborado um plano de testes*. Apesar dos conhecimentos em programação para dispositivos móveis iOS, por mim adquiridos durante o primeiro semestre, a pouca experiência consolidada no seu desenvolvimento, motivou, por vezes, mais algum tempo na implementação da aplicação.

Estes atrasos levaram a que o tempo útil para a fase de testes não fosse o ideal, obrigando a algum esforço adicional. Saliento, contudo, que todos os testes previstos foram executados e aprovados.

Terminada a fase de testes, foi compilada e escrita a documentação necessária, e por fim, iniciada a elaboração deste relatório de estágio.

Apresentam-se nas figuras seguintes as calendarizações referentes às tarefas previstas e às tarefas realizadas durante as duas grandes milestones.

Tarefas	Set/11	Out/11	Nov/11	Dez/11	Jan/12	Fev/12
T1						
T2						
T3						
T4						
T5						
T6						
T7						

Figura 23 – Planeamento inicial para o 1º semestre

Tarefas	Set/11	Out/11	Nov/11	Dez/11	Jan/12	Fev/12
T1						
T2						
T3						
T4						
T5						
T6						
T7						

Figura 24 – Calendarização das tarefas realizadas no 1º semestre

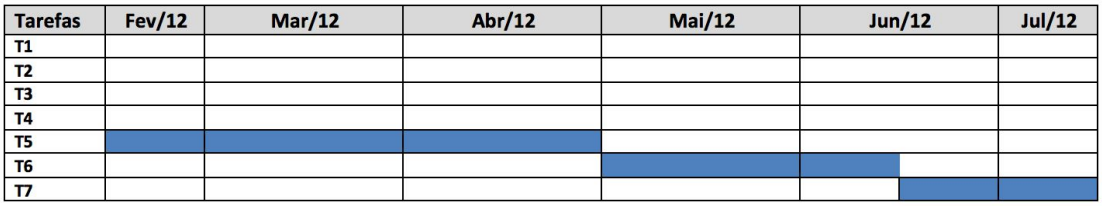
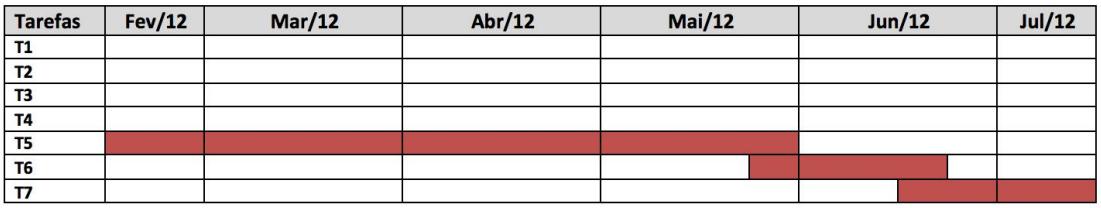


Figura 25 – Planeamento inicial para o 2º semestre



1

Figura 26 – Calendarização das tarefas realizadas no 2º semestre

Capítulo 7: Conclusão

7.1. Principais desafios

A integração na equipa do OOBIAN e o trabalho sobre uma plataforma num estado de desenvolvimento bastante avançado requer algum tempo de adaptação. Durante este tempo de adaptação é necessário esforço no sentido de minimizar esse mesmo tempo e de atingir um ritmo de trabalho autónomo.

O primeiro desafio prendeu-se com o estudo e compreensão do funcionamento da ferramenta OOBIAN, que exigiu bastante tempo de análise para perceber os diferentes componentes da ferramenta e que funcionalidades deveria um cliente para dispositivos móveis ter. Aliado a este factor, houve a aprendizagem de uma nova linguagem não muito comum, o Objective-C e de um novo ambiente de desenvolvimento, o Xcode, de novas tecnologias e de novas metodologias de desenvolvimento de *software*. Esta aprendizagem foi feita autonomamente, representando um desafio mais elevado mas ultrapassado devido à experiencia adquirida durante o percurso académico.

Outro desafio foi a pouca experiencia da empresa em desenvolvimento para dispositivos móveis, tendo sendo este o primeiro desenvolvimento de um produto para este tipo de dispositivos. Apesar da equipa OOBIAN ser extensa, os elementos da mesma não possuíam experiencia neste tipo de desenvolvimento e foi preciso também preparar o servidor e os pedidos para dispositivos móveis (com maiores exigências a nível do tamanho dos dados transmitidos).

7.2. Produto final e trabalho futuro

Analisando o estado atual da aplicação cliente iOS e observando todos os requisitos implementados, constata-se que todos os requisitos considerados no documento de especificação foram implementados. A aplicação passou em todos os testes a que foi submetida e encontra-se atualmente à espera do resultado da submissão na AppStore.

Em termos de futuro, existe muito trabalho possível de ser realizado sobre a aplicação. Em primeiro lugar, implementar os requisitos não considerados no documento de especificação, em particular as notificações, que terão que ser implementadas usando o centro de notificações da Apple.

Outro possível caminho para evolução do OOBIAN iOS, é a possibilidade implementar um chat (possibilidade em estudo também para o cliente Web) que facilitasse a comunicação entre os colaboradores de uma empresa que fizesse uso da plataforma OOBIAN. Outro aspeto interessante seria melhorar a pesquisa com base nos serviços de localização e permitir, por exemplo, pesquisar as empresas que efetuassem um determinado serviço presentes na zona da localização atual do utilizador e automaticamente gravar os seus contactos no dispositivo móvel.

7.3. Considerações finais

O estágio marca o início do fim do curso e representa o primeiro contacto com a realidade empresarial. O contacto com esta nova realidade veio-se a revelar uma experiência bastante positiva, posso afirmar sem sombra de dúvidas que foi um ano enriquecedor para a minha futura vida profissional. Não descurando a parte tecnológica, a área em que mais aprendi foi nos chamados *soft skills* e no trabalho em equipa. Esta experiência num ambiente empresarial revelou-se muito preciosa e fez-me ficar mais preparado para os novos desafios profissionais que se avizinham.

Em relação ao OOBIAN iOS e à parte tecnológica do estágio, foi um projeto bastante interessante e que me permitiu entrar numa área onde ainda não tinha muita experiencia, o desenvolvimento para dispositivos móveis. Além dos conhecimentos ganhos nesta área, foi importante por em prática outros conhecimentos adquiridos ao longo do meu percurso académica, conhecimentos em engenharia de *software* e em SoA (Service Oriented Architecture), por exemplo.

Este estágio, e consequentemente a aplicação desenvolvida, permitiu igualmente à empresa a sua primeira experiência em desenvolvimento de software para dispositivos móveis, trabalho que será continuado. Em relação à plataforma OOBIAN, fica claramente mais rica e interessante do ponto de vista comercial, com um cliente para dispositivos móveis iOS.

As dificuldades surgiram, como seria de esperar, porque o estágio obrigou a contactar com novas pessoas, tecnologias, uma nova cidade e até uma nova casa. No entanto cada uma das dificuldades encontradas foi uma nova lição aprendida que será uma mais-valia para o futuro.

Referências

[1] – OOBIAN. Acedida a 10 de Julho de 2012. Disponível em http://www.oobian.com/.

[2] – Morgan Stanley Research. *The Mobile Internet Report*. December 15, 2009.

[3] – Maisis. Acedida a 10 de Julho de 2012. Disponível em http://www.maisis.pt/.

[4] - Portugal Telecom Inovação, SA. Acedida a 10 de Julho de 2012. Disponível em http://www.ptinovacao.pt/.

[5] – iOS. Acedida a 10 de Julho de 2012. Disponível em http://www.apple.com/ios/.

[6] – Apple. Acedida 10 de Julho de 2012. Disponível em http://www.apple.com/.

[7] – Mac OS. Acedida a 10 de Julho de 2012. Disponível em http://www.apple.com/macosx/.

[8] – Darwin OS. Acedida a 10 de Julho de 2012. Disponível em http://www.puredarwin.org/.

[9] – iPhone 5. Acedida a 10 de Novembro de 2012. Disponível em http://www.apple.com/iphone/

[10] – Xcode. Acedida a 10 de Julho de 2012. Disponível em http://developer.apple.com/technologies/tools/.

[11] – Xcode Developer Tools. Acedida a 10 de Julho de 2012. Disponível em http://developer.apple.com/technologies/tools/features.html.

[12] – PhoneGap. Acedida a 10 de Julho de 2012. Disponível em http://phonegap.com/.

[13] – HockeyKit. Acedida a 10 de Julho de 2012. Disponível em http://hockeykit.net/.

[14] – FIELDING, R. T. 2000. *Architectural Styles and the Design of Network-based Software Architectures*, 76-106.

[15] – FIELDING, R. T. 2000. *Architectural Styles and the Design of Network-based Software Architectures*, 107-147.

[16] – RICHARDSON, L.;RUBY, S. 2007. *RESTful Web Services*. O'Reilly.

[17] – TILKOV, S. 10 de Dezembro de 2007. *A Brief Introduction to REST* [Internet]. Acedida a 6 de Julho de 2012. Disponível em http://www.infoq.com/articles/rest-introduction.

[18] – JSONKit. Acedido a 5 de Novembro de 2012. Disponível em https://github.com/johnezang/JSONKit.

[19] – TouchJSON. Acedido a 5 de Novembro de 2012. Disponível em https://github.com/TouchCode/TouchJSON.

[20] – NextiveJSON. Acedido a 5 de Novembro de 2012. Disponível em https://github.com/nextive/NextiveJson.

[21] – SBJSON. Acedido a 5 de Novembro de 2012. Disponível em http://stig.github.com/json-framework/.

[22] – *JSON Libraries for iOS Comparison.* Acedido a 10 de Julho de 2012. Disponível em http://www.bonto.ch/blog/2011/12/08/json-libraries-for-ios-comparison-updated/.

[23] – BSON. © http://bsonspec.org/.

[24] – Google. *Protocol Buffers – Google's data interchange format* [Internet].Acedida a 10 de Julho de 2012. Disponível em http://code.google.com/p/protobuf/.

[25] – XML. Acedido a 5 de Novembro de 2012. Disponível em http://www.w3schools.com/xml/.

[26] – *thrift-protobuf-compare*. Acedido a 10 de Julho de 2012. Disponível em http://code.google.com/p/thrift-protobuf-compare/wiki/Benchmarking.

[28] – Apple. *iOS Human Interface Guidelines* [Internet]. Acedido a 10 de Julho de 2012. Disponível em http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html.

[28] – Balsamiq Mockups. Acedido a 10 de Julho de 2012. Disponível em http://www.balsamiq.com/products/mockups.

[30] – GRADY, R. *Practical Software Metrics for Project Management and Process Improvement*. Prentice-Hall, 1992

[31] – MYERS, J. 1979. *The Art Of Software Testing*. Glenford.

[32] – DIJKSTRA, E. W. 1972. *Notes On Structured Programming*.

[33] – *iOS Human Interface Guidelines*. Acedido a 10 de Julho de 2012. Disponível em http://developer.apple.com/library/ios/#DOCUMENTATION/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html

[34] – *App Store Review Guidelines*. Acedido a 10 de Julho de 2012. Disponível em https://developer.apple.com/appstore/guidelines.html