



UNIVERSIDADE DE ÉVORA

ESCOLA DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

Desenvolvimento de uma aplicação para dispositivos móveis - Gestor de conteúdos e workflows, denominado iScriptor

Ricardo Jorge Cerveira Dias

Orientação: Vítor Manuel Beires Pinto Nogueira

Mestrado em Engenharia Informática

Dissertação

Évora, 15 de julho de 2015



UNIVERSIDADE DE ÉVORA

ESCOLA DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

Desenvolvimento de uma aplicação para dispositivos móveis - Gestor de conteúdos e workflows, denominado iScriptor

Ricardo Jorge Cerveira Dias

Orientação: Vítor Manuel Beires Pinto Nogueira

Mestrado em Engenharia Informática

Dissertação

Évora, 15 de julho de 2015

Sumário

Atualmente a “invasão” de dispositivos móveis no nosso dia-a-dia, dotados de alto processamento, sensores e funcionalidades, capazes de substituir, em muitos casos, o PC para uso pessoal e profissional, tornou imperativa a migração de produtos informáticos para este novo paradigma, que de outra forma, serão, conseqüentemente, ultrapassados por produtos da concorrência. O Scriptor Server é um desses casos, tornando-se, deste modo, fundamental, a sua recriação para este novo paradigma, denominado mobile. O desafio foi proposto pela empresa Viatecla, com o objetivo de desenvolver um protótipo funcional do Scriptor Server, orientado para dispositivos móveis. Além de se falar das metodologias adotadas na criação deste tipo de aplicativos e de se fazer um estudo minucioso das tecnologias e ferramentas de programação existentes, é também descrito, em pormenor, o desenvolvimento da recriação do Scriptor Server numa App denominada iScriptor, analisando-se os seus requisitos e arquitetura. Enumerar-se-ão, ainda, as principais dificuldades sentidas na criação da App e apresentar-se-ão as conclusões e as melhorias passíveis de serem introduzidas no sistema.

*Developing an application for mobile devices - iScriptor:
content and workflow manager*

Abstract

The current "invasion" of mobile devices in our day-to-day lives, equipped with high speed processors, sensors and features capable of replacing, in many cases, a computer for personal and/or professional use, makes the migration of computer products towards this new paradigm imperative, otherwise they will be replaced by competitive products. The Scriptor Server is a crucial example for the current mobile paradigm. The challenge presented by Viatecla aims to develop a working prototype of the Scriptor Server for mobile devices. Besides discussing the methodologies adopted in the creation of such applications and after conducting a thorough study of existing programming tools and technologies, a detailed description is presented regarding the development and recreation of a Scriptor Server App called iScriptor. Such a description focuses on the analysis of its requirements and architecture. A list of the main difficulties in the creation of this App will be presented by the end of this work, as well as conclusions and improvements that can be introduced into the system.

aos meus pais

Agradecimentos

Gostaria de expressar o meu sincero agradecimento pelo apoio durante a realização e desenvolvimento desta dissertação:

Ao meu orientador, Prof. Doutor Vítor Manuel Beires Pinto Nogueira, pela sua disponibilidade, orientação, críticas construtivas e por todo o apoio oferecido, ao longo deste processo de crescimento profissional e pessoal que foi a elaboração desta dissertação.

À Empresa Viatecla Soluções Informáticas e Comunicações Lda, personificada no Eng.º Rui Estêvão, Eng.º João Vieira, entre muitos outros, pelo apoio e compreensão demonstrados, bem como o pronto auxílio prestado, por toda a paciência demonstrada e pelo conhecimento transmitido.

Aos meus colegas da Engenharia Informática e principalmente à Marília Santos e Domingos Martins pelo companheirismo e apoio.

Aos meus pais, José Dias e Maria Dulce, por serem os pais maravilhosos que são, por me apoiarem em cada etapa da minha vida e por terem sempre acreditado em mim, incentivando-me a continuar a estudar.

E claro, ao meu amor, Mafalda Costa, por ter sido o meu braço direito ao longo destes doze anos, ajudando-me a ultrapassar todos os obstáculos, não me deixando nunca desistir e ajudando-me sempre a tomar as melhores decisões. Obrigada pelo teu apoio, paciência, compreensão e companheirismo.

Acrónimos

- API** Application Programming Interface
- APP** Aplicação movel
- BD** Base de Dados
- B2B** Business to business
- CERN** European Organization for Nuclear Research
- CPU** Central Processing Unit
- CSS** Cascading Style Sheets
- GPS** Global Positioning System
- HTML** HyperText Markup Language
- HTTP** Hypertext Transfer Protocol
- ISO** International Standard Organization
- JS** JavaScript
- MVC** Model-view-controller
- PC** Personal Computer
- PHP** Hypertext Preprocessor
- QUIS** Questionnaire for user interface satisfaction
- REST** Representational State Transfer
- RWD** Responsive Web Design
- SO** Sistema Operativo

SOAP Simple Object Access Protocol

SUMI Software Usability Measurement Inventory

SSL Camada de transporte segura

UI User Interface

WHATWG Web Hypertext Application Technology Working Group

W3C World Wide Web Consortium

XML Extensible Markup Language

Conteúdo

Sumário	i
Abstract	iii
Lista de Conteúdo	xiii
Lista de Figuras	xvi
Lista de Tabelas	xvii
1 Introdução	1
1.1 Objetivos	2
2 Estado de Arte	5
2.1 Introdução	5
2.2 Dados de Mercado sobre Sistemas Operativos Móveis	6
2.3 Abordagens ao Desenvolvimento	8
2.3.1 Nativa	8
2.3.2 Web	8
2.3.3 Híbrida	9
2.4 Nativa vs Web vs Híbrida	9
2.4.1 Desempenho	9
2.4.2 Compatibilidade	9
2.4.3 <i>Offline</i>	10
2.4.4 Tempo de desenvolvimento e Distribuição	10
2.4.5 Segurança	10

2.4.6	Custos	11
2.5	HTML5	11
2.5.1	Web-Storage	12
2.5.2	<i>Offline</i> Web	13
2.5.3	Geolocalização	14
2.5.4	Vídeo	14
2.5.5	Canvas	14
2.5.6	JavaScript	14
2.5.7	CSS	15
2.6	Adaptive and Responsive Web Design	16
2.6.1	Grid flexível	17
2.6.2	Conteúdos flexíveis de imagem e vídeo	18
2.6.3	Media Queries	20
2.7	Testes de Usabilidade	22
2.8	Futuro	27
2.8.1	Tizen	28
2.8.2	Firefox OS	28
2.8.3	Ubuntu Phone OS	28
3	Especificação do Projeto	31
3.1	Introdução	31
3.2	Scriptor Server	31
3.3	Casos de Estudo	34
3.3.1	Metronic	35
3.3.2	Keyhole Gestor Identidades	36
3.3.3	FT Financial Times Journal	37
3.3.4	Conclusões	39
3.4	iScriptor	40
3.5	Web Service	41
3.6	Análise de Requisitos e de Utilizadores	42
3.7	Arquitetura	48
3.8	Mockups	49
3.9	Execução de Testes de Usabilidade	53
4	Implementação do Projeto iScriptor	57
4.1	Introdução	57

CONTEÚDO	xiii
4.2 Abordagem	57
4.3 Projeto iScriptor	59
4.4 Interface Responsive	59
4.5 Armazenamento	64
4.6 Modo <i>Offline</i>	65
4.7 Sincronização com o Servidor	68
5 Conclusões	71
5.1 Trabalho Futuro	72
Referências bibliográficas	73

Lista de Figuras

2.1	Sistemas Operativos mobile no Mundo	6
2.2	Android vs Apple apps na Alemanha e Inglaterra em 2015	7
2.3	Quota de mercado dos Sistemas Operativos mobile em Portugal	7
2.4	Representação do HTML5	12
2.5	Grid Flexible consoante o tipo de dispositivo	17
2.6	Adaptive Image em diferentes dispositivos	19
2.7	Exemplo de uma Media Query	20
2.8	Exemplo código html	21
2.9	Regras de acordo com o tamanho do viewport	21
2.10	Problemas de Usabilidade detetados consoante o número de avaliadores	26
2.11	Gráfico de utilização de três SO mobile	27
2.12	Ubuntu Phone OS	29
3.1	Interface Scriptor Server com a descrição de alguns componentes	32
3.2	Interface Scriptor Server, formulário de um determinado Conteúdo	33
3.3	Interface do Site do cliente da Plataforma Scriptor Server. Lista de ofertas da Agência de Viagens	33
3.4	Resultado final da inserção de um novo Conteúdo (oferta de viagem)	34
3.5	Interface Metronic para Tablet e PC	35
3.6	Interface Metronic para Smartphone	36
3.7	Interface da web App FT	39
3.8	Resposta JSON do Scriptor Server a um pedido <i>'channelList'</i> feito pela aplicação iScriptor	42
3.9	Arquitetura iScriptor e Scriptor Server	49

3.10	Login da aplicação no iPad	50
3.11	Lista de Canais e Conteúdos vista através de um Tablet	51
3.12	Lista de canais vista através de um Smartphone	51
3.13	Vista de um Conteúdo através de um smartphone	52
3.14	Vista de um Conteúdo através de um Tablet	52
3.15	Vistas, Filtros e Ordenação na lista de Conteúdos num Tablet	53
3.16	Formulário de avaliação de teste sobre ícones	54
3.17	Diagrama de Estados do iScriptor	55
3.18	Avaliação heurística ao iScriptor. Problemas detetados	56
4.1	Processo de desenvolvimento da App iScriptor	58
4.2	iScriptor vista Tablet	61
4.3	Lista de Conteúdos widget para seleccionar colunas	61
4.4	vista de um Conteúdo	62
4.5	Editar formulário do Conteúdo	62
4.6	Tipo de Vista da lista de Conteúdos, agrupada por uma determinada coluna	63
4.7	iScriptor vista Smartphone	63
4.8	Conteúdo visto através de um smartphone, sem possibilidade de edição	64
4.9	Sistema de Armazenamento do iScriptor	65
4.10	Ficheiro manifest que armazena os ficheiros indicados em cache	67
4.11	Gravar web App nos dispositivos mobile	68
4.12	Sincronização com o Servidor sem ligação à Internet	69
4.13	Sincronização com o Servidor com ligação à Internet	70

Lista de Tabelas

3.1	Autenticação do utilizador na App iScriptor.	43
3.2	Utilizador escolhe Conteúdo na App iScriptor.	44
3.3	Utilizador visualiza Conteúdo na App iScriptor.	44
3.4	Utilizador actualiza Conteúdo na App iScriptor.	45
3.5	Utilizador pesquisa sobre Conteúdos na App iScriptor.	45
3.6	Utilizador ordena lista de Conteúdos através da coluna pretendida na App iScriptor.	46
3.7	Utilizador escolhe vista de conteúdos na App iScriptor.	46
3.8	Utilizador edita Conteúdo na App iScriptor.	47
3.9	Utilizador filtra as colunas que pretende visualizar na lista de Conteúdos na App iScriptor.	48
3.10	Utilizador faz Logout na App iScriptor.	48

Capítulo 1

Introdução

Vivemos numa era em que os dispositivos móveis fazem parte do nosso dia-a-dia, o seu poder de processamento e as suas capacidades funcionais aumentam, exponencialmente, a cada dia que passa, assim como os modelos de negócio que envolvem estes dispositivos. O aparecimento de comunidades adeptas da utilização destes dispositivos e, nomeadamente da troca e partilha de ideias, tem contribuído para o seu desenvolvimento.

Porém, nem tudo é assim tão simples, pois desenvolver aplicações para Smartphones é diferente de desenvolver aplicações para Desktops, devido às características dos dispositivos móveis, nomeadamente o tamanho do monitor. Outro dos grandes problemas é a quantidade de diferentes dispositivos e os seus sistemas operativos (iOS, Android, Windows Phone, etc.) incompatíveis uns com os outros, chegando mesmo a haver aplicações incompatíveis com o mesmo SO, mas de versões diferentes (o SO Android é um bom exemplo, dado que aplicações que trabalham na versão 2.2 não trabalham na versão 2.3). A construção de aplicações nativas para cada dispositivo e SO torna grandes os gastos na programação dessas aplicações, visto que é preciso criar uma nova aplicação para cada um dos sistemas operativos, bem como alterar o código fonte para as diferentes versões do mesmo SO.

Por tudo isto, o aparecimento de linguagens de programação, que conseguissem melhorar a compatibilidade dessas aplicações com os diversos sistemas, foi inevitável. Estas novas linguagens baseadas em tecnologias web vieram colmatar o fosso que existia entre as diferentes plataformas. Conseguindo, estas, atingir todas as plataformas, fazendo correr as aplicações em qualquer *browser* instalado no dispositivo móvel.

Importa saber, então, todas as suas vantagens e desvantagens, para se tentar perceber se realmente estas novas tecnologias vieram para ficar e substituir os métodos até hoje

utilizados no desenvolvimento de aplicações.

1.1 Objetivos

O objetivo do projeto, realizado na empresa Viatecla, era desenvolver uma aplicação para dispositivos móveis, denominada iScriptor, a partir da aplicação existente, Scriptor Server, usada apenas em Desktops. Neste sentido, o objetivo primordial desta dissertação é explicitar detalhadamente o desenvolvimento da App iScriptor.

Assim, depois da Introdução (primeira parte deste trabalho), na segunda parte, far-se-á uma abordagem teórica sobre os assuntos mais relevantes no desenvolvimento de uma aplicação para dispositivos móveis. Em primeiro lugar, será feito um estudo relativamente aos dados de mercado sobre sistemas operativos. Depois, far-se-á um resumo das abordagens existentes para o desenvolvimento de um projeto deste género, sendo elas: Nativa, Web, Híbrida, seguindo-se, então, uma comparação das três abordagens no que respeita ao desempenho, compatibilidade, trabalho *offline*, tempo de desenvolvimento e distribuição, segurança e custos. Seguir-se-á uma explicação sobre a tecnologia HTML5. Neste ponto falar-se-á, também, do Web-Storage, Offline Web, Geolocalização, Vídeo, Canvas, JavaScript e CSS. O ponto seguinte desenvolverá o Adaptive and Responsive Web Design e os seus elementos constituintes: Grid Flexível, Conteúdos Flexíveis de imagem e vídeo, e Media Queries. Segue-se, depois, uma abordagem aos Testes de Usabilidade, que se referem, de uma maneira geral, a um conjunto de métodos que se alicerçam em examinar os aspetos de usabilidade de uma *interface*, por parte de avaliadores/futuros utilizadores, visando encontrar problemas de usabilidade na mesma e, conseqüentemente, em arranjar forma de os solucionar. Para terminar, falar-se-á do futuro ao nível dos sistemas operativos para dispositivos móveis. De referir que neste ponto se irá falar daquilo que era futuro aquando da realização do projeto e não do que será futuro tendo como ponto de partida esta dissertação.

Na terceira parte desta dissertação, referente à Especificação do Projeto, apresentar-se-ão a plataforma Scriptor Server e o planeamento da App iScriptor. Serão também, apresentados os Casos de Estudo: Metronic, Keyhole Gestor de Identidades e FT Financial Times Journal. Aqui retirar-se-ão, claro está, algumas conclusões relativamente à implementação, comunicação ao servidor, linguagens, sistemas de armazenamento, modo *offline*, etc. Falar-se-á, ainda, da Análise de Requisitos, da Arquitetura, dos Web Service, dos Mockups e da Execução dos Testes de Usabilidade.

Para concluir, na quarta parte, que diz respeito à Implementação do iScriptor, será, inicialmente, feita uma abordagem geral do desenvolvimento da aplicação. Falar-se-á, em seguida, de como foi criada e desenvolvida a Interface Responsive do iScriptor. Segue-se uma descrição do tipo de Armazenamento e da possibilidade de trabalho *Offline*. Por fim, far-se-á uma descrição da Sincronização da App com o servidor.

Posto isto, é necessário referir que este trabalho contribui para se perceber que existem di-

ferentes tipos de abordagem no desenvolvimento de uma aplicação móvel (Nativa, Híbrida ou Web), quando se deve optar por uma em detrimento de outra, de acordo com o(s) objetivo(s) que se pretende(m) alcançar. Permite também perceber que ao serem feitas escolhas é necessário entender que existem várias maneiras de atingir as funcionalidades pretendidas e que também no caso das estratégias e arquiteturas existem umas melhores do que outras, devendo ser escolhidas segundo os objetivos estabelecidos. Considero que este trabalho, contribui, ainda, para compreender se o HTML5 é uma tecnologia capaz de ombrear com outras tecnologias usadas na criação de aplicações móveis.

Capítulo 2

Estado de Arte

2.1 Introdução

O desenvolvimento de aplicações móveis tem tido um crescimento exponencial, devido à popularidade e massificação dos dispositivos móveis, o que tem impulsionado ainda mais a área da informática.

Neste capítulo será dada uma perspectiva do mercado de vendas, quer de dispositivos, quer de aplicações móveis, para se tentar perceber quais os SOs mais utilizados e porquê.

Abordar-se-ão, também, temas relacionados com o desenvolvimento e modelação de aplicações móveis. Começar-se-á, então, por analisar três abordagens: o desenvolvimento de aplicações Nativas, aplicações Web e aplicações Híbridas.

De seguida, falar-se-á das novas funcionalidades do HTML5, tecnologia desenvolvida pelo W3C, a partir do HTML.

Na secção seguinte, irá apresentar-se o Adaptive and Responsive Web Design que veio dar uma ajuda preciosa na criação de páginas web que respondem ao tamanho de qualquer ecrã, através de um conjunto de normas e ferramentas.

Serão, seguidamente, abordadas algumas das tecnologias mais utilizadas no desenvolvimento de aplicações Web e os cuidados que se têm no desenvolvimento de interfaces para o utilizador, nomeadamente os Testes de Usabilidade.

Por fim, falar-se-á do futuro dos sistemas operativos que, como foi mencionado nos objetivos desta dissertação, se refere ao futuro tendo em conta aquilo que era futuro aquando da realização do projeto e não do que será futuro partindo da data desta dissertação.

2.2 Dados de Mercado sobre Sistemas Operativos Móveis

Hoje em dia, é possível ter alguns exemplos de más estratégias seguidas, que passam, entre outros casos, pela aposta num determinado sistema operativo ou numa determinada tecnologia, como é o caso do BlackBerry OS e do HTML5, com o Facebook.

Por tudo isto, torna-se sensato pensar que a escolha na estratégia é um dos passos mais importantes no desenvolvimento de qualquer App móvel, pois um passo em falso poderá resultar em custos avultados para a empresa, tornando-se, assim, necessário, um estudo aprofundado do caminho a seguir.

O SO Android é líder destacado a cada ano que passa como se pode verificar na Figura 2.1. Uma das explicações poderá passar pelo número de Apps existentes para este SO, ultrapassando já as 1 500 000. Deste modo, o número de Apps pode explicar o baixo uso de dispositivos com Windows Phone, que apresentam um número bem inferior de Apps, comparando com o número anteriormente referido. Por outro lado, não explica a diminuição do uso do iOS a favor do Android, sendo que, neste caso, a explicação poderá ser outra, como por exemplo o reduzido número de modelos e o elevado custo dos dispositivos da Apple, comparados com as inúmeras opções de modelos e preços mais em conta dos dispositivos com o Android.

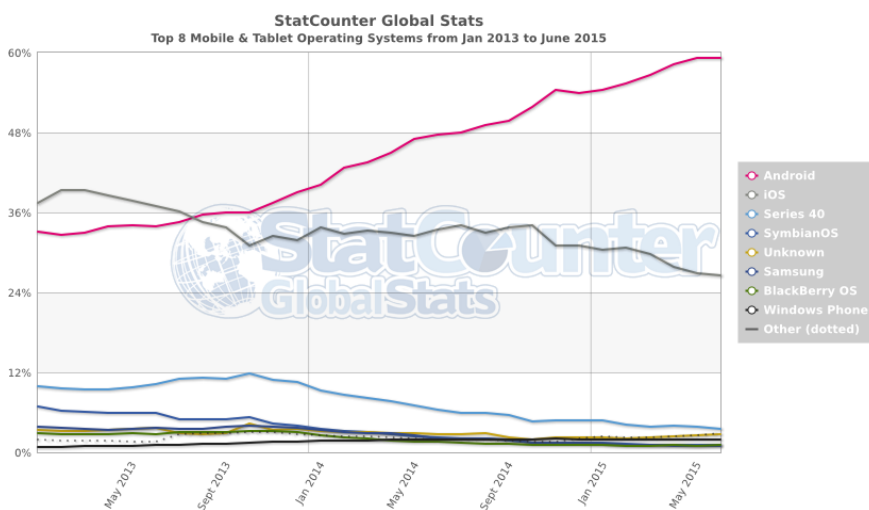


Figura 2.1: Sistemas Operativos mobile no Mundo Fonte:<http://statcounter.com>, consultado a 2 de junho de 2015.

Perante os números apresentados na Figura 2.1 há quem não queira seguir a tendência do mercado, apontando como grande defeito o facto dos utilizadores do Android não estarem dispostos a comprar Apps, tornando-se desinteressante o desenvolvimento de Apps para esse SO. Ao longo dos tempos esta afirmação foi verdadeira, no entanto, presentemente, começa a tornar-se falsa, como é possível verificar através da Figura 2.2 que mostra um aumento na compra de Apps no SO Android por parte dos seus utilizadores comparati-

vamente com o seu opositor iOS (utilizadores dos sistemas da Apple são conhecidos por não se preocuparem em despendere dinheiro para ter acesso a uma determinada aplicação). Podendo, assim, afirmar-se que o Android terá "roubado" clientes ao iOS ou, então, as Apps do Android serem menos dispendiosas do que as do iOS.

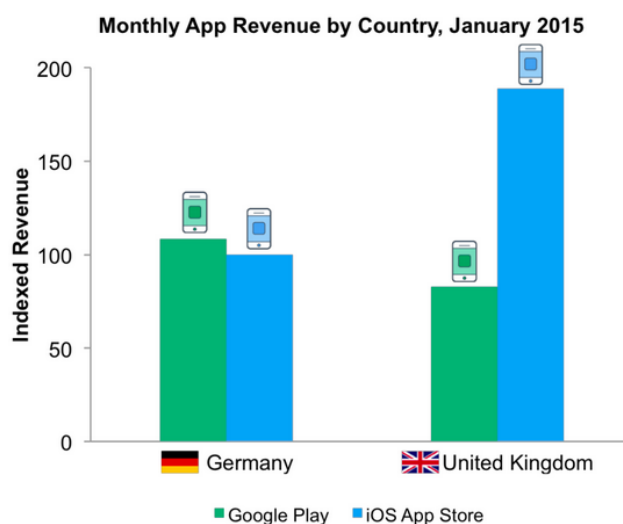


Figura 2.2: Android vs Apple Apps na Alemanha e Inglaterra em 2015. Fonte: <http://venturebeat.com>, consultado a 1 de junho de 2015.

Em Portugal, como é possível verificar através da Figura 2.3, o cenário não difere do do resto do mundo, o Android lidera, sendo que em Portugal a margem entre os dois principais SOs não é tão acentuada, uma vez que o iOS tem uma quota de mercado acima dos trinta por cento.

2.3 Abordagens ao Desenvolvimento

2.3.1 Nativa

Atualmente, existem três gigantes na indústria de dispositivos móveis que ditam as regras: a Apple, a Samsung e a Google. A Apple é líder de vendas de Tablets e obtém a segunda posição de vendas de Smartphones, utilizando nos seus dispositivos um SO proprietário denominado iOS. A Samsung lidera as vendas de Smartphones, utilizando o Android como SO. A Google, por sua vez, é líder destacada no mercado de SOs utilizados em dispositivos móveis também com o Android.

Tanto a Apple como a Google criaram lojas online próprias denominadas App Stores, para venda e distribuição de aplicações móveis. Estas lojas revolucionaram o mercado do *software* criando um modelo de negócio inovador na sua venda e capazes de beneficiar monetariamente tanto o Programador como o Consumidor.

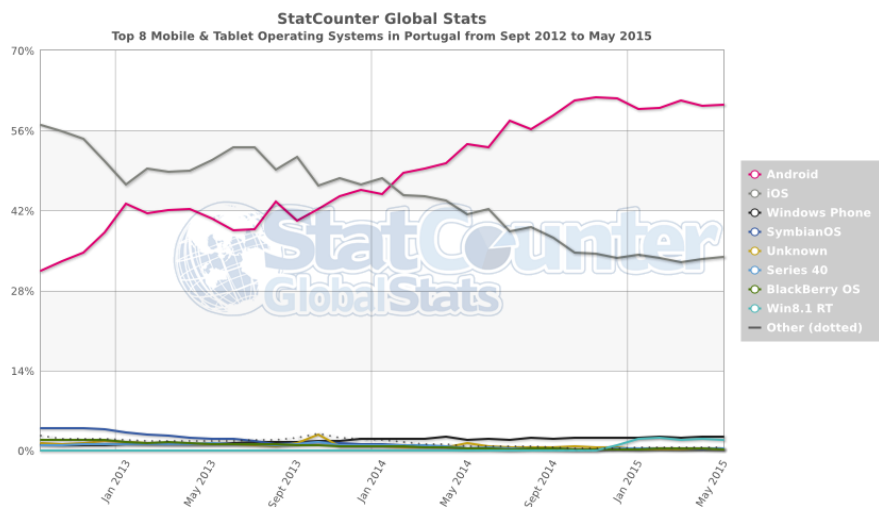


Figura 2.3: Quota de mercado dos Sistemas Operativos mobile em Portugal
 Fonte: <http://statcounter.com>, consultado a 2 de junho de 2015.

Estas aplicações são criadas propositadamente para um determinado SO, utilizando a mesma linguagem do SO do dispositivo, permitindo, desta forma, o acesso completo aos recursos do seu *hardware*, não sendo, contudo, possível o uso da App noutra SO. O uso de uma aplicação implica a sua transferência e instalação. É das três abordagens aquela que tem maior nível de desempenho, mas, em contrapartida, é também a mais complexa de desenvolver, acarretando tempo e custos elevados.

2.3.2 Web

Uma aplicação Web para dispositivos móveis é simplesmente uma página web e, como tal, utiliza tecnologias web no seu desenvolvimento. A tecnologia mais usada no desenvolvimento destas aplicações é o HTML5.

Este tipo de aplicações não é mais do que uma página web visualizada através de um *browser*, desde que este aceite HTML5. O servidor deteta, a cada pedido, qual o tipo de dispositivo móvel de que se trata e ajusta-se mediante as suas características, sejam Smartphones com um pequeno ou grande ecrã ou Tablets.

Hoje em dia, quase todos os *browsers*, pelo menos os mais utilizados (Chrome, Firefox, Opera, Safari) nas suas versões mais recentes, aceitam este tipo de tecnologia, não sendo, portanto, necessária qualquer instalação da aplicação no dispositivo. É das três abordagens a que tem menor nível de desempenho, sendo, em contrapartida, a menos complexa de desenvolver e com maior compatibilidade entre os vários SOs acarretando, assim, menor tempo e custos.

2.3.3 Híbrida

As aplicações Híbridas são a conjugação das duas abordagens anteriores (Nativas e Web), tentando aproveitar o melhor de cada uma. Tomando uma visão simplista do que é uma aplicação Híbrida, pode dizer-se que no seu desenvolvimento existem duas camadas: camada principal e menos complexa, desenvolvida em tecnologias Web (HTML5, CSS3, JS), facilitando o uso desse código para os outros SO, diminuindo os custos no seu desenvolvimento; e camada desenvolvida na linguagem nativa do SO do dispositivo móvel que fica encarregue da ligação às funcionalidades do dispositivo, assegurando, assim, o acesso a todos os sensores e funcionalidades do mesmo.

Uma aplicação Híbrida pode muito bem passar por uma aplicação Nativa, pois, por exemplo é possível fazer o *download* da aplicação Híbrida, através das App Stores e posteriormente instalá-la, tal como acontece com as aplicações Nativas. Para o utilizador é quase indistinguível se se trata de uma App Nativa ou de uma App Híbrida.

2.4 Nativa vs Web vs Híbrida

2.4.1 Desempenho

Perante as informações recolhidas, as Aplicações Nativas são, sem dúvida, as que têm melhor desempenho, aguentando gráficos melhor do qualquer uma das outras abordagens. No caso da criação de jogos indubitavelmente que a melhor abordagem é a Nativa, por tudo o que implica: gráfico, fluidez e processamento. As Web são as mais lentas das três abordagens, sendo a velocidade da rede o principal obstáculo no seu desempenho. As Híbridas são um compromisso entre as outras duas, estando, assim, vocacionadas para Apps de empresas dadas as suas potencialidades, ao tentarem obter o melhor dos dois mundos (Nativo e Web).

2.4.2 Compatibilidade

Quando se fala em compatibilidade pensa-se em fazer uma App que sirva a todos os SO, sem necessitar de qualquer tempo e custo extras para o seu funcionamento. É aqui que as Nativas apresentam o seu principal problema, devido ao qual se equaciona, frequentemente, a desistência na sua implementação. O desenvolvimento de uma App Nativa para um dado SO não possibilita o seu uso noutro SO, pois são incompatíveis entre si e, visto que existem dois grandes SOs usados (iOS e Android), que se distinguem da restante concorrência, torna-se relevante pensar e ponderar a implementação de uma App Nativa. Aliado a tudo isto, chegam novas notícias dando conta que, num futuro próximo, novos SOs poderão ser criados e lançados no mercado, tendo, cada um, as suas armas para tentar destronar os já instalados. Assim, a aposta recai nas aplicações Web e Híbrida que ganham força perante este cenário.

Apesar das Aplicações Híbridas necessitarem de código Nativo, este é muito menor e menos complexo comparando com o de uma App Nativa, podendo sempre aproveitar-se todo o código produzido em HTML5/CSS3/JS. Já as Web Apps são as mais compatíveis entre os vários SOs, necessitando apenas da informação do tipo de dispositivo, de modo a ajustar a aplicação ao ecrã deste.

2.4.3 *Offline*

É muito importante que uma App móvel trabalhe em modo *Offline*, visto que, hoje em dia, o acesso à Internet, em dispositivos móveis, através da rede móvel, é pago, nomeadamente o tráfego que é contabilizado, sendo que quanto maior for esse tráfego, maior será o seu custo. Deste modo, é de extrema importância conseguir trabalhar em modo *Offline*. Qualquer das três abordagens referidas consegue trabalhar desta forma, embora no caso das Web Apps esta funcionalidade seja limitada pelo *browser* usado (AppCache, Web Storage).

2.4.4 Tempo de desenvolvimento e Distribuição

As Web Apps, em média, são as que gastam menos tempo na sua implementação graças às tecnologias usadas. A sua distribuição é a mais rápida das três abordagens, bastando apenas lançar a App *online* na Internet. Todavia, em termos de visibilidade é pior comparativamente com as outras abordagens, visto que, nos nossos dias, as App Stores revolucionaram a procura e venda de Apps e, estando estas inibidas de entrar nessas lojas, torna-se difícil a sua visibilidade. No entanto, hoje existem Web Stores que começam a fazer alguma concorrência às App Stores da Apple e da Google. A este nível as Apps Híbridas levam vantagem sobre as outras duas, pois, apesar de levarem mais tempo no seu desenvolvimento do que as Web, conseguem ter os dois métodos de distribuição (Web, App Store). Já as Apps Nativas são as mais complexas e com uma implementação mais morosa, sendo o seu método de distribuição através das App Stores.

2.4.5 Segurança

Tanto nas Web Apps como nas Apps Híbridas os problemas de segurança são quase os mesmos, apesar de já existirem preocupações a esse nível por parte dos *browsers*, que têm em conta alguns tipos de ataque (*malicious scripts* e *cross domain requests*). No entanto, se se quiser, as Apps Híbridas podem usar *native controls*, como é o caso da *uiWebView* (iOS) e *WebView* (Android), oferecendo, assim, menor vulnerabilidade a ataques como:

- Cross site scripting
- Cross site Request Forgery
- SQL Injections

- Ataques à Base de Dados local (BD do dispositivo não se encontra encriptada)

2.4.6 Custos

Os custos são um dos pontos mais importantes na escolha de uma determinada abordagem. Neste aspeto, as Web e Híbridas ganham vantagem face às Nativas. O tempo de implementação é menor, equivalendo a menos horas de trabalho podendo mesmo ser usado e reutilizado esse código noutros SO móveis. Por sua vez, para implementar uma App Nativa nos principais SO móveis (Android, iOS, Windows Phone, BlackBerry) é necessário escrever quatro códigos diferentes, levando à elaboração de quatro aplicações diferentes, sendo, deste modo, os custos multiplicados por quatro. Quanto aos custos de manutenção, também aqui, nas Apps Nativas, o cenário não difere, sendo que os custos continuam bastante superiores. Enquanto que nas aplicações Web e Híbridas o código feito em Tecnologias Web se mantém inalterado, devido aos *standards*, requerendo apenas pequenos ajustes aquando do lançamento das novas versões dos SOs, nas Nativas há ainda, outro problema, isto é, requerem um trabalho maior e profundo quando novas versões dos SOs móvel são lançadas.

2.5 HTML5

O HTML (Hyper Text Markup Language), criada por Tim Berners Lee, em 1989, no CERN [18], é uma linguagem utilizada para estruturar e representar conteúdo, através de uma página Web. Esta Linguagem foi-se desenvolvendo, ao longo dos tempos, através da World Wide Web Consortium (W3C), com o propósito de assegurar o sucesso da Internet.

O HTML passou por diversas versões até ser tornada um *standard* internacional ISO/IEC, sendo a versão em causa o HTML 4.01, publicado no ano de 1999.

Em 2004, novamente pelas mãos da W3C, começou a ser desenvolvida uma nova versão HTML, esta denominada agora HTML5. Porém, só em 2008, com a ajuda de algumas empresas que formavam o Web Hypertext Application Technology Working Group (WHATWG) e, logo, as principais interessadas na evolução do HTML e nas tecnologias ligadas à mesma, foi lançado ao grande público.

Em julho de 2012, a WHATWG e a W3C decidiram estabelecer um grau de separação, de forma a que a W3C ficasse responsável pela especificação do HTML5 e a WHATWG pela sua standardização, a que deram o nome de “Living Standard”. Esta decisão teve como objetivo fazer evoluir uma tecnologia sem que esta esteja alguma vez completa, podendo ser sempre atualizada e melhorada, pela adição de recursos, mas sem nunca serem removidas as suas funcionalidades. Esta quinta versão não deixou de ser fiel às suas anteriores versões, sendo uma linguagem de marcação, usada para estruturar e apresentar conteúdo na web.

O HTML5 não funciona apenas com HTML necessitando, igualmente, de Javascript e CSS para definir estilos, dinamizar e ainda usar as APIs disponibilizadas. O CSS é utilizado para especificar a aparência e a formatação de um documento HTML, promovendo a separação entre a formatação e o conteúdo de um documento. A utilização do Javascript marcou o início da era do *client-side scripting* das páginas web. Aumentando o seu potencial, permitindo, de forma dinâmica, editar, criar ou remover informação contida numa página HTML. Esta quinta versão, além das novas funcionalidades, trouxe consigo mudanças importantes, nomeadamente a nível da semântica e acessibilidade, incorporando novos recursos, que anteriormente só eram possíveis recorrendo a outras tecnologias.

Em Outubro de 2014, o HTML5 foi designado como *Recommendation*. Esta nova versão do HTML não necessita de qualquer instalação de *plug-ins*, fornecendo um conjunto de funcionalidades, de entre as quais se podem realçar: Canvas; Video; *Offline Web*; Geolocalização; Web Storage.



Figura 2.4: Representação do HTML5 Fonte:<http://churchm.ag/>, consultado a 5 de junho de 2015.

As fases de desenvolvimento, definidas para as funcionalidades são:

- First Draft - fase inicial;
- Working Draft - fase inicial, mas mais madura relativamente ao *First Draft*;
- Last Call - fase bastante avançada, mas existe *feedback* a ser processado;
- Awaiting implementation feedback - fase avançada, mas pode ser alterada em função da resposta dos programadores;
- Implemented and widely deployed - fase de conclusão.

2.5.1 Web-Storage

Antigamente, o armazenamento de informação era feito através dos *Cookies*. Este tipo de armazenamento era muito utilizado nos *logins*, sendo possível fazer a autenticação de forma automática. Contudo, o seu uso apresentava algumas falhas graves:

- Capacidade limitada de 4Kb. Poderia ser suficiente no início da Internet, mas hoje em dia, com ligações à internet cada vez mais rápidas e utilização de aplicações Web cada vez mais pesadas, o uso de *Cookies*, tornou-se insuficiente.
- Falhas graves de segurança, nomeadamente pela ausência do protocolo SSL (Camada de Transporte Segura) de alguns sites, ficando possível descriptar a informação guardada em *Cookies*.
- Redundância de informação nos pedidos HTTP ao servidor.

Assim, o Web-Storage[17] veio tentar colmatar estas falhas, através de um armazenamento com maior capacidade advindo de um sistema *Key-value pair*.

Houve, então, um aumento na capacidade de armazenamento, nomeadamente de 10Mb para o Chrome, Firefox, Safari e Internet Explorer.

Passando, agora, a existir dois tipos de armazenamento: local (*localStorage*) e de sessão (*sessionStorage*). No *localStorage* a informação continuará guardada mesmo que se feche o *browser*. Por sua vez, no *sessionStorage* só estará disponível na navegação enquanto o *browser* estiver ativo, não passando essa informação a qualquer outro *browser*.

O Web Storage encontra-se na fase Working Draft [16].

2.5.2 Offline Web

As páginas web são carregadas e renderizadas quando se está *online*, sendo que o seu carregamento implica a ligação à Internet. Então, como será possível carregar páginas estando *offline*? Não é possível! A não ser que sejam carregadas e guardadas quando se está *online*, sendo, basicamente, assim, que funciona. Isto torna-se possível através da AppCache.

AppCache

A utilização da AppCache[5] dá a possibilidade de:

- Offline Browsing: os utilizadores conseguem navegar através do *browser* sem ligação à Internet.
- Maior velocidade: o uso da informação em cache torna o seu acesso bastante mais rápido.
- Redução da utilização do Servidor: o *browser* apenas faz o *download* dos recursos que foram modificados no servidor.

As páginas *offline* funcionam através de um ficheiro *manifest*, que se trata de um ficheiro de texto, guardado no servidor, onde são indexados o nome dos ficheiros que o *browser* necessita guardar. Este, quando tiver uma ligação *online* à Internet irá ler esse ficheiro *manifest* e fará o *download* dos recursos necessários, armazenando ou atualizando a informação no *browser*, de maneira a que, quando se aceda à aplicação, mesmo estando *offline*, se consiga obter a informação atualizada. O caso inverso também é possível, sendo as alterações efetuadas *offline* e guardadas, assim, quando este voltar a ter uma ligação *online*, terá capacidade para sincronizar essa informação com o servidor.

2.5.3 Geolocalização

A geolocalização possibilita, opcionalmente, partilhar a nossa localização com outras pessoas. Existem várias formas de obter esta informação, nomeadamente através do IP ou da rede à qual estamos ligados à Internet ou, ainda, do sensor GPS, caso o dispositivo o tenha. Como foi referido, esta informação só pode ser transmitida ao servidor caso o utilizador o autorize.

2.5.4 Vídeo

A utilização de *plug-ins* como Quicktime, RealPlayer ou Flash surgiu da necessidade de ver vídeos *online* em páginas web, visto o HTML 4.01 não suportar este tipo de media. Se nos PCs tudo fica resolvido com o aparecimento destes *plug-ins*, nos dispositivos móveis é bem diferente, devido à falta de *plug-ins* compatíveis. Com o HTML5 e apenas utilizando a tag `video` é possível visualizar vídeos, sem serem necessários *plug-ins*, embora continuem a existir alguns problemas, nomeadamente devido ao formato usado por cada *browser*.

2.5.5 Canvas

É o elemento usado na criação de gráficos através da tag `canvas`. É usado para renderizar gráficos, jogos ou outros elementos visuais em tempo real. Este elemento não é mais que um retângulo capaz de ser manipulado através do JavaScript, de modo a conseguir obter a forma desejada. O retângulo é um sistema bidimensional de coordenadas x e y de tamanho $x1$ e $y1$, sendo o ponto $(0,0)$ o canto superior esquerdo e o ponto $(x1,y1)$ o canto inferior direito.

2.5.6 JavaScript

O JavaScript é uma linguagem de programação para a Internet criada pela Netscape em setembro de 1995. Tornou-se uma das linguagens mais usadas na construção de web sites e aplicações web. Hoje em dia esta linguagem permite criar *interfaces* interativas com

usabilidade e comportamento semelhantes aos de uma aplicação nativa, focando-se na dinâmica e nos aspetos funcionais. A inclusão do AJAX veio facilitar a troca de dados com o servidor, através de chamadas assíncronas ao servidor, tornando as páginas mais interativas para o utilizador. A existência de várias livrarias e *frameworks*, como o jQuery, Prototype, MooTools, BootStrap facilitam o processo de desenvolvimento. O JavaScript tornou-se, assim, uma tecnologia indispensável a qualquer programador Web. Contudo, existem problemas ao nível do suporte nos *browsers*, visto que nem todas as funcionalidades estão disponíveis.

2.5.7 CSS

O CSS é uma linguagem de estilo que define a apresentação e formatação de documentos escritos em HTML. Esta tecnologia foi criada para possibilitar a separação entre a formatação e o conteúdo de um documento, a partilha da mesma formatação em múltiplas páginas e a aplicação de diferentes formas de apresentação de conteúdo, dependendo do dispositivo em que está a ser visualizada a página HTML. As particularidades desta tecnologia trazem maior flexibilidade, acessibilidade e controlo à especificação das características de apresentação de páginas de Internet. Um dos principais problemas, à semelhança do JavaScript, é ao nível do suporte entre *browsers*. A versão mais recente, o CSS3, está dividida em módulos que incluem a especificação anterior CSS2, o que permite a compatibilidade com implementações mais antigas e adiciona novas funcionalidades.

Os módulos mais importantes que constituem o CSS3 são:

- Media Queries - São expressões que verificam determinadas condições da página, aplicando diferentes regras de CSS para diferentes cenários [11]. Assim, torna-se possível obter diferentes estilos para diferentes tamanhos de ecrãs, por exemplo, uma página pode utilizar uma determinada cor de fundo ou um determinado tamanho de letra, quando visualizada num ecrã com uma determinada dimensão e pode ter outra cor de fundo e outro tamanho de letra num ecrã com outra dimensão, isto sem que seja necessário alterar o seu conteúdo.
- Selectors - especificam padrões que permitem a seleção de elementos, por atributos, que se pretendem estilizar.
- Background - implementa várias propriedades para o fundo do ecrã, como o controlo do tamanho da imagem de fundo, a área de posicionamento das imagens de fundo e a utilização de várias imagens ao mesmo tempo.
- Borders - possibilita a criação de contornos arredondados, adição de sombras a caixas e utilização de imagens como contorno.
- Text Effects - as novas características de texto são: a capacidade de aplicação de sombras no texto, quebras de linha quando uma palavra não cabe numa área e

possibilidade de utilizar a fonte de texto pretendida bastando, para tal, dizer qual o ficheiro de fontes a utilizar.

- 2D e 3D Transformations - possibilitam a aplicação das seguintes transformações: translação, rotação, escala e perspetiva a elementos, em duas ou três dimensões.
- Animations - permite a transição gradual entre estilos e a animação de elementos através da utilização de *keyframes*, regras que definem a alteração de estilo.
- Multiple Column Layout - consiste na apresentação de múltiplas colunas de texto, podendo definir o número de colunas, o espaçamento entre elas, a forma de preenchimento e a largura.
- User Interface - possibilita que o utilizador altere, à sua vontade, o tamanho dos elementos.

2.6 Adaptive and Responsive Web Design

A interação do utilizador à web, através de vários tipos de dispositivos com variados tamanhos de ecrã, mudou a forma como acedemos e partilhamos a informação no mundo digital, bem como, na criação dessas páginas web. É, assim, cada vez mais importante oferecer experiências de utilização consistentes para os mais diversos tipos de utilizadores.

O Responsive e o Adaptative Web Design vêm, assim, ajudar através de um conjunto de normas e ferramentas capazes de criar páginas web que respondem ao tamanho de qualquer ecrã.

Uma vez que seria impensável desenhar uma versão do mesmo site para o tamanho de ecrã de um determinado dispositivo, o Responsive Web Design (RWD) surgiu como uma das soluções técnicas para esse problema. Portanto, este consegue dar resposta à necessidade de adaptação do website a qualquer dispositivo. O conceito e as técnicas de Responsive Web Design surgiram pela primeira vez em 2010, no artigo *Responsive Web Design*, elaborado pelo designer gráfico e programador Ethan Marcotte, e publicado no seu blog *A List Apart*.

O Responsive Web Design é um conceito que se relaciona com a capacidade de desenvolver websites que se adaptem ao tamanho e à resolução do ecrã do dispositivo que se está a utilizar. Para tal, são necessárias três noções fundamentais na sua implementação: layouts flexíveis (através de grids, com dimensões relativas); conteúdos flexíveis de imagem e vídeo (através de redimensionamento dinâmico); e media queries e media query listeners (o uso de media queries permite suportar diferentes resoluções nos dispositivos sem ser feita qualquer alteração no conteúdo). Como refere Marcotte no seu artigo *Responsive Web Design* [7] estes são os três ingredientes para o Responsive Web Design: *Fluid grids, flexible images, and media queries are the three technical ingredients for responsive web design, but it also requires a different way of thinking*. Com o Responsive Web Design a utilização de *layouts*

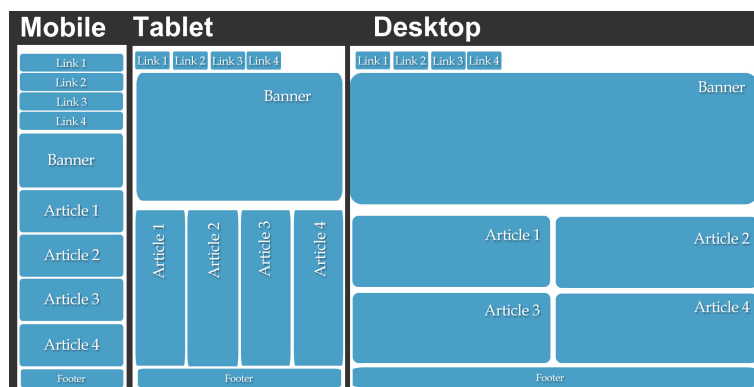


Figura 2.5: Grid Flexível consoante o tipo de dispositivo Fonte:<http://gridbyexample.com/> consultado a 5 de junho de 2015.

passa a ser dimensionada através de percentagens e não de pixéis, como acontecia até aí. Também o tamanho do texto passa a ser escalável. As imagens e outros tipos de media são redimensionados automaticamente dependendo do dispositivo em que são carregados.

Assim, através da junção de grids flexíveis, imagens de tamanho escalável e media queries conseguem-se criar aplicações *responsive* flexíveis e dinâmicas.

2.6.1 Grid flexível

De uma forma simplificada pode dizer-se que o grid flexível é um conjunto de linhas de base que fornecem a estrutura do *layout* (estruturam o conteúdo, subdividindo a página vertical e horizontalmente em margens, colunas e blocos de conteúdo). Segundo Boulton [1]: *In the context of graphic design, a grid is an instrument for ordering graphical elements of text and images.*

O sistema de grids surgiu inicialmente nas páginas de livros e jornais. Porém, com a sua exploração, descobriu-se que é possível transformar a base estrutural, que até então utilizava medidas em pixéis, numa estrutura fluída com medidas percentuais. Deste modo, o sistema de grides forma, atualmente, o *layouts* das páginas web.

Nem todos os autores estão de acordo relativamente à utilização de grides. Para alguns, o seu uso torna mais rápido o design das páginas, garantindo a coerência visual entre páginas relacionadas, especialmente quando se tem um website com várias páginas. Todavia, para os mais cétricos, a utilização de grids limita a criatividade dos designers. Boulton refere isso mesmo no seu artigo *Five simple steps to designing grid systems – Preface* [1]: *It is often said of grid systems that they limit the scope for creativity or leave no freedom. Karl Gerstner, one of Switzerland's preeminent graphic designers, was aware of this conflict with the designers adoption of grid systems.*

2.6.2 Conteúdos flexíveis de imagem e vídeo

Tendo em conta que flexível significa adaptar-se, a ideia por detrás das imagens flexíveis é a de que as mesmas se adaptem ao dispositivo onde estão a ser utilizadas. A ideia é manter as imagens no tamanho máximo em que poderão ser usadas. Não são definidos quaisquer valores para a largura e altura das mesmas. Cabe ao *browser* redimensionar o tamanho das imagens quando necessário, usando CSS para orientar o tamanho relativo de cada uma. Assim, as imagens apresentam as suas dimensões originais sempre que a sua largura não exceda a do contexto em que se inserem, pois, caso contrário, serão redimensionadas.

A adaptação das imagens ao tamanho do dispositivo consegue-se através de um `max-width` de 100%, a mesma solução pode ser utilizada nos vídeos. Deixar a responsabilidade de escalar imagens para o *browser* pode ser, então, a forma mais simples de transformar imagens de tamanho fixo em imagens de tamanho flexível. Todavia, nem sempre isto resulta, uma vez que, por exemplo, as versões mais antigas do Internet Explorer não suportam a `max-width`, embora este seja um problema menor, uma vez que existem alguns *workarounds* capazes de solucionar este problema.

É na questão das imagens flexíveis que existem mais críticas no que se refere ao Responsive Web Design, sendo que o principal problema reside na utilização de imagens com alta resolução, que são carregadas pelos *browsers* mesmo quando os utilizadores têm pouca largura de banda disponível. Alguns autores, portanto, defendem que carregar imagens com alta ou baixa resolução deve ser uma opção do utilizador.

Presentemente existem várias ferramentas que funcionam como alternativas na obtenção de imagens fluídas, contornando os inconvenientes associados à utilização de imagens com grande resolução. Entre essas ferramentas temos: Adaptive Images, Sencha.io Src, Picturefill e FitVids, sendo que todas pretendem que seja dado, à imagem, o tamanho mais apropriado, tendo em conta o tamanho do ecrã que está a ser utilizado.

- Adaptive Images – criada por Matt Willcox – *deliver small images to small devices* [4]. É uma componente PHP que deteta o tamanho do ecrã e que posteriormente cria, armazena em cache e devolve automaticamente uma versão da imagem com o tamanho mais apropriado. Esta é uma ótima opção para sites onde não há possibilidade de reestruturar o código, dado que não é preciso utilizar *markup* extra, mas apenas incluir os ficheiros necessários.
- Sencha.io Src - criada por James Pearce, trata-se de um serviço que desenvolve versões otimizadas das imagens consoante o tamanho do dispositivo. Para o utilizar é necessário atribuir, no atributo fonte (Src) da imagem, o endereço do Sencha.io Src, seguido do endereço da imagem, como por exemplo:
'http://src.sencha.io/http://site.pt/imagens/teste.jpg'

Esta ferramenta utiliza 'user agentes' para descobrir qual é o tamanho do dispositivo, redimensionando a imagem adequadamente, porém, por omissão, a imagem é



Figura 2.6: Adaptive Image em diferentes dispositivos Fonte:<http://adaptive-images.com/> consultado a 5 Junho de 2015.

redimensionada para que ocupe 100% do comprimento do ecrã do dispositivo. Com esta ferramenta também é possível redimensionar as imagens para um tamanho específico, sendo, para tal, necessário, adicionar um outro parâmetro com o tamanho pretendido (400px), por exemplo:

```
'http://src.sencha.io/400/http://site.pt/imagens/teste.jpg'
```

Além disto, o Sencha.io Src consegue, ainda, fazer cache dos pedidos, evitando que a imagem seja gerada cada vez que a página é carregada.

- Picturefill - desenvolvida por Scott Jehl, simula o comportamento do elemento 'picture', recorrendo a elementos compatíveis com os browsers atuais. Esta é, por exemplo, a ferramenta utilizada pelo site da Microsoft para apresentar imagens com diferentes resoluções, de acordo com o ecrã do dispositivo.
- FitVids (para vídeos) - desenvolvido por Chris Coyler, Trent Walton, Dave Rupert e Reagan Ray. Trata-se de um *plug-in* jQuery, leve e fácil de utilizar, em que os vídeos respondem aos diferentes tamanhos do ecrã, mantendo sempre uma relação de proporção original.

No caso dos vídeos torna-se, também, necessário redimensionar o tamanho do texto consoante o dispositivo utilizado, para que o mesmo possa ser lido pelos utilizadores. Assim sendo, o CSS3 introduziu novas medidas tipográficas baseadas no tamanho do *viewport*.

Desta forma, as unidades de viewport-percentage referem-se ao tamanho inicial do contexto onde o texto se encontra. O tamanho da fonte é redimensionado assim que o comprimento e/ou largura do elemento que a contém é alterado.

2.6.3 Media Queries

As Media Queries foram desenvolvidas pelo W3C e fazem parte das especificações do CSS3. Permitem identificar os tipos de media com os quais se está a lidar e também inspecionar as características físicas dos dispositivos e *browsers*. Com a utilização das media queries torna-se possível direcionar o design para um conjunto específico de dispositivos, sem que o conteúdo seja alterado, embora tal possa acontecer.

Segundo o W3C, uma media query consiste numa media type e zero ou mais expressões que verificam as condições de determinados tipos de media features [10]. A media query é uma expressão lógica que pode ser verdadeira ou falsa, sendo que é verdadeira se a media type da media query combinar com a media type do dispositivo onde o documento está a ser exibido e, assim, todas as media query são verdadeiras.

Todavia, para se perceber melhor o que são media queries é importante definir também media types e media features.

- Media types – refere-se aos diferentes tipos de media, isto é, os media types permitem criar regras dirigidas para um determinado tipo de media ou dispositivo. Deste modo, os media types permitem especificar como o documento é apresentado nos diferentes tipos de media ou dispositivos (incluindo dispositivos táteis em braille).
- Media features – permitem inspecionar as propriedades específicas do dispositivo, orientando, assim, as regras aplicadas conforme essas características. Algumas media features são: *widht* (largura do elemento), *height* (altura do elemento), *color* (cor), *device-width* (largura do dispositivo), *device-height* (altura do dispositivo), *orientation* (orientação do dispositivo). Combinando media types e media features constroem-se, então, os media queries, que são um dos elementos fundamentais do Responsive Web Design.

Declaração de media queries

Uma media query contém dois componentes:

1. Um media type;
2. Uma query propriamente dita, declarada entre parêntesis, contendo uma específica media feature, seguido do valor alvo.

Considere-se, então, o seguinte exemplo de uma media query.

```
<link rel="stylesheet" type="text/css" media="screen and (max-device-width: 480px)" href="sheetland.css" />
```

Figura 2.7: Exemplo de uma Media Query.

Cada media query é iniciada pelo media type selecionado, que neste exemplo é *screen*. Segue-se a query propriamente dita, declarada entre parênteses (`max-device-width: 480px`). A query combina dois componentes: o nome da característica - `max-device-width` e o seu valor `480px`. A media query que compõe o exemplo interroga o *browser* sobre 1) se este se trata de um ecrã (media type *screen*) e, em caso afirmativo, 2) se a resolução horizontal é igual ou inferior a `480px`. Se o *browser* passar nestes dois critérios, ou seja, se virmos o nosso trabalho no pequeno ecrã de um dispositivo móvel, como por exemplo o iPhone, então os estilos definidos dentro da media query são aplicados e o dispositivo carrega o `shetland.css`. Caso contrário, o link é completamente ignorado.

À medida que o *layout* vai sendo escalado e os seus elementos redimensionados, o uso das media queries pode corrigir as imperfeições visuais que vão aparecendo, sendo que também podem auxiliar na otimização do conteúdo do site para ir de encontro às necessidades de cada dispositivo, criando, assim, *layouts* alternativos para diferentes gamas de resolução. Para detetar o *layout* que deve ser apresentado a estratégia é determinar o valor *viewport*, para controlar o tamanho da área de visualização do *browser* disponível, definindo que o seu tamanho corresponde ao tamanho do ecrã (`width=device-width`).

Para tal, deve definir-se no cabeçalho HTML o seguinte:

```
<meta name="viewport" content="width=device-width">
```

Figura 2.8: Exemplo código html.

Podem, assim, aplicar-se regras de acordo com o tamanho do *viewport*, consoante o valor de `width`, `min-width` e `max-width`:

```
@media screen and (width: 400px)    {...}  
@media screen and (max-width: 600px) {...}  
@media screen and (min-width: 1200px){...}
```

Figura 2.9: Regras de acordo com o tamanho do viewport.

Importa referir que, uma vez que as media queries consistem numa *feature* de CSS3, as versões de *browsers* que não são compatíveis com CSS3 não suportam media queries. Estes *browsers* ignoram as regras definidas em media queries, podendo causar falhas de formatação dos elementos e, por conseguinte, incoerências nas *interfaces*. Todavia, já existem ferramentas que conseguem superar esta limitação dos *browsers*, interpretando media queries, como é o caso da `Respond.js` e da `css3.mediaqueries.js`.

2.7 Testes de Usabilidade

A usabilidade tem uma relação direta com a *interface*. Esta juntamente com o sistema computacional interativo e o utilizador constituem os três principais componentes da interação Homem-Computador. Os primeiros testes de usabilidade surgiram, claro está, no âmbito da investigação sobre esta interação.

Os testes de usabilidade, de um modo geral, referem-se a um conjunto de métodos que se alicerçam em examinar os aspetos de usabilidade de um documento/sistema/*interface*, por parte de avaliadores/futuros utilizadores. Visam encontrar problemas de usabilidade no projeto, fazendo as recomendações necessárias para que os mesmos sejam eliminados.

A usabilidade é uma qualidade que deve ser inerente ao documento, possibilitando, deste modo, ao utilizador um bom uso do mesmo, uma vez que o *software* pode estar bem concebido relativamente à funcionalidade, mas o mesmo será rejeitado pelo utilizador se a sua usabilidade não for boa.

O conceito de usabilidade suscita diferentes opiniões, nomeadamente no que respeita aos parâmetros para medir a usabilidade de um documento/sistema. Shackel [13], por exemplo, define quatro parâmetros para fazer tal medição: eficácia, aprendizagem, flexibilidade e atitude do utilizador. Por sua vez, Nielson [8] [9] considera cinco parâmetros para o efeito: fácil de aprender, eficiente para usar, fácil de lembrar, pouco sujeita a erros, e agradável de usar. Com efeito, Smith e Mayers [14] propõem apenas três parâmetros para medir a usabilidade, mas que, de certa forma, englobam as propostas anteriormente mencionadas. Assim, para estes autores a usabilidade deve ser medida através de: facilidade de aprendizagem, facilidade de utilização e satisfação no uso do sistema pelo utilizador, quer, então, isto dizer que um documento multimédia só será facilmente aceite pelo utilizador se for, para este, fácil de aprender a usar, de saber usá-lo e se o mesmo se sentir satisfeito ao usá-lo.

- Facilidade de aprendizagem – será, talvez, o atributo mais importante da usabilidade, pois é ele que pode levar a que futuros utilizadores optem por usar este documento/sistema em detrimento de outro(s). É um atributo que deve ser medido mesmo em relação a utilizadores pouco experientes, devendo, portanto, apresentar um acesso à “ajuda” (instruções claras e concisas), que oriente o utilizador passo a passo, mas que não se trate de uma descrição extensa sobre todo o processo, pois isso torna-se aborrecido para o utilizador e perde a eficiência de uma ajuda passo a passo. O utilizador deve, então, compreender facilmente a *interface*, os diversos percursos e o que pode fazer no documento/sistema.
- Facilidade de utilização – refere-se à facilidade com que o utilizador deve conseguir trabalhar com o documento/sistema depois de ter aprendido a interagir com ele.
- Satisfação no uso do sistema pelo utilizador – o utilizador deve sentir-se satisfeito ao navegar pelo documento/sistema, devido à sua *interface*, ao conteúdo, à estrutura do

documento, às ajudas disponíveis, ao processo de interação e navegação, etc. Foram criados vários testes para medir a satisfação do utilizador, tais como o SUMI (Software Usability Measurement Inventory) e o QUIS (Questionnaire for User Interface Satisfaction).

Os testes de usabilidade avaliam o desempenho e as preferências do utilizador, sendo que a usabilidade é medida em relação a determinado tipo de utilizadores e tarefas. É necessário, portanto, ter em atenção as expectativas do sujeito relativamente à utilidade do produto, à facilidade em aprender a usá-lo, a usá-lo e a instalá-lo. Pode, assim, dizer-se que avalia a comunicabilidade, ou seja, a comunicação entre projetista e utilizador, se a *interface* expressa o modelo de interação previsto pelos projetistas e, também, como o conhecimento do utilizador evolui através da interação.

Torna-se, portanto, fundamental planear bem o teste, começando por explicitar a necessidade da sua existência e referindo-se o que se vai medir e o porquê de se o fazer. Colocam-se as questões a responder e, seguidamente, define-se o perfil de utilizador. Posteriormente, seleciona-se o método, indicam-se as tarefas, definem-se instrumentos e, logo depois, selecionam-se as técnicas de recolha de dados. É também de grande importância especificar o equipamento necessário (tipo de computador e quantidade de *software*), indicar os critérios necessários para se considerar o teste terminado, especificar os dados a serem recolhidos e o modo como vão ser tratados. Por fim, deve ser elaborado um relatório com as propostas de alteração a serem realizadas futuramente.

Estes testes devem ser realizados durante todo o processo de desenvolvimento, incluindo a fase inicial de criação da *interface*, sendo que existem diferentes testes para cada fase do processo de desenvolvimento do projeto, que serão abordados em seguida.

- Teste exploratório – é um teste inicial que deve ser feito no começo do processo, quando se define e concebe o serviço ou recurso, uma vez que é necessário saber como os utilizadores respondem a determinados serviços, sendo que, entre outras coisas pode: analisar se a interface representa bem classes de objetos, se as relações entre esses objetos são facilmente compreendidas, se as especificação de pré-requisitos para usar o documento ou o produto são necessárias e, ainda, se a tabela de conteúdos está organizada de tal forma que agrade a utilizadores iniciados e a utilizadores experientes [2].
- Testes sobre ícones – devem ser feitos na fase inicial de desenvolvimento do protótipo, podendo, porém, ser feitos em papel. Devem ser apresentados aos utilizadores, para que mencionem o que representa cada um dos ícones, sendo possível, assim, verificar se estes transmitem ou não a ideia pretendida. Posteriormente, numa fase mais avançada, mas em que o documento esteja ainda em construção, pode ser apresentada uma imagem de uma página protótipo, assim como uma breve descrição da função de cada ícone, solicitando-se, claro está, ao sujeito, para que identifique o ícone com a descrição. Os resultados destes testes serão medidos através da proporção de ícones descritos/identificados corretamente.

- Teste de avaliação – deve, também, ser feito numa fase inicial, podendo servir para expandir os resultados do teste exploratório. Nesta fase deve ser constituída uma equipa multidisciplinar para executar um determinado serviço e para discutir questões de usabilidade. Além disso, aqui já se pede ao utilizador que realize tarefas, em vez de apenas percorrer o produto e comentar sobre os ecrãs. Podem, através deste teste, ser recolhidos dados quantitativos.
- Teste de validação/verificação – contrariamente aos testes anteriores, que ocorrem durante a fase de desenvolvimento do processo, este só pode ser realizado na fase final, pois tem como objetivo verificar a usabilidade do serviço e a eficácia dos recursos de aprendizagem. Neste sentido: revê-se a consistência do sistema e a interface, compara-se o sistema com os “*standards*” de usabilidade com orientações gerais e outros serviços relacionados.

Rubin [12], refere ainda outro teste – de comparação – a desenvolver em qualquer fase do processo, que consiste em comparar dois ou mais designs alternativos. Segundo Amorim [2], este teste permite analisar qual dos estilos de interface se aproxima mais do modelo concetual da população alvo. Identificar em qual deles se sentirá o utilizador mais orientado e em que tarefas é necessário disponibilizar a “ajuda”. Este teste pode ser usado em conjunção com qualquer um dos outros testes.

Quanto aos avaliadores estes devem ser de dois tipos: especialistas em interação Homem-Computador e multimédia e uma amostra do público-alvo a quem se destina o projeto.

Embora haja algum consenso relativamente a este aspeto, alguns autores que apresentam propostas diferentes.

Tessmer [15], por exemplo, considera que devem existir três tipos de avaliadores:

- Um especialista – que revê o sistema, podendo detetar facilmente problemas de inconsistência, tarefas pobres, *interfaces* confusas, etc.
- Um observador e um utilizador – enquanto o observador observa, o utilizador vai apontando as dificuldades por ele anunciadas. Este tipo de recolha de dados tem um problema, que consiste na alteração de comportamentos por parte dos utilizadores ao saberem que estão a ser observados (efeito de Hawthorne). Pode resolver-se este problema recorrendo a câmaras de vídeo para registar movimentos de mãos, expressões faciais, etc.
- Pequeno grupo de utilizadores – neste caso compara-se o desempenho dos membros do grupo.

Há que ter muita atenção na seleção da amostra de utilizadores, uma vez que as suas características têm de coincidir com as dos futuros utilizadores.

No que concerne aos métodos de avaliação e técnicas de recolha de dados há consenso em aceitar quatro métodos para o efeito, que serão explicitados em seguida, será, porém, dada

maior relevância à avaliação heurística, por ser esta a utilizada no processo de desenvolvimento do projeto iScriptor.

- Observação – é feita em laboratório, direta ou indiretamente (através de câmaras) e pretende-se verificar a interação do sujeito com a *interface*. Utilizando-se este método é fundamental que o observador utilize um guião para saber o que vai observar e para saber o que se pretende que seja observado em particular.
- Sondagens – podem ser utilizadas duas técnicas de recolha de dados, isto é questionários e entrevistas.
- A avaliação heurística – criada por Jakob Nielsen - é uma das formas de avaliar a usabilidade mais económica (não necessita de laboratórios ou equipamentos) e prática (leva apenas um ou dois dias para a aplicar) e pode ser aplicada em qualquer fase do ciclo de desenvolvimento do *software*, permitindo apoiar o desenvolvimento de projetos; porém é aconselhável que seja realizada nas fases iniciais, onde a *interface*, às vezes, se restringe a um esboço descrito em papel.

Este tipo de avaliação permite detetar problemas na fase de desenvolvimento da *interface*, evitando, deste modo, erros, que ao serem detetados após a implementação gerariam custos desnecessários à empresa.

É realizada por peritos, que podem ser engenheiros da usabilidade, programadores, designers ou utilizadores. Em alguns casos pode, ainda, ser feita com colegas de trabalho para reduzir custos, mas nunca deve ser feita individualmente, pois uma só pessoa não tem a capacidade de levantar todas as questões heurísticas.

Assim, envolve sempre um grupo de avaliadores que examina a *interface* e julga as suas características, face a reconhecidos princípios de usabilidade. Como referido anteriormente, as questões de heurística podem ser levantadas por qualquer pessoa e permitem simular o comportamento do utilizador perante a *interface*. Como não é possível detetar todos os problemas de usabilidade através da heurística é frequente a realização de outros testes, como os testes com utilizadores, que permitem validar os testes de avaliação heurística e corrigir outros erros que não tinham, até então, sido detetados.

Jacob Nielsen, na sua obra *Usability Engineering* [8] apresenta um gráfico que permite ver exatamente o número de problemas de usabilidade encontrados numa avaliação heurística em função do número de observadores, onde é facilmente verificável que existe um número ideal, 5 a 6 avaliadores, capazes de detectar 75% dos problemas de Usabilidade.

Este gráfico permite que a empresa estabeleça uma relação entre custos e eficácia, permitindo encontrar um número de observadores adequado à dimensão do projeto e ao orçamento previsto para os testes de usabilidade.

Nielsen, em 1995, [3] definiu 10 heurísticas de usabilidade, sendo elas:

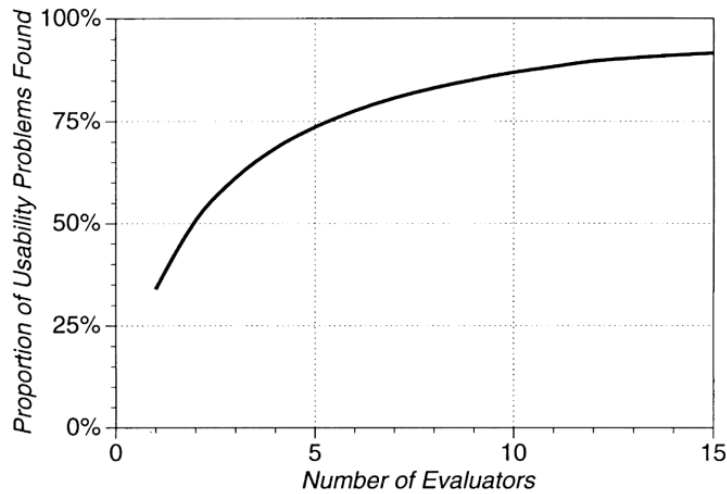


Figura 2.10: Problemas de Usabilidade detetados consoante o número de avaliadores.

1. Visibility of system status - The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2. Match between system and the real world - The system should speak the users language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3. User control and freedom - Users often choose system functions by mistake and will need a clearly marked 'emergency exit' to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4. Consistency and standards - Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5. Error prevention - Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
6. Recognition rather than recall - Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7. Flexibility and efficiency of use - Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8. Aesthetic and minimalist design - Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9. Help users recognize, diagnose, and recover from errors - Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. Help and documentation - Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

2.8 Futuro

Aquando da realização do projeto, em 2013, estava previsto, para meados desse mesmo ano, o lançamento de novos SOs móveis. Neste documento será dada maior importância a três deles, os quais se pensou virem a ter algum impacto no mercado.

Pouco se falou do Windows Phone, talvez por, até agora, não terem sido dadas provas de suas reais possibilidades, para justificar a afirmação e uma fraca adesão, como se pode verificar na Figura 2.11. Não obstante, é necessário ter algum cuidado antes de descartar, deste setor, o Windows Phone devido a ser um produto desenvolvido pela Microsoft, gigante mundial e líder do SO mais usado em todo o mundo em computadores pessoais.

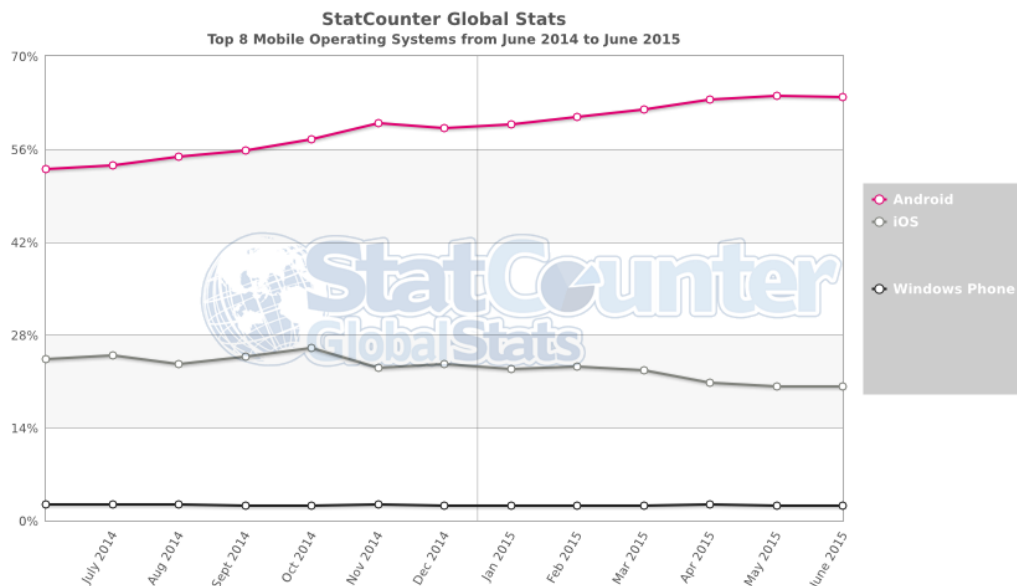


Figura 2.11: Gráfico de utilização de três SO mobile Fonte:<http://statcounter.com> consultado a 5 de junho de 2015.

Novas estratégias estão a ser postas em prática, entre as quais o uso do HTML5 no desenvolvimento de aplicações para o Windows 8, nomeadamente para a *interface* Metro, desenvolvida, principalmente, para dispositivos móveis (Tablets). Pode muito bem ser este o primeiro passo para um crescimento sustentável, com a integração do HTML5 no

desenvolvimento de aplicações, fazendo aumentar o número de aplicações para dispositivos móveis, que é um dos pontos fracos apontados ao SO da Microsoft.

2.8.1 Tizen

Um grupo de companhias, incluindo a Samsung (um dos líderes mundiais de venda de dispositivos móveis) e a Intel, criou um projeto denominado Tizen, desenhado para ser usado em Smartphones, Tablets e Netbooks. Nascido com base do sistema operativo MeeGo Linux, o Tizen é desenvolvido através do HTML5 e de outras Tecnologias Web, o que leva ao desenvolvimento de aplicações, utilizando as mesmas ferramentas que usam no desenvolvimento de Web Apps. A visão da Samsung é criar um SO com o qual consiga ter maior controlo nos seus dispositivos e conseguir, com o uso dessas tecnologias Web, angariar programadores para desenvolverem Apps para esse mesmo SO.

Em Janeiro de 2013, a Samsung confirmou à Bloomberg o uso do Tizen em inúmeros dispositivos da marca.

2.8.2 Firefox OS

É um SO móvel criado pela empresa Mozilla, desenvolvido em tecnologias Web, como o HTML5. Tem por base o SO Android, mas não será possível utilizar as Apps do Android. No lugar disso, poderão instalar-se Web Apps através da App Store da Firefox, denominada por Firefox Marketplace. O objetivo passa por criar um SO *open source*, bem mais aberto que o Android, em que uma comunidade contribua para o seu desenvolvimento. A Mozilla tenta, assim, criar um produto com a mesma receita com a qual criou o Firefox, que hoje em dia compete, lado a lado, com os gigantes da indústria de *software*. A Mozilla garante, após os primeiros testes, um desempenho bastante rápido e já com inúmeros aplicativos desenvolvidos. Dotado de argumentos inovadores capazes de fazer face à concorrência, o que já vem sendo um hábito nos seus produtos. Este SO entra no mercado brasileiro no início de 2013, numa primeira fase, seguindo-se, depois, o resto do Mundo.

2.8.3 Ubuntu Phone OS

É um SO móvel desenvolvido pela Canonical, com base no Ubuntu, mas sem o recurso ao Java Virtual Machine, aumentando o seu desempenho através do aproveitamento de todas as potencialidades do *hardware*.

O seu objetivo é criar uma experiência única em dispositivos de baixo processamento, tentando ganhar muitos utilizadores neste segmento e, ao mesmo tempo, tornar a experiência dos super Smartphones diferente, ou seja, tornando um Smartphone, que cabe no bolso, num Desktop, como é possível ver pela Figura 2.12.



Figura 2.12: Ubuntu Phone OS Fonte:<http://siliconangle.com>, consultado a 14 de junho de 2015.

Capítulo 3

Especificação do Projeto

3.1 Introdução

Neste capítulo será apresentado o produto Scriptor Server, que serviu de base à criação da aplicação móvel referente a esse mesmo produto, o iScriptor. Ao longo deste capítulo, além da apresentação do Scriptor Server, será apresentado o planeamento da App iScriptor. Serão, ainda, apresentados três casos de estudo - Metronic, Keyhole Gestor de Identidades e FT Financial Times Journal - e retiradas, claro está, algumas conclusões relevantes para a tomada de decisões no desenvolvimento do projeto. Será descrito todo o processo realizado ao nível da análise de requisitos, a sua arquitetura, o uso de Web Services e a criação de um protótipo (Mockups), nomeadamente para a realização de Testes de Usabilidade, que permitiram desenvolver adequadamente a App.

3.2 Scriptor Server

O Scriptor Server é uma plataforma B2B (Business to Business) criada pela empresa Viatecla, é utilizada na gestão de conteúdos e processos das empresas, *backoffice* de sites, bem como em aplicações intranet, extranet, e *ecommerce*. O Scriptor trata-se de uma plataforma multi tarefa e transversal, na medida em que pode ser utilizado para gestão documental e de conteúdos da empresa e na construção e gestão de processos, tais como projetos, gestão de recursos humanos e *backoffice* de sites. Uma das principais utilizações desta plataforma é a de *backoffice*, contando com um largo portfólio de sites.

A flexibilidade do Scriptor Server permite às organizações a implementação, desenvolvimento e operação de processos de negócio complexos, entregando um conjunto de benefícios

com impacto direto na performance do próprio negócio.

Com o Scriptor Server a gestão de informação assenta em processos ágeis e seguros, com mecanismos simples que permitem normalizar uma vastíssima gama de tipos de informação, bem como os respetivos ciclos de vida. A criação e gestão de processos e conteúdos é uma tarefa fácil e segura. A sua utilização permite a publicação de conteúdo e informação em múltiplos canais, garantindo a reutilização de informação, mas aplicando *layout* distinto para diferentes ambientes.

A plataforma Scriptor Server permite gerir tanto sites de Internet como ambientes intranet, que sirvam de ponto de orquestração colaborativa dos processos e *workflow* internos de uma organização. Também permite disponibilizar as ferramentas necessárias para suporte à disponibilização de informação em *front-ends* do mais elevado nível de exigência. Através de uma API simples e versátil é possível o desenvolvimento de novas funcionalidades, assim como a integração com outros sistemas, com facilidade e rapidez.

Ao olharmos para as Figuras 3.1, 3.2, 3.3 e 3.4 é possível compreender melhor o funcionamento da plataforma Scriptor Server. O utilizador através da *Interface* Scriptor Server pode criar e editar conteúdos no seu site. Estas Figuras demonstram o processo de um Cliente do Scriptor Server, neste caso uma Agência de Viagens, a introduzir uma nova oferta no seu site.



Figura 3.1: Interface Scriptor Server com a descrição de alguns componentes.

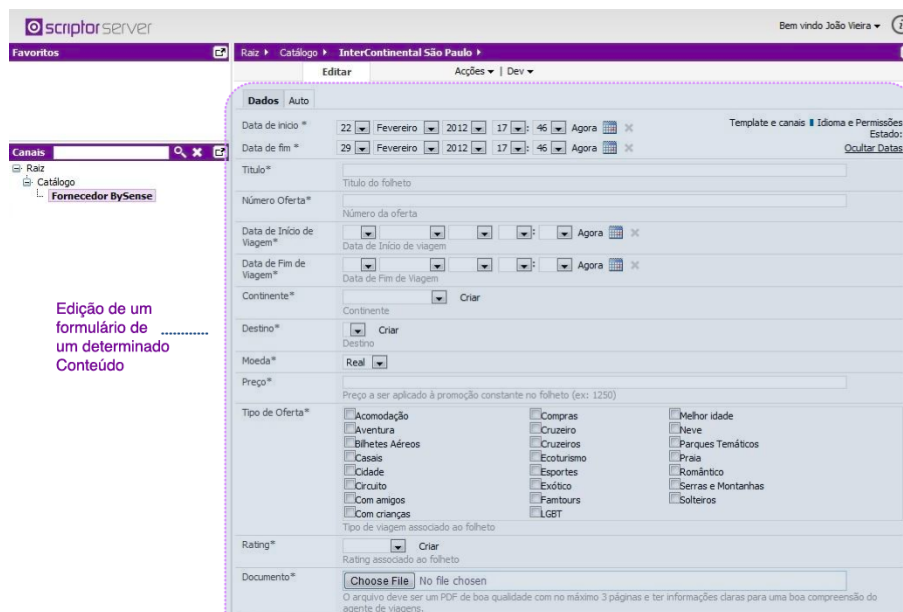


Figura 3.2: Interface Scriptor Server, formulário de um determinado Conteúdo.

As Figuras 3.1 e 3.2 mostram a *interface* Scriptor Server existente e dimensionada para PC. O Agente de Viagens, através dos Canais, seleciona a lista onde pretende introduzir um novo Conteúdo (oferta de viagem). O resultado final é possível verificar-se através das Figuras 3.3 e 3.4.

Código	Fornecedor	Miniatura	Título	Continte	Destino	Tipo de Oferta	Preço	Data de Início	Data de Fim
41	Alliance Travel Turismo Ltda		Orlando e New York - Julho	América do Norte	Estados Unidos da América	Acomodação Bilhetes Aéreos Compras Com crianças Parques Temáticos	USD\$ 3.995,00	11-07-2012	26-07-2012
40	Alliance Travel Turismo Ltda		Semana Santa com Guia Brasileiro	América do Norte	Estados Unidos da América	Acomodação Parques Temáticos Compras	USD\$ 2.695,00	26-03-2012	07-04-2012
38	Fenix Operadora Turistica Ltda		Semana Santa - Santiago - 5 dias - Saída 04/04	América do Sul	Chile	Acomodação Casais Cidade Com amigos Com crianças Melhor idade Solteiros	USD\$ 691,00	04-04-2012	08-04-2012
39	Eurorentlei Rent a Car Portugal Lda		Reservas por modelo de Carro e não Grupo!	Europa	Portugal	Acomodação Aluguel de carro	€ 18,00	02-03-2012	31-05-2012
37	Fenix Operadora Turistica Ltda		Promoção Baixa Temporada	América do Sul	Argentina	Acomodação Cidade Com amigos Com crianças Compras Melhor idade	USD\$ 438,00	01-03-2012	15-06-2012
36	Calcos Brasil Operadora Turistica		Buenos Aires - Super promoção LAN	América do Sul	Argentina	Casais Acomodação	USD\$ 424,00	01-03-2012	10-06-2012

Figura 3.3: Interface do Site do cliente da Plataforma Scriptor Server. Lista de ofertas da Agência de Viagens.



Figura 3.4: Resultado final da inserção de um novo Conteúdo (oferta de viagem).

3.3 Casos de Estudo

Sendo o objetivo deste projeto desenvolver uma aplicação web para dispositivos móveis, foram, para tal, estudadas e analisadas várias aplicações, de modo a tentar mitigar, neste projeto, erros e estratégias falhadas de outros projetos. Além disso, pretendeu-se, com este estudo, ganhar alguma percepção e espírito crítico, no que diz respeito a *interfaces* e arquiteturas, e daí retirar ilações úteis, que pudessem ser aplicadas aquando do desenvolvimento da aplicação.

Serão, assim, analisadas em seguida, algumas Web Apps e será feita uma comparação com as versões Desktop e mobile, incidindo-se principalmente no que se refere à usabilidade e ao modo como a informação é organizada e as funcionalidades que tiveram de ser sacrificadas por imposição tecnológica ou devido ao tamanho do monitor, por ser inferior ou por não ter a relevância necessária para aquele contexto.

No primeiro caso de estudo será analisada a *interface* da aplicação Metronic, similar à que se pretende desenvolver. Trata-se de uma solução para administração de websites *backoffice*, da qual apenas será analisada a sua componente gráfica, pois existem poucos dados sobre a sua implementação e arquitetura.

Por sua vez, o segundo caso, Keyhole Gestor de Identidades, trata-se de uma aplicação de gestão de identidades e presenças que é uma das características que o produto Scriptor, hoje em dia, oferece. Neste caso o alvo de análise será a implementação e a arquitetura, uma vez que a *interface* é simples e demasiado simplista, não trazendo mais valias ao projeto.

Por último, o terceiro caso é referente à aplicação do Financial Times Journal (FT), uma das aplicações com maior tráfego de utilizadores e que foi uma das primeiras soluções com

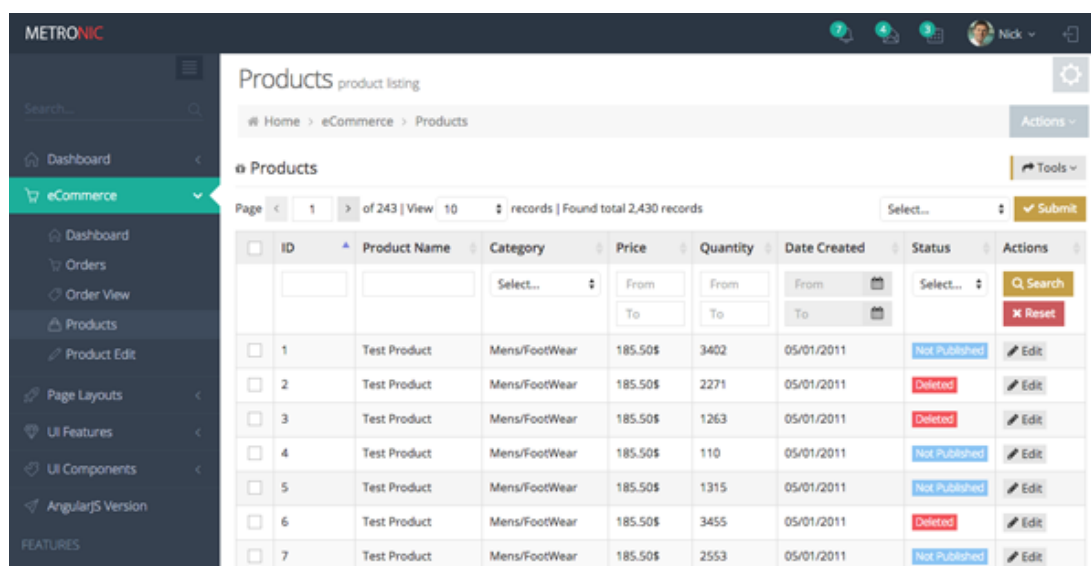
maior êxito na utilização destas tecnologias. Tal como no segundo caso, também neste, será analisada a implementação e arquitetura da aplicação. O Blog existente sobre o desenvolvimento desta aplicação, facilitou bastante a compreensão da estratégia e solução seguidas.

3.3.1 Metronic

Metronic é o *template* de uma aplicação capaz de gerir qualquer tipo de website *backoffice*. Tem uma *interface responsive* bastante apelativa, sendo que esta foi dada como exemplo a seguir, pelo facto de a disponibilização dos conteúdos ser a pretendida. Esta aplicação tem uma disposição apelativa e é fácil de entender por parte do utilizador, sendo, além disso, bastante similar à usada no Scriptor Server.

Nas versões Desktop e Tablet é possível identificar três áreas: do lado esquerdo, a Barra de Navegação com todos os canais disponíveis; no topo, a Identificação da Aplicação e Botões com determinadas funcionalidades e avisos; e a parte que abrange maior superfície de ecrã é a área onde os conteúdos estão dispostos, quer em forma de Tablets ou de formulários de conteúdos.

À medida que o monitor diminui (versão mobile), também a sua *interface* e disposição são alteradas, passando a ter três áreas na mesma, mas apenas duas são visíveis, alternando através da escolha do utilizador entre a Área de Conteúdos e a Barra de Navegação. O utilizador, através do Botão assinalado, escolhe a área pretendido como é possível verificar através das figuras 3.1 e 3.2.



The screenshot displays the Metronic eCommerce dashboard. On the left is a dark sidebar with navigation options: Dashboard, Orders, Order View, Products, Product Edit, Page Layouts, UI Features, UI Components, and AngularJS Version. The main content area is titled 'Products product listing' and shows a breadcrumb trail: Home > eCommerce > Products. Below this is a table with 7 columns: ID, Product Name, Category, Price, Quantity, Date Created, Status, and Actions. The table contains 7 rows of test products. The first row has ID 1, Product Name 'Test Product', Category 'Mens/FootWear', Price '185.50\$', Quantity '3402', Date Created '05/01/2011', and Status 'Not Published'. The second row has ID 2, Product Name 'Test Product', Category 'Mens/FootWear', Price '185.50\$', Quantity '2271', Date Created '05/01/2011', and Status 'Deleted'. The third row has ID 3, Product Name 'Test Product', Category 'Mens/FootWear', Price '185.50\$', Quantity '1263', Date Created '05/01/2011', and Status 'Deleted'. The fourth row has ID 4, Product Name 'Test Product', Category 'Mens/FootWear', Price '185.50\$', Quantity '110', Date Created '05/01/2011', and Status 'Not Published'. The fifth row has ID 5, Product Name 'Test Product', Category 'Mens/FootWear', Price '185.50\$', Quantity '1315', Date Created '05/01/2011', and Status 'Not Published'. The sixth row has ID 6, Product Name 'Test Product', Category 'Mens/FootWear', Price '185.50\$', Quantity '3455', Date Created '05/01/2011', and Status 'Deleted'. The seventh row has ID 7, Product Name 'Test Product', Category 'Mens/FootWear', Price '185.50\$', Quantity '2553', Date Created '05/01/2011', and Status 'Not Published'. Each row has an 'Edit' button in the Actions column. Above the table, there are search and filter options, including a search bar, a 'Submit' button, and a 'Reset' button.

ID	Product Name	Category	Price	Quantity	Date Created	Status	Actions
1	Test Product	Mens/FootWear	185.50\$	3402	05/01/2011	Not Published	Edit
2	Test Product	Mens/FootWear	185.50\$	2271	05/01/2011	Deleted	Edit
3	Test Product	Mens/FootWear	185.50\$	1263	05/01/2011	Deleted	Edit
4	Test Product	Mens/FootWear	185.50\$	110	05/01/2011	Not Published	Edit
5	Test Product	Mens/FootWear	185.50\$	1315	05/01/2011	Not Published	Edit
6	Test Product	Mens/FootWear	185.50\$	3455	05/01/2011	Deleted	Edit
7	Test Product	Mens/FootWear	185.50\$	2553	05/01/2011	Not Published	Edit

Figura 3.5: Interface Metronic para Tablet e PC Fonte:<http://www.keenthemes.com>, consultado a 2 de Junho de 2015.

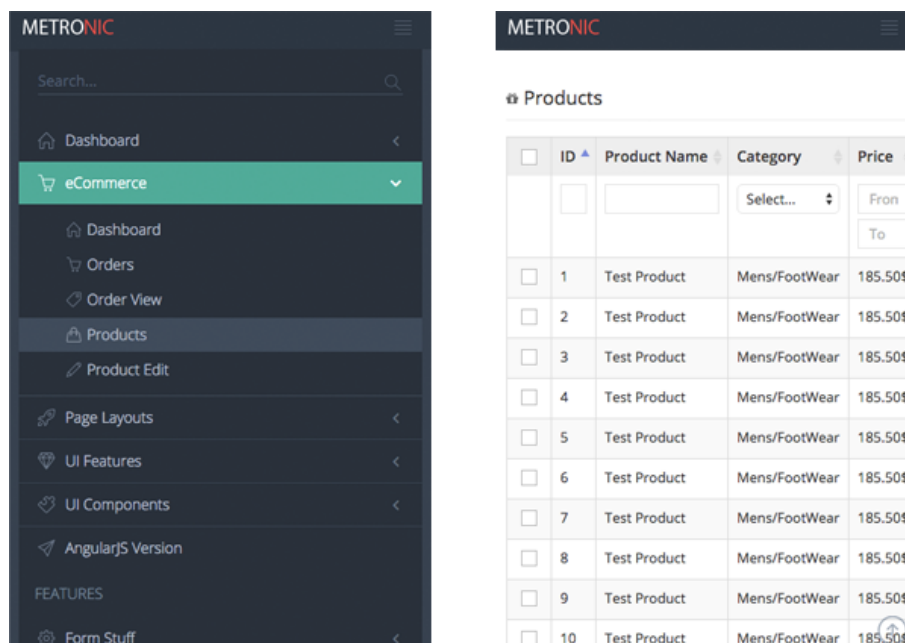


Figura 3.6: Interface Metronic para Smartphone Fonte:<http://www.keenthemes.com>, consultado a 2 de Junho de 2015.

Existem, no entanto, alguns pormenores sobre os quais foi decidida uma abordagem diferente, nomeadamente a possibilidade da Barra de Navegação e a Área de Conteúdos ser independente, isto é, o utilizador poderá ter a possibilidade de movimentar cada Área de Conteúdos, ficando estática a Barra de Navegação e vice-versa. Também a área no texto da aplicação, denominada por *Header*, deverá ficar fixa em qualquer tipo de dispositivo.

3.3.2 Keyhole Gestor Identidades

A Keyhole Software desenvolveu um gestor de identidades (colaboradores) e presenças. Esta aplicação pode ser acessada por qualquer tipo de dispositivo, advindo a sua portabilidade da utilização do HTML5. A empresa apostou no HTML5, pois este tem vindo a ser adotado por todos os *browsers*, devido às novas características como: localStorage, canvas, suporte total do CSS3, APIs de localização, novos serviços de áudio e vídeo, otimização dos mobile *browsers*, melhoria da performance de processamento do JavaScript. Foram, essencialmente, as suas características que motivaram a escolha desta tecnologia por parte do grupo de trabalho.

A análise desta aplicação será dividida em duas partes: o lado servidor (Server Side) e a *interface* (Front End).

O Server Side foi desenvolvido em Java EE com um sistema de armazenamento através de uma base de dados relacional MySQL. O acesso à base de dados MySQL foi implementado através de web services, acessados através de pedidos URL, via HTTP.

O Front End foi todo desenvolvido em HTML, CSS e JavaScript, com auxílio de algumas

frameworks:

- Bootstrap - é uma framework open source, para melhoria dos estilos de UI (*interface* Utilizador).
- jQuery - é uma biblioteca JavaScript, código aberto (open source), de modo a simplificar os scripts/funções desenvolvidos pelo programador. Esta biblioteca facilita o desenvolvimento de pedidos em AJAX, facilitando a interação Front End e Server.
- Backbone.js - é uma framework MVC (model - view - controller). O modelo consiste em separar lógica/funções, regras do negócio e vistas, possibilitando a reutilização do código para diferentes funcionalidades.

O papel do JavaScript é fundamental, uma vez que é ele o gerador de pedidos AJAX, com os web services implementados no servidor e, ao mesmo tempo, tem facilidade e rapidez na interpretação e decodificação dos objectos JSON, através dos seguintes métodos:

- JSON.parse() - cria objetos em JSON.
- JSON.stringify() - transforma os objetos JSON em Strings.

O JSON, por sua vez, é a linguagem de intercâmbio de dados, super leve, conseguindo mesmo superar o XML neste aspeto, consumindo, deste modo, menos dados e tornando, claro está, as transferências mais rápidas.

3.3.3 FT Financial Times Journal

Inicialmente foi criada uma aplicação nativa para iPad e iPhone, mas com o aumento de dispositivos Android, a FT sentiu-se obrigada a criar uma solução para lidar com este problema. Então, numa primeira fase, considerou criar duas aplicações, uma para iPhone e iPad e outra para Android. Contudo, a criação de mais uma aplicação nativa obrigava à contratação de uma equipa para o desenvolvimento e a contratação de duas equipas de administração especializadas para cada uma das aplicações, o que tornaria os custos elevadíssimos. Neste sentido, acabaram por optar por uma solução menos dispendiosa, isto é, a criação de uma multi-plataforma, usando tecnologias HTML5, que diminuiria os custos em 80% e aumentaria as receitas, pois não seria necessária a publicação nas lojas de aplicações que retirariam 20% do valor gerado pelos utilizadores.

Ao nível da *interface* tentaram reduzir ao máximo o uso de JavaScript no conceito de *responsive design*, dando principal relevância ao CSS através da Flexbox (sistema de caixas flexíveis ou fixas consoante a necessidade), dando a possibilidade ao programador a habilidade de declarar os elementos flexíveis quer estejam dispostos na vertical quer estejam na horizontal.

Usaram algumas Frameworks no desenvolvimento da *interface* como jQuery e Bootstrap. Devido à complexidade da Web App, implementaram a aplicação através de módulos, usando o conceito MVC (model - view - controller). Este conceito foi criado através da Framework - Backbone.

Por se tratar de uma aplicação de notícias, o uso de imagens é fundamental, pois uma das estratégias para que o utilizador leia ou escolha uma determinada notícia é precisamente a imagem, pois esta consegue mais facilmente captar a atenção do leitor/utilizador. Porém, a utilização de imagens coloca dois problemas, uma vez que é necessário dimensionar a imagem para um determinado dispositivo e, ao mesmo tempo, tornar o ficheiro o mais pequeno possível, de modo a reduzir o tráfego de dados e o espaço de armazenamento. A FT adotou o uso de *softwares* especializados na redução de imagens degradando o mínimo possível a imagem.

Também ao nível do *scroll* foi implementada uma biblioteca capaz de fazer movimentos na vertical e na horizontal em caixas/conteúdos independentes, tornando a utilização da Web App mais fluída e parecida com a experiência numa aplicação nativa.

Esta aplicação tem a possibilidade de trabalhar *online* e *offline* através de uma base de dados do *browser*. Inicialmente a FT pensou que a utilização da AppCache fosse a solução para todas as necessidades de armazenamento, dado que é possível aumentar a sua capacidade de acordo com as necessidades. Todavia, revelou-se um falhanço, uma vez que qualquer ficheiro que fosse atualizado, originava uma atualização de todos os ficheiros armazenados, levando o utilizador a alguns minutos de espera. No entanto, a AppCache não foi descartada, pois o seu uso é fundamental, visto que é ela a única que dá a possibilidade de trabalhar em modo *offline*. Assim, é utilizada para as definições básicas, como código HTML e algumas imagens que não necessitam de ser atualizadas muitas vezes, uma vez que sempre foram atualizadas no ficheiro. A aplicação sofrerá uma atualização total que, em alguns casos, mediante o tamanho dos ficheiros, poderá levar alguns minutos.

O uso do localStorage serve para armazenar todo o código JavaScript e CSS, que necessita de ser mais vezes atualizado.

Por último, usam dois sistemas de armazenamento, com uma capacidade de armazenamento acima da do localStorage, onde guardam todas as imagens e artigos do jornal. O uso de dois sistemas deve-se ao facto do *browser* Safari suportar apenas o Web SQL, que é um sistema idêntico ao SQL, mas que entretanto foi abandonado pela W3C, deixando de dar suporte a esta tecnologia, passando a desenvolver um sistema de armazenamento denominado IndexedDB. Este trata-se de um sistema com performance superior, que os restantes *browsers* adotaram como sistema de armazenamento de grande capacidade, levando a FT a desenvolvê-lo para a sua aplicação.

O uso destes dois sistemas de armazenamento não só permite a utilização da aplicação *offline*, como diminui bastante o tempo de resposta às solicitações dos utilizadores, dado que deixam de ser necessários alguns pedidos ao servidor através da Internet, buscando-se

esses conteúdos nos sistemas de armazenamento do *browser*.

Sempre que é necessário um novo conteúdo e interação com o servidor, a aplicação faz um pedido HTTP, através de um determinado URL. O servidor, através de um Web Service, responde um Array de objetos JSON, que a App FT transforma em String e guarda no sistema de armazenamento do *browser*.



Figura 3.7: Interface da web App FT Fonte:<http://www.ft.com>, consultado a 2 de Junho de 2015.

3.3.4 Conclusões

Após uma primeira análise, onde foram identificadas as várias abordagens possíveis para o uso do produto Scriptor Server em dispositivos *mobile*, optou-se por uma que pudesse servir todos os tipos de dispositivos independentemente do sistema operativo, recaindo, portanto, a escolha pelo desenvolvimento de uma aplicação em HTML5, capaz de garantir essa necessidade. Foi realizada uma nova reunião com os colaboradores da Viatecla e, com base nestes Casos de Estudo, foram discutidos e retiradas as seguintes conclusões:

- como desenvolver uma *interface* capaz de se adaptar a todos os tipos de dispositivos, ficando decidido o uso do conceito de Responsive Web Design e as suas técnicas inerentes, como a utilização de media queries e flexboxes.
- como fazer a comunicação ao servidor, na obtenção e utilização de conteúdos, onde ficou decidido o uso da API REST e criação e melhoramento dos Web Services criados de modo a servir a nova aplicação iScriptor.
- qual a linguagem de transporte desses dados que melhor se adequa ao conceito mobile XML ou JSON. A escolha recaiu na utilização do JSON, uma vez que para

estes casos é bastante superior, visto ser mais leve do que o XML. Tem um tempo de transferência inferior, ocupando, por isso, menos espaço nas bases de dados dos *browsers*, poupando, assim, espaço para outros conteúdos, já que o espaço destes é limitado. Além disso, é fácil de guardar e interpretar através do JavaScript.

- qual o melhor ou melhores sistemas de armazenamento a usar, consoante determinadas características e funcionalidades, ficando decidido o uso do `localStorage`, visto ser o caminho mais simples e com menor risco, pois o Web SQL foi descontinuado pela W3C e o IndexedDB encontra-se em fase de desenvolvimento.
- como conseguir trabalhar a aplicação em modo *offline*, usando a `AppCache` e como camuflar a página web parecendo uma aplicação nativa, isto é, criar ícones nos dispositivos e abrindo diretamente a aplicação, sem ser necessário abrir qualquer *browser*, nem escrever um endereço URL.
- como editar os conteúdos em modo *offline* usando o mecanismos de armazenamento dos *browsers* (`localStorage`).

3.4 iScriptor

O Projeto iScriptor diz respeito ao objetivo de ser criada uma aplicação para dispositivos móveis com características específicas. O principal objetivo desta aplicação é fornecer, aos utilizadores da plataforma Scriptor, um novo meio para a leitura, pesquisa e edição dos seus conteúdos.

A plataforma conta com uma *interface* web desenvolvida para computadores pessoais, sendo que esta deixa de ser funcional aquando da sua utilização por um dispositivo móvel (Smartphone, Tablet). Assim, a aplicação terá de ultrapassar algumas barreiras impostas por estes dispositivos como é o caso do reduzido tamanho dos monitores e o uso da aplicação em modo *Offline*, devido à perda de sinal de Internet, seja ela por 3G, 4G ou Wi-Fi, muito característica nestes dispositivos.

Para tal, a aplicação terá de conseguir comunicar com o servidor Scriptor, permitindo a um determinado utilizador aceder aos seus próprios conteúdos, assim como editá-los. É importante que estes conteúdos sejam guardados no próprio dispositivo, de modo a tornar-se possível a sua visualização e uso quando o dispositivo perder a ligação à Internet.

A *interface* do Scriptor Server para PC é dividida em três áreas. No topo, o denominado *header*, com a identificação do produto e com um botão no canto superior direito. No lado esquerdo, a lista de canais onde estão disponíveis todos os canais que o utilizador tem privilégios para visualizar. Estes canais são controlados e filtrados através de grupos e cada canal tem uma lista de grupos associada, caso o utilizador se encontre num desses grupos terá o privilégio de visualizar e editar os conteúdos desse canal. Cada canal, pode ter canais filhos e ao mesmo tempo ter uma lista de conteúdos. Essa lista de conteúdos é disponibilizada na área com maior espaço, ocupando mais de 70% do ecrã. O utilizador

poderá escolher o conteúdo pretendido da lista de conteúdos, sendo, assim, aberto um formulário que foi definido pelo criador do canal.

3.5 Web Service

O uso de Web Services potencia as funcionalidades dos dispositivos móveis, uma vez que é possível criar variadas aplicações com um grau de complexidade menor e com um potencial elevado. Esta abordagem é a ideal em aplicações que necessitem de consultar um servidor para recolher informação ou que necessitem de fornecer dados a um servidor, como por exemplo a edição de um determinado conteúdo. Isto vai de encontro a um dos objetivos propostos para a nova aplicação móvel, iScriptor.

Antes deste projeto foi criada uma *interface* REST, no Scriptor Server, com o objetivo de servir algumas aplicações Web. Esta Interface foi, assim, aproveitada e reformulada permitindo a troca de informação estruturada, usando HTTP (Hyper Text Transfer Protocol).

A utilização dessa API REST é feita por duas operações importantes que são o GET e o POST. O GET será utilizado em cada pedido efetuado pelo utilizador ao Scriptor, de modo a conseguir obter o conteúdo pretendido. O método POST será utilizado em cada edição que o utilizador efetue num dado conteúdo. Para esta troca de informação é utilizada uma notação, num formato JSON[6]. É uma formatação leve de troca de dados que para um ser humano é de fácil leitura e escrita, como se pode verificar através da Figura 3.8. Para as máquinas é fácil de interpretar e gerar, tornando-se, assim, um formato ideal de troca de dados, ficando os seguintes web services definidos:

- `validateUser` - valida o utilizador através de um *user* e uma *password* válida. É-lhe atribuída uma chave de autenticação 'token'. É através dessa chave de autenticação que são realizados todos os outros web services aqui mencionados.
- `channelList` - fornece uma lista com todos os canais 'filhos' de um determinado canal 'pai'. É possível escolher a profundidade pretendida.
- `fieldList` - fornece uma lista com todos os campos de um determinado conteúdo (ContentID). Ao mesmo tempo, um canal pode ou não ter conteúdos.
- `contentList` - fornece uma lista com todos os conteúdos de um determinado Canal.
- `contentListConditional` - consoante determinadas condições é possível obter variadas formas de uma lista de conteúdos.
- `getContent` - fornece o valor de cada campo de um determinado Conteúdo.
- `saveContent` - pedido de edição de conteúdo, bastando para tal indicar o valor do campo de um determinado Conteúdo.

- `getView` - fornece todos os dados necessários para implementação de vistas. A vista pode ou não ser um agrupamento de conteúdos. Este serviço dá as informações necessárias da existência do número e nome das vistas, bem como a estrutura de cada uma, nomeadamente se é ou não uma vista agrupada e, em caso afirmativo, qual o campo de agrupamento.

```

- {
  "Result": "OK",
  - "ChannelData": {
    "Id": "7CF6D26B-C2EB-4B03-A63F-32E2FFA97149",
    "ParentId": "",
    "ParentName": "",
    "Name": "Viatecla",
    "Desc": "1",
    "Path": "/Viatecla",
    "BackofficeLayout": "2",
    - "Descendents": [
      - {
        "Id": "3B6A8CAC-23DF-4B70-BC16-7EA369AF9D8C",
        "Name": "Viatecla Nova Imagem",
        "Desc": "",
        "Childs": "15",
        "Path": "/Viatecla/Viatecla Nova Imagem",
        "BackofficeLayout": "2"
      },
      - {
        "Id": "C68C5E2D-5B8C-4E9C-B3E0-38FFE1E20A33",
        "Name": "e-jobs",
        "Desc": "",
        "Childs": "3",
        "Path": "/Viatecla/e-jobs",
        "BackofficeLayout": "1"
      },
      - {
        "Id": "2B7B0786-E7F0-41D0-80B4-CA92178B55C5",
        "Name": "Repositorio",
        "Desc": "",
        "Childs": "1",
        "Path": "/Viatecla/Repositorio",
        "BackofficeLayout": "1"
      }
    ]
  }
}

```

Figura 3.8: Resposta JSON do Scriptor Server a um pedido `'channelList'` feito pela aplicação iScriptor.

3.6 Análise de Requisitos e de Utilizadores

Serão apenas feitas as descrições dos vários casos de uso do Utilizador (Mobile User) da App iScriptor. A plataforma Scriptor pertence a um nível superior, controlado inteiramente pela empresa que a criou e que a gere, a Viatecla, pelo que a descrição dos casos de utilização deste não se tornam importantes para a caracterização da App iScriptor.

As Funcionalidades que o Utilizador poderá efetuar serão:

- Autenticação
- Escolher Canal
- Escolher Conteúdo
- Pesquisar Lista de Conteúdos
- Visualizar Conteúdo
- Atualizar Conteúdo
- Terminar Sessão (Logout)

Existem, no entanto, outras funcionalidades que apenas fazem sentido se forem utilizadas em dispositivos onde o ecrã é mais generoso, como é o caso dos Tablets. Serão implementadas algumas funcionalidades criadas apenas para Tablets, nomeadamente sobre a organização e forma como os conteúdos são apresentados, bem como a sua edição.

As funcionalidades que o Utilizador do Tablet terá, além das anteriores, serão:

- Editar Conteúdo
- Escolher Vistas (Colunas Visíveis, Filtrar, Ordenar, Agregar, Pagar)
- Ordenação por Colunas
- Filtros (Colunas)

Função	Autenticação
Âmbito	Web App iScriptor para a plataforma Scriptor Server
Finalidade	Permite ao utilizador autenticar-se perante a Plataforma. O utilizador necessita de colocar um Username, uma Password esta operação só é efectuada na primeira vez que executa a App
Utilizadores	Utilizador da plataforma Scriptor
Pré-Condições	O utilizador terá de estar previamente registado no servidor e terá de ter uma ligação activa à Internet
Pós-Condições	O utilizador fica identificado, sendo-lhe atribuído um id de sessão.
Fluxo Típico de Eventos	1. O Utilizador coloca a App em execução no seu dispositivo. 2. O utilizador insere o Username a Password e carrega no botão Entrar. 3. O sistema fará um pedido de validação apenas se o dispositivo tiver uma ligação de Internet activa.
Fluxo Alternativo	O utilizador insere mal algum dos três parâmetros e surgirá uma mensagem de alerta informando que existe algum erro nos dados fornecidos.

Tabela 3.1: Autenticação do utilizador na App iScriptor.

Função	Escolher Conteúdo
Âmbito	Web App iScriptor para a plataforma Scriptor Server
Finalidade	Permite ao utilizador escolher um Conteúdo da lista com os vários conteúdos
Utilizadores	Utilizador da plataforma Scriptor
Pré-Condições	O utilizador terá de efectuar a Função Escolher Canal. A Lista de Conteúdos será carregada através da informação guardada no dispositivo. Sempre que o dispositivo tenha Internet, fará pedidos HTTP GET ao servidor, de modo a actualizar a informação sempre que consiga.
Pós-Condições	Sempre que um pedido HTTP GET seja bem sucedido, fará uma actualização na base de dados do dispositivo.
Fluxo Típico de Eventos	1. O Utilizador selecciona um determinado Canal da lista de canais. 2. Será apresentado uma lista de Conteúdos. Na versão Smartphone cada Conteúdo será apresentado com o título do conteúdo a sua data de criação e da última actualização. Na versão Tablet cada Conteúdo é apresentado com todos os campos seleccionados no servidor. 3. O Utilizador escolhe o conteúdo pretendido
Fluxo Alternativo	

Tabela 3.2: Utilizador escolhe Conteúdo na App iScriptor.

Função	Visualizar Conteúdo
Âmbito	Web App iScriptor para a plataforma Scriptor Server
Finalidade	Permite ao utilizador visualizar os vários parâmetros de um dado Conteúdo.
Utilizadores	Utilizador da plataforma Scriptor
Pré-Condições	O utilizador terá de efectuar a Função Escolher Conteúdo.
Pós-Condições	Nada é alterado no sistema o utilizador tem acesso aos vários parâmetros de um dado Conteúdo.
Fluxo Típico de Eventos	1. O Utilizador selecciona um determinado Conteúdo da lista de Conteúdos. 2. Será apresentado o Conteúdo seleccionado com os vários parâmetros.
Fluxo Alternativo	

Tabela 3.3: Utilizador visualiza Conteúdo na App iScriptor.

Função	Actualizar Conteúdo
Âmbito	Web App iScriptor para a plataforma Scriptor Server
Finalidade	Permite ao utilizador com uma sessão validada obter a última versão de um determinado Conteúdo em tempo real
Utilizadores	Utilizador da plataforma Scriptor
Pré-Condições	O utilizador terá de seleccionar o Conteúdo pretendido. Terá de ter acesso à Internet de modo a efectuar um pedido GET ao servidor de modo a obter a última versão de um dado Conteúdo.
Pós-Condições	É apresentado o Conteúdo actualizado. Será substituída a informação guardada no dispositivo com a informação mais actualizada.
Fluxo Típico de Eventos	1. O sistema apresenta o formulário do Conteúdo. 2. O Utilizador selecciona o botão para actualizar o Conteúdo 3. O Sistema fará um pedido HTTP com o servidor onde fará um pedido GET à sua API REST. 4. O sistema gerará o novo Conteúdo através da resposta do servidor.
Fluxo Alternativo	Ao clicar no botão de Actualizar caso o dispositivo não tenha uma ligação de Internet activa, aparecerá uma mensagem de alerta reportando ao Utilizador a falta de Internet e o pedido não será efectuado.

Tabela 3.4: Utilizador actualiza Conteúdo na App iScriptor.

Função	Pesquisa lista Conteúdos
Âmbito	Web App iScriptor para a plataforma Scriptor Server
Finalidade	Permite ao utilizador pesquisar sobre um determinado atributo numa determinada coluna, bastando para isso inserir no topo de uma determinada coluna o que pretende pesquisar
Utilizadores	Utilizador da plataforma Scriptor
Pré-Condições	O utilizador terá de efectuar a Função Escolher Canal com sucesso.
Pós-Condições	São apresentados os resultados da pesquisa em forma de lista com todos os Conteúdos que contenham essa String.
Fluxo Típico de Eventos	1. O Utilizador introduz o nome do Conteúdo que procura, sobre um determinado atributo 2. Será feita uma pesquisa por todos os Conteúdos que tenham essa String nesse atributo. 3. O sistema gerará uma lista de Conteúdos com os resultados da pesquisa. O utilizador poderá fazer Scroll sobre a lista de Conteúdos e seleccionar o que pretende
Fluxo Alternativo	Poderá não existir nenhum Conteúdo com esse nome, será exibido um alerta ao Utilizador a avisar que não foi encontrado nada com esse nome, ficando a visualizar o mesmo Menu em que está.

Tabela 3.5: Utilizador pesquisa sobre Conteúdos na App iScriptor.

Função	Ordenação por Colunas
Âmbito	Web App iScriptor para a plataforma Scriptor Server
Finalidade	Permite ao Utilizador que esteja a usar um Tablet como dispositivo, escolher Ordenar uma data coluna de uma dada lista de Conteúdos
Utilizadores	Utilizador da plataforma Scriptor
Pré-Condições	O utilizador terá de ter efectuado a Função Escolher Canal.
Pós-Condições	O utilizador verá a sua lista de Conteúdos Ordenada pela Coluna seleccionada.
Fluxo Típico de Eventos	1. O utilizador perante a sua lista de Conteúdos terá de clicar no topo da coluna, de modo a ordenar a lista consoante a sua escolha. 2. Escolhida a coluna, a lista de conteúdos será disposta consoante a opção.
Fluxo Alternativo	

Tabela 3.6: Utilizador ordena lista de Conteúdos através da coluna pretendida na App iScriptor.

Função	Escolher Vistas
Âmbito	Web App iScriptor para a plataforma Scriptor Server
Finalidade	Permite ao Utilizador que esteja a usar um Tablet como dispositivo, escolher o tipo de vista (escolher colunas, filtrar, ordenar, agregar, pagnar) que pretende para uma dada lista de Conteúdos
Utilizadores	Utilizador da plataforma Scriptor
Pré-Condições	O utilizador terá de ter efectuado a Função Escolher Canal.
Pós-Condições	O utilizador verá a sua lista conforme a vista que seleccionou
Fluxo Típico de Eventos	1. O utilizador perante a sua lista de Conteúdos terá de clicar no botão Vistas, podendo escolher os vários tipos de vistas possíveis para os seus Conteúdos. 2. Escolhido a vista que pretende, os seus conteúdos serão dispostos consoante a escolha.
Fluxo Alternativo	

Tabela 3.7: Utilizador escolhe vista de conteúdos na App iScriptor.

Função	Editar Conteúdo
Âmbito	Web App iScriptor para a plataforma Scriptor Server
Finalidade	Permite ao Utilizador alterar o ou os parâmetros de um determinado Conteúdo
Utilizadores	Utilizador da plataforma Scriptor
Pré-Condições	O utilizador terá de ter efectuado a Função Visualizar Conteúdo.
Pós-Condições	Os parâmetros de um dado Conteúdo que forem modificados serão efectuados as seguintes alterações: Offline- A actualização será gravada no dispositivo numa queue, quando o dispositivo tiver uma ligação de Internet activa, será enviado para o servidor um pedido HTTP com o método POST com as alterações. Online- Será enviado para o servidor um pedido HTTP método POST com as alterações efectuadas, caso obtenha: - Resposta positiva do servidor, o pedido foi efectuado com sucesso é feito um pedido HTTP GET com a informação actualizada e é guardado no dispositivo. - Resposta negativa do servidor, o pedido não foi efectuado, o Utilizador será alertado para actualizar primeiro o Conteúdo e só depois submeter alguma alteração.
Fluxo Típico de Eventos	1. O Utilizador selecciona o botão editar quando se encontra a Visualizar o Conteúdo. 2. O Utilizador efectua as alterações necessárias. 3. Depois de editar todos os parâmetros que pretende, terá de clicar no botão Submeter alterações. 5. Utilizador carrega no botão Guardar 6. O sistema verificará se existe uma ligação à Internet activa 7. Caso exista ligação efectuará um pedido HTTP POST com as alterações ao servidor. 8. Caso a data de referência seja igual á do servidor o pedido será efectuado com sucesso.
Fluxo Alternativo	O utilizador poderá carregar no botão de retrocesso e será enviado para o Menu do Visualizar Conteúdo e as alterações efectuadas não serão tidas em conta. O utilizador poderá carregar num outro canal na lista de Canais e será enviado para os Conteúdos desse canal e as alterações não serão efectuadas. O dispositivo não tendo uma ligação de Internet disponível guardará as alterações de conteúdos numa queue de actualizações no dispositivo. Tendo o dispositivo Internet a queue de trabalho será lida e será feito um pedido HTTP POST para cada elemento da queue, com referência à data em que o ficheiro foi recebido, existindo assim vários tipos de colisões: - O Conteúdo foi alterado e tem uma data superior à do pedido, é então avaliado o histórico do Conteúdo para saber exactamente o parâmetro que foi alterado caso o parâmetro não seja o mesmo, será feita a actualização (Merge). - O Conteúdo foi alterado e tem uma data superior à do pedido, é avaliado o histórico do Conteúdo se o parâmetro que foi alterado é o mesmo, não será efectuado a actualização o servidor dará uma resposta da não actualização do Conteúdo. O dispositivo gerará um alerta ao Utilizador a informar que o pedido não foi bem-sucedido e que para concretizar a operação deve actualizar primeiro o Conteúdo, de modo avaliar as alterações já efectuadas.

Tabela 3.8: Utilizador edita Conteúdo na App iScriptor.

Função	Filtros
Âmbito	Web App iScriptor para a plataforma Scriptor Server
Finalidade	Permite ao utilizador escolher as colunas que pretende que fiquem visíveis da lista de Conteúdos.
Utilizadores	Utilizador da plataforma Scriptor
Pré-Condições	O utilizador terá de efectuar a Função Autenticação com sucesso.
Pós-Condições	É apresentado as colunas que o utilizador seleccionou.
Fluxo Típico de Eventos	1. O Utilizador carrega no botão Colunas. 2. O Utilizador desseleciona as colunas que não pretende. 3. Será feita uma pesquisa no ficheiro guardado no dispositivo por todos os Canais e Conteúdos que tenham essa String no Título. 4. O sistema gerará uma lista de Canais e Conteúdos com os resultados da pesquisa. 5. O utilizador poderá fazer Scroll sobre a lista de conteúdos e seleccionar o que pretende
Fluxo Alternativo	Seleciona outro local da tela para o widget das colunas desaparecer

Tabela 3.9: Utilizador filtra as colunas que pretende visualizar na lista de Conteúdos na App iScriptor.

Função	Sair(Logout)
Âmbito	Web App iScriptor para a plataforma Scriptor Server
Finalidade	Permite ao Utilizador sair da App iScriptor
Utilizadores	Utilizador da plataforma Scriptor
Pré-Condições	O utilizador terá tido uma Autenticção com sucesso.
Pós-Condições	O utilizador verá a sua lista conforme a vista que seleccionou
Fluxo Típico de Eventos	1. O utilizador selecciona o botão Sair no topo direito da App iScriptor. 2. A App fecha e o utilizador depara-se com o seu ambiente de trabalho.
Fluxo Alternativo	

Tabela 3.10: Utilizador faz Logout na App iScriptor.

3.7 Arquitetura

A arquitetura do Scriptor Server pode ser definida através de três camadas: Front End ou Client Side, Middleware e Camada de Armazenamento.

A Front End ou Client Side, usada pelos utilizadores da aplicação, é desenvolvida em HTML e JavaScript tratando-se de uma *interface* criada apenas para ser utilizada em Desktops e portáteis. Foi, então, necessário criar uma *interface* capaz de ser utilizada por todos os dispositivos, que deu origem a este projeto e culminou nesta dissertação de Mestrado. Os detalhes da sua implementação e estratégias seguidas serão abordadas no capítulo 4.

A segunda camada ou camada intermédia é responsável pela segurança e interação entre a camada Front End e a camada de Armazenamento. Nesta camada, a Middleware, existem dois tipos de web services. Inicialmente existia um web service onde eram gerados XML'S, de modo a servir os pedidos solicitados pelos utilizadores (Front End), posteriormente, foi criada uma API REST para servir outras aplicações.

A terceira camada é uma camada de Armazenamento, composta por uma base de dados relacional SQL Server, desenvolvida pela Microsoft. Este sistema tem sido alvo de críticas por parte dos utilizadores devido à lentidão na execução dos pedidos por parte dos utilizadores.

Com o desenvolvimento deste projeto será criada uma nova camada que ficará paralela e ao mesmo nível da camada Front End existente, sendo esta utilizada pelos dispositivos móveis, de forma a tentar colmatar a necessidade que existe em servir esse tipo de dispositivos. A nova arquitetura do Scriptor Server ficará definida como se pode verificar na Figura 3.9.

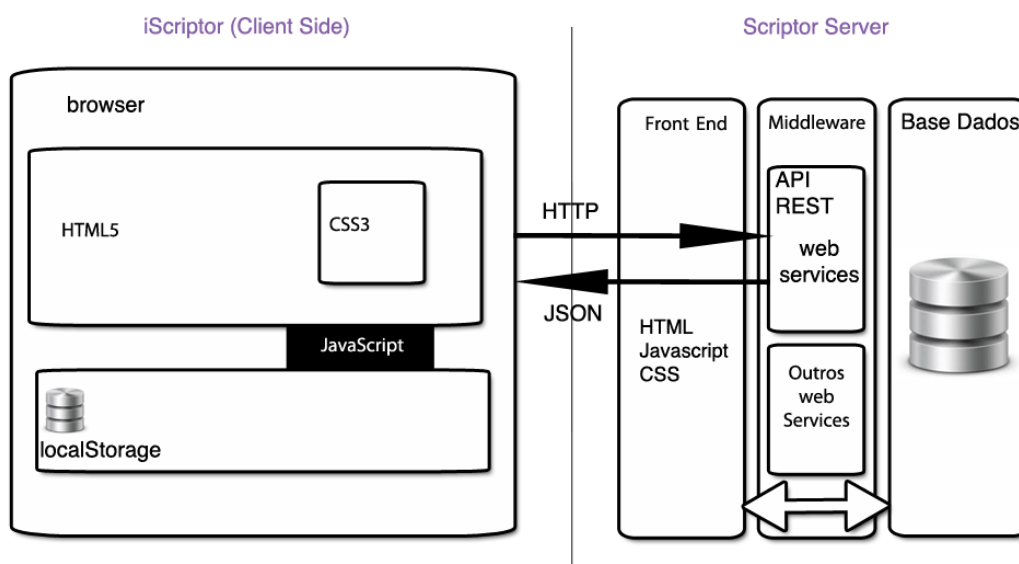


Figura 3.9: Arquitetura iScriptor e Scriptor Server.

3.8 Mockups

O projeto iScriptor consiste no desenvolvimento de uma *interface* UI que sirva o produto Scriptor Server, uma vez que tem uma *interface* apenas para dispositivos com monitor de grande dimensão. Foi, portanto, criado um conjunto de Mockups para serem, depois analisados, discutidos e validados pela equipa técnica do Scriptor Server e pelos designers da Viatecla. Os Mockups foram criados através de Photoshop, uma ferramenta de manipulação e edição de imagem. Estes Mockups incidiram na definição e disposição de várias áreas da *interface*, bem como na criação e disposição de vários ícones, que executam

determinadas funcionalidades.

A criação destes Mockups serviu, também, para diferenciar a disposição das várias áreas da *interface*, o tamanho da letra e a disposição dos conteúdos, consoante o tipo de dispositivo utilizado, Smartphone ou Tablet. Serviu, também, para perceber se todas as funcionalidades pretendidas em Tablets, fariam sentido em serem implementadas em dispositivos que têm ecrãs reduzidos, como é o caso dos Smartphones.

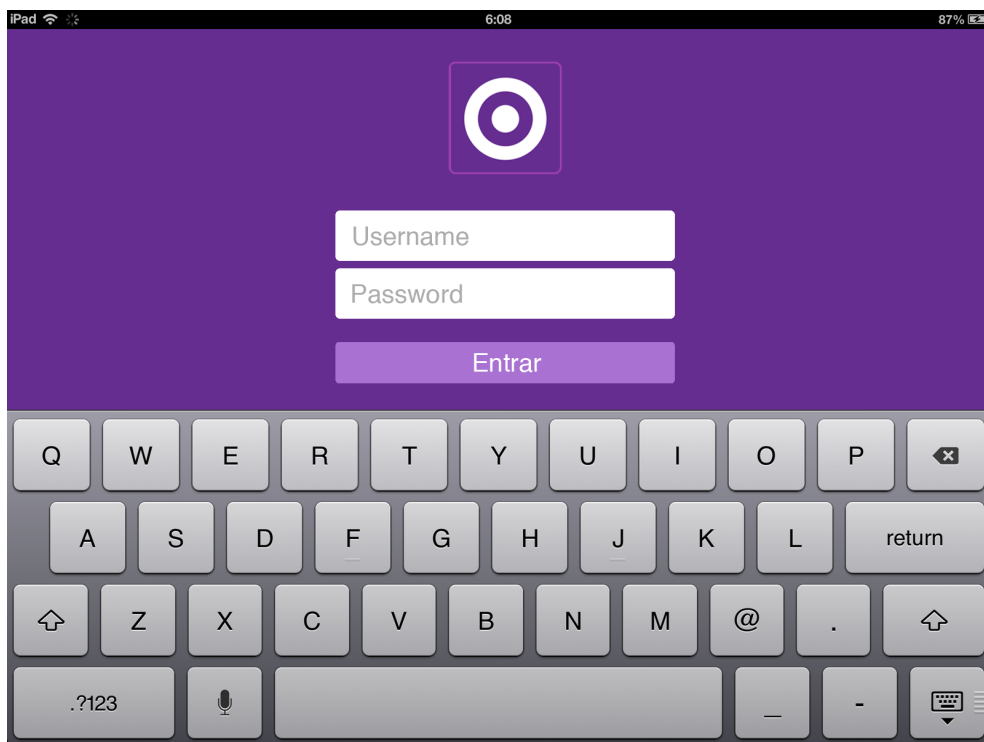


Figura 3.10: Login da aplicação no iPad.

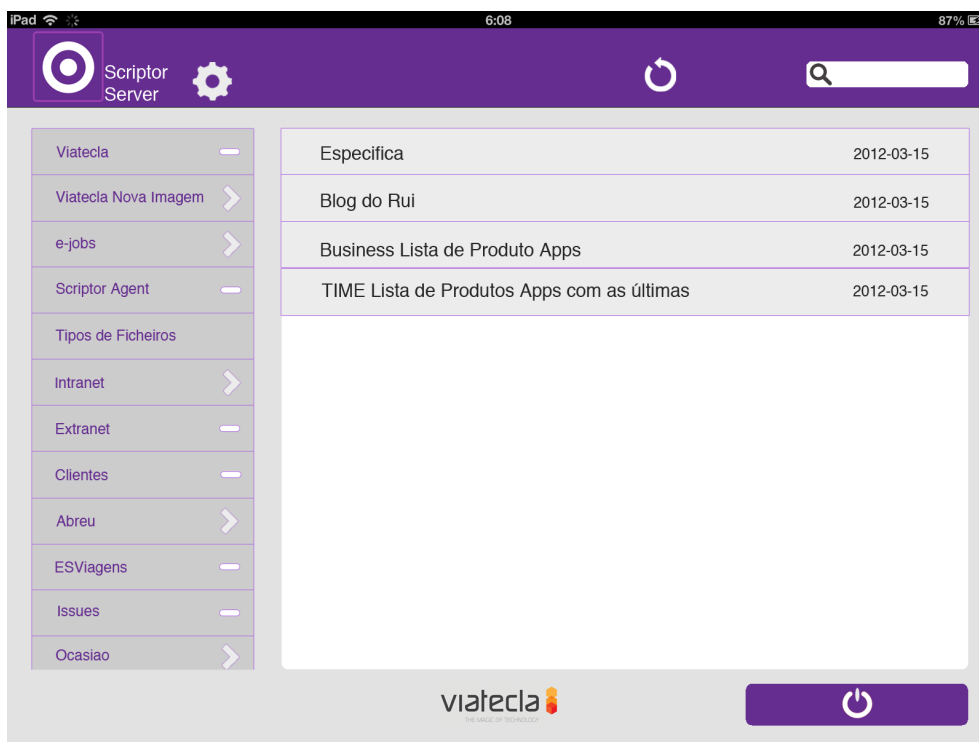


Figura 3.11: Lista de Canais vista através de um Tablet.

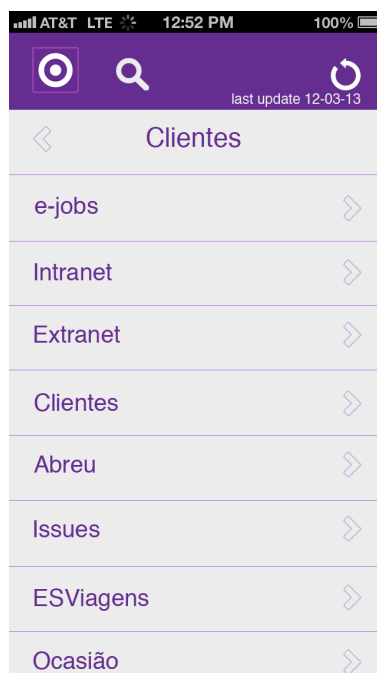


Figura 3.12: Lista de canais vista através de um Smartphone.

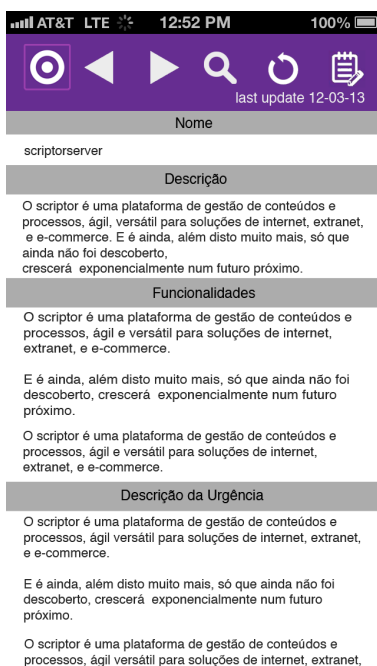


Figura 3.13: Vista de um Conteúdo através de um Smartphone.

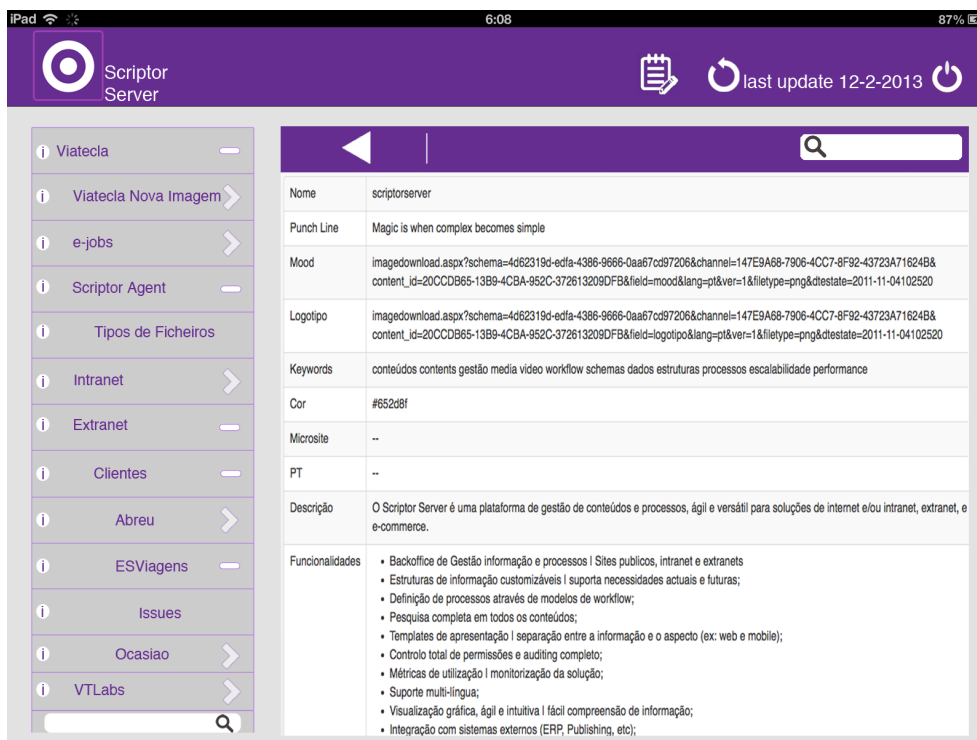


Figura 3.14: Vista de um Conteúdo através de um Tablet.



Figura 3.15: Vistas, Filtros e Ordenação na lista de Conteúdos num Tablet.

3.9 Execução de Testes de Usabilidade

Como referido na secção anterior, foram criados Mockups com o intuito de serem discutidos em reunião com a equipa técnica responsável pelo desenvolvimento do Scriptor e pelos designers da Viatecla, de modo a avaliar, testar e validar a nova *interface* do Scriptor Server para dispositivos móveis.

Tendo em conta o referido na secção 2.7 desta dissertação, onde foram resumidos os vários tipos de testes de usabilidade existentes para a avaliação da *interface*, podemos considerar que, neste projeto, se realizaram dois tipos de avaliação: um teste sobre ícones e uma avaliação heurística.

Seguidamente, será descrito o processo seguido durante a reunião de testes.

- Testes sobre ícones - através dos vários Mockups criados, foram explicados os vários ícones existentes, assim como a funcionalidade de cada um deles. A cada avaliador foi distribuída uma tabela, de modo a avaliar cada item. É possível verificar essa tabela através da Figura 3.16.

	Botões	Avaliação de 1 a 5	Observações
Header	Actualizar canal		
	Editar Conteúdo		
	Pesquisa (opção 1)		
	Retroceder		
	Configurações (opção 1)		
	Logout (opção 1)		
Content Barra de Navegação	Escolher Canal		
	Retroceder na lista de Canais		
	Verificar se o canal tem conteúdos		
Content	Conteúdos em lista		
	Retroceder		
	Pesquisa (opção 2)		
	Vistas		
	Filtros		
	Ordenação Colunas		
	Lista Simples		
Footer	Configuração (opção 2)		
	Logout (opção 2)		

Figura 3.16: Formulário de avaliação de teste sobre ícones.

- Avaliação heurística - este teste consistiu em identificar problemas e falhas de usabilidade da *interface*. A primeira parte da avaliação consistiu na descrição do projeto e dos vários Mockups, bem como na apresentação de um diagrama de estados (Figura 3.17), de modo a que os avaliadores compreendessem todas as funcionalidades e estados existentes na nova aplicação. Na segunda parte deste teste foi entregue a cada avaliador um guião, de forma a seguirem determinados passos e identificarem, ao longo do caminho, possíveis falhas e problemas relativamente à usabilidade do iScriptor.

No final, foram discutidos e analisados todos os problemas e sugestões apontadas pelos avaliadores, tentando-se encontrar soluções para esses problemas. Foi criada uma tabela com os problemas, como é possível verificar pela Figura 3.18 e foram discutidas possíveis soluções.

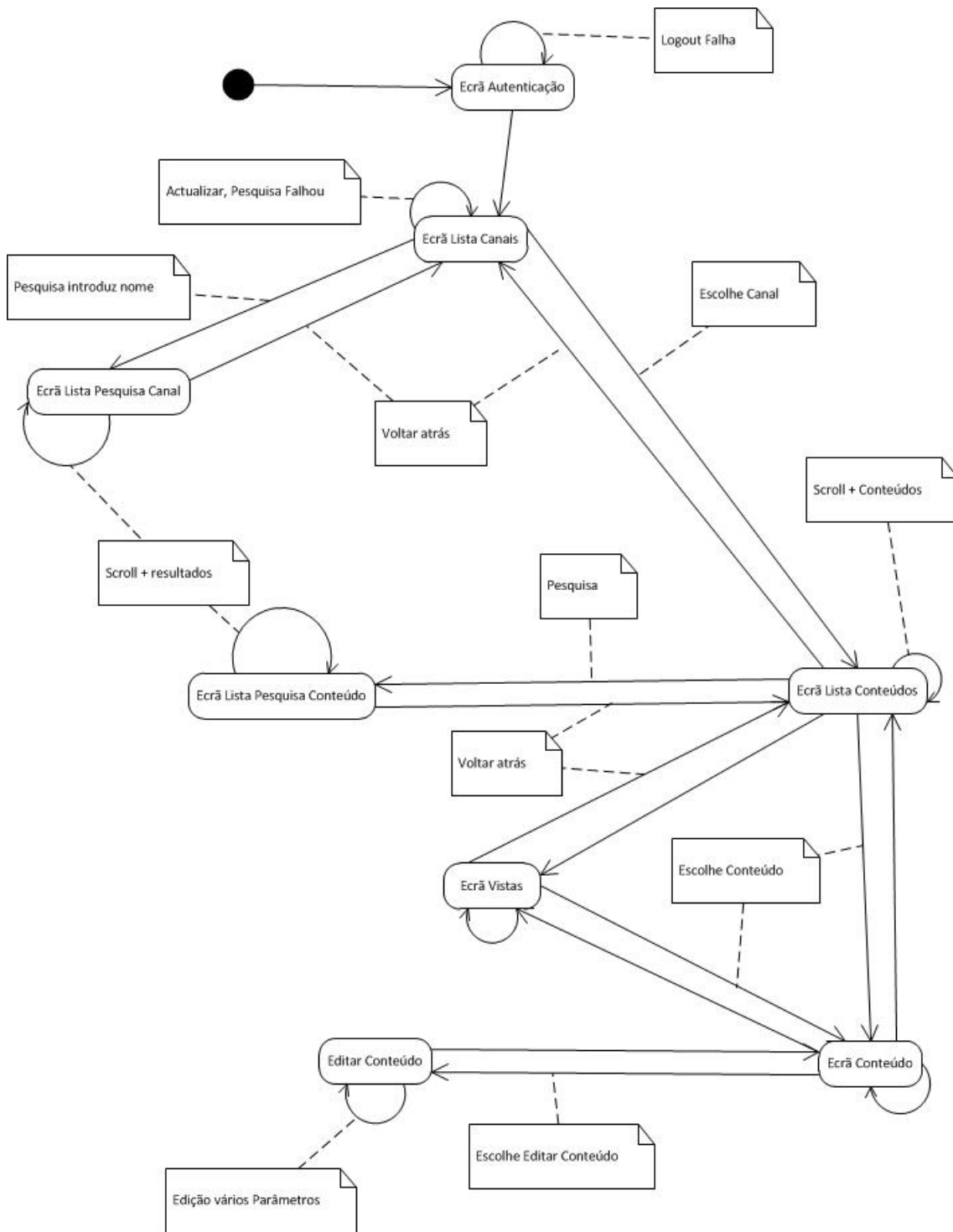


Figura 3.17: Diagrama de Estados do iScriptor.

#id	Descrição do Problema	Módulos
1	Demasiada altura do header	Header
2	Retrocesso Barra de Canais	Canais
3	Níveis na Barra de Canais	Canais
4	Identificar canal que está a ser visualizado	Canais
5	Pesquisa de Conteúdos deve fazer parte do content e não do header	Content
6	Lista de Conteúdos deve ter as colunas (campos do conteúdo)	Lista Conteúdos
7	Lista de Conteúdos deve ter cores alternadas para melhor identificação	Lista Conteúdos
8	Botão de edição e novo deve fazer parte da barra do content	Lista Conteúdos
9	Retirar Footer ocupa demasiado ecrã	Footer
10	Passar o botão de logout para o topo do ecrã	Header
11	Não faz sentido botão configurações simplificar na 1ª fase desenvolvimento	Footer
12	Retirar Funcionalidades na versão Smartphone	Smartphone

Figura 3.18: Avaliação heurística ao iScriptor. Problemas detetados.

Capítulo 4

Implementação do Projeto iScriptor

4.1 Introdução

Neste capítulo vão ser abordados os aspetos gerais e a forma como decorreu todo o processo de desenvolvimento da aplicação móvel iScriptor. Começar-se-á, então, por apresentar a abordagem selecionada para desenvolvimento da App, isto é, o modelo em espiral, que revelou ser o mais adequado para o projeto. Serão, também, descritas as fases do projeto e as ferramentas usadas para o desenvolver. Em seguida, sendo o armazenamento um aspeto muito importante neste projeto, será também abordado, neste capítulo, podendo-se, desde já, adiantar que se optou pelo `localStorage`, visto ser o único sistema suportado por todos os *browsers*. Posto isto, será apresentada a solução encontrada para se conseguir interagir com a aplicação quando não há Internet, ou seja, em modo *Offline*. Para terminar este capítulo será descrita a forma de sincronização com o servidor.

4.2 Abordagem

De um modo geral, o processo de desenvolvimento foi baseado no modelo em espiral, embora com algumas modificações. Na figura 4.1 pode-se observar um diagrama geral do processo de desenvolvimento, onde estão representadas as suas etapas principais.

No diagrama podemos observar que é um modelo de desenvolvimento sequencial, cujas etapas estão dependentes umas das outras, embora se possa retroceder para melhorar ou implementar novas funcionalidades numa etapa anterior.

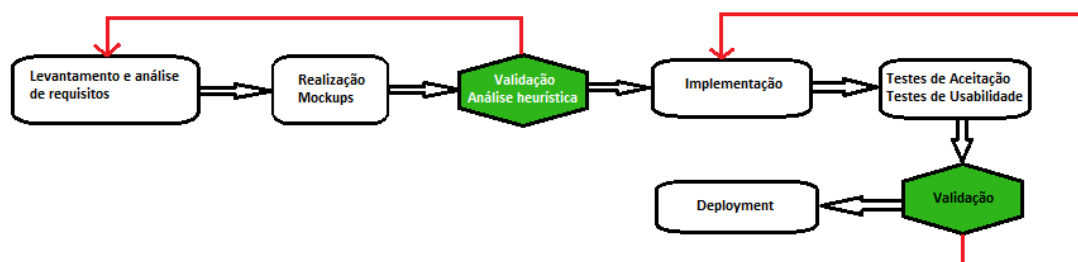


Figura 4.1: Processo de desenvolvimento da App iScriptor.

Como apontado anteriormente, o objetivo principal deste projeto é obter uma versão móvel, com determinadas características, de uma aplicação existente na empresa Viatecla, denominada Scriptor Server. Pretende-se, deste modo, fornecer aos utilizadores da plataforma Scriptor um novo meio de acesso, pesquisa e edição dos seus conteúdos. A plataforma conta com uma *interface* web desenvolvida para computadores pessoais, sendo que esta deixa de ser funcional aquando da sua utilização por um dispositivo móvel (Smartphone, Tablet).

O projeto desenvolveu-se em três grandes fases:

A primeira fase consistiu num estudo aprofundado de todas as abordagens e tecnologias existentes na conceção de aplicações móveis. Neste estudo foram apontados os pontos fortes e fracos de cada abordagem e tecnologia. Esta fase serviu, ainda, para determinar qual a melhor abordagem, que servisse os interesses e características propostos (*Online/Offline*; Compatibilidade; Baixo custo).

Na segunda fase foi feita uma análise de requisitos e criado um protótipo, para que fosse realizado um Teste de Usabilidade, no sentido de se prosseguir com o projeto e/ou alterar o que fosse necessário. O protótipo consistiu num conjunto de mockups desenhados, para uma melhor conceção do projeto a implementar.

Por fim, a terceira fase compreendeu toda a implementação do projeto. Esta fase incluiu o estudo de alguns processos mais complexos, que serão alvo de análise e descrição ao longo deste capítulo.

Importa referir que o processo de desenvolvimento da App iScriptor não foi totalmente cumprido, ficando a fase de validação, por cumprir, visto o tempo definido para a implementação desta aplicação ter sido ultrapassado em alguns meses. Decidiu-se, então, que seria concluída a sua implementação e ficaria por concluir a fase de validação e *deployment*, ficando estas fases a cargo da equipa de desenvolvimento do Scriptor Server da Viatecla.

4.3 Projeto iScriptor

Antes de se proceder à implementação da nova *interface* do iScriptor foi necessário escolher as ferramentas de implementação.

O IDE escolhido foi o NetBeans, uma ferramenta *open source* com suporte à construção de aplicações Web em HTML5. Além de incluir suporte para as tecnologias Web a serem usadas no projeto como HTML5, CSS3 e JavaScript, o NetBeans tem a facilidade de incorporar as bibliotecas de JavaScript pretendidas, tais como o jQuery, base da framework jQuery Mobile. O NetBeans oferece assistência de código, *debugging*, informação sobre o suporte dos *browsers* para cada método/função, e um servidor web e HTTP *built-in* para pré-visualização e testes do projeto.

A criação do iScriptor levou a grandes alterações, nomeadamente no Middleware, por parte da equipa técnica do Scriptor Server. Os web services já implementados foram alvo de alterações, tentando-se restringir a informação mínima necessária, otimizando ao máximo as mensagens entre o servidor e a nova *interface*. Foram, também, criados novos web services para fazer face às funcionalidade pretendidas.

4.4 Interface Responsive

A criação de uma *interface* capaz de alcançar os objetivos propostos foi um dos pontos principais e, portanto, muito ponderado, ficando, à partida, decidido que a *interface* teria disposições e funcionalidades diferentes consoante o tamanho de ecrã. Existiria uma determinada disposição para dispositivos com uma largura de ecrã superior a 500px e uma disposição diferente e ausência de algumas funcionalidades para dispositivos com uma largura de ecrã inferior a 500px, como é o caso dos Smartphones. Assim, para se criar uma *interface* capaz de cumprir os objetivos estabelecidos é necessário ter em conta diversas variáveis:

- Tamanho do monitor/tipo de dispositivo
- Forma como o utilizador interage com a *interface* (toque ou rato);
- Suporte em todos os *browsers*

Aplicar o conceito Responsive Web Design pareceu o mais adequado para fazer face às diversas variáveis. O uso das suas técnicas como a utilização de grids flexíveis e media queries facilitou o processo de adaptação da *interface* a qualquer monitor.

A utilização do jQuery Mobile facilitou o processo de criação de uma *interface responsive*, uma vez que este possibilita a criação de aplicações web, orientadas aos dispositivos móveis, através de componentes flexíveis. Uma página definida com o jQuery Mobile é constituída por três componentes: header, content e footer.

- O header - barra de navegação situada no topo da página, com possibilidade de ficar fixa mesmo quando o utilizador faz *scroll*. Pode conter botões, imagens e texto. No caso do iScriptor, o header tem o logótipo identificador do produto e três botões, sendo eles: Full Screen - que dá a possibilidade ao utilizador de ver uma das áreas em ecrã inteiro; Alerta Edição - um botão com alerta dos número de conteúdos, cujo servidor não efetuou as alterações e/ou que o utilizador submeteu por qualquer motivo; Logout - termina a sessão.
- O content - tal como o nome indica, é onde se localiza o conteúdo da página. No iScriptor esse content está dividido em duas áreas: uma com uma abrangência de 15% para área de Barra de Navegação ou Barra de canais e os restantes 85% para a lista e formulário dos conteúdos. Esta percentagem é sempre igual até um determinado tamanho de ecrã, pois no caso de monitores reduzidos como os dos Smartphones, a disposição das áreas é diferente e o funcionamento também, para tal o uso de media queries foi fundamental.
- O footer - assemelha-se ao header, localizando-se, porém, no fundo da página. Optou-se por um footer intermitente, isto é, o footer só aparece em determinados casos, visto que ocupa espaço e este é um bem necessário em monitores de pequenas dimensões.

Esta *framework* oferece também *widgets* que ajudam a formatar a informação, em listas, formulários, colunas ou barras de navegação. Alguns desses *widgets* foram uma ajuda bastante valiosa, uma vez que a sua adaptação na *interface* vem auxiliar e facilitar a seleção dos conteúdos por parte do utilizador.

As media queries foram também muito importantes, pois são as responsáveis pela definição da disposição da *interface* para Tablets e Smartphones, bem como pelo acesso e restrição a determinadas funcionalidades, consoante o tipo de dispositivo. Como foi referido na análise de requisitos, a utilização de certas funcionalidades quando se usa um dispositivo móvel como é o caso dos Smartphones, deixam de ter sentido, dado que o tamanho do monitor dificulta a execução da tarefa.

A utilização de Flexboxes foi também testada, mas o seu desempenho mostrou-se algo deficiente, na medida em que o carregamento do código fonte da *interface* era lento, além de ser necessário usar diferentes sintaxes consoante o tipo de *browser*.

Project Name	Status	Feature ID	Developer	Tester / QA Lead
BackOffice Aplications - Envio Emails	Released	Envio Emails	Pedro Lameiras	Pedro Lameiras
BackOffice Aplications - Wiki	Released	Wiki	Pedro Lameiras	Pedro Lameiras
Scriptor Server - Build Listener	Draft	Build Listener	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server - Multiple file upload	Draft	Multiple file upload	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server - Object Generator (Entity Framework)	Draft	Object Generator (Entity Framework)	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server - Resources Add-in	Draft	Resources Add-in	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server - Scriptor Agent	Draft	Scriptor Agent	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server - Setup	Draft	Setup	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server - Template Generator	Draft	Template Generator	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server - Video upload	Draft	Video upload	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server - Document Generator	Draft	Document Generator	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server - Excel Add-in	Draft	Excel Add-in	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server - Scriptor Licenses Keys [EN]	Developed	Scriptor Licenses Keys	João Vieira	
Scriptor Server - Scriptor Licenses Keys [PT]	Developed	Scriptor Licenses Keys	João Vieira	
Scriptor Server - Template VS Add-in	Draft	Template VS Add-in	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server - Word Add-in	Draft	Word Add-in	Ricardo Rodrigues	Ricardo Rodrigues
Scriptor Server BackOffice Aplications	Released	Aplications	Pedro Lameiras	Pedro Lameiras
Scriptor Server BackOffice Social Networks	Release	1.0		

Figura 4.2: iScriptor vista Tablet.

Name	Tipo	Função	Team
Entidade Externa	VT Extern		
João Carmona	VT Intern		
Pedro Melgueira	VT Extern		
João Pires	VT Extern		
Rafael Paulo	VT Intern		
Emanuel Costa	VT Intern		
José Júnior	VT Intern		
José Miguel Pereira	VT Extern		
Daniel Freitas	VT Intern		
Paulo Primavera	VT Intern		Álvaro Ribeiro
Hugo Patito	VT Extern		Álvaro Ribeiro
João Fernandes	VT Extern		
Adboulaye Gueye	VT Extern		Jorge Araujo
Duarte Peres	VT Intern		Ricardo Raminhos
Teresa Santos	VT Intern		Elsa Casimiro
Nelson João	VT Intern		Daniel Freitas

Figura 4.3: Lista de Conteúdos widget para seleccionar colunas.

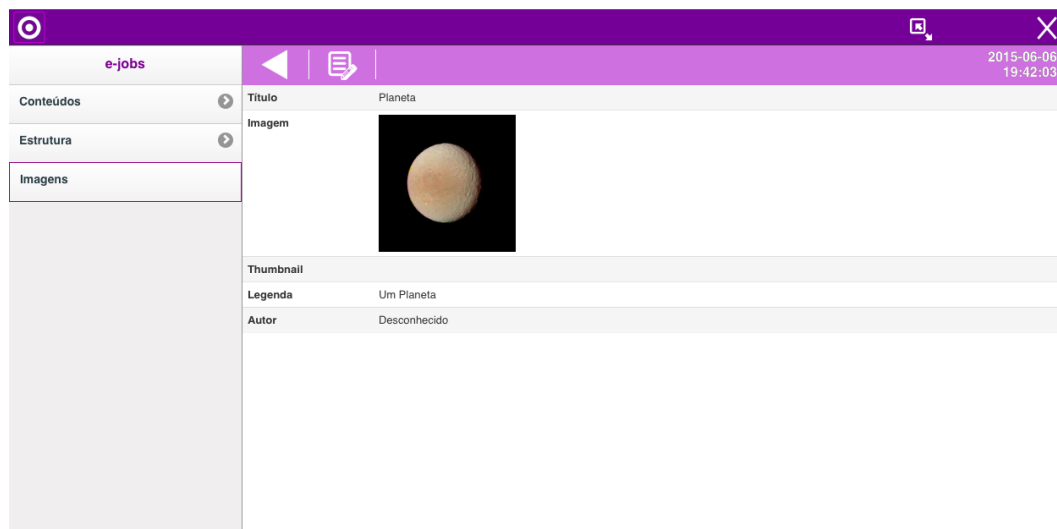


Figura 4.4: vista de um Conteúdo.

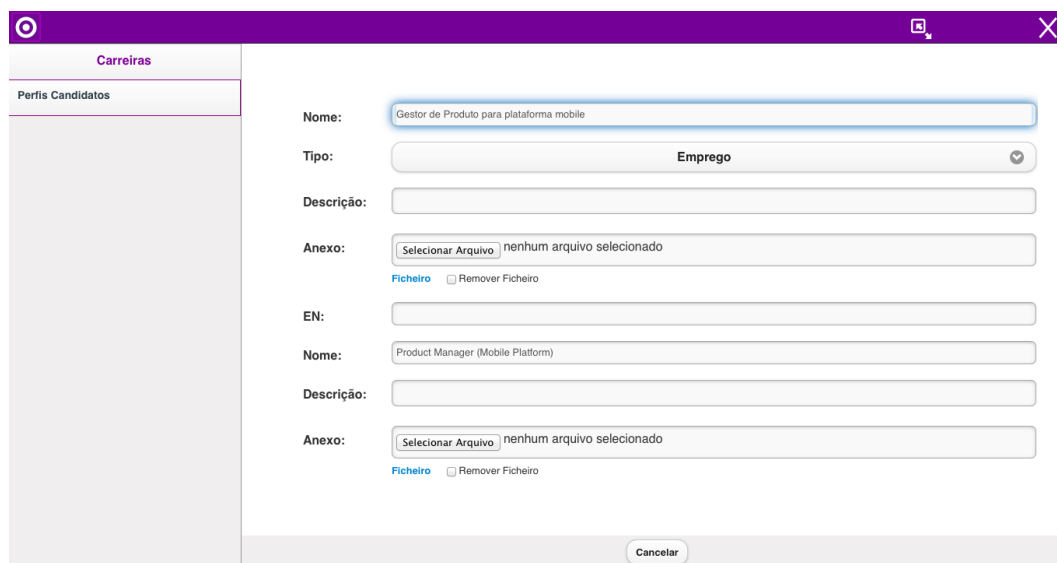


Figura 4.5: Editar formulário do Conteúdo.

Scriptor Server Team					
Novo		BOs por Máquina		Pesquisar	
2015-06-06 20:15:16					
Instalações de Backoffices	Acetrax				
Gerador de Documentos	Url	Cliente	Tipo	Versão de BO	Observações
Scriptor Server Network	Apenas rede interna	Acetrax	Bridge4Media	Scriptor Server Vistas	
ScriptorAppStore	Ariel02				
	Ariels NLB				
	BPI				
	CDM				
	Cosmos				
	DGOTDU				
	EP				
	Évora				
	Url	Cliente	Tipo	Versão de BO	Observações
	http://tv.evoradistritodigital.pt/bo	ADRAL	FutureBox	AlaMoana	
	FCG				
	GMS				
	McKinsey				
	previous	1	next		

Figura 4.6: Tipo de Vista da lista de Conteúdos, agrupada por uma determinada coluna.

Viatecla Nova Imagem		
A Viatecla		
Aplicações		
Artigos		
Carreiras		
Emprego		
Eventos		
Homepage		
Inovação		
Mercados		
Microsites		

Pesquisar		
Title	Data	Número
news2	12.03.2012	2
news3	10.04.2012	3
news3teste		
news4	15.05.2012	4
news7	07.08.2012	7
news8	08.09.2012	8
news8teste		

Figura 4.7: iScriptor vista Smartphone.

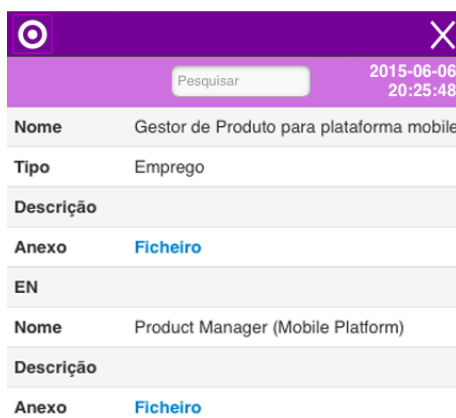


Figura 4.8: Conteúdo visto através de um smartphone, sem possibilidade de editar.

4.5 Armazenamento

A decisão pelo `localStorage` facilitou, de certa forma, o trabalho, pois uma vez que era o único sistema suportado por todos os *browsers*, fez com que a equipa de trabalho se focasse apenas nele, não sendo necessário desenvolver dois tipos de sistema de Armazenamento.

O `localStorage` é uma base de dados *key-value* (chave-valor), quer isto dizer que precisa de um identificador *key* e de um valor *String*. Assim, sempre que o utilizador tiver Internet, a aplicação fará uma comunicação ao servidor através de um pedido HTTP URL, o servidor ao receber um determinado tipo de pedido dará uma resposta (objeto JSON). Consoante o tipo de pedido, assim é guardado ou não.

Se for para guardar é necessário transformar o objeto JSON num `String`, em JavaScript. Essa operação é bastante facilitada se for usado o método `JSON.stringify` (objetoJSON), que transforma o objeto JSON numa `String`. Antes de introduzir a informação no `localStorage` é necessário verificar a quota de limite do mesmo, caso seja atingida, o utilizador é alertado e alguns registos serão apagados, criando-se, assim, espaço para a nova informação. Existindo espaço a inserção da `String` é realizada usando o método `localStorage.setItem` (`String`).

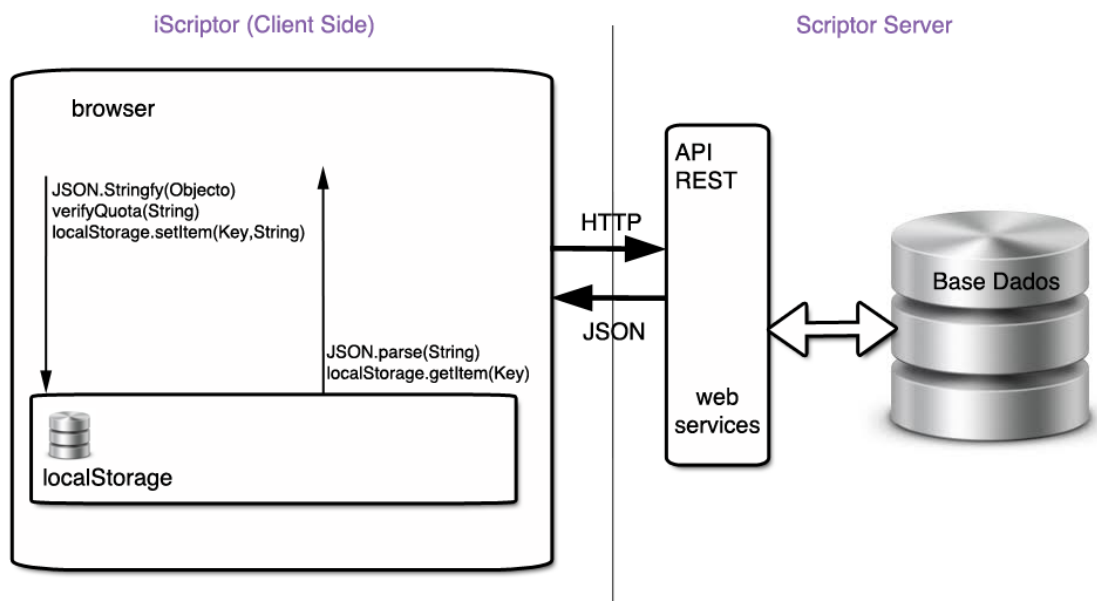


Figura 4.9: Sistema de Armazenamento do iScriptor.

Sempre que é necessário aceder à informação é necessário saber qual é o identificador, que é dado através do nome da div criada em HTML, JSON.Parse (String).

4.6 Modo *Offline*

Um dos objetivos propostos seria a possibilidade de o utilizador interagir com a aplicação tendo ou não ligação à Internet, conseguindo aceder e até mesmo editar os seus conteúdos estando no estado *Offline*. Para cumprir este objetivo foi necessário entender como ultrapassar algumas adversidades:

- Como disponibilizar a aplicação em modo *Offline*?
- Como camuflar a aplicação dando a perceção de ser uma aplicação nativa?
- Como pode o utilizador aceder e editar um determinado conteúdo?

A funcionalidade Application Cache permite ao utilizador continuar a usar a aplicação iScriptor sem necessidade de internet conseguindo-se usar a aplicação em modo *Offline*, usando uma versão mais limitada, isto é o utilizador poderá aceder aos conteúdos guardados na Application Cache.

A possibilidade de conseguir aceder a conteúdos pelo *browser* sem a necessidade de internet é graças à Application Cache. Esta, para além de permitir armazenar informação e

trabalhar em modo *Offline*, também garante uma maior rapidez no carregamento de recursos, uma vez que é possível armazenar os recursos e poder acedê-los sem a necessidade de comunicar com o servidor, reduzindo, assim, a carga do servidor.

Quanto aos ficheiros que irão ficar disponíveis, quando não existe ligação à Internet, têm de ser especificados num ficheiro manifest que deve encontrar-se do lado do servidor, contendo os nomes dos recursos que o *browser* deverá guardar localmente e está referenciado com o atributo manifest na tag HTML, como é possível verificar através da Figura 4.10.

O ficheiro manifest está dividido em três partes: Cache Manifest, Network e Fallback.

A primeira, Cache Manifest, deve estar obrigatoriamente no ficheiro e os nomes que o seguem consistem nos ficheiros que ficarão disponíveis após serem descarregados na primeira vez que o utilizador interage com a aplicação e fica sem ligação à Internet. Na segunda parte do ficheiro deverá vir o Network, que contém os nomes dos ficheiros que nunca ficarão em cache e que será sempre necessário fazer um pedido ao servidor. Por fim, virá o Fallback, que representa a página para onde o utilizador é redirecionado caso um recurso não se encontre disponível, contendo, então, dois parâmetros: o primeiro permite informar o caminho que não está disponível e o segundo para onde é redirecionado.

A Figura 4.10 mostra a estruturação e especificação dos ficheiros que devem estar em cache, conforme descrito anteriormente. Depois do Cache Manifest, encontra-se representada uma linha iniciada com #, seguida da data e uma versão, sendo esta uma breve abordagem para posteriormente o *browser* saber se o *manifest* foi ou não atualizado.


```
CACHE MANIFEST
#version 1.1 25-6-2013

index.html
js/main.js
js/requests.js
js/localStorage.js
js/channel.js
js/content.js
js/forms.js
js/contentList.js
js/pagination.js
js/jquery-1.8.1.min.js
js/jquery.mobile-1.3.1.min.js
js/bootstrap.min.js
js/jquery.mobile.iscrollview.js
js/iscroll.js

img/back_left.png
img/close_white.png
img/edit_button.png
img/iScriptor_logo.png
img/refresh.png
img/scriptor_icon_favicon.ico
img/warning.png

css/1.3.0.css
css/bootstrap.min.css
css/layout.css
css/images/ajax-loader.gif
css/images/icons-18-black.png
css/images/icons-18-white.png
css/images/icons-36-black.png
css/images/icons-36-white.png

NETWORK:
*
```

Figura 4.10: Ficheiro manifest que armazena os ficheiros indicados em cache.

Em relação à informação que será apresentada ao utilizador, assim como quaisquer alterações feitas pelo mesmo durante o período *Offline*, é da responsabilidade do implementador decidir e implementar soluções para preservar os dados e sincronizá-los com o servidor nos períodos em que existe ligação à Internet.

Para disponibilizar a aplicação iScriptor em modo *Offline* foi necessário guardar os ficheiros mínimos, de modo a aceder à página principal da aplicação. No caso do iScriptor optou-se por guardar todos os ficheiros que possibilitam a renderização da *interface* web da aplicação, ou seja todos os ficheiros responsáveis pela estrutura, marcação e estilo da *interface*. Os conteúdos, que necessitam de atualização e que são editados ficam a cargo do localStorage, sendo sincronizados com o servidor sempre que a aplicação volte novamente a ter ligação à Internet

Alguns *browsers*, nomeadamente os mais utilizados nos dispositivos móveis, Safari e Chrome, possibilitam a criação de um ícone que ficará disponível na tela principal do dispositivo juntamente com outros ícones de aplicações nativas. Ao selecionar o ícone, o *browser* abrirá a aplicação em *full screen*, carregando os ficheiros guardados na Application Cache, escondendo a barra de URL e os botões, dando ao utilizador a ilusão de que está a usar

uma aplicação nativa.

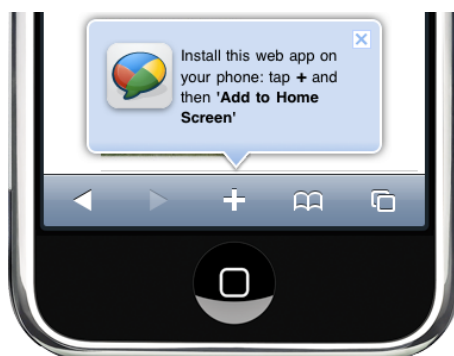


Figura 4.11: Gravar web App nos dispositivos mobile.

4.7 Sincronização com o Servidor

Sempre que é criado um novo conteúdo é necessário interagir com o servidor. Esta comunicação é feita através de um pedido ao mesmo. O Pedido é feito e é enviado um objeto JSON, esse objeto é criado pela aplicação iScripator com o sys-id do Canal onde é pretendida a criação do novo conteúdo. Cada objeto JSON é composto com o sys-id do canal com a informação correspondente de cada campo. Sempre que não existe ligação à Internet (*Offline*) o novo conteúdo é guardado no `localStorage` para que, assim que aplicação volte a interagir com o servidor, seja criado automaticamente o pedido de novo conteúdo ao servidor. Ao realizar esta operação com sucesso o registo criado no `localStorage` é apagado e o canal é atualizado no `localStorage`.

No caso da edição de um determinado conteúdo o processo de sincronização com o servidor é diferente e mais complexo, existindo 4 tipos de cenários:

- Existe Internet e consegue-se fazer a edição do conteúdo.
- Existe Internet e o servidor recusa a alteração e guarda a edição do utilizador.
- Não existe Internet, os conteúdos alterados são guardados e assim que existir Internet sincroniza com o servidor e tem sucesso.
- Não existe Internet, os conteúdos alterados são guardados e assim que existir Internet sincroniza com o servidor e não tem sucesso.

Sempre que é editado um determinado conteúdo é necessário interagir com o servidor. Esta comunicação é feita através de dois pedidos ao servidor. O primeiro pedido pergunta ao servidor se um determinado conteúdo sofreu alguma atualização após uma determinada data e hora. O segundo pedido é um objeto JSON, com o campo ou os campos que o utilizador alterou. Este segundo pedido está dependente da resposta do primeiro pedido. Consoante o tipo de resposta, o segundo é ou não executado e também no tipo de resposta

reside a diferença do primeiro e segundo cenários na edição e sincronização de conteúdos com o servidor. Caso o servidor diga que determinado conteúdo não sofreu alterações executa o pedido de alteração. Caso o conteúdo tenha sofrido alteração o pedido com as alterações não é executado e a edição é guardada no `localStorage`, sendo o utilizador avisado. A alteração é guardada com o intuito do utilizador, através do botão de falha na sincronização, consiga aceder ao conteúdo e perceba quais os campos que foram alterados pelo outro utilizador, conseguindo, assim, comparar o conteúdo atual com a alteração que no passado tentou submeter.

Nos casos em que a edição é feita sem ligação à Internet (*Offline*) o processo é idêntico ao segundo cenário. A edição é guardada no `localStorage`, assim que seja possível interagir com o servidor, isto é a aplicação dispõem de ligação à Internet (*Online*), executa um primeiro pedido a perguntar se existiu alguma alteração após a data de armazenamento do conteúdo. Caso não existam alterações, efetua um segundo pedido com as alterações pretendidas e o conteúdo é atualizado no `localStorage`. Caso contrário, guarda as alterações submetidas pelo utilizador no `localStorage` e avisa-o de que não foi possível executar a alteração pretendida. O utilizador, através do botão de falha de sincronização, consegue aceder ao conteúdo atualizado e por baixo de cada campo em que efetuou alterações existe um “Comentário” com as alterações que o utilizador efetuou e que não foram submetidas.

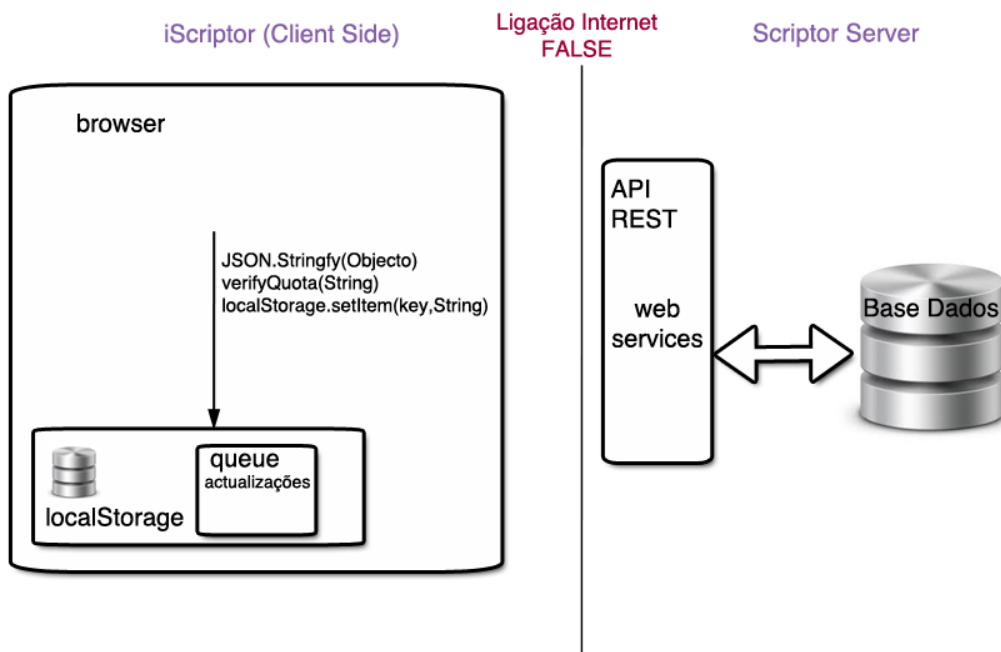


Figura 4.12: Sincronização com o Servidor sem ligação à Internet.

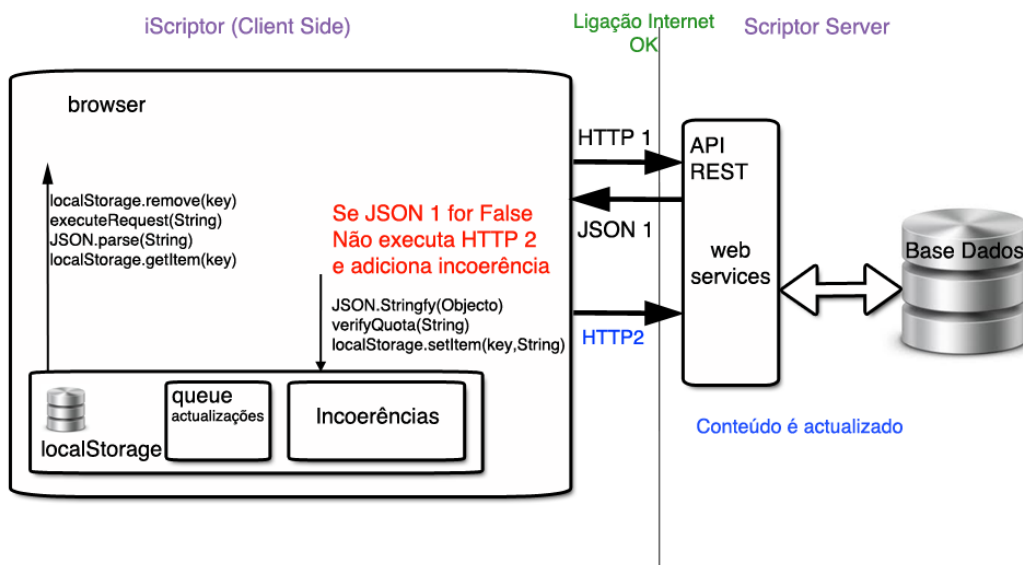


Figura 4.13: Sincronização com o Servidor com ligação à Internet.

Capítulo 5

Conclusões

O desenvolvimento de aplicações para dispositivos móveis está repleto de desafios, nomeadamente qual o melhor caminho a seguir e qual a melhor estratégia a adotar. Como foi referido, o tamanho do ecrã, a compatibilidade e os gastos no desenvolvimento destas aplicações são os maiores desafios no seu desenvolvimento. O menor tamanho do monitor obriga a uma escolha minuciosa dos conteúdos, bem como à criação de uma *interface* interativa com grande usabilidade e acessibilidade, aliada a uma fluidez e desempenho na interação utilizador/dispositivo.

Na altura em que este Projeto foi desenvolvido o HTML5 ainda se encontrava em fase de desenvolvimento e mesmo assim foi capaz de cumprir, com sucesso, os objetivos definidos. O HTML5, neste momento, encontra-se já em fase de *Recommendation*, com as suas APIs mais maduras, robustas e definidas, tendo tudo para se tornar uma séria ameaça às abordagens Nativa e Híbridas, visto que o custo de desenvolvimento e suporte aliado à sua compatibilidade com todos os *browsers* são as suas maiores armas. Com o cumprimento dos objetivos definidos é demonstrado que o HTML5 é capaz de ombrear com as restantes abordagens, sendo necessário adaptar as várias estratégias e arquiteturas possíveis aos objetivos pretendidos. De referir, ainda, que a maioria dos novos Smartphones lançados no mercado contém poderosos motores de processamento, facilitando o uso do HTML5.

Não existem respostas exatas ou estratégias infalíveis sobre a abordagem a adotar, essa resposta está dependente de muitas variáveis, entre as quais: o que é que se pretende alcançar; quais os custos suportados; os objetivos; e a que utilizadores se destinam. Existem certezas quanto à evolução dos dispositivos móveis, às várias atualizações e à entrada de novos SOs móveis, sabendo-se, também, que o mercado móvel terá um crescimento exponencial, sendo a mudança a única constante. O tempo para a chegada de qualquer aplicação móvel é, então, importante e decisivo, e quanto mais rápido melhor. Considerando as inúmeras

variáveis no desenvolvimento de Apps é prudente pensar que a melhor estratégia passará por um investimento, inicialmente, mínimo, reduzindo os custos no desenvolvimento e tornando-a o mais multi-plataforma possível, de modo a conseguir abranger o maior número de utilizadores e tentando, assim, consolidar um número mínimo de utilizadores. Assim sendo, a evolução de qualquer App será progressiva e sustentável.

5.1 Trabalho Futuro

Com a criação da App iScriptor a Plataforma Scriptor Server fica com duas *interfaces* para os seus clientes, uma *interface* para PC, ou seja para ecrãs de grandes dimensões e uma aplicação (iScriptor) para dispositivos móveis. No futuro seria importante existir apenas uma *interface* com todas as Funcionalidades.

A sincronização da App iScriptor com o servidor, nomeadamente na edição de conteúdos deveria ser melhorada, através de *logs*. Seria, assim, possível verificar qual o campo que foi alterado de um determinado conteúdo, ao invés daquele que foi implementado. Caso um determinado Conteúdo seja alterado, apesar do campo editado não coincidir com a última atualização do Conteúdo, não é possível efetuar a sua edição e atualização no servidor.

Outra das melhorias poderia passar pela possibilidade dos utilizadores escolherem quais os conteúdos que querem armazenar (Gestão da sua BD no seu dispositivo) ou, então, existir uma espécie de gestão do localStorage pela App iScriptor, atualizando a data do registo sempre que este seja solicitado para leitura ou para edição, conseguindo-se, assim, remover os registos mais antigos.

Seria, também, importante melhorar a resposta da API Rest aos pedidos do utilizador, nomeadamente no que diz respeito aos pedidos de Vistas, de forma a melhorar a performance das mesmas.

Referências bibliográficas

- [1] Mark Boulton. Five simple steps to designing grid systems - preface. Website, 2005. <http://www.markboulton.co.uk/journal/five-simple-steps-to-designing-grid-systems-preface>, pag. vista em: 15/04/2015.
- [2] Ana Amélia Amorim Carvalho. Testes de usabilidade: exigência supérflua ou necessidade? Book, 2006.
- [3] NNG Nielsen Norman Group. Tim berners lee. Website, 2012. <http://www.nngroup.com/reports/>, pag. vista em: 22/03/2014.
- [4] Adaptive Images. Adaptive images. Website. <http://adaptive-images.com>, pag. vista em: 15/04/2015.
- [5] Patonnier Jeremie. Using the application cache. Website, 2014. https://developer.mozilla.org/pt-PT/docs/HTML/Using_the_application_cache, pag. vista em: 10/03/2014.
- [6] JSON. Introducing json. Website, 2012. <http://www.json.org>, pag. vista em: 10/03/2014.
- [7] Ethan Marcotte. Responsive web design. Website, 2010. <http://alistapart.com/article/responsive-web-design>, pag. vista em: 15/04/2015.
- [8] J. Nielsen. Usability engineering. Book, 1993.
- [9] J. Nielsen. Multimedia and hypertext: the internet and beyond. Book, 1995.
- [10] Media Queries. Media queries. Website. <http://www.w3.org/TR/css3-mediaqueries/>, pag. vista em: 20/04/2015.
- [11] Florian Rivoal. Media queries. Book, 2012.
- [12] J. Rubin. Handbook of usability testing. Book, 1994.

- [13] B. Shackell. Ergonomics in design usability. Book, 1986.
- [14] C. Smith. Telematics application for education and training: Usability guide. Book, 1996.
- [15] M. Tessmer. Planning and conducting formative evaluations. Book, 1993.
- [16] W3C. W3c 5.6 offline web applications. Website, 2011. <http://www.w3.org/TR/2011/WD-html5-20110525/offline.html>, pag. vista em: 10/03/2014.
- [17] W3C. Web storage. Website, 2013. <http://www.w3.org/TR/webstorage/>, pag. vista em: 10/03/2014.
- [18] wikipedia. 10 usability heuristics for user interface design. Website, 2012. http://pt.wikipedia.org/wiki/Tim_Berners-Lee, pag. vista em: 04/05/2015.