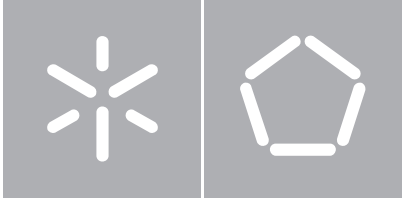
Universidade do Minho
Escola de Engenharia

Samuel da Costa Rodrigues

Estudo e Implementação de Interfaces Web em HTML5

Janeiro de 2014



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Samuel da Costa Rodrigues

Estudo e Implementação de Interfaces Web em HTML5

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de

João Miguel Lobo Fernandes

Janeiro de 2014

Agradecimentos

O trabalho aqui realizado teve a colaboração de diferentes pessoas. Desta forma cumpre-me endereçar um agradecimento a quem me ajudou, nomeadamente:

- Professor João Miguel Fernandes e Eng. Telma Mota que me lideraram, motivaram e dedicaram o seu tempo em prol deste trabalho;
- Henrique Martins, Ricardo Macedo e Vasco Amaral meus amigos e colegas de trabalho sempre disponíveis para darem contributos nas alturas em que necessitei;

Por último reservo um especial agradecimento aos meus pais e irmã, que ao longo de todo o meu percurso humano e académico foram suporte indispensável nos bons e maus momentos.

Resumo

A web está em constante evolução. A evolução tecnológica fez com que as aplicações web estivessem cada vez mais presentes no mercado e surgissem novos padrões arquiteturais, novos dispositivos e novas experiências de utilização das aplicações. Tudo isto com o propósito de satisfazer as exigências que o mercado e os utilizadores finais impõem.

Toda esta evolução potenciou o aparecimento de uma nova versão do HTML, o HTML5, a qual está em constante progresso. Não se pode esperar que o HTML5 esteja totalmente concluído para implementar interfaces com esta tecnologia, podendo assim ser considerado um *living standard*, porque já existem funcionalidades suficientemente maduras e que inclusivamente já são utilizadas em produtos no mercado. Devido ao facto do HTML ser uma linguagem de marcação, torna-se indispensável associar o HTML ao JavaScript, de modo a poder disponibilizar dinamismo e funcionalidade às páginas web. Existiu por isso, tal como o HTML, uma grande evolução relativa à linguagem, mais propriamente às frameworks JavaScript de desenvolvimento web existentes.

Devido ao progresso destas tecnologias, esta dissertação pretende analisar as várias funcionalidades disponibilizadas pela tecnologia HTML5 e as frameworks JavaScript existentes no mercado. Para tal, estas tecnologias serão usadas, de forma experimental, num projeto de desenvolvimento de software na empresa onde decorre este trabalho. Os resultados focam-se em identificar as funcionalidades HTML5 implementadas e apresentar uma comparação entre as frameworks JavaScript em estudo, segundo um conjunto específico de critérios. É do interesse da empresa onde este trabalho foi realizado aplicar as funcionalidades HTML5 e frameworks JavaScript, de modo a identificar as vantagens e desvantagens de cada tecnologia para, num futuro próximo, as aplicar em aplicações web.

O objetivo deste trabalho é assim medir quantitativa e qualitativamente, segundo os critérios considerados para análise, o impacto da introdução do HTML5 e das frameworks JavaScript em produtos de software, caso estas substituam as que estão hoje em dia em utilização.

Abstract

The web is continuously evolving. This evolution increased the presence of web applications in the market, created new architectural patterns, new devices and new ways of user experience. All of this with the goal to meet the requirements that the market and the final users require.

Those developments have caused the creation of a new version of HTML, HTML5, which is in constant progress. No one can expect that HTML5 is a finished specification, to start creating interfaces only with this technology, but it can be considered a standard living because there are features mature enough to be developed in web applications available for the market. Due to the fact HTML is a markup language, it is essential to link HTML to JavaScript, in order to provide dynamism and functionality to web pages. For this reason there was a major evolution on the language, more specifically in JavaScript frameworks existing for web development.

The progress of these technologies has led this dissertation to analyze the multiple features provided by HTML5 technology and the existing JavaScript frameworks. For such, these technologies will be used, experimentally, in a software development project in the company where this work follows. The results are focused on identifying the features made with HTML5 and display a comparison of JavaScript frameworks studied, according to a specific set of criteria. It is the company's interest where this work was carried out, apply the HTML5 features and JavaScript frameworks, in order to identify the pros and cons of each technology, to in the near future, develop them in web applications.

The goal of this work is to measure quantitatively and qualitatively, according to the criteria considered for analysis, the impact of the introduction of HTML5 and JavaScript frameworks in web software products, if they replace those in use today.

Índice

Índice de Figuras

Índice de Tabelas

1 Introdução

1.1 Enquadramento

Este documento técnico, designado de dissertação, foi desenvolvido no âmbito da unidade curricular Dissertação lecionada no curso de Mestrado em Engenharia Informática (MEI) da Universidade do Minho (UM).

Este módulo é o culminar de um ciclo de cinco anos (três de licenciatura e dois de mestrado), e por isso mesmo, tem como principal função a aplicação e verificação de alguns dos conhecimentos, capacidades e competências desenvolvidos ao longo da licenciatura e do mestrado. Para além da aplicação das competências adquiridas, este módulo tem também uma vertente autodidata que se revela como uma forma adequada de adquirir novos conhecimentos e de consolidar os já existentes.

1.2 Objetivos

O crescimento do uso das aplicações web, o aparecimento da nova versão do HTML e o desenvolvimento de várias frameworks JavaScript transformaram o mercado, tanto em funcionalidades como em competências técnicas necessárias para criar aplicações web. Com esta dissertação pretende-se então realizar o estado da arte do HTML5, identificando quais as funcionalidades desta tecnologia, desde os desafios existentes nos navegadores até às limitações na implementação das funcionalidades. E pretende-se apresentar as vantagens e desvantagens das várias frameworks JavaScript existentes no mercado, o que permite às equipas de desenvolvimento web reconhecer as que conseguem satisfazer as suas necessidades ou quais estão neste momento com mais perspetivas de vingar no mercado, já que este se encontra ainda confuso.

Resumindo, este trabalho tem então o objetivo de identificar quais as oportunidades existentes com a especificação da nova versão do HTML e quais as funcionalidades das frameworks JavaScript, de modo a criar uma base de conhecimento sólida para o futuro desenvolvimento de aplicações web na PT Inovação.

1.3 Motivação e desafios

Como referido anteriormente, esta dissertação tem como propósito levantar o estado da arte do HTML5, o que permite aumentar o conhecimento, tanto pessoal ou para quem consulta a dissertação, sobre esta tecnologia.

Com o desenvolvimento e evolução das tecnologias web, as novas aplicações que têm sido desenvolvidas têm maior parte do seu código do lado do navegador (código JavaScript, CSS e HTML) sendo por vezes suficiente para criar aplicações web de elevada complexidade. Com o respetivo aumento deste tipo de aplicações, que apenas recorriam a bibliotecas JavaScript, apareceu a necessidade das bibliotecas se tornarem frameworks JavaScript. Este mercado encontra-se ainda muito confuso, dada a quantidade de frameworks existentes, por isso a motivação está em analisar este mercado de modo às entidades interessadas possam ser clarificadas, evitando assim aplicar frameworks que não correspondem às necessidades das aplicações. De um ponto de vista pessoal, a dissertação permitiu aplicar uma série de técnicas para perceber se estas tecnologias são compensáveis e em que situações. Tecnicamente possibilitou aprofundar o conhecimento sobre tecnologias que hoje são indispensáveis para construir qualquer aplicação baseada na web. Este trabalho vai permitir, do ponto de vista da PT Inovação, possuir um estudo para ajudar a escolher estas tecnologias em futuros projetos de desenvolvimento de aplicações web.

1.4 Metodologia do trabalho desenvolvido

A primeira parte da dissertação tem como objetivo enquadrar o leitor na tecnologia HTML, JavaScript e tecnologias adjacentes. De seguida, e de um modo mais profundo, é descrito o HTML5 e são analisadas as novas funcionalidades, descrevendo quais as suas vantagens e desvantagens do ponto de vista, por exemplo, do programador ou até do navegador.

O objetivo da segunda parte da dissertação é de efetuar um estudo comparativo de várias frameworks JavaScript e analisar o impacto do HTML5 na implementação interfaces web. Estas tecnologias e frameworks serão incluídas num produto de software para a aplicação e comprovação do estudo efetuado. Este estudo comparativo está divido em duas fases, em que na primeira se classificam as frameworks

segundo um conjunto de critérios, e na fase seguinte, são analisadas as melhores frameworks conforme os pontos de vista existentes no desenvolvimento de aplicações web.

1.5 Organização da tese

Este documento encontra-se divido em várias partes, em que o planeamento do trabalho para a concretização da dissertação organizou-se de acordo com as seguintes partes:

Parte 1 – Contexto (capítulo 2)

Este capítulo permite contextualizar o leitor no tema da dissertação. São explicados os conceitos base que permitem entender a restante tese, são abordados temas como HTML, JavaScript, CSS, jQuery e JSON.

Parte 2 – Análise (capítulo 3)

A parte da análise contém a investigação feita de modo a efetuar um correto desenvolvimento nas fases posteriores. Tem como objetivo de descrever a tecnologia HTML5 e as frameworks JavaScript existentes no mercado.

Parte 3 – Especificação e desenvolvimento do caso de estudo (capítulo 4)

Esta fase permitiu especificar o caso a desenvolver de modo a entender as funcionalidades da aplicação. Foram também abordadas questões mais técnicas como arquitetura, design e mockups para verificar a influência das tecnologias em estudo na fase de especificação de software.

Parte 4 – Conclusões (capítulos 5, 6)

Após o desenvolvimento do caso de estudo foram obtidas conclusões relativas à aplicação do HTML5 e às frameworks JavaScript. As conclusões relativas às frameworks foram divididas duas fases, em que a primeira (capítulo 5.2.1) permitiu conhecer as frameworks e classificá-las segundo um conjunto de critérios, e numa segunda fase (capítulo 5.2.2) que permitiu analisar as frameworks segundo os *stakeholders* de aplicações web.

Finalmente o capítulo 6 contém as conclusões da dissertação.

2 Contexto

Este capítulo permite contextualizar o leitor no tema da dissertação. São abordados os conceitos e tecnologias principais da dissertação, nomeadamente HTML5 e JavaScript, como as tecnologias que lhes são relacionadas, como por exemplo JSON, jQuery e CSS.

2.1 HTML

2.1.1 História do HTML

A primeira versão do HTML nasceu no ano de 1989 quando o físico Tim Berners-Lee, que trabalhava no CERN, propôs um protocolo para que os investigadores pudessem trocar documentos científicos. No ano seguinte, Berners-Lee especificou o HTML e desenvolveu um navegador para conseguir interpretar a linguagem e, em 1991, a primeira versão do HTML era publicada. Seguidamente foram publicadas várias novas versões do HTML, sendo a primeira oficial o HTML 2.0 publicada no ano de 1994. Também em 1994, Berners-Lee cria a *World Wide Web Consortium* (W3C) de que é ainda o diretor.

Na massificação do HTML, surgiu uma "guerra entre navegadores", nomeadamente entre o Netscape e o Internet Explorer, que viriam a marcar a fase inicial deste novo mercado. A próxima versão do HTML a ser especificada seria a 3.2 no ano de 1996, mas que não chegou a marcar diferença na história do HTML. A versão 4.01 foi publicada em 1999, sendo ainda a versão mais utilizada nas páginas web nos dias de hoje. No que respeita a navegadores, no início da década de 2000, o Internet Explorer dominou o mercado, enquanto que o Netscape entrou em queda acentuada. Entretanto, o W3C lançou a primeira versão do XHTML, baseado nas ideias do XML (no ano de 1998), tendo em conta a previsão de que esta linguagem teria grande utilização no futuro. Neste contexto, a comunidade científica questionou se o W3C tinha desistido do HTML. A resposta foi a seguinte:

"Foi acordado que estender o HTML 4.0 seria complicado, enquanto que converter o HTML 4.0 para XML seria mais simples. Para ultrapassar estas restrições o HTML deve ter um novo começo, baseado num conjunto de tags XML" (W3C, 1998).

Então, em 2001, desenvolveu-se o XHTML, que adicionava poucas funcionalidades à versão em uso na altura, sendo a principal diferença que o XHTML não permitia erros de sintaxe ao contrário do HTML. Esta versão foi tão pouco utilizada que no ano de 2004 o *Web Hypertext Application Technology Working Group*

(WhatWG), grupo de entusiastas que decidiu trabalhar independentemente na nova versão do HTML, especificou fora dos procedimentos da W3C e em conjunto com outras empresas (tal como a Apple, Mozilla Foundation e Opera) o HTML5, que se resolveu os pontos-chave da versão anterior.

Com o sucesso que o WhatWG beneficiou ao implementar uma nova versão do HTML, o W3C anunciou que ambas as entidades iriam trabalhar em conjunto. Passados 4 anos, é publicada a primeira versão do HTML5 escrita por Ian Hickson. Já nesta fase os navegadores começaram a ser compatíveis com certas funcionalidades do HTML5. A partir deste momento, o HTML5 foi constantemente atualizado e as empresas começaram a tomar partidos sobre esta versão. A Amazon, Apple, YouTube ente muitas outras perceberam as potencialidades do HTML5 e investiram nesta tecnologia que acreditam ser o "futuro" das aplicações web, enquanto que outras preferiram manter as suas aplicações web com versões antigas do HTML.

2.1.2 O que é o HTML

O HTML (HyperText Markup Language) é a linguagem de marcação interpretada pelos navegadores para poder representar informação. Esta linguagem está presente em todas as páginas web que existem atualmente, e estas têm a extensão *html* ou *htm*. O HTML é composto por elementos HTML que contêm a informação a apresentar. Estes elementos são sempre compostos por uma tag de início e de fim, como podemos ver no exemplo seguinte:

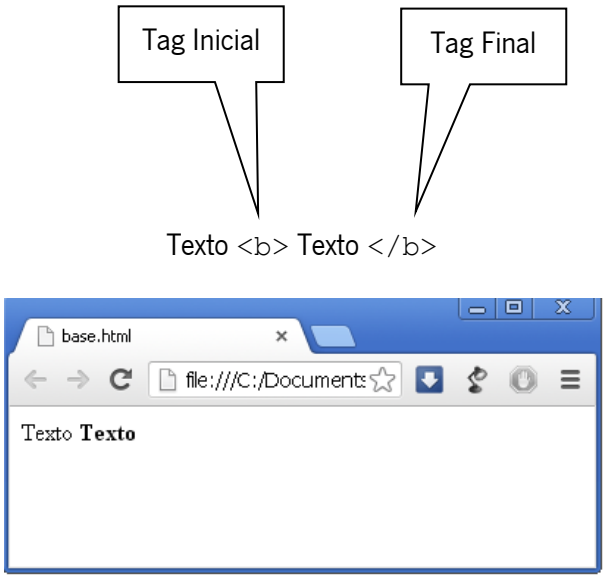


Figura 1 – Representação de código HTML no navegador

O propósito do navegador web é de interpretar o documento HTML recebido do servidor, e apresentar a informação que este contém. Como se pode constatar, o navegador não apresenta as tags HTML, mas utiliza-as para interpretar o conteúdo da página. Neste exemplo, o navegador identificou a tag `<b>` e apresenta o conteúdo a negrito.

As tags HTML foram especificadas para seguir certas regras, em que algumas são apresentadas de seguida:

- As tags são compostas por palavras-chave entre os símbolos '<' e '>', como por exemplo `<html>`.
- As tags HTML estão normalmente presentes em pares, como `<b>` e `</b>`.
- A tag de fim é escrita de maneira idêntica à tag inicial. Tem a particularidade de ter uma barra lateral antes do início do nome da tag, como `</b>`.
- As tags podem ainda conter atributos, como por exemplo a tag <a> que é composta pelo atributo *href* que contém uma hiperligação: `<a href="google.com">Google</a>`.

Hoje em dia, o HTML está associado a duas outras tecnologias, CSS e JavaScript, que permitem que as páginas sejam dinâmicas e tenham estilos associados. Algumas entidades (incluindo o W3C) até defendem que o HTML5 é a junção do HTML com essas duas tecnologias. Estas tecnologias são apresentadas neste documento e atualmente chegam a ser tão importantes como o HTML5 em si.

2.1.3 O porquê do HTML5

Hoje em dia, muitas das aplicações existentes situam-se na *cloud*, isto é, os recursos e as infraestruturas situam-se em servidores partilhados e interligados através da internet. Isto permite que, a partir de qualquer terminal e em qualquer lugar, o utilizador tenha acesso à informação, aos arquivos e aos programas no momento do acesso à *cloud*. Sendo assim, o mercado envolvente preocupou-se em criar aplicações cada vez mais ricas em vários aspetos (conteúdo, design, funcionalidades, etc.) e para acompanhar essa vontade, o mercado criou várias alternativas ao HTML, como por exemplo o Flash. A versão 5 do HTML permitiu ao mundo Web tornar-se num ambiente propício para desenvolver aplicações

de software, por permitir criar páginas altamente dinâmicas com a ajuda do JavaScript, e por ter retirado as dependências a extensões de terceiros.

Pode por isso constatar-se que, na web moderna, a ideia do que era uma aplicação e o que são páginas foi alterada. As aplicações podem ser acedidas a partir de diferentes dispositivos, o espaço em disco deixa de ser necessário para instalar aplicações, as atualizações de programas são transparentes ao utilizador, entre outras vantagens. Assim, a web passou a ser um das plataformas mais usadas para disponibilizar programas informáticos aos utilizadores. Alguns dos objetivos definidos pelo WhatWG (Hickson, 2013), para criar a especificação do HTML5 foram:

- novas funcionalidades baseadas no HTML, CSS, JavaScript e DOM;
- redução da necessidade de uso de extensões, como por exemplo o Flash;
- melhor tratamento de erros;
- mais *tags* e menos *scripting*;
- independência relativamente ao dispositivo.

2.2 CSS

O Cascading Style Sheets (CSS) é utilizado para especificar a aparência e a formatação de um documento HTML. O seu principal benefício é promover a separação entre a formatação e o conteúdo de um documento. No enquadramento do HTML5 e do JavaScript o CSS não é um elemento indispensável, isto é, podem ser implementadas funcionalidades HTML5 e JavaScript sem recorrer ao CSS. No entanto, o CSS permite apresentar estas funcionalidades de uma maneira mais agradável ao utilizador. Adicionalmente, facilita a mudança do estilo de apresentação e reduz o tamanho dos ficheiros HTML, já que os elementos relativos à apresentação são removidos.

2.3 JavaScript

Nos tempos iniciais da internet, o acesso à web era efetuado pela consulta de páginas HTML simples. Caso o utilizador desejasse consultar outro conteúdo ou até ter algum tipo de interação com o próprio, era necessário efetuar um novo pedido ao servidor, que resultava numa nova página HTML. Dominava então uma web totalmente estática. Existindo a necessidade de interação do utilizador com a página HTML, nasceu a tecnologia JavaScript. A primeira versão do JavaScript, inicialmente denominada de *LiveScript* foi

criada em maio de 1995 por Brendan Eich, que trabalhava na Netscape, e que atualmente é membro da equipa da Mozilla.

Os anos seguintes da tecnologia foram marcados pela alteração no nome para JavaScript e pela intervenção da Microsoft ao criar uma tecnologia similar, o *JScript*. As duas tecnologias tinham os mesmos objetivos. O que as diferenciavam eram essencialmente os componentes que estavam desenvolvidos para determinados navegadores, isto é, certas funcionalidades da linguagem funcionavam corretamente no NetScape e não no Internet Explorer, ou vice-versa.

Mais tarde, e para resolver os problemas de incompatibilidade entre navegadores, a linguagem JavaScript foi tornada padrão pela *European Computer Manufacturers Association* (ECMA). Esta padronização tornou a linguagem formalmente denominada de ECMAScript. Na prática a comunidade chama JavaScript a essa tecnologia, independentemente da versão da linguagem ou do navegador em questão.

Quando Jesse James Garrett introduziu o conceito de *Asynchronous JavaScript and XML* (Garrett, 2005), tecnologia mais conhecida por *AJAX* que permite que a informação seja atualizada sem necessidade de carregar toda a página HMTL, existiu um grande aumento do uso do JavaScript nas aplicações web. Este aumento de popularidade veio amadurecer a comunidade e fez com que fossem criadas bibliotecas de modo a aumentar a facilidade de desenvolver código JavaScript

O JavaScript marcou o início da era do *client side scripting* das páginas web. Permite transformar o HTML em algo mais rico e com mais potencial, e é apenas com esta tecnologia que algumas das potencialidades do HTML5 podem ser exploradas. Em conjunto com o HTML5, o JavaScript consegue simplificar algumas tarefas, que eram com as versões anteriores, complexas e morosas de implementar, tal como incorporar um vídeo numa página ou validar formulários.

O JavaScript permite, de uma maneira breve, alterar informação de uma página HTML de uma maneira dinâmica e fluída, como por exemplo:

- Introduzir código HTML na página de uma forma dinâmica, como por exemplo para validar um formulário e, caso existam inconsistências, alertar o utilizador. Este tipo de técnicas evita assim chamadas desnecessárias ao servidor.
- Atribuir eventos a elementos, permitindo, por exemplo, que com um simples clique num botão seja colocado conteúdo disponível ao utilizador.

- Criar *cookies*, de modo a que os utilizadores possam ter uma experiência mais personalizada nas próximas vezes que usufruírem da aplicação web.

2.4 JSON

A tecnologia JSON é um formato padrão de troca de informação entre aplicações, podendo ser comparado ao XML. Permite de uma forma simples e legível representar estruturas de dados de forma independente da linguagem. O formato JSON é utilizado quando determinada aplicação faz pedidos de informação a servidores, quer a APIs quer em chamadas AJAX. Muitas das bibliotecas usadas em aplicações web suportam nativamente este tipo de estrutura.

O exemplo seguinte representa um objeto JSON que caracteriza uma sala (delimitado por '{' e '}') que contém um conjunto (delimitado por '[' e ']') de equipamentos, em que estes últimos também são considerados objetos JSON.

```
{
    "id": 1,
    "nome": "Sala01",
    "lugares": 30,
    "Equipamentos": [
      {
        "id": "1",
        "descricao": "projetor"
      },
      {
        "id": "2",
        "descricao": "Quadro"
      }
}
```

2.5 JQuery

O jQuery é uma biblioteca que permite simplificar o código JavaScript existente nas páginas HTML. Facilita a criação de animações, manipulação do DOM, pedidos AJAX e outro tipo de processos, e permite um simples e rápido desenvolvimento de código cliente. O alto nível de abstração que oferece torna-se na principal vantagem no uso desta biblioteca.

O jQuery permite abreviar tarefas comuns, que em JavaScript puro requerem várias linhas de código, em simples chamadas à biblioteca. Muitas das frameworks JavaScript analisadas dependem desta biblioteca que se tornou um elemento essencial no que toca a desenvolvimento de funcionalidades que recorrem a código JavaScript.

2.6 Adaptação dos navegadores ao HTML5

No início do HTML, existiu uma intensa luta entre navegadores, em que os principais competidores, como já foi referido, foram o Internet Explorer e o Netscape. A ideia que estava por detrás desta "guerra" era que cada navegador implementava as suas funcionalidades para que quando o programador construísse aplicações web, estas serem usufruídas apenas num navegador. Com o passar dos anos veio-se a mudar, felizmente, essa ideia já que os programadores de aplicações web passaram a preocupar-se em implementar aplicações que funcionassem independentemente do terminal.

Entretanto, este conflito terminou e atualmente os navegadores procuram satisfazer ao máximo quer os programadores, quer os utilizadores das aplicações web. O Firefox, o Internet Explorer, o Chrome, o Safari e o Opera são os navegadores mais utilizados e mais avançados, que garantem atualizações contínuas e são mais ricos em funcionalidades. Estes procuram implementar as funcionalidades disponíveis no HTML5, estando já algumas delas totalmente funcionais.

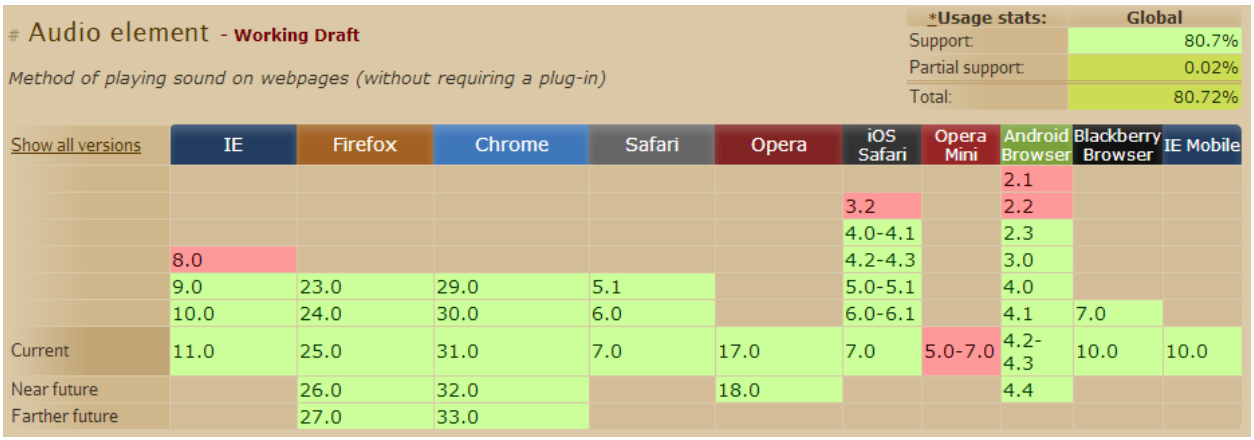
Uma das questões que se coloca quando se fala de suporte do navegador é a capacidade de determinado navegador ser compatível com a tecnologia HTML5. Esta é uma dúvida errada a ser colocada, porque o HTML5 é um grande conjunto de novas funcionalidades e na realidade nenhum navegador é compatível com a totalidade das funcionalidades HTML5. O mais correto a perguntar é se uma determinada funcionalidade HTML5 é compatível com determinada versão do navegador, já que estas vão sendo implementadas faseadamente nos navegadores.

Ao longo deste trabalho foi-se concluindo que muitas funcionalidades ainda não estão desenvolvidas em alguns navegadores e constatou-se que os que mais suportam o HTML5 seriam o Opera e o Google Chrome, sendo este último o mais utilizado nos exemplos descritos ao longo do documento.

Existem, por isso, várias ferramentas para verificar a existência destas funcionalidades para os vários navegadores, sendo *modernizr* uma das ferramentas mais referenciadas na literatura e das mais utilizadas

pelos programadores. Trata-se de uma biblioteca JavaScript que deteta funcionalidades HTML5 e CSS3 no navegador do utilizador.

Existem também várias aplicações web que permitem verificar quais as funcionalidades existentes nas diferentes versões dos navegadores. A página web `caniuse.com` permite verificar se uma versão de um navegador suporta determinada funcionalidade HTML5 ou CSS3. A fig. 2 mostra as versões dos diversos navegadores que suportam o elemento HTML5 `<audio>` e pode constatar-se que está suportado em 80.7% das várias versões dos navegadores existentes no mercado.



Audio element - Working Draft

Method of playing sound on webpages (without requiring a plug-in)

*Usage stats:	Global
Support:	80.7%
Partial support:	0.02%
Total:	80.72%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
						3.2		2.2		
						4.0-4.1		2.3		
	8.0					4.2-4.3		3.0		
	9.0	23.0	29.0	5.1		5.0-5.1		4.0		
	10.0	24.0	30.0	6.0		6.0-6.1		4.1	7.0	
Current	11.0	25.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	10.0
Near future		26.0	32.0		18.0			4.4		
Farther future		27.0	33.0							

Figura 2 – Página caniuse.com que indica o suporte dos navegadores ao elemento <audio> HTML5

Nas várias funcionalidades descritas ao longo da dissertação será feita uma pequena referência ao suporte da funcionalidade, a partir da página web apresentada.

As extensões dos navegadores foram um elemento muito importante para o desenvolvimento da Web e permitiram que aplicações web proporcionassem aos utilizadores experiências únicas, tais como jogos e vídeos. O problema com estas extensões é a necessidade de uma instalação manual e a constante atualização das possíveis falhas de segurança. Este processo não é trivial para todos os utilizadores pondo assim em risco a segurança dos mesmos.

O momento de viragem aconteceu em 2010 quando a Apple anuncia que não incorpora nos seus dispositivos móveis o Flash devido aos riscos de segurança e aos consumos de bateria, e indica que o futuro é o HTML5 (Jobs, 2010). Após este anúncio, o HTML5 começou a ser encarado com mais seriedade pela comunidade web.

As funcionalidades avançadas obtidas com as extensões eram complexas, senão impossíveis, de implementar com a versão 4 do HTML, por isso na especificação do HTML5 foi considerado evitar o uso destas extensões. Na especificação atual do HTML5, foram implementadas alternativas viáveis como as tag HTML `<video>` e `<canvas>` que são descritas no capítulo de análise.

3 Análise

Neste capítulo é efetuada a análise à tecnologia HTML5 e às frameworks JavaScript. A análise da tecnologia HTML5 permite identificar quais as funcionalidades disponíveis e quais as vantagens para programadores e para utilizadores finais. A análise das frameworks JavaScript tem o objetivo de escolher qual o conjunto de frameworks a utilizar no desenvolvimento do caso de estudo, identificando quais as frameworks JavaScript existentes no mercado, analisando os conceitos que cada uma suporta e comparando os aspetos positivos e negativos de cada uma delas.

3.1 Funcionalidades HTML5

3.1.1 Elementos globais

Uma das primeiras diferenças do HTML5 relativamente ao HTML4, e que é provavelmente a mais básica mas útil para os programadores, é a alteração do elemento que define o tipo de documento HTML presente. A tag DOCTYPE na sua última versão seria definida da seguinte maneira:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional// EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Com o HTML5 esta tag foi simplificada transformando-se apenas no seguinte:

```
<!DOCTYPE html>
```

Não existem problemas com versões antigas de navegadores porque estes são capazes de interpretar a nova tag. A tag que define o tipo de caracteres existentes na página também foi encurtada para simplificar o processo de construção da página.

No HTML4:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

No HTML5:

```
<meta charset="utf-8">
```

Quando desenvolvido código JavaScript numa página HTML, o processo de definição do tipo de linguagem já não se torna necessário. Basta que seja utilizada a tag `<script>`, enquanto que no HTML4 era necessário incluir o atributo `type="text/javascript"`.

3.1.2 Elementos removidos

Alguns elementos do HTML4 tornar-se-ão obsoletos quando o HMTL5 se tornar padrão. Os navegadores ainda os poderão vir a interpretar, mas a sua utilização deve ser evitada quando o HTML5 for publicado.

Os principais elementos removidos são o `<big>`, `<blink>`, `<center>`, `<font>` e `<applet>`. Para alguns destes elementos, a alternativa recomendada será o uso de CSS, enquanto que para outros, como o `<applet>`, deverão ser substituídos por novos elementos.

3.1.3 Elementos estruturais

Foram especificados novos elementos que tornam o processo de construir páginas web mais simples. Nas versões anteriores ao HTML5, recorria-se ao elemento `<div>` para criar várias secções das páginas tal como o cabeçalho ou a barra lateral.

Com os novos elementos HTML5, cada secção está incluída num elemento próprio, o que simplifica a implementação de uma página web, como se pode constatar na figura 3.

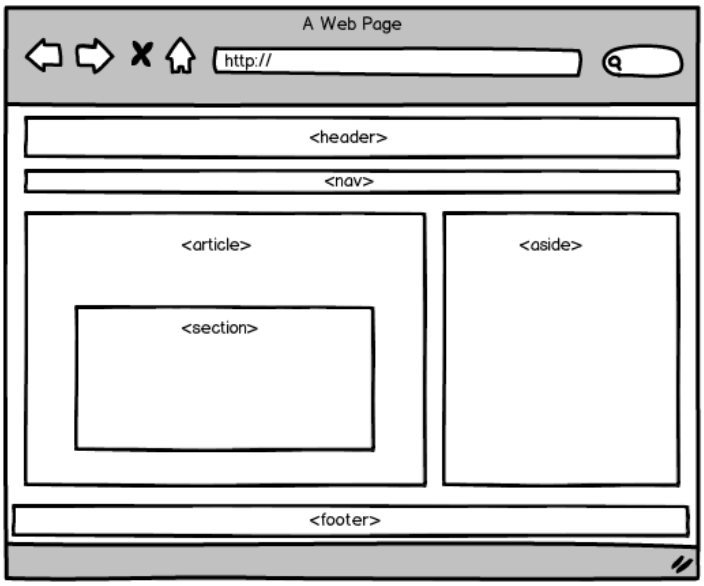


Figura 3 – Possível estruturação de uma página HTML a partir de novos elementos estruturais

De notar que a posição destes elementos não tem que ser a presentada na imagem, nem são de presença obrigatória, apenas é uma abordagem para construir uma página clara e bem estruturada.

Como apresentado acima, e como é usual na maioria das páginas web, o elemento <header> é o primeiro elemento da página. Poderá conter o título da página, imagens, entre outros. Por exemplo:

```
<header>
  <a href="/"><img src=logo.png alt="home"></a>
  <h1>A minha página</h1>
</header>
```

O elemento <nav> usualmente contém uma lista de hiperligações para páginas existentes na aplicação web. Este elemento até pode ser inserido na tag <header> ou estar incluído noutro elemento estrutural. Exemplo de aplicação do elemento <nav>:

```
<nav>
   <ul>
      <li>Home</li>
      <li><a href="noticias.html">Notícias</a></li>
      <li><a href="documentos.html">Documentação</a></li>
   </ul>
</nav>
```

O elemento <aside> permite incluir informação adicional ao conteúdo principal, tal como publicidade ou outro tipo de informação relacionada.

Os elementos <article> e <section> permitem definir um bloco principal de informação, como por exemplo uma notícia. Dentro do elemento <article> pode ser definido um elemento <header> que fica sempre no topo do <article>. Para concluir, existe o elemento <footer> que permite conter hiperligações secundárias na parte inferior da página, como por exemplo as ligações "contactos", "sobre nós" ou até informação legal. Como referido anteriormente, estes elementos podem ser incluídos uns com os outros, como demonstra a figura 4.

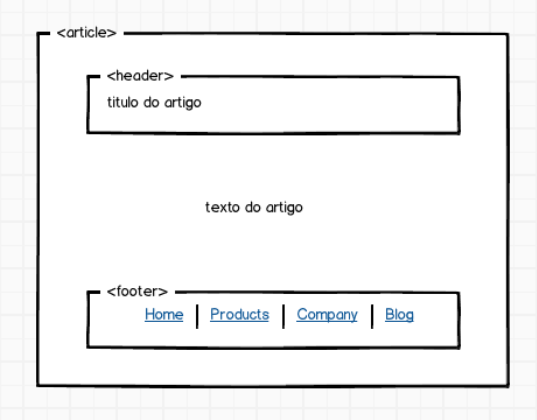


Figura 4 – Possível composição da tag <article>

3.1.4 Formulários

Em HTML5, a implementação dos formulários tornou-se mais simples e mais uniformizada. Por exemplo, no projeto de uma unidade curricular do mestrado em que esta dissertação se insere, o HTML5 permitiu ultrapassar as diferenças nas validações que cada programador implementava. Permite também ao programador poupar tempo na implementação dos mesmos, já que a maioria das validações é feita definindo apenas o tipo de *input* que se pretende.

A especificação do HTML5 não define a maneira como os navegadores devem representar as mensagens de erro que lançam. Diferentes navegadores representam a interface do utilizador de maneira diferente, como podemos constatar na figura 5.

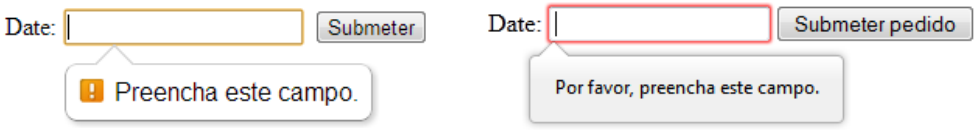


Figura 5 – Avisos HTML5 provenientes de diferentes navegadores

Estes avisos podem ser modificados, mas ao implementar essas modificações perde-se uma das vantagens que foi referenciada - a facilidade de produzir formulários. A validação dos dados é efetuada através de atributos introduzidos no HTML, por exemplo o atributo *required* permite validar a presença de qualquer input no controlo.

```
<input type="text" name="text" required>
```

Para incluir apenas números na entrada do formulário pode ser utilizado o tipo "number", em que apenas é permitida a introdução de caracteres numéricos no controlo. Esta funcionalidade tem a particularidade de estar implementada nos dispositivos móveis, em que o teclado apresentado é específico para introduzir números.

Por vezes, pode ser útil ao utilizador ajudar a preencher o controlo. Na versão 4 do HTML esta ajuda era implementada recorrendo a código JavaScript, mas o HTML5 tornou o desenvolvimento da funcionalidade muito mais simples.

```
<input type="text" name="textbox" placeholder="Introduza o texto">
```

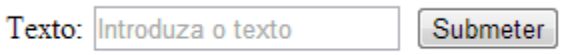


Figura 6 – Exemplo do atributo placeholder

Em certos casos é necessário validar, para além do tipo de dados, o conteúdo que irá ser introduzido. Com o atributo *pattern*, o programador pode, por exemplo, validar um número de 9 dígitos:

```
<input type="text" id="username" required pattern="[0-9]{9}">
```

Figura 7 – Mensagem de aviso caso input esteja incorreto

O tipo *email* permite validar a presença de um email no controlo, mas esta validação difere entre os diferentes navegadores. Por exemplo o Chrome valida um email do tipo "email@dominio", o Opera por outro lado valida logo que é introduzido um '@'.

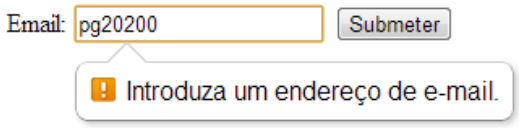


Figura 8 – Mensagem de aviso caso seja introduzido um e-mail incorreto

Com base nesta apresentação das principais funcionalidades dos formulários HTML5, pode concluir-se que é preciso ter em atenção o suporte dos navegadores para determinada funcionalidade, desde o seu tipo de validação até à não existência da funcionalidade. É preciso, no entanto, ter em consideração que são sistematicamente lançadas atualizações tanto aos navegadores como à especificação do HTML5.

3.1.5 Canvas

O *canvas* é um dos componentes mais discutidos do HTML5 e permite construir elementos gráficos para serem utilizados em imagens, jogos, entre outros. Em termos de HTML, este componente apenas especifica uma nova tag `<canvas>` que pode ser considerada como uma área da página que tem uma dimensão e que, no interior, se podem desenhar figuras, linhas, etc.

Para iniciar um "desenho" é necessário utilizar a tag `<canvas>` e definir a área deste.

```
<canvas id="b" width="300" height="225"></canvas>
```

A título de exemplo, pode ser definido o seguinte código JavaScript a aplicar no *canvas*:

```
function draw_b() {
    var b_canvas = document.getElementById("a");
    var b_context = b_canvas.getContext("2d");
    b_context.fillRect(50, 25, 150, 100);
}
```

Neste excerto de código é carregado o contexto de desenho "2d" que inicia a API no elemento definido. De seguida, pode ser iniciada a conceção dos elementos, em que neste caso em particular é desenhado um retângulo utilizando a função *fillRect.*

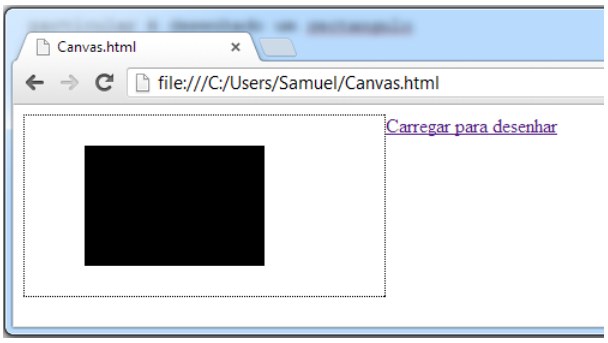


Figura 9 – Exemplo de desenho de um retângulo utilizando a API canvas

Com esta API é possível definir qualquer efeito que o programador deseja, desde cores fixas a gradientes. É possível ainda carregar imagens e definir texto para ser apresentado no *canvas*. No caso de curvas pode ser utilizado o método *arc* que permite criar curvas perfeitas, bem como construir círculos ou curvas mais elaboradas com base nas curvas de Bézier. Esta é das componentes do HTML5 mais complexas de utilizar, existindo livros focados apenas nesta API. É possível com os elementos apresentados obter uma alternativa válida ao *flash*, apenas utilizando HTML5 e JavaScript.

Caso seja necessário recorrer ao 3D, é recomendada a utilização de bibliotecas JavaScript, como por exemplo o *WebGL* ou o *KineticJS* que facilitam a criação de animações, camadas, eventos entre outras funcionalidades baseadas no elemento `<canvas>`.

3.1.6 Vídeo & áudio

Com o passar dos anos, o vídeo e o áudio passaram a ser cada vez mais utilizados. Um caso de sucesso foi o YouTube, que tem como funcionalidade principal permitir ao utilizador visualizar vídeos pela internet. Contudo, do lado do programador, a colocação de vídeo e áudio numa página web não é um processo trivial. Na anterior versão do HTML, era necessário recorrer a extensões para visualizar e ouvir este tipo de conteúdos, o que obrigava o utilizador a instalar uma extensão, como por exemplo o Flash, o Quicktime ou o Real Player. Na versão 4 do HTML, para adicionar um vídeo é necessário utilizar os elementos `<object>` e `<embed>`.

Dado que existem várias extensões de áudio e vídeo, é necessário o utilizador final descodificar o formato através dos codecs corretos (o codec é programa que codifica e descodifica determinados formatos existentes). Por isso, o desafio com esta nova versão do HTML é o tratamento do vídeo/áudio com diferentes codecs. As tabelas 1 e 2 mostram quais os codecs vídeo e áudio incorporados nos navegadores.

Tabela 1 – Suporte aos Codecs de vídeo nos diferentes navegadores

Codec	IE	Firefox	Safari	Chrome	Opera	IPhone	Android
Theora+Vorbis+Ogg	.	3.5+	.	4.0+	10.5+	.	.
H.264+ACC+MP4	9.0+	.	3.2+	4.0+	.	3.2+	2.1+
WebM	.	4.0+	.	6.0+	10.6+	.	2.3+

Tabela 2 - Suporte aos Codecs de áudio nos diferentes navegadores

Codec	IE	Firefox	Safari	Chrome	Opera	IPhone	Android
Vorbis+Ogg	.	3.6+	.	6.0+	10+	.	2.3+
MP3	9.0+	.	5.0+	6.0+	.	4.0+	2.3+
WAV	.	3.6+	5.0+	9.0+	10+	4.0+	.

Como se pode constatar, diferentes navegadores descodificam diferentes formatos, pelo que o desafio para o programador é conseguir que o vídeo e o áudio sejam produzidos nas várias versões dos navegadores. O exemplo seguinte mostra como apresentar um vídeo utilizando o elemento HTML5 `<video>`, para o caso do áudio basta substituir a tag para `<audio>`.

```
<video src="pr6.webm" width="320" height="240"></video>
```

Existem vários atributos que permitem personalizar o vídeo, tal como o atributo *controls* que atribui um leitor por omissão para os vídeos. Este leitor difere de navegador para navegador. O exemplo seguinte apresenta um vídeo a ser executado no Chrome na versão 23.

```
<video src="pr6.webm" width="320" height="240" controls></video>
```

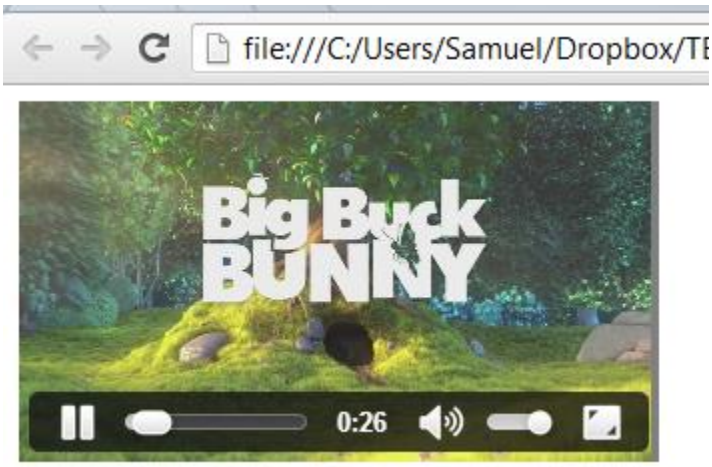


Figura 10 – Exemplo de leitor de vídeo HTML5 com controlos incorporados

Existem ainda dois atributos que permitem definir o comportamento do vídeo no arranque da página. O primeiro é o *preload* que permite ao navegador descarregar o vídeo logo que a página é completamente carregada. O segundo é a opção *autoplay* que executa o vídeo sem o consentimento do utilizador, ou seja, executa o vídeo após a página estar carregada. No exemplo seguinte o vídeo é descarregado automaticamente mas este só é executado quando o utilizador desejar.

```
<video src="pr6.mp4" width="320" height="240" preload autoplay=”none”>
</video>
```

Como indicado anteriormente, o desafio para o programador, é de conseguir reproduz o vídeo nos vários navegadores. Como solução, pode ser necessário indicar o mesmo vídeo com diferentes *codecs*, para este se tornar compatível com um leque superior de navegadores.

```
<video width="320" height="240" controls>
     <source src="pr6.mp4" type='video/mp4; codecs="avc1, mp4a.40.2">
     <source src="pr6.webm" type='video/webm; codecs="vp8, vorbis"'>
     <source src="pr6.ogv" type='video/ogg; codecs="theora, vorbis"'>
</video>
```

A tag `<video>` é interpretada pelos navegadores dos dispositivos móveis, mas a resolução dos vídeos não é adequada quando comparada com o tamanho do ecrã, por isso é recomendada uma resolução adequada quando os dispositivos destino forem telefones ou *tablets*. Para resolver o problema é adotada uma solução que recorre ao CSS bastando definir o atributo `media` no elemento `<source>` do vídeo.

```
<video width="320" height="240" controls>
    <source src="hi-pr6.mp4” media=”(min-device-width:¬ 800px)”>
    <source src="low-pr6.webm" '>
</video>
```

Uma das barreiras à introdução deste elemento no mercado é a publicidade colocada nos vídeos, que no caso do YouTube continua a ser apresentada em Flash.

3.1.7 Drag & Drop

Uma das funcionalidades cada vez mais vistas nas aplicações web é a possibilidade de arrastar e largar um ficheiro para o servidor. Alguns serviços recorrem a extensões ou bibliotecas JavaScript para disponibilizar esta funcionalidade. Na versão 5 do HTML, qualquer elemento passa a poder ser arrastável, simplificando o desenvolvimento de funcionalidades com base no drag & drop.

O exemplo que servirá de explicação para demonstrar esta funcionalidade consiste no arrastamento de uma imagem para uma área que está preparada para receber a imagem.

Figura 11 – Caso de estudo da funcionalidade Drag & Drop

Primeiro, é necessário que o elemento seja arrastável, adicionando-lhe o atributo *draggable* e definindo a função a ser invocada no atributo *ondragstart*.

```
<img id="drag1" src="img.png" draggable="true" ondragstart="drag(event)" width="100" height="100">
```

É necessário também definir a função invocada depois dos elementos serem largados, através do atributo *ondrop*.

```
<div id="div1" ondrop="drop(event)" style="float: left">
```

Para concluir este exemplo, é posteriormente necessário implementar o código JavaScript para definir o comportamento dos eventos.

3.1.8 Armazenamento

O armazenamento do HTML5 tornou-se uma alternativa aos *cookies*. Os *cookies* são ficheiros que contêm informação de uma aplicação web e estão localizados no disco do utilizador. Assim, foi necessário "reinventar" a maneira como os dados eram guardados do lado do cliente, e com o HTML5 veio um novo método de armazenamento, o *web storage*.

Comparando com os *cookies*, este método é mais rápido e mais seguro. É necessário ter em conta que este tipo de armazenamento não deve ser substituído por um servidor para guardar a totalidade da informação, apenas é uma alternativa aos métodos de armazenamento do lado do cliente, e que deve ser utilizado para dados não críticos. Este armazenamento, tal como os *cookies*, não é independente do terminal, sendo que os dados apenas são guardados localmente.

O *web storage* divide-se em duas categorias, o *localstorage* e *sessionstorage*. O primeiro permite que os dados sejam acedidos em diferentes sessões do navegador ou até depois de ser fechado, idêntico ao comportamento de um cookie. O *sessionstorage* apenas armazena os dados numa janela do navegador, e depois de esta ser fechada os dados serão apagados.

A API torna simples as operações de dados. O métodos *setItem* e *geItem* definem respetivamente o valor a guardar e o valor a obter. A título de exemplo, no seguinte código JavaScript irá ser guardado o texto 'teste':

```
localStorage.setItem('valor','teste');
```

Para verificar o valor guardado, podem ser acedidas ferramentas de programação dos navegadores. Na figura 12 é mostrada a ferramenta de programador do Chrome. Podemos verificar que a variável 'valor' que tem atribuído o valor 'teste' foi guardado com sucesso e pode ser acedido através da função *localStorage.getItem('valor').*

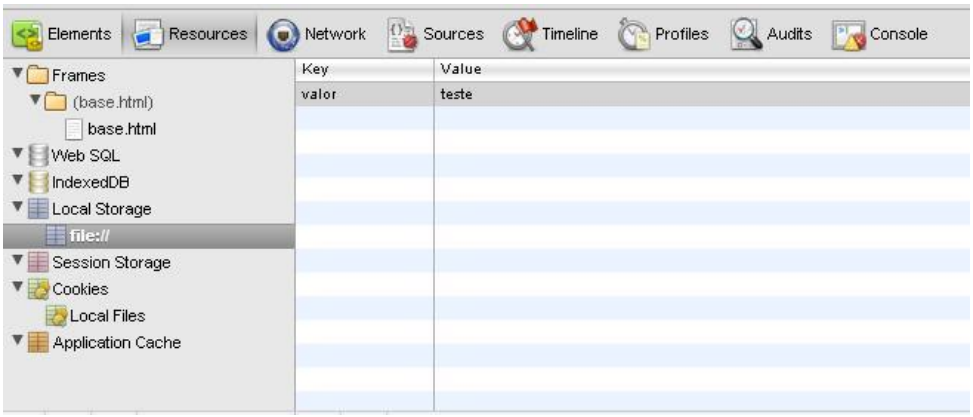


Figura 12 – Ferramenta de programação do Google Chrome

Um dado importante de referir é que a funcionalidade de armazenamento HTML5 já pode ser explorada em todos os principais navegadores, como se pode constatar na tabela 3, e consequentemente um número razoável de aplicações web utilizam esta funcionalidade.

Tabela 3 - Disponibilidade da funcionalidade de armazenamento nos diferentes navegadores

IE	Firefox	Chrome	Safari	Opera	IOS	Android
8.0+	16.0+	23.0+	5.1+	12.1+	3.2+	2.1+

Como alternativa ao *web storage*, existe o *Indexed Database*, que é uma API mais complexa mas que permite guardar informação de uma maneira hierárquica, como por exemplo um objeto JSON. Existe também a API de acesso a ficheiros que pode ser útil, por exemplo, para guardar imagens ou ficheiros PDF.

3.1.9 Offline

As aplicações web sempre têm como pressuposto a existência de uma ligação à internet, e hoje em dia grande parte das aplicações necessitam de uma ligação à rede. A funcionalidade *offline* do HTML5 não foi especificada para tornar as aplicações independentes da internet, mas foi desenhada para dar ao utilizador uma experiência agradável quando existe perda de ligação, o que é conveniente para utilizadores de dispositivos móveis.

Por isso, o objetivo da aplicação que implementa esta funcionalidade é de realizar o descarregamento de determinados ficheiros indicados pelo programador e guardá-los na cache do navegador. Quando é realizada uma tentativa de aceder à aplicação sem ligação à rede o navegador carrega automaticamente os ficheiros locais, permitindo assim ao utilizador continuar a sua interação com a aplicação. Podem ser feitos cache de ficheiros CSS, JS e HTML.

Em relação aos dados alterados pelo utilizador, quando a aplicação está a ser executada sem acesso à internet, é da responsabilidade do programador implementar as soluções para não existir perda de dados.

Os ficheiros que ficarão disponíveis, quando não existe ligação à internet, têm de ser especificados num ficheiro *manifest* que se encontra no servidor. O ficheiro *manifest* é um simples ficheiro de texto que indica quais os recursos que o navegador deve guardar localmente, este ficheiro deve ser referenciado como atributo da tag `<html>` no início da página HTML.

```
<html manifest=http://localhost/manifest.mf>
…
</html>
```

O ficheiro *manifest* poderá estar estruturado em 3 partes, cada uma das quais descrita de seguida.

Cache

Secção em que são especificados os ficheiros que ficarão disponíveis caso não exista acesso à internet. Esta é a única secção das 3 referenciadas que é de uso obrigatório. Exemplo:

```
CACHE MANIFEST
index.html
stylesheet.css
images/logo.png
scripts/main.js
```

Network

Nesta secção são definidos os ficheiros que requerem obrigatoriamente uma ligação para o servidor, ou seja, ficheiros que caso seja feito um pedido forçado ao servidor por parte do utilizador estes têm que ser acedidos via internet e não localmente. No caso apresentado de seguida, todos os ficheiros que não estão indicados na secção Cache serão pedidos ao servidor remoto.

```
NETWORK:
*
```

Fallback

Secção que especifica para onde é redirecionado o utilizador caso um recurso não esteja acessível. O primeiro parâmetro define a fonte a ser acedida e o segundo é o destino do reencaminhamento caso não haja ligação.

```
FALLBACK:
/ /offline.html
```

Neste caso, o utilizador é redirecionado para a página offline.html caso tenha sido feito um pedido a um recurso não disponível na secção Cache do ficheiro manifest.

Esta componente, juntamente com a de armazenamento, permite que os utilizadores tenham uma experiência de utilização agradável quando não possuem ligação à rede. De referir que esta funcionalidade está bem presente nos dispositivos móveis que são muitas vezes submetidos a falhas de ligações.

3.1.10 Geolocation

Esta API permite obter a posição do utilizador da aplicação com relativa precisão. Esta funcionalidade é uma especificação da W3C e, como se pode constatar na tabela 4, já está disponível na maioria dos navegadores.

Tabela 4 - Disponibilidade da funcionalidade Geolocation nos diferentes navegadores

IE	FIrefox	Chrome	Safari	Opera	IOS	Android
9.0+	16.0+	23.0+	5.1+	12.1+	3.2+	3.1+

Quando é chegado o momento de partilhar a localização do utilizador, a segurança deste é tomada em questão. Segundo a especificação "um *user agent* não deve enviar informação da localização para web sites sem a explícita permissão do utilizador" (W3C, 2013). Por isso, quando é necessária a localização do utilizador, o navegador pede permissão para aceder à mesma. Na figura 13 é apresentado um exemplo típico de mensagem a requisitar a localização do utilizador, neste caso no Google Chrome.

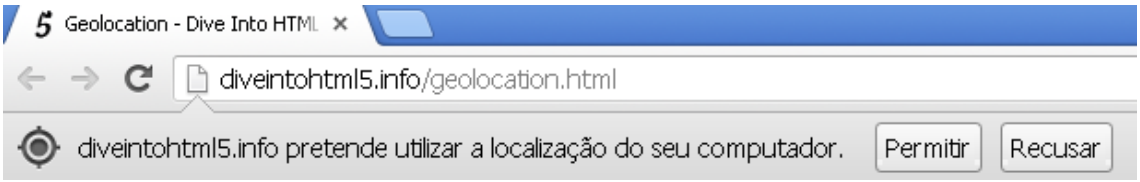


Figura 13 – Mensagem de requisição de localização no Chrome

A obtenção da localização torna-se muito útil para escolher que tipo de informação deve ser apresentada ao utilizador, como, por exemplo, os locais ou eventos de interesse próximos, as principais notícias da zona em questão ou até para apresentar publicidade de estabelecimentos próximos.

Para aceder à localização, apenas é necessário fazer uma chamada à função *getCurrentPosition(successFunction,errorFunction)* da API.

Existem vários métodos de deteção da localização, sendo os mais conhecidos e utilizados apresentados de seguida.

- GPS - Este método é o mais preciso que existe. Utiliza o sistema GPS para obter a localização mas é o menos utilizado no mercado, devido ao facto de estar apenas disponível em dispositivos móveis e este ser o mais lento para obter resposta.
- A-GPS (Assistive GPS) - Este método utiliza a triangulação entre as torres da rede móvel para determinar a localização, não é tão preciso como o GPS mas suficiente para a maioria das ocasiões.
- Wi-Fi – Este é provavelmente o mais utilizado para dispositivos moveis, é rápido e relativamente preciso.

- IP – Pelo IP do utilizador é possível obter uma posição relativa deste, mas é o menos preciso dos métodos apresentados, sendo por vezes apenas possível obter o país e a zona através deste método.

Esta API torna-se útil para determinar que tipo de informação pode ser apresentada pelo utilizador, ou como acontece nas redes sociais, onde é partilhada determinada informação. Esta API é relativamente simples de utilizar e acompanha a tendência da massificação dos dispositivos móveis.

3.1.11 Web Sockets

Hoje em dia a web baseia-se no protocolo HTTP, que consiste no processamento de pedido/resposta por parte do cliente (normalmente browser) e do servidor. Este protocolo torna a navegação entre páginas web estática. O aparecimento do AJAX permitiu uma navegação mais dinâmica mas sempre baseado neste tipo de paradigma. Existe atualmente uma tecnologia que permite que o servidor envie dados para o cliente sempre que achar necessário (sem que o cliente faça primeiro o pedido), denominada de "Comet", mas que nunca foi considerada uma opção nativa e de simples implementação.

Sendo assim tornou-se necessário desenvolver uma opção nativa nascendo assim a especificação do web socket. Esta tecnologia permite uma conexão persistente entre o cliente e o servidor em que ambos podem enviar informação a qualquer altura (sem que exista um primeiro pedido forçado por uma das partes).

Para implementar este tipo de tecnologia é necessário criar primeiro a conexão:

```
var socket = new WebSocket('ws://meuservidor.com/updates');
```

De seguida para enviar informação é necessário implementar a função de quando é aberta a conexão e enviar os respetivos dados através da função `send()` do objeto WebSocket:

```
connection.onopen = function () {
  connection.send('Ping'); // Envio do texto 'Ping' para o servidor
};
```

Para receber informação basta implementar a função `onmessage` e receber a informação vinda do servidor:

```
connection.onmessage = function (e) { // mensagem vinda do servidor
  console.log('Server: ' + e.data);
};
```

Esta tecnologia é particularmente útil em casos em que é necessária uma baixa latência, aproximando-se de uma aplicação a funcionar em tempo real. Pode ser aplicada em casos como:

- Jogos online multiplayer;
- Chat e aplicações de troca de mensagens;
- Log de eventos em tempo real;

É necessário referir que a arquitetura do lado do servidor terá que ser pensada, dado estes novos padrões de comunicações entre cliente/servidor. Existem já frameworks para servidores que suportam nativamente esta tecnologia tal como Socket.IO, jWebSocket, PlayFramework!, entre outras.

3.1.12 Workers

Sendo o JavaScript executado num ambiente *single-threaded* (onde apenas se executa um script de cada vez), e caso o código JavaScript a ser executado contenha funções complexas este pode interromper o funcionamento do navegador levando ao aparecimento da seguinte mensagem:

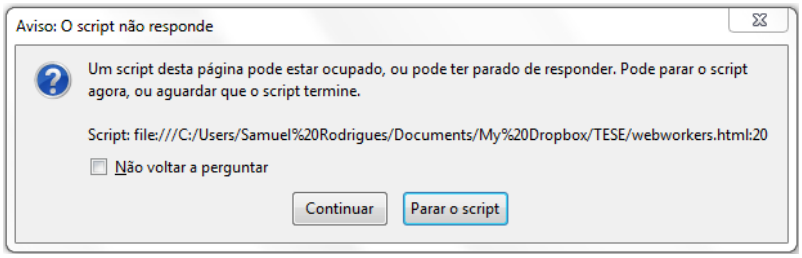


Figura 14 – Mensagem de aviso caso o script interrompa o funcionamento do navegador

Os web workers permitem evitar que situações como estas ocorram, ou seja, proporciona que vários scripts sejam executados ao mesmo tempo permitindo que sejam processadas grandes quantidades de dados e sejam feitas várias chamadas a APIs sem que o utilizador seja privado de navegar na aplicação web.

Esta componente está implementada nos vários navegadores existentes, incluído a versão 10 do Internet Explorer. Existem dois tipos de web workers:

- *Dedicated Workers* - Estes estão ligados apenas ao script "pai", ou seja, o script que criou o worker.
- *Shared Workers* - Estes podem comunicar entre workers e não dependem apenas do script "pai".

De uma maneira típica o código relativo aos workers deverá estar incluído num ficheiro JavaScript, a ser chamado pelo JavaScript da página HTML.

```
var worker = new Worker('task.js');
```

Caso o ficheiro exista, o navegador cria um novo fio de execução para o ficheiro em questão; caso contrário o worker não é executado. A comunicação entre workers é feita através da passagem de mensagens, sendo estas recebidas e enviadas através do evento *onMessage*. Existem duas maneiras de terminar um worker:

1. Chamando a função *worker.terminate()* a partir do script principal;
2. Chamando a função *self.close()* dentro do próprio worker;

Os web *workers* são ainda muito recentes e pouco presentes no mercado. A maioria dos exemplos encontrados envolve o cálculo de funções matemáticas, mas podem ser utilizados em muitos mais casos, tais como:

- Processamento de grande quantidade de dados;
- Análise de vídeo ou áudio;
- Processamento de imagens;
- Processamento de listas de dados ou respostas vindas de *web services;*
- Cifragem/decifragem de dados;

3.1.13 HTML5 nas aplicações móveis

Com o grande aumento da utilização dos dispositivos móveis na navegação da internet, a construção das aplicações web tem uma componente móvel cada vez mais importante. Existe já no mercado uma grande discussão relativa a ter aplicações que recorrem ao HTML5 ou a aplicações nativas (nomeadamente iOS e Android). A grande vantagem do HTML5 para estas aplicações é a possibilidade delas serem executadas nos vários tipos de dispositivos, o que implica um menor custo de produção.

A desvantagem do HTML5 é que as aplicações nativas têm um leque mais alargado de funcionalidades no que toca ao uso dos recursos físicos do dispositivo. O uso da câmara ou do acelerómetro são exemplos em que é necessário recorrer ao código nativo. A velocidade das aplicações HTML5 para dispositivos móveis sempre foi criticada, sendo um grande exemplo as aplicações do Facebook para Android e iOS que eram extremamente lentas. Mark Zuckerberg, CEO do Facebook, afirmou que "foi um erro ter apostado em demasia no HTML5" (techcrunch, 2012). Em 2013, já foram lançadas aplicações nativas que melhoraram a aplicação em termos de velocidade.

Para resumir existem principalmente 3 tipo de abordagens – nativa, HTML5 e híbridas. O esquema seguinte permite identificar onde situa cada abordagem no que toca à capacidade de uso do hardware e a plataforma para o qual se destina a aplicação.

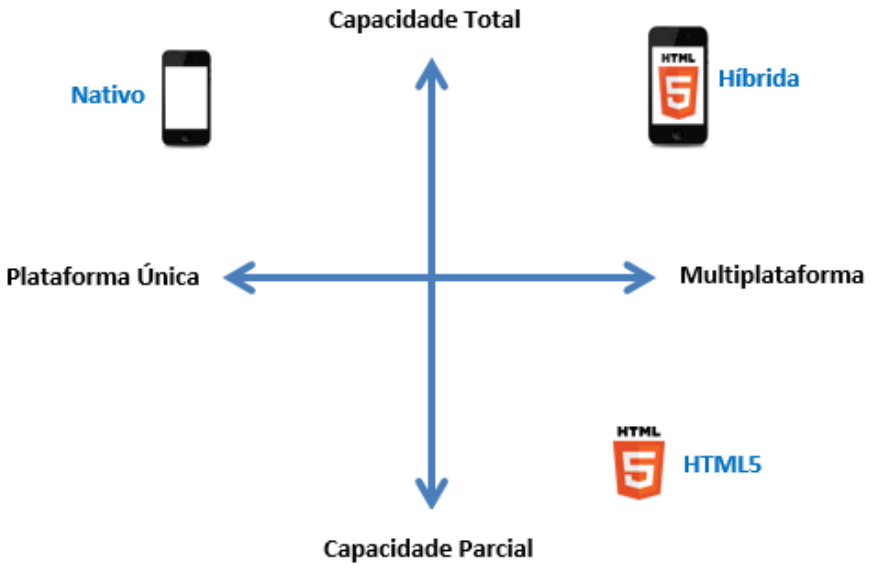


Figura 15 – Esquema que representa onde se inserem os vários tipos de aplicações móveis

As aplicações nativas são específicas de uma tecnologia/plataforma (Objective-C para iOS e Java para Android), e as aplicações HTML5 utilizam tecnologias padrão como HTML, CSS e JavaScript. As aplicações HTML5 podem ser executadas em diversas plataformas, mas são limitadas em termos de aproveitamento do hardware. Existem várias ferramentas que permitem criar aplicações deste tipo tais como o *jQueryMobile* ou o *Sencha Touch*. Pode ser também abordada uma solução *responsive* que permite que a página se adapte às várias resoluções de ecrãs. A figura 16 apresenta a página web da Microsoft segundo várias resoluções de ecrã, podendo constatar diferenças no design de cada uma.

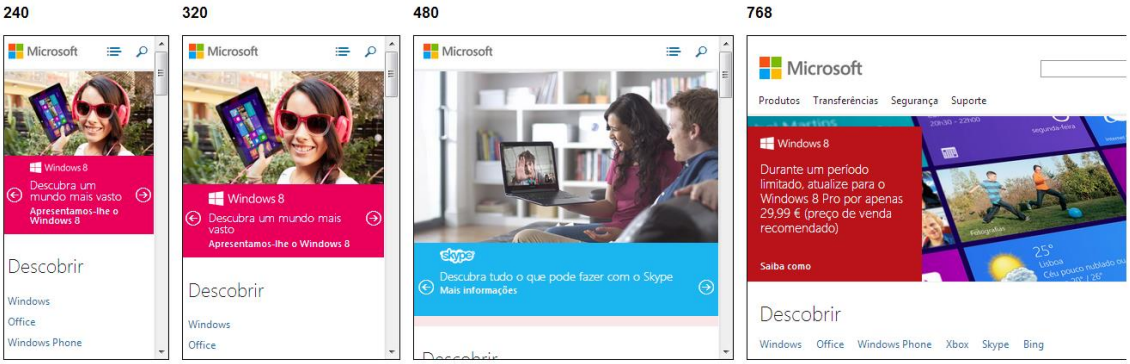


Figura 16 – Exemplo de uma página web responsive nas várias resoluções de ecrãs

A alternativa híbrida permite juntar o melhor (e o pior) de ambas as anteriores soluções. Uma solução deste tipo é composta por ficheiros HTML, CSS, JavaScript e encapsuladas num pacote nativo, o que possibilita por exemplo aceder aos recursos do dispositivo e a aplicação estar disponível na respetiva loja de aplicações. O *phonegap* e o *trigger.io* são ferramentas que permitem fazer essa encapsulação.

Para concluir, o mercado das aplicações móveis está a adaptar-se às necessidades dos consumidores e empresas. Existem várias soluções no mercado que permitem tirar proveito do HTML5 e cabe a cada entidade saber qual a melhor opção a tomar.

3.2 Frameworks JavaScript

Atualmente a tecnologia JavaScript tornou-se essencial para qualquer aplicação web. Esta tecnologia evoluiu o suficiente para serem criadas aplicações web complexas recorrendo quase unicamente ao JavaScript. Algumas bibliotecas existentes revelaram-se algo limitadas no que toca a código JavaScript complexo e extenso, levando assim ao aparecimento de alternativas como frameworks que fornecem as

funcionalidades necessárias para os programadores desenvolverem aplicações que possam, por um lado, satisfazer as necessidades do utilizador e, por outro, criar um código simples, estruturado e legível. As bibliotecas JavaScript para manipulação DOM, como por exemplo o jQuery, não são suficientemente eficazes para aplicar a casos complexos dado que código produzido é confuso, complexo de implementar e de relativo baixo nível. As frameworks permitem adicionar um nível de abstração que torna mais simples a manipulação de dados e permite separar a UI dos dados.

Este tipo de frameworks é, na maioria das vezes, utilizado em *single-page applications* (SPA). Este conceito consiste em aplicações compostas apenas por uma página web e tem por objetivo dar uma melhor experiência ao utilizador através de atualizações parciais da página, o que as torna úteis em aplicações de larga escala. Em aplicações SPA, tipicamente, todo o código necessário (HTML, CSS, JS) é descarregado no arranque inicial da aplicação e são feitas chamadas a servidores apenas para manipulação de dados. Estas aplicações são aplicáveis em diversos casos, como por exemplo, aplicações que comuniquem com servidores baseados em serviços (APIs ou arquiteturas orientada a serviços). O melhor exemplo de uma aplicação deste tipo é o Gmail (mail.google.com) que descarrega os ficheiros necessários (neste caso depois do login efetuado) e que, de seguida, apenas faz alterações parciais à página para apresentar a informação necessária ao utilizador, sem necessitar de redesenhar toda a página.

Padrão MVC

O padrão *Model-View-Controller* (MVC), bastante conhecido no mundo das aplicações web, foi introduzido a fim de gerir o aumento da complexidade das aplicações web. Tornou-se fundamental a separação entre os dados e a interface, pelo que alterações feitas no design da página não afetam a manipulação de dados, e vice-versa. Este padrão (representado na figura 17) tem como conceitos chave:

- **Modelos** - Os Modelos são o coração de qualquer aplicação, representam uma entidade do domínio, bem como uma parte da lógica que o rodeia: conversões, validações, propriedades e controlo de acessos.
- **Vistas** - As vistas no padrão MVC têm dois objetivos: Ouvir os eventos lançados pelo DOM na página HTML, e apresentar os modelos ao utilizador. A ideia geral da vista é de representar os dados, e em que estes podem sofrer edição.
- **Controlador** - O MVC resolve o problema da separação das camadas de acesso aos dados, regras de negócio e apresentação/interação com o utilizador, introduzindo um componente entre os dois:

o controlador. Este normalmente é desenvolvido com o intuito de manipular os modelos consoante os pedidos efetuados pelo utilizador (através da vista).

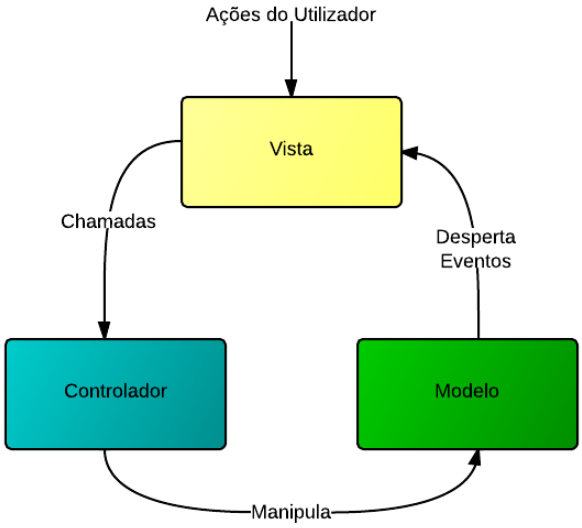


Figura 17 – Padrão MVC em frameworks JavaScript

O MVC é muito aplicado em aplicações Web, onde a vista é geralmente a página HTML, o controlador contém o código que implementa as regras do negócio e manipula os modelos, por fim o modelo é representa o conteúdo/objeto em si, geralmente armazenado em base de dados.

As aplicações que são construídas com as frameworks JavaScript analisadas têm uma arquitetura MVC que é aplicada do lado do cliente (Figura 18). São usualmente denominadas de frameworks MV*, porque são compostas pelo modelo e pela vista, mas o conceito do controlador difere de framework para framework. Concluindo, cada uma tem a sua maneira de implementar o padrão MVC.

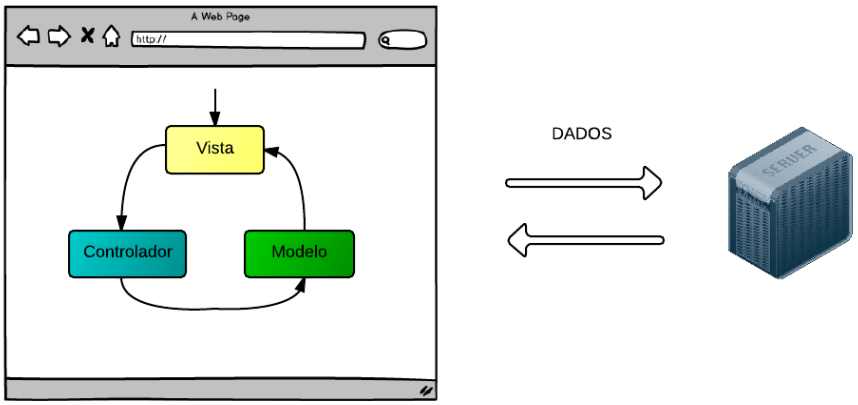


Figura 18 – Padrão MVC do lado do cliente

Antes de escolher uma framework JavaScript para o desenvolvimento de uma aplicação web, é conveniente efetuar uma análise às vantagens e desvantagens. Por isso, foi necessário efetuar um levantamento das frameworks existentes no mercado, que teve como base conteúdo retirado de pesquisas efetuadas, nomeadamente na internet, já que não existe ainda bibliografia específica sobre o mercado em questão. Devido ao elevado número de amostras a analisar, cada uma das frameworks não foi testada com casos práticos, o que permitiria uma maior precisão na análise mas que tornaria este processo de análise lento e demorado.

A quase totalidade das frameworks baseiam-se no mesmo padrão MVC, sendo as maiores diferenças a aplicação do controlador e a maneira como é efetuado a ligação dos dados nas páginas HTML. Existem ainda outras framework que não foram analisadas por vários fatores, tal como documentação inexistente, imaturidade da framework, comunidade praticamente inexistente e ausência do padrão MVC.

É necessário relembrar que muitas das frameworks apresentadas podem ser incorporadas com outras bibliotecas para procurar resolver os pontos menos positivos de cada uma. A listagem seguinte apresenta as frameworks JavaScript identificadas através de pesquisas no github (github.com), comunidades JavaScript, e artigos espalhados por diversas páginas da internet dedicadas ao desenvolvimento web. Apenas 9 das 15 apresentadas foram testadas com o caso de estudo.

- Agility
- Angular
- Backbone

- Batman
- Canjs
- Cappuccino
- Ember
- Extjs
- Javascriptmvc
- Knockout
- Meteor
- Sammy
- Spine
- Sproutcore
- YUI app frameworks

4 Desenvolvimento do caso de estudo

Este capítulo procura especificar e apresentar o caso de estudo desenvolvido, de modo a aplicar, num produto de software, as funcionalidades HTML5 e as frameworks referenciadas no capítulo de análise. O desenvolvimento do caso de estudo foi efetuado e acompanhado pela PT Inovação de modo a avaliar os resultados da implementação deste caso de estudo.

O caso de estudo depois de desenvolvido permitirá identificar quais as limitações e vantagens de cada framework para, no final, conseguir construir uma matriz que diferencie e destaque as várias frameworks segundo um conjunto de critérios.

4.1 Visão geral do caso de estudo

Esta aplicação tem como principal objetivo possibilitar ao utilizador definir quais as competências que este domina. De uma maneira simples, o utilizador escolhe de uma lista de competências aquelas que ele entende que possui, possibilitando assim à organização saber quais as competências dos seus colaboradores.

Esta aplicação proporciona ainda a sugestão de novas competências ao utilizador, com base nas competências anteriormente escolhidas pelo utilizador. Por essa razão, as várias competências deverão estar associadas entre si através, por exemplo, de tags. Estas tags são elementos que descrevem as competências, podendo várias competências estar associadas à mesma tag.

Um exemplo seria adicionar a competência ‘Oracle’ à lista pessoal, que tem como tags ‘base_de_dados’ e ‘comercial’. A figura 19 mostra competências relacionadas com estas tags.

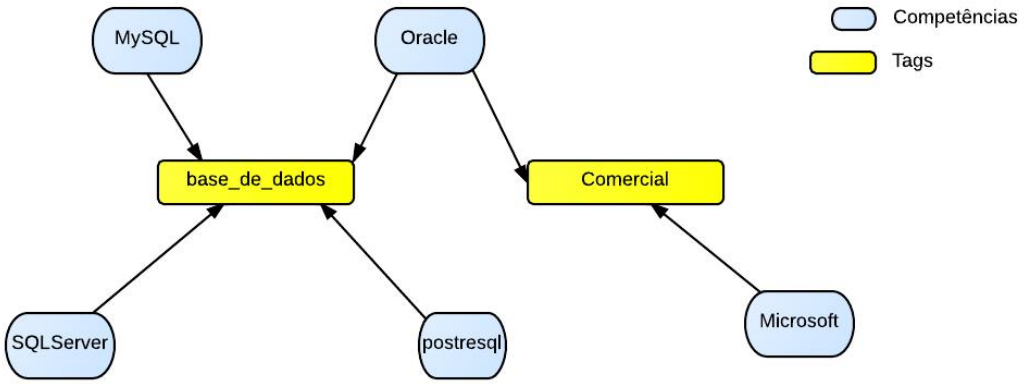


Figura 19 – Exemplo de relacionamento entre competências

Assim, poderiam ser sugeridas as competências 'SQLServer' e 'Microsoft' porque estão relacionadas através das tags da competência 'Oracle' (tags 'base_de_dados' e 'comercial'). A aplicação também tem que permitir visualizar quais as competências escolhidas pelos utilizadores da aplicação e com que nível de domínio, de forma que, de um ponto de vista administrativo, se possa ter uma melhor perceção das competências dos utilizadores que usufruem da aplicação

Existirão três níveis dentro da lista de competências pessoais, sendo o nível 1 as competências que o utilizador domina menos e o nível 3 o máximo de conhecimento de determinada competência.

4.2 Funcionalidades do caso de estudo

Para qualquer aplicação que exija interações com o utilizador, torna-se útil identificar as funcionalidades existentes na aplicação, assim como identificar os atores que irão interagir com o sistema.

Sendo esta uma aplicação relativamente simples, apenas um tipo de utilizador é considerado e este tem disponibilizadas as seguintes funcionalidades:

- Escolher competências
- Remover competências
- Iniciar sessão
- Visualizar competência dos utilizadores
- Alterar nível de uma competência
- Pesquisar competências

4.3 Requisitos do caso de estudo

De modo a que a aplicação possa ser coerente, abordam-se as operações CRUD (criar, ler, editar e apagar) sobre os vários modelos da aplicação. Este tipo de operações é corrente nas aplicações, por isso torna-se importante implementar estas funcionalidades para efetuar uma correta avaliação das frameworks. Sendo assim, partindo das funcionalidades identificadas no capítulo anterior, a aplicação nesta fase inicial tem como requisitos:

Número: 1
Título: Escolher competências
Descrição: O utilizador tem a possibilidade de escolher, de uma lista de competências, aquelas que achar que domina.
Racional: De modo a responder ao propósito principal da aplicação, o utilizador tem acesso à lista de competências existentes no sistema, escolhendo as que deseja para se tornarem suas.

Número: 2
Título: Alterar nível da competência
Descrição: O utilizador pode alterar o nível das competências que lhe estão designadas.
Racional: De modo a diferenciar as competências conhecidas do utilizador, estas possuem um dado valor (1, 2 ou 3) de modo a diferenciar o nível de conhecimento.

Número: 3
Título: Remover competências
Descrição: O utilizador pode remover competências da sua lista pessoal de competências.
Racional: Caso o utilizador se tenha enganado a escolher competências ou achar que estas deixaram de ser apropriadas, pode remover competências da sua lista pessoal.

Número: 4
Título: Adicionar competências sugeridas
Descrição: O utilizador pode adicionar competências à sua lista competências pessoais, a partir de um conjunto de sugestões que lhe é apresentado.
Racional: O algoritmo de escolha de sugestões tem que apresentar as sugestões de modo a aumentar a probabilidade de serem escolhidas. As sugestões são apresentadas por uma ordem e, para tal, o algoritmo baseia-se numa contagem das tags existentes nas competências pessoais, para depois possibilitar a ordenação das competências que têm tags idênticas às já escolhidas pelo utilizador. Por exemplo, caso seja escolhida a tag *base_dados* 5 vezes e a tag *comercial* 2 vezes, as competências com a primeira tag deverão aparecer no topo da lista de sugestões de modo a aumentar a probabilidade de aceitação da sugestão. Este algoritmo apenas permite o limite máximo de 10 sugestões ao utilizador, de modo a garantir uma maior eficiência do algoritmo.

Número: 5
Título: Iniciar sessão
Descrição: O utilizador autentica-se para ter acesso às funcionalidades da aplicação.
Racional: O utilizador apenas pode aceder à aplicação depois de iniciar a sessão na aplicação e caso as respetivas credenciais estejam corretas. Esta validação permite que a informação relativa às competências do utilizador seja corretamente armazenada e que não possam ser alteradas por outros utilizadores da aplicação.

Número: 6
Título: Pesquisar competências
Descrição: De modo a facilitar a tarefa de escolher determinada competência que o utilizador possui, a aplicação permite a pesquisa de competências existentes na aplicação.
Racional: São incorporados 2 controlos na página de forma a facilitar a pesquisa de competências. (1) Input que filtra as competências enquanto o utilizador introduz caracteres (conhecido por *filter as you type*) e (2) Paginação que permite que seja possível avançar pelas páginas das competências.

Número: 7
Título: Visualizar competências dos utilizadores
Descrição: O utilizador consulta as tags escolhidas pelos restantes utilizadores.
Racional: De modo a identificar quais as competências dos utilizadores, será possível em determinada área da aplicação visualizar os utilizadores que escolheram determinada competência. Isto é útil para, por exemplo, obter ajuda às pessoas indicadas sobre determinada competência.

Número: 8
Título: Impedir competências repetidas
Descrição: O sistema não permitirá elementos repetidos nas listas de competências
Racional: De modo a evitar competências repetidas na lista pessoal de competências, o sistema não poderá permitir duplicações, isto é, quando determinada competência for escolhida esta não poderá aparecer na lista de competências do sistema, nem nas sugestões.

4.4 Modelo de dados

O modelo de dados procura representar os modelos existentes na aplicação a desenvolver. Para este projeto existem 5 tabelas, descritas a seguir ao modelo de dados apresentado (Figura 20):

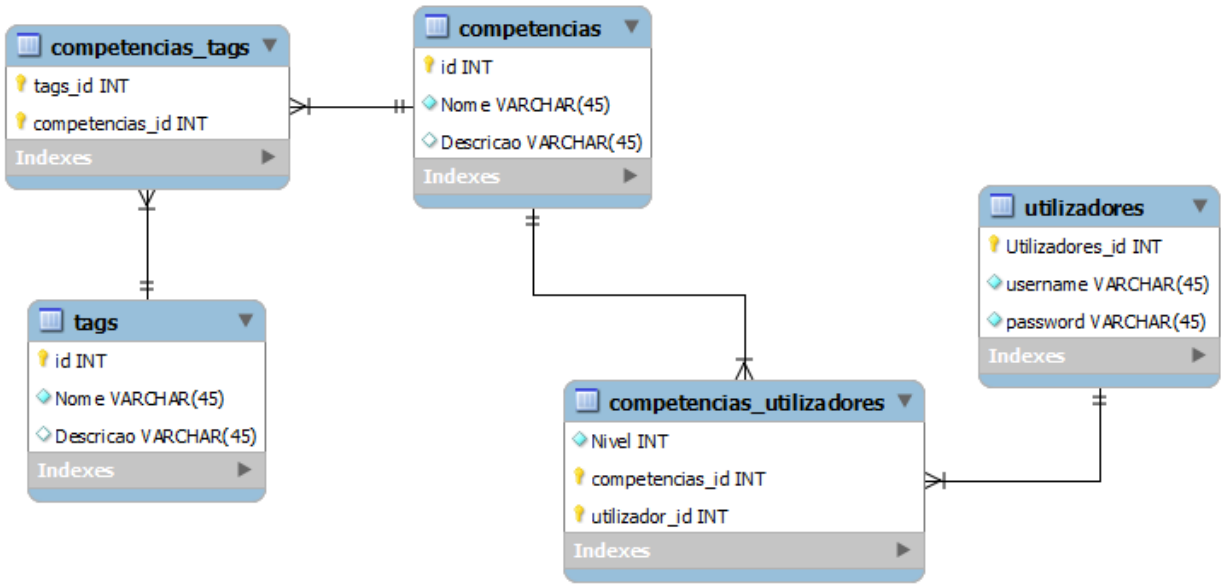


Figura 20 – Modelo de dados da aplicação

- A tabela 'competencias' contém informação sobre todas as competências.
- A tabela 'competencias_utilizadores' armazena as competências de cada utilizador.
- A tabela 'tags' armazena todas as tags do sistema
- A tabela 'utilizadores' permite identificar quem são os utilizadores da aplicação.
- A tabela 'competencias_tags' contém as relações entre competências e tags, possibilitando a sugestão de competências ao utilizador.

É importante referir que a passagem da informação proveniente do servidor até à aplicação cliente será feita através de estruturas em JSON que são propícias para este tipo de frameworks.

4.5 Arquitetura

Este tipo de aplicações tipicamente requer um tipo de arquitetura dividido em 2 partes. O lado do servidor que trata de fornecer ao cliente a informação relativa ao negócio, através por exemplo de uma API REST que é responsável pela persistência dos dados. Pelo lado do cliente, os dados relativos ao negócio são normalmente recebidos em JSON ou XML, e a respetiva framework trata de fornecer as funcionalidades pretendidas, desenhar a interface e manipular o DOM. A figura 21 permite entender a arquitetura típica de aplicações desenvolvidas com este tipo de frameworks

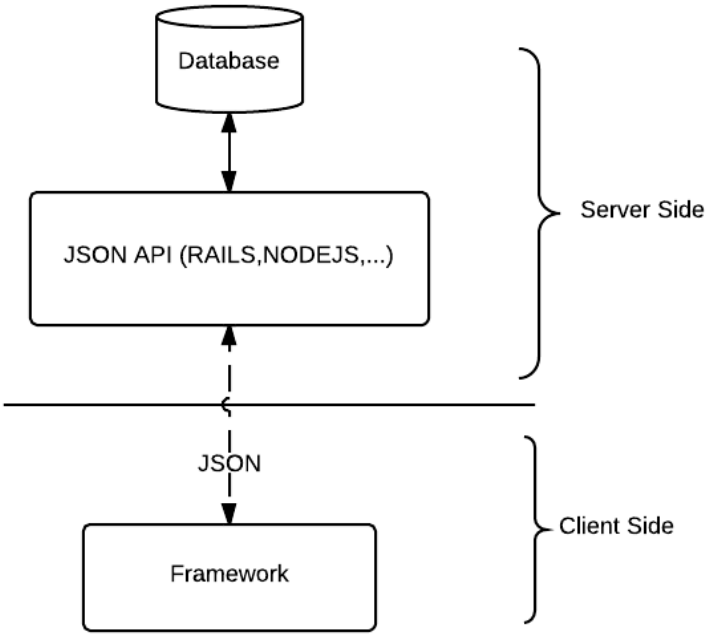


Figura 21 – Arquitetura típica de aplicações JavaScript

Neste caso de estudo e dado que o objetivo passa por aplicar funcionalidades HTML5 e estudar as frameworks JavaScript, a componente servidor deixa de ser necessária. A persistência de dados será desenvolvida do lado do cliente, usufruindo das capacidades do HTML5.

4.6 Mockups

De modo a especificar a interface da aplicação, foram criados vários mockups, que são apresentados de seguida. No momento em que o utilizador acede à aplicação, será abordado com uma janela modal

(janela filho que requer interação com o utilizador antes que este aceda à página mãe), que apenas permite ao utilizador efetuar o login, como é ilustrado na figura 22. A validação feita às credenciais introduzidas é nesta fase muito primitiva, pois apenas é verificado num JSON a existência dessas credenciais. Depois do login efetuado com sucesso, são carregados os dados do utilizador em questão. Caso as credenciais estejam erradas é apresentada uma *alert dialog* a indicar o erro de autenticação.

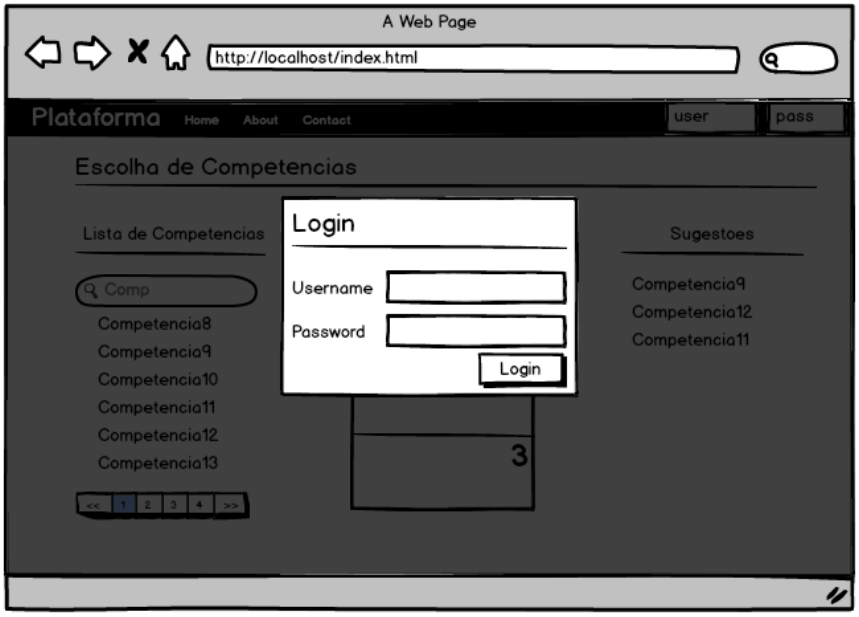


Figura 22 – Mockup de login

Quando as credenciais estão erradas, o utilizador é notificado; caso contrário, a janela modal é retirada e são carregadas as 3 listas de competências:

- Lista de todas as competências existentes no sistema
- Lista das competências pessoais do utilizador (dividida em vários níveis)
- Lista de sugestões baseada nas competências anteriormente escolhidas pelo utilizador

A Figura 23 exibe a interface inicial da aplicação em que podemos constatar a existências de 3 listas de competências. Do lado esquerdo está apresentada a lista de competências existente no sistema, e no centro da página é apresentada a lista de competências pessoais, dividida em 3 níveis. Por último, à

direita, é apresentada a lista de sugestões. Esta última é modificada quando um elemento é adicionado à lista de competências pessoais. A Figura 24Figura 25 permite identificar as ações que o utilizador pode efetuar nessa mesma interface.

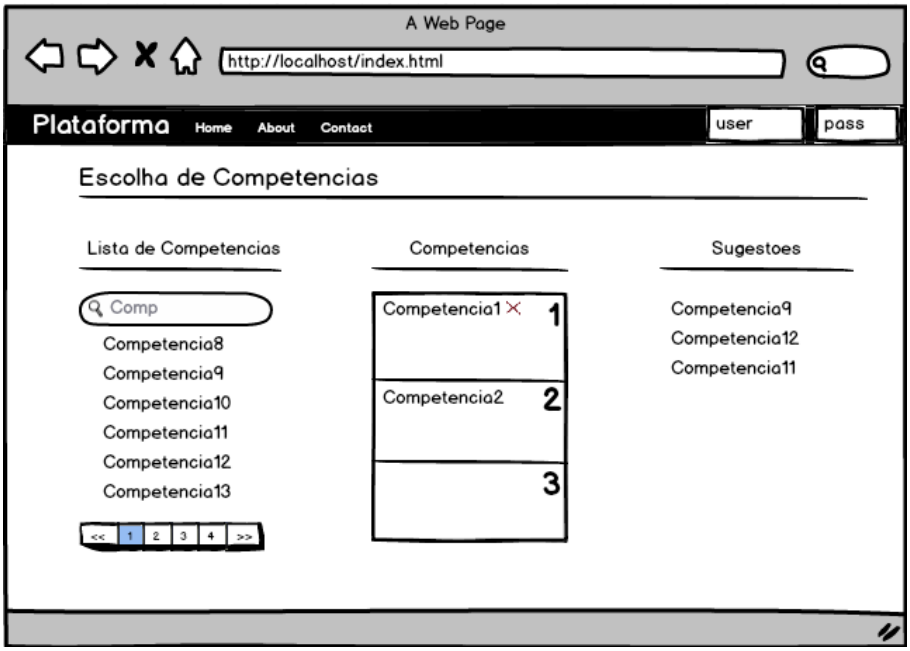


Figura 23 – Mockup da página inicial

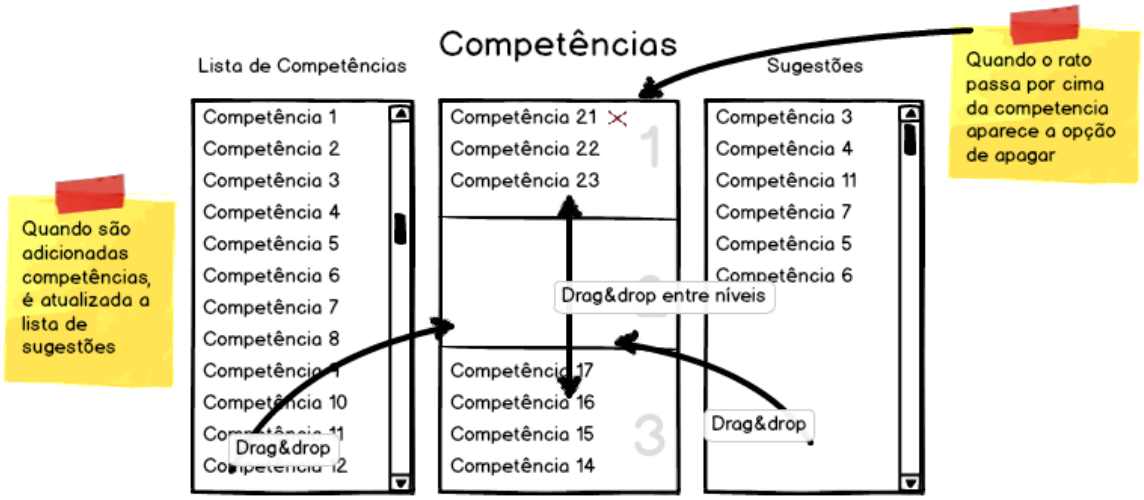


Figura 24 – Interações possíveis entre as listas existentes

A lista de sugestões da aplicação (coluna da esquerda) poderá ser extensa, dificultando assim a tarefa ao utilizador de escolher competências que ele entende que possui. De modo a respeitar os requisitos foram introduzidos os 2 controlos que facilitam a tarefa de pesquisa de sugestões (como se pode confirmar na Figura 23), em que no topo da coluna existe um *input* que filtra as competências enquanto o utilizador introduz os caracteres. Na parte inferior da coluna foi colocada paginação para facilitar a consulta das competências.

Como se pode constatar na figura 24, as competências são arrastáveis para qualquer um dos níveis. Depois de largados acontecem um conjunto de eventos descritos de seguida:

- É adicionado um registo na lista de competencias_utilizadores, com o nível e competência desejados.
- É removido o elemento na lista de competências de modo a não possibilitar competências repetidas.
- É gerada a lista de sugestões (i.e. são apresentadas as competências ainda não escolhidas que têm tags idênticas com as existentes na lista pessoal)

É importante referir que estes eventos são transparentes ao utilizador e quase instantâneos, já que todo o processo é executado do lado do cliente. Podem ainda ser alterados os níveis das competências existentes na lista de competências do utilizador. Internamente é feito uma edição ao registo com o nível escolhido pelo utilizador.

O mockup seguinte pretende demonstrar a animação que permitirá ao utilizador ter perceção de quais as sugestões que aparecem ou desaparecem depois de adicionada uma competência pessoal. A ideia é de quando a lista de sugestões é alterada o utilizador ter noção de quais as competências sofreram alterações. A solução passa pelas novas sugestões surgirem da esquerda para a direita e ao contrário caso as competências sejam removidas da lista de sugestões. A funcionalidade foi proposta para testar as frameworks e verificar qual a sua flexibilidade no que toca a funcionalidades que dizem respeito a questões de design (linguagem CSS) e animação.

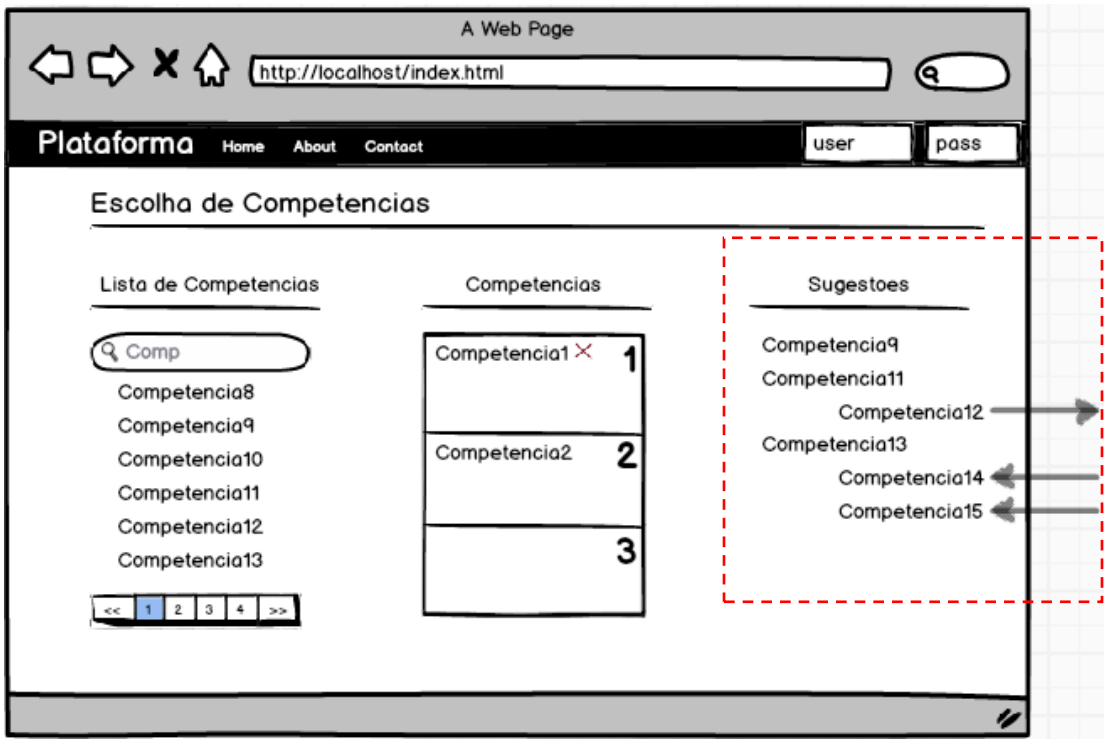


Figura 25 – Animação quando existem alterações nas sugestões

De modo a abordar os restantes requisitos, é criada uma nova "área" HTML, que permite que seja abordada a componente de roteamento das frameworks, isto é, sem sair da página HTML (senão seria necessário carregar outra vez os ficheiros JavaScript da aplicação) outro tipo de informação será apresentada, tal como por exemplo no GMail em que o utilizador nunca sai da página mas o conteúdo é alterado dinamicamente.

De modo a cumprir os requisitos, escolheu-se apresentar as tags numa forma de nuvem que é composta pelas tags escolhidas por **todos** os utilizadores, e em que são realçadas as tags mais escolhidas. Caso seja escolhida determinada tag, no lado direito da página são apresentados os utilizadores que escolheram a tag selecionada, como se pode constatar na figura 26. Esta funcionalidade permite manipular os modelos 'utilizador' e 'tag', e as respetivas relações entre as tabelas.

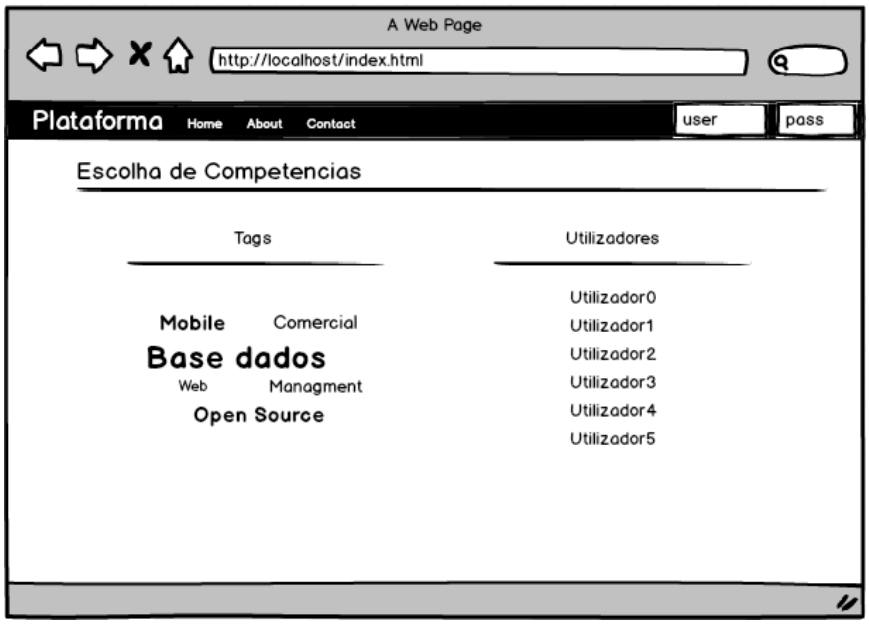


Figura 26 – Área que permite visualizar as competências escolhidas por todos os utilizadores

4.7 Bibliotecas utilizadas

Foi utilizado um conjunto de técnicas e ferramentas para que certas características da aplicação fossem iguais entre as frameworks testadas, isto é, o que diferencia as frameworks é a aplicação dos conceitos/funcionalidades das mesma e não, por exemplo, estilos. Isto permite ter uma análise mais independente, sem existirem elementos que perturbem a análise das frameworks.

4.7.1 Bootstrap

O Bootstrap é uma framework desenvolvida pelo Twitter para a criação de *front-ends* de uma forma simples e rápida. Esta framework é constituída por um conjunto específico de ficheiros CSS, HTML e JavaScript que utilizam uma série de padrões e/ou convenções para formulários, botões, tabelas, etc. Esta framework, permite que um programador web construa *front-ends* sofisticados com pouco esforço, podendo dar mais atenção a questões de usabilidade e *look&feel*, questões estas que, nem todos os programadores dominam.

4.7.2 Underscore

O Underscore é uma biblioteca JavaScript que disponibiliza métodos que facilitam certas operações em listas, tal como pesquisa, interseções, remoção de elementos por atributos, ordenação, intersecções, etc.

Esta biblioteca torna-se útil em vários casos, em que se pode reduzir várias linhas de código para apenas uma chamada à biblioteca.

4.7.3 JQuery

Como apresentado no capítulo de contexto, esta biblioteca é indispensável no que toca a desenvolvimento web do lado do cliente, e até é requisito obrigatório para algumas frameworks em análise, daí ser escolhida para facilitar o desenvolvimento da aplicação.

5 Resultados do desenvolvimento

O desenvolvimento do caso de estudo permitiu obter conclusões sobre as tecnologias em estudo na dissertação. Estes resultados estão divididos em 2 partes; a primeira é relativa ao uso da tecnologia HTML5 e às suas vantagens, e a segunda parte é relativa às frameworks JavaScript em estudo indicando quais as vantagens de cada uma no desenvolvimento de aplicações web.

5.1 Resultados HTML5

A aplicação desenvolvida possui funcionalidades que permitiram implementar várias componentes da especificação HTML5, tanto disponibilizadas ao utilizador ou transparentes a este. A equipa especificou que a escolha de competências deverá ser feita através da funcionalidade *drag & drop*, ou seja, as competências serão arrastadas e largadas para as áreas destinadas a receber estes elementos. O uso do *drag & drop* permite avaliar a flexibilidade das frameworks, quando sujeitadas a paradigmas de utilização alternativos (ao contrario de típicas hiperligações HTML). Esta funcionalidade *drag & drop*, que se tornou nativa com o HTML5, evitou a inclusão de mais uma biblioteca na aplicação web, que torna a aplicação menos lenta a carregar e reduz a quantidade de dependências. No caso de estudo desenvolvido, as competências podem ser arrastáveis para áreas próprias (figura 27) e ao serem largadas são acionados eventos definidos pelo programador.

Figura 27 – Drag & drop nativo na aplicação

Como a especificação do HTML5 indica (capítulo 3.1.7), qualquer elemento pode ser arrastável, portanto no elemento *label* (que contém o nome da competência) foram definidos os atributos *draggable* e

ondragstart (este último permite definir o método a ser chamado quando o arrastamento inicia). No elemento destino, neste caso uma *div,* são definidos os atributos *ondragover* e *ondrop* (este último define o método a ser chamado quando o arrastamento acaba). De seguida é apresentado o código HTML e JavaScript que permitiu implementar a funcionalidade descrita anteriormente.

Código HTML:

```
<label id="{{competencia.id}}" draggable="true"
ondragstart="drag(event)">{{competencia.nome}}</label>


<div id="competenciasNivel" ondrop="drop(event)"
ondragover="allowDrop(event)">
```

Código JavaScript:

```
function drag(ev) {
    ev.dataTransfer.setData("Text", ev.target.id);
}


function allowDrop(ev) {
    ev.preventDefault();
}


function drop(ev, nivelDestino) {
   ev.preventDefault();
   var idCompetencia = ev.dataTransfer.getData("Text");
   scope.addCompetenciaUtilizador(idCompetencia, nivelDestino);
   scope.updateListaSugestoes();
}
```

Foram introduzidos, no formulário de login, os atributos que validam os inputs do utilizador, isto é, foi declarado o atributo *required* que obriga a inserção de dados e foi também aplicado o atributo *type* no campo *password,* que permite que os caracteres não sejam apresentados quando estes são introduzidos no controlo, como se pode demonstrar nas figuras 38 e 39.

Username:

```
<input type="text" required />
```

Password:

```
<input type="password" required />
```

Figura 28 – Excerto do formulário que contem os atributos HTML5

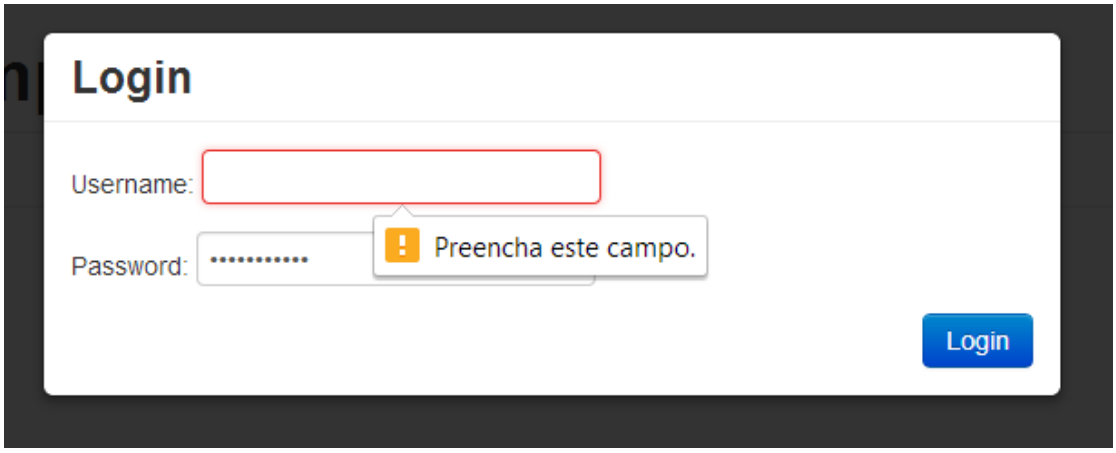


Figura 29 – Formulário HTML5

Dada que toda a lógica de negócio e interface do utilizador é processada pelo navegador, constatou-se que não existia a necessidade de configurar um servidor, em que este seria apenas destinado à persistência de dados. Por isso foi adaptada uma aproximação local para tratar desta questão, sendo a funcionalidade *localStorage* a mais adequada e transparente ao utilizador. Sendo o objetivo que quando determinado utilizador escolhesse uma competência, o respetivo JSON fosse gravado no navegador através das funções *setvalue* e *getvalue,* dispensando assim, para a fase de desenvolvimento, a configuração de servidores. Pode-se concluir que a aplicação é totalmente *client-side*, e dispensa de qualquer servidor, em que os dados armazenados apenas estarão disponíveis no navegador do utilizador em questão.

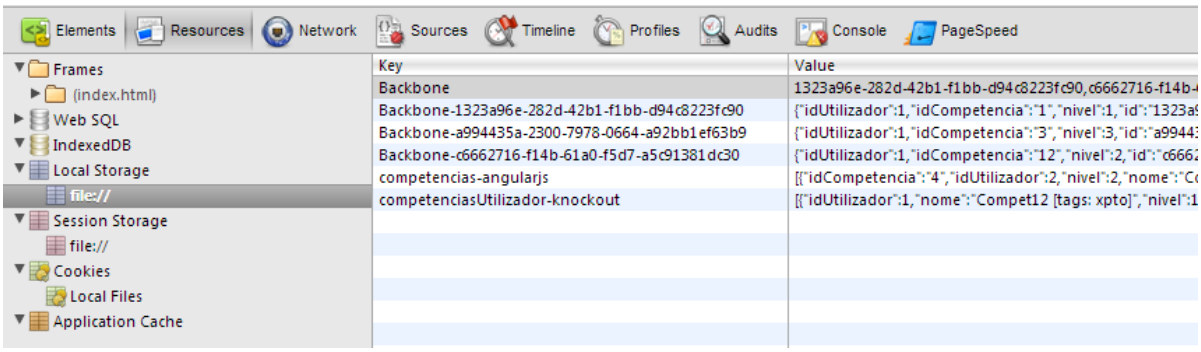


Figura 30 – Exemplo de informação armazenada no navegador Google Chrome

De uma maneira mais discreta, mas que permite uma boa organização do código HTML é a introdução das tags semânticas HTML5, que permitiram estruturar de uma maneira mais correta o HTML de cada lista de competência (Figura 31). A diferença ao utilizar estas tag semânticas situa-se ao nível do programador já que para o utilizador final esta funcionalidade é transparente. Ao programador fornece um código HTML muito mais legível, o que se torna útil ao longo do desenvolvimento.

Figura 31 – Esquema que representa a estruturação do código HTML das listas de competências

As funcionalidades apresentadas anteriormente permitiram que o código produzido seja mais legível, evitou configuração de servidores, e por fim evitou-se a importação de mais uma extensão, adotando o drag & drop nativo. A tabela abaixo apresenta o suporte destas funcionalidades HTML5 nas diferentes versões dos navegadores mais utilizados do mercado.

Tabela 5 – Suporte das funcionalidades HTML5 nos diferentes navegadores

	IE	Firefox	Chrome	Safari	Opera
Drag & Drop	8.0+	22.0+	28.0+	5.1+	16.0+
LocalStorage	8.0+	22.0+	28.0+	5.1+	16.0+
Formulário	10.0+	22.0+	28.0+	5.1+	16.0+
Tags semânticas	9.0+	22.0+	28.0+	5.1+	16.0+

5.2 Resultados frameworks JavaScript

O desenvolvimento do caso de estudo permite concluir como as frameworks funcionam e quais as suas limitações, pretendendo-se assim apresentar as vantagens e obstáculos identificados ao longo do desenvolvimento. Com um conjunto de critérios será feita a classificação de cada uma das frameworks de modo a conseguir diferenciar cada uma e dar a possibilidade ao leitor a escolha da que deseja consoante os critérios apresentados na tabela de comparação.

5.2.1 Fase 1

Neste capítulo, as frameworks irão ser abordadas individualmente para identificar as suas vantagens e desvantagens quando aplicadas num produto de software. É necessário relembrar que para avaliar as frameworks e de modo a que o trabalho não fosse de extrema extensão e moroso apenas os requisitos 1,2,4 e 8 foram implementados num conjunto especifico de frameworks de modo a identificar as suas características base. As frameworks escolhidas para o desenvolvimento tiveram como base a lista de frameworks estudadas no capítulo de análise, elegendo assim as que possuíssem uma comunidade minimamente ativa e que existisse um conjunto de boas práticas e tutoriais disponíveis de modo a iniciar o desenvolvimento. A lista resultou no seguinte conjunto de nove frameworks:

- Agility
- Angular
- Backbone
- Batman
- Ember
- JavaScriptMVC
- Knockout
- Sammy
- Spine

Depois de executada esta fase foram escolhidas as 4 melhores frameworks em que as restantes funcionalidades foram desenvolvidas de modo a tirar conclusões mais claras e precisas sobre o modo de funcionamento das mesmas.

5.2.1.1 Agility

Esta framework tem um tamanho reduzido (4kb - licença MIT) e é dependente da biblioteca jQuery. Tem por objetivo de possibilitar o desenvolvimento rápido de aplicações web, enquanto mantém o equilíbrio através do padrão MVC. Tem uma pequena comunidade e a documentação poderia ter melhor qualidade. Revelou possuir uma arquitetura estritamente MVC de maneira a manter a pilha de processamento simples, ou seja, o *core* é mais pequeno oferecendo menos funcionalidades ao programador de modo a evitar processamento extra criado por outros componentes.

Numa fase inicial do desenvolvimento da aplicação, a página oficial da framework tornou-se útil para entender os conceitos e boas práticas da framework, a API e os seus conceitos estão bem descritos e bem exemplificados o que facilita a sua aprendizagem. Ao longo do desenvolvimento tornou-se complexo obter ajuda na comunidade já que esta é pequena.

Concluindo, a framework Agility.js revelou-se pouco madura e com uma pequena comunidade. Esta é das frameworks apresentadas a mais recente, mas a sua página oficial tornou-se uma mais-valia na sua aprendizagem.

5.2.1.2 Angular

A framework Angular é desenvolvida pela Google (77kb – licença MIT), o que transmite confiança na evolução da framework a maneira como esta foi desenhada. A página oficial da framework é relevante para obter qualquer tipo de informação, em que são explicados todos os conceitos através de vídeos, exemplos e guias de desenvolvimento, etc. Tornou-se o melhor recurso para o desenvolvimento da aplicação.

É relativamente recente (a versão 1.0.0 foi lançada em Junho de 2012) mas apresenta uma maturidade elevada. A sua comunidade é considerável, embora com este caso prático não tenha sido necessário recorrer com frequência à mesma. É importante referir que está incluída na framework uma componente de testes, mas que não foi aplicada no caso de estudo. Como ponto positivo da framework, o código final tem uma apresentação limpa e clara, muito devido ao seu nível elevado de abstração, ao contrário por exemplo do Backbone.

É composta por uma arquitetura MVC típica com roteamento e persistência de dados incluída. Tem como grande extensão o *angular-seed* que permite criar testes unitários, criar árvore de ficheiros, entre outras funcionalidades. Tem como ferramentas disponíveis, uma extensão para o Chrome que permite explorar os modelos enquanto é feito o depuramento da aplicação, e outra extensão que facilita o uso da framework de teste *jasmine*.

5.2.1.3 Backbone

O Backbone.js é a framework JavaScript MVC mais conhecida no mercado, possui um tamanho reduzido (6.3kb - licença MIT), existindo por isso um vasto leque de aplicações web que usufruem desta framework. Existe uma comunidade enorme (boa documentação, elevado auxílio na internet, etc.), que revela ser o seu ponto mais forte. Na fase inicial da implementação do caso de estudo foi necessária alguma elasticidade na procura de boas práticas, porque a página oficial da framework não fornece tutoriais, isto é, apenas é descrita a API (com exemplos). Mas ao procurar soluções na internet, o programador depara-se com excelentes vídeos, tutoriais, soluções, etc.

Tem a particularidade de ser adaptável, isto é, pode-se recorrer a extensões feitas pela comunidade (existentes em número elevado) para satisfazer determinada necessidade. Por exemplo, neste projeto foi adicionada uma extensão que permitisse a relação *many-to-many* entre os modelos *tag* e *competências* (a framework só tem de raiz as relações um para um e um para muitos).

O Backbone tem dependência com a biblioteca Underscore.js, esta biblioteca possui utilitários e funções JavaScript que facilitam a operações em listas, para manipulação de DOM é aconselhável incluir a biblioteca jQuery. Tem como conceito base a *collection*, que é essencialmente uma lista de modelos, em que é possível atribuir eventos, para que esta seja notificada de alguma mudança. Com a atual arquitetura da framework, que é tipicamente MVC, é necessário que o programador escreva mais código padrão, tornando-a de mais baixo nível quando comparado com Ember, Angular ou Knockout.

Concluindo o Backbone é uma framework altamente flexível, e até recomendável a sua incorporação com outras bibliotecas JavaScript (como por exemplo para efetuar pedidos ao servidor). A sua grande vantagem é a sua extensa comunidade, que disponibiliza exemplos/tutoriais e que facilita a resolução de problemas.

5.2.1.4 Batman

Framework desenvolvida em CoffeeScript (linguagem que tem como base o JavaScript), mas em que pode ser utilizada a linguagem JavaScript para desenvolver as aplicações web. A framework é recomendada se a equipa de desenvolvimento entender a linguagem CoffeeScript para dar continuidade à filosofia da framework e para poder tirar maior proveito das potencialidades desta. A documentação é extensa mas existe falta de tutoriais na página oficial da framework. Muito utilizada em programadores que trabalhem com *Ruby on Rails* (RoR) e CoffeeScript, possui uma arquitetura tipicamente MVC (persistência de dados e mapeamento de URLs incluída).

Desenvolvendo o caso de estudo com a framework Batman.js, a aplicação não foi implementada com sucesso. Na fase inicial do desenvolvimento, rapidamente se constatou que a maioria da documentação e tutoriais estavam escritos em CoffeeScript, e concluiu-se que não seria trivial o desenvolvimento da aplicação aplicando este tipo de linguagem. Optou-se então por cessar o desenvolvimento por existir um grande obstáculo na linguagem utilizada.

Pôde-se concluir que esta framework está focada para programadores CoffeeScript em que a sua comunidade está apenas focada neste tipo de linguagem.

5.2.1.5 Ember

O Ember.js oferece uma grande variedade de funcionalidades, o que resulta no seu tamanho elevado (47kb - licença MIT). Tem uma comunidade e documentação úteis, mas mais pequenas quando comparada com frameworks mais maduras, como o Backbone. A framework possui várias funcionalidades incorporadas, o que permite ao programador escrever menos código base ao contrário do Backbone, mas que torna a aplicação menos controlada pelo programador. O Ember.js não requer outras extensões por ser uma framework que por si só oferece várias funcionalidades.

Ao longo do desenvolvimento constatou-se que a framework de desenvolvimento Ember.js, possui conceitos muito próprios, que fazem o tempo de aprendizagem ser maior quando comparado com frameworks com conceitos mais óbvios e simples. No que respeita ao padrão MVC, a frameworks contém os 3 componentes típicos do padrão. A vista contém o código HTML e os respetivos eventos, em que qualquer ligação feita a partir da vista irá acionar um método do controlador que processará o pedido para

alterar o modelo e/ou a vista. Tem como componente principal o roteador, que permite mapear os URLs feitos à aplicação com os respetivos métodos do controlador.

Tendo uma filosofia própria, notou-se que a documentação poderia estar mais completa, isto é, existem falta de tutoriais na página oficial que permitam conhecer as boas práticas da framework e existem módulos mal documentados. A alternativa passou por consultar a comunidade em que se encontrou mais apoio e tutoriais que permitiram o desenvolvimento da aplicação.

A implementação do caso de estudo foi mais complexa do que em frameworks mais simples, mas em contrapartida o código final tem um aspeto relativamente simples e com menos linhas de código quando comparado com as restantes frameworks em análise.

5.2.1.6 JavascriptMVC

Das frameworks apresentadas neste documento, esta é provavelmente a mais complexa. Tem um tamanho elevado (50kb) porque inclui uma serie de componentes que permitem estruturar o código, gerar automaticamente partes de código, testar a aplicação, gerar documentação, entre outros. Estas extensões acabam por tornar complexa aprendizagem da framework. Tem uma filosofia diferente das restantes e estrutura de ficheiros própria. A página oficial tem toda a documentação necessária o que permitiu que a fase inicial do desenvolvimento da aplicação decorresse com normalidade.

A framework tem, tal como o Ember, muitas funcionalidades disponíveis que possibilitam realizar tarefas complexas. Existem módulos incorporados que permitem a criação de testes unitários e criação de documentação do código, por isso, é recomendada para projetos grandes onde a componente de testes e documentação da API são um fator preferencial. Estes módulos não foram testados por não ser, nesta fase do projeto, um critério determinante na escolha das frameworks.

Esta é a mais antiga das frameworks apresentadas, tendo por isso uma comunidade sólida e fazendo com que a framework esteja num nível de maturidade aceitável. É recomendada para projetos grandes onde a componente de testes e documentação da API são um fator preferencial. De notar que estes componentes adicionados podem ser excluídos ou aproveitados noutras frameworks.

5.2.1.7 Knockout

O Knockout.js tem um tamanho razoável (14.3kb - licença MIT), uma boa comunidade e uma documentação extensa, tem como ponto forte a sua página oficial. A página indica boas práticas e abordagens de como se devem desenvolver as aplicações, existe ainda um tutorial que permite em tempo real criar código para testar funcionalidades da framework. A sua documentação permitiu que a fase inicial do desenvolvimento da aplicação decorresse sem problemas.

O padrão aplicado nesta framework é Model-View-ViewModel (MVVM). Os modelos tal como as restantes frameworks são responsáveis pela logica de negócio, validação e persistência de dados. Os *view-model* são a camada responsável por dar funcionalidades à aplicação, pode corresponder ao controlador na arquitetura MVC tradicional. Por exemplo, um *view-model* pode conter métodos para adicionar, editar, e remover itens de uma lista. De notar que não tem incorporada a componentes de roteamento ou acesso a dados em servidores, a solução passa por utilizar bibliotecas externas.

Ao longo do desenvolvimento o knockout revelou ser uma framework bastante completa e de fácil compreensão o que resultou um rápido desenvolvimento. A aplicação final tem um código muito claro e simples de perceber, podendo assim concluir-se que a framework é simples e relativamente poderosa.

5.2.1.8 Sammy

A frameworks tem um pequeno tamanho (16kb – licença MIT) e é de fácil aprendizagem, fornecendo uma estrutura básica para desenvolver aplicações JavaScript. Tem uma documentação razoável tendo em conta a dimensão da framework e é dependente do jQuery. A arquitetura é composta por vistas, eventos e roteador, a componente de eventos permite separar a lógica das páginas HTML, e em que pode ser considerada a junção com outras frameworks (por exemplo Backbone para definição de modelos).

Aplicando a framework de desenvolvimento Sammy.js, a aplicação não foi implementada com sucesso. A ideologia desta framework revelou não ser a mais indicada e tornou-se pouco flexível para este tipo de aplicações. A filosofia consiste em atribuir eventos a URLs, isto é, enquanto que é pretendida em certas aplicações eventos serem acionados depois do utilizador carregar em botões ou arrastar elementos, com esta framework apenas são respondidos eventos quando são feitos novos pedidos HTTP à aplicação. Esta framework pode ser utilizada em aplicações web em que os urls seguem o conceito REST.

Consequentemente o caso de estudo não segue este tipo de convenções, o que resultou na não implementação completa do caso de estudo. Concluindo, esta é uma frameworks que revelou ser muito simples e pequena no que toca ao leque de funcionalidades desejadas. A sua página oficial descreve bem os conceitos da frameworks e tem como contra a sua pequena comunidade.

5.2.1.9 Spine

A framework Spine foi inspirada no Backbone, ou seja, procura ser tipicamente MVC e bastante flexível, possibilitando também a incorporação de extensões, tem uma considerável documentação e tamanho reduzido (7kb - licença MIT). A frameworks foi desenvolvida em CoffeeScript, mas em que pode ser aplicado o JavaScript para desenvolver aplicações. Na fase inicial do desenvolvimento, todo o conhecimento foi obtido através a página da framework que contem vídeos, tutoriais e descrição da API. É importante referir que a documentação pode ser consultada em JavaScript ou CoffeeScript, ao contrário da framework Batman que apenas tem documentação escrita em CoffeeScript. O grande objetivo da framework é de permitir alterações assíncronas aos dados para permitir uma interface sem travagens e mais fluida possível.

É necessário incluir no projeto (à escolha do programador) uma biblioteca para incorporação de dados no HTML, como por exemplo o *mustache* (mustache.github.com) ou *jQuery.tmpl* (github.com/jquery/jquery-tmpl). Este exemplo mostra perfeitamente a preocupação que a framework tem em manter o seu core o mais simples possível. No desenvolvimento em si da aplicação, o programador tem tendência para recorrer ao jQuery para implementar certas funcionalidades, o que permite concluir que existem poucas funcionalidades incorporadas na framework.

Tem o *Spine.app* como grande extensão, que permite construir a árvore de ficheiros e código base para qualquer aplicação, é recomendável o seu uso na página web da framework. Também fornece a extensão Spine mobile (2.6kb) para desenvolvimento de aplicações como destino dispositivos móveis. Em relação à arquitetura esta é tipicamente MVC, com roteador incluído. A comunidade é pequena e dividida em programadores CoffeeScript ou JavaScript, o que pode ser considerado um contratempo na procura de soluções.

5.2.1.10 Comparação

Para concluir esta primeira fase do projeto e tirar corretas conclusões sobre as diferenças entre as frameworks foi feito um resumo do estudo que possibilitasse a escolha das melhores frameworks. Para isso a tabela 8 pretende agrupar um conjunto de critérios para avaliar as frameworks e poder diferenciá-las. Nestes resultados já foram excluídas as frameworks Batman e Sammy, que foram as frameworks em que o caso de estudo não foi implementado com sucesso.

5.2.1.11 Critérios

Os critérios considerados para a avaliação da framework estão descritos em seguida, bem como a escala atribuída a cada um.

- Funcionalidades: Critério que define a quantidade de funcionalidades que estão incluídas na framework. O valor atribuído varia desde "1" em que existem as funcionalidades básicas para este tipo de aplicações MVC e REST (*binding* de dados, acesso a APIs, camadas MVC), até "4" em que estão incluídas várias funcionalidades úteis ao programador (como por exemplo componente de roteamento e integração com jQuery).
- Aprendizagem: Curva de aprendizagem da framework, isto é, avaliar se estudar e implementar os conceitos da framework foi um processo rápido ou lento. Os valores atribuídos têm o valor máximo de "4", que representa uma aprendizagem de cerca de um dia (aproximadamente 8 horas de trabalho), o que permite perceber os conceitos da framework e ter capacidade de desenvolver funcionalidades. O valor "1" representa uma aprendizagem de mais de 2 dias até entender os conceitos e desenvolver funcionalidades. Entre cada valor a diferença é de 0,5 dias de trabalho (aproximadamente 4 horas).
- Comunidade: Facilidade com que o programador se depara em encontrar soluções, vídeos, tutoriais, entre outros conteúdos. Em que "1" representa uma pequena comunidade, e "4" uma grande e prestável comunidade. Este critério é calculado através da soma do número de questões colocadas no *stackoverflow* (stackoverflow.com) e as estrelas atribuídas no *github* (github.com) sobre a framework em questão.

Tabela 6 – Classificação do critério comunidade

Comunidade	Valor correspondente
<1000	1
1.000-9.999	2
10.000-20.000	3
>20.000	4

- Nº linhas: Representa o número linhas de código fonte. É importante referir que o código é todo da mesma linguagem, está idêntico e normalizado em todas as frameworks comparadas de modo existir uma correta comparação. A tabela seguinte mostra a valor atribuído ao número aproximado de linhas de código da aplicação.

Tabela 7 - Valores atribuídos ao número de linhas de código da aplicação

Nº linhas	Valor correspondente
<100	4
100-124	3
125-150	2
>150	1

- Extensões: Suporte da framework a extensões quer implementados pela equipa que desenvolve a frameworks ou a comunidade em si. Em que "0" representa que não é possível adicionar extensões ou não existem; "1" a framework possibilita a inclusão de extensões mas estas são quase inexistentes. E o valor "2" que representa a facilidade de as incorporar e existe uma razoável quantidade disponível. Esta escala é mais pequena para dar uma menor importância quando comparado com os restantes critérios.

5.2.1.11.1 Resultados

A tabela seguinte mostra os valores atribuídos às frameworks seguindo os critérios definidos no ponto 5.2.1.11.

Tabela 8 – Pontuações atribuídas às frameworks segundo os critérios indicados

Frameworks	Funcionalidades	Aprendizagem	Comunidade	Nº linhas	Plugins
Agility	1	4	1	2	1
Angular	3	3	3	4	1
Backbone	2	4	4	2	2
Ember	4	2	3	3	1
JavascriptMVC	4	1	1	3	1
KnockOut	3	3	3	3	1
Spine	1	4	2	2	1

Pode-se constatar com a tabela anterior que as frameworks com maior número de funcionalidades têm uma aprendizagem mais lenta, dado que parte do tempo inicial de aprendizagem serve para interpretar e perceber os conceitos de cada framework.

Somando os valores atribuídos (Tabela 9), pode-se constatar que existem 2 grupos distintos de frameworks. As frameworks Backbone, Ember, Knockout e Angular situam-se no grupo das mais pontuadas, sendo as mais equilibradas considerando os critérios em questão.

As frameworks menos pontuadas foram a JavascriptMVC, muito por causa da sua aprendizagem lenta e a sua comunidade quase inexistente. As frameworks Agility e Spine obtiveram baixas pontuações por terem poucas funcionalidades e por possuírem uma pequena comunidade, quando comparadas com as restantes frameworks.

Tabela 9 - Somatório das pontuações atribuídas na tabela 8

Frameworks	TOTAL
Angular	14
Backbone	14
Ember	13
KnockOut	13
JavasCriptMVC	10
Spine	10
Agility	9

Chegou-se assim à conclusão de escolher as frameworks de desenvolvimento Angular, Backbone, Ember e Knockout para serem analisadas na próxima fase do projeto.

5.2.2 Fase 2

Para a próxima fase de desenvolvimento foram implementados os restantes requisitos da aplicação, possibilitando assim aprofundar os conhecimentos obtidos sobre as 4 mais bem classificadas frameworks da fase 1. Isto permite obter conclusões mais claras e precisas no que toca a certas quer questões técnicas, quer questões de mais alto nível. De seguida está descrito o resultado deste desenvolvimento e as respetivas conclusões.

5.2.2.1 Angular

A frameworks de desenvolvimento Angular, tal como a Ember, comporta-se mais como uma framework do que as anteriores apresentadas (Backbone e Knockout), porque são compostas por mais funcionalidades, têm conceitos e ecossistemas muito próprios, o que permite criar aplicações completas sem necessitar de recorrer a extensões extras, embora estas existam.

Ao longo deste estudo constatou-se que existiu um grande aumento de artigos e de adeptos à framework o que pode condicionar a sua escolha em projetos futuros, sendo a framework que obteve mais crescimento durante da realização da dissertação.

De referir de que esta framework teve o maior aumento de numero de linhas de código no desenvolvimento do caso de estudo em questão (cerca de 4 vezes maior), mas que continua a ser aquela que resulta num menor numero de linhas de código final. Concluindo, esta frameworks incorpora a pilha completa de componentes para criar aplicações web, e que pode ser considerada caso o desenvolvimento em questão seja de aplicações de complexidade elevada.

5.2.2.2 Backbone

Ao desenvolver as restantes funcionalidades com esta framework, constatou-se que é a de mais baixo nível quando comparada com as restantes desta fase do projeto, isto é, esta comporta-se de um modo mais idêntico a uma biblioteca do que com uma framework completa.

Uma particularidade encontrada é a facilidade de incorporar bibliotecas JavaScript na framework, ou seja, qualquer biblioteca que seja adicionada, à partida não existirão conflitos com as funcionalidades do Backbone e poderão ser usufruídas sem problemas, tornando-a numa framework altamente flexível.

O que se pode constatar com o aumento de funcionalidades desenvolvidas recorrendo ao uso desta framework, é que caso o programador não tenha cuidado em manter o seu código estruturado e caso seja uma aplicação com muito código *cliente-side* este poder-se-á tornar de leitura e interpretação complexa e confusa. O código final é constituído por 406 linhas de código (aumento 3.25 vezes superior ao código inicial)

5.2.2.3 Ember

A framework possui um ecossistema mais reservado, funcionando como uma *caixa negra,* tendo o programador por isso algumas dificuldades em incluir bibliotecas que não foram desenvolvidas com o propósito de serem aplicadas na frameworks Ember.

De notar que esta é provavelmente a framework menos madura das consideradas na segunda fase do estudo por ter o componente de acesso a dados (*ember data*) ainda em claros desenvolvimentos. Considerando apenas a 2ª fase do estudo, com esta frameworks obteve-se o maior número de linhas do código, muito devido à elevada complexidade de uso da framework.

5.2.2.4 Knockout

O knockout tornou-se uma framework que permite construir interfaces robustas com relativa simplicidade. De modo a implementar a totalidade das funcionalidades, nomeadamente o requisito número 7, foi necessário incluir a extensão Sammy.js (biblioteca indicada na página oficial da framework) que permitisse assim tornar a aplicação numa *single page aplication*. Esta biblioteca permite mapear URLs a funções desenvolvidas pelo programador, tornando-se assim na componente de roteamento da aplicação.

Concluindo, o Knockout é uma framework que se comporta mais como uma biblioteca que auxilia a criação de interfaces ricas e que pretendem responder da melhor maneira ao utilizador. O código resultante do desenvolvimento nesta segunda fase é composto por 377 linhas de código JavaScript o que faz um aumento de 3.22 vezes quando comparado com o número de linhas do código inicial.

5.2.2.5 Perspetivas

De modo a perceber em que situações determinadas frameworks são mais apropriadas, são explicados os pros e contras das frameworks em estudo na segunda fase do projeto, encarando de diferentes pontos de vista das entidades que desenvolvem aplicações web. Foram consideradas então 4 tipo de entidades que usualmente são intervenientes e partes interessadas no desenvolvimento de aplicações web. São elas:

- Designer – Entidade que tem como responsabilidade de desenvolver o aspeto visual da aplicação web, preocupa-se em existir uma boa interface, é importante para o designer existir separação das camadas CSS, HTML, JS.
- Tester – Entidade que testa a aplicação, é necessário ter em conta se existem condições ou obstáculos na frameworks para realizar testes.
- Projeto – Cada projeto tem o seu tempo e metodologia de desenvolvimento, podendo assim ter intervenção na escolha da framework.
- Programador – entidade responsável por desenvolver as funcionalidades pretendidas para a aplicação final. O programador tem que estar confortável e ter acesso a conteúdo que o permitem desenvolver código de maneira rápida, correta e que seja de fácil manutenção.

Designer

Existem dois métodos de efetuar o bind dos dados no HTML através do JavaScript. O primeiro passa por aplicar atributos às tags HTML para que a frameworks identifique estes atributos e possa substituir por informação no HTML. No excerto de código HTML seguinte (com uso da framework Knockout), o atributo *data-bind* da *div,* mostra que irá executar um ciclo para cada sugestão, e na *label* em si, irá ser feita a ligação do id e do nome na *label*. Das frameworks selecionadas na segunda fase apenas o Knockout e Angular permitem a utilização deste método.

```
<div id="sugestoesList" data-bind="foreach: { data: sugestoes } ">
     <label data-bind="attr: { id: id}, text: nome" draggable="true"
     ondragstart="drag(event)" ></label>
</div>
```

Para o caso do Ember e do Backbone é recorrido a *template engines*. Estes templates são scripts existentes nas páginas que contem identificadores especiais que são substituídos futuramente pela informação desejada, como demonstra o excerto de código seguinte.

```
<script type="text/template" id="competencias_template">
      <% competencias).each(function(competencia) { %>
            <label id="<%= competencia.id %>" draggable="true"
            ondragstart="drag(event)"> <%= competencia.nome %></label>
      <% }); %>
</script>
```

De um modo geral é muito mais agradável, quer para um designer ou programador ter a tag dentro do HTML que correspondem à logica de negócio, do que scripts, em que caso existam em número elevado tornam a página difícil de analisar.

Tester

De um ponto de vista de um *tester*, a framework Angular tem larga vantagem porque tem incluída de origem uma componente de testes que funciona em completa sintonia com as restantes funcionalidades disponibilizadas pela framework. Isto não significa que o código desenvolvido para as restantes framework não possa ser testado, mas tem que ser incorporada uma biblioteca especial de teste JavaScript, o que pode provocar alguns contratempos, nomeadamente numa frameworks como a Ember em que existe por vezes alguma dificuldade em conseguir incluir na perfeição este tipo de bibliotecas.

Projeto

Considerando a característica de ciclo de vida de um projeto, mais precisamente a metodologia adotada para desenvolver determinada aplicação web, a escolha da frameworks poderá ser decisiva. Por exemplo, um projeto em que a arquitetura e requisitos não sejam bem definidos na fase inicial, (adotando uma metodologia mais ágil como o SCRUM) a framework de desenvolvimento Ember poderá não ser a melhor escolha, por não se ter o conhecimento necessário se o ecossistema Ember irá corresponder às espectativas. Com uma metodologia mais rigorosa (como por exemplo o RUP) torna-se mais simples a escolha da framework, já que se pode constatar corretamente se os requisitos são coexistentes com os conceitos da framework. No caso de uma metodologia mais ágil, o facilitismo de incorporar bibliotecas para realizar determinadas tarefas pode ser preponderante na escolha das framework.

Caso inicialmente seja necessária uma frameworks que contem a pilha completa de desenvolvimento e cujas funcionalidades da aplicação final será principalmente o CRUD de dados através de uma API REST, a framework Ember poderá ser a melhor escolha.

Programador

Cada programador tem a sua preferência no que toca a filosofia de tecnologias ou frameworks. Como já foi referido existem frameworks mais simples de rápida compreensão e outra mais complexas de lenta compreensão mas mais capaz de responder a necessidades complexas, cabe por isso, à entidade responsável do projeto ter em consideração a sua equipa de desenvolvimento, já que será este que será responsável por desenvolver um produto funcional ao utilizador/cliente.

6 Conclusões

Este capítulo que pretende resumir o trabalho efetuado ao longo do projeto, referenciando os principais objetivos alcançados e o trabalho futuro a realizar.

6.1 Síntese do trabalho

Este trabalho foi desenvolvido com o objetivo de oferecer uma base de conhecimento que permita à PT Inovação desenvolver aplicações web que sejam modernas em termos tecnológicos e arquiteturais.

De modo a adquirir esse conhecimento, foi efetuado um estudo que permite conhecer as funcionalidades HTML5 e as frameworks JavaScript existentes, permitindo assim que se desenvolva aplicações web que suportem as várias tendências, arquiteturas e dispositivos (desktop e móvel) existentes, de modo a acompanhar a evolução das aplicações web.

No estudo do HTML5, conclui-se que a nova versão da linguagem tem várias vantagens tanto para quem utiliza as aplicações, como para quem as desenvolve. Permite ao utilizador reduzir a quantidade de extensões instaladas no computador, o que melhora a experiencia de utilização das aplicações (principalmente em dispositivos móveis). Permite simplificar o desenvolvimento das aplicações de vários modos, tal como tornar o código fonte mais legível, facilitar a validação dos formulários, facilitar a criação de animações/jogos e aproveitar uso de funcionalidades nativas (localização e drag & drop) de modo a evitar a constante inclusão de extensões de desenvolvimento.

Como foi referido ao longo da dissertação, com o aumento do uso destas tecnologias que são executadas no navegador apareceram as frameworks JavaScript. O estudo realizado a este respeito permitiu inicialmente identificar as vantagens e qual a filosofia em que estas frameworks se inserem, resultando em novas arquiteturas e filosofias. Este estudo foi dividido em duas fases descritas de seguida:

- Na primeira fase foram estudadas as várias frameworks existentes na web, já que existe alguma dificuldade, para as entidades que desenvolvem aplicações com este tipo de frameworks, escolher qual a melhor. Com base num conjunto de critérios, chegou-se à conclusão que existem 4 frameworks maduras e completas (Angular, Backbone, Ember e Knockout) para que o desenvolvimento de aplicações web baseadas nestas arquiteturas seja bem sucecido.

- Na segunda fase foram desenvolvidas as restantes funcionalidades, recorrendo às frameworks mais bem classificadas na fase 1, e conclui-se que a escolha da framework depende da aplicação a desenvolver e dos respetivos recursos envolventes ao projeto.

Cabe assim a quem pretender utilizar estas novas tecnologias analisá-las e identificar quais as funcionalidades, já que disponibilizam funcionalidades de permitem cobrir as várias partes do desenvolvimento de aplicações web.

6.2 Trabalho futuro

É possível enriquecer o trabalho já realizado, com linhas de atuação diferentes. No caso do HTML5, algumas funcionalidades não foram implementadas no caso de estudo, já que esta tecnologia é de âmbito alargado e algumas funcionalidades não se inseriam no caso de estudo a desenvolver. Esta nova implementação permitiria ao leitor ter uma ideia mais concreta de algumas funcionalidades HTML5 e de qual o seu real propósito (como por exemplo o vídeo e áudio HTML5). Assim sendo, seria possível numa fase futura aplicar as restantes funcionalidades, de modo a apresentar ao leitor mais factos de como o HTML5 é vantajoso.

Bibliografia

Armeli-Battana, S. (2013). *MVC Applies to JavaScript.* Developer.Press.

Bell, P. (12 de Junho de 2012). A survey of client MVC frameworks. IBM. Obtido de http://www.ibm.com/developerworks/web/library/wa-clientmvc/wa-clientmvc-pdf.pdf

Brad Green, S. S. (2013). *AngularJS.* O'Reilly Media.

Bruce Lawson, R. S. (2011). *Introducing HTML5 (Voices That Matter).* New Riders.

Garrett, J. J. (18 de Fevereiro de 2005). Ajax: A New Approach to Web Applications.

Hempton, G. L. (13 de Janeiro de 2012). *The Top 10 Javascript MVC Frameworks Reviewed*. Obtido de codebrief: http://codebrief.com/2012/01/the-top-10-javascript-mvc-frameworks-reviewed/

Hickson, I. (2013). *HTML Standard*. Obtido de whatwg: http://www.whatwg.org/specs/web-apps/current-work/#introduction

Jobs, S. (Abril de 2010). *Thoughts on Flash*. Obtido de http://www.apple.com/hotnews/thoughts-on-flash/

Larsen, R. (Abril de 2011). *HTML5, CSS3, and related technologies*. Obtido de http://www.ibm.com/developerworks/library/wa-webstandards/index.html

Longman, A. W. (1998). *A history of HTML*. Obtido de W3C: http://www.w3.org/People/Raggett/book4/ch02.html

Lubbers, P. (2010). *Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development.* Apress.

Lubbers, P. (2011). *HTML5 Programming* (2 ed.). Apress.

McKeachie, C. (19 de Setembro de 2013). *Choosing a JavaScript MVC Framework*. Obtido de http://www.funnyant.com/choosing-javascript-mvc-framework/

Osmani, A. (27 de Julho de 2012). *Journey Through The JavaScript MVC Jungle*. Obtido de smashingmagazine: http://coding.smashingmagazine.com/2012/07/27/journey-through-the-javascript-mvc-jungle/

Osmani, A. (2013). *Developing Backbone.js Applications.* O'Reilly Media.

Pieters, S. (2012). *html4-differences*. Obtido de http://dev.w3.org/html5/html4-differences/

Pilgrim, M. (2010). *HTML5: Up and Running.* O'Reilly Media.

Sanderson, S. (1 de Agosto de 2012). *Rich JavaScript Applications – the Seven Frameworks* . Obtido de http://blog.stevensanderson.com/2012/08/01/rich-javascript-applications-the-seven-frameworks-throne-of-js-2012/

Simpson, K. (2013). *JavaScript and HTML5 Now.* O'Reilly Media.

techcrunch. (11 de Setembro de 2012). *Mark Zuckerberg: Our Biggest Mistake Was Betting Too Much On HTML5*. Obtido de techcrunch: http://techcrunch.com/2012/09/11/mark-zuckerberg-our-biggest-mistake-with-mobile-was-betting-too-much-on-html5/

W3C. (Maio de 1998). *Shaping the Future of HTML*. Obtido de http://www.w3.org/MarkUp/future/#summary

W3C. (2012). *HTML 5.1*. Obtido de http://www.w3.org/TR/html51/

W3C. (October de 2013). *Geolocation API Specification*. (A. Popescu, Editor) Obtido de w3: http://www.w3.org/TR/geolocation-API/#privacy_for_uas

Walls, C. (18 de Outrubro de 2012). Client-Side UI Smackdown, SpringOne 2GX. Washington, D.C. Obtido de http://www.infoq.com/presentations/JavaScript-Frameworks-Review

Way, J. (2012). *Decoding HTML5.* Rockable Press.