Projeto de Mestrado em Engenharia Informática – Computação Móvel

AnyKB: Aplicação Web de Base de Conhecimento Focada em Tecnologias HTML Emergentes

Telmo Filipe Moreira Marques

Leiria, setembro de 2013

Projeto de Mestrado em Engenharia Informática – Computação Móvel

AnyKB: Aplicação Web de Base de Conhecimento Focada em Tecnologias HTML Emergentes

Telmo Filipe Moreira Marques

Projeto de Mestrado realizado sob a orientação do Professor Doutor Patrício Domingues, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, setembro de 2013

Agradecimentos

Agradeço em primeiro lugar ao Instituto Politécnico de Leiria e à Escola Superior de Tecnologia e Gestão de Leiria por disponibilizar as condições de estudo e investigação necessárias ao desenvolvimento deste projeto.

Agradeço em especial ao Professor Doutor Patrício Domingues por todo o apoio e motivação ao longo do meu percurso como estudante, e em particular pela orientação científica deste projeto. Agradeço também a todos os docentes do Departamento de Engenharia Informática, em especial aos docentes com quem tive a oportunidade de interagir diretamente.

Agradeço ao colega Celso Santos pelo apoio disponibilizado na conceção de elementos visuais do projeto, à colega Rita Silva e Mestre Lee Reis pelo acompanhamento académico e científico, e a todos os colegas que se disponibilizaram para a realização dos testes.

Agradeço à empresa Webtuga, Lda. pela pronta disponibilização da toda a infraestrutura que atualmente suporta tecnologicamente este projeto.

Agradeço por fim ao meu pai, mãe e irmã pelo apoio incondicional em todas as escolhas pessoais e profissionais, e à minha namorada pelo apoio emocional e paciência ao longo de todo este percurso.

Resumo

A *web* é atualmente a plataforma baseada em tecnologias abertas mais popular e mais acessível a nível mundial, através da Internet. Neste contexto, o *browser* tem a função de interpretar a *web* sempre com o mesmo resultado, independentemente do sistema operativo, do *hardware* e do próprio *browser* utilizado. Este conceito de "interpretação multiplataforma" não é novidade, existindo linguagens de programação cujo objetivo principal é exatamente esse. No entanto, a forma como uma aplicação *web* é disponibilizada ao utilizador é significativamente diferente da forma como uma aplicação nativa equivalente é disponibilizada. Um exemplo é a completa dependência a um determinado servidor remoto, e os constrangimentos que isso acarreta. Mesmo os componentes visuais de uma aplicação, que não são de qualquer modo interpretados pelo servidor, estão dependentes deste para serem servidos ao *browser* do utilizador. Estes constrangimentos resultam comparativamente num desempenho e fluidez (rapidez) inferiores em aplicações *web.* Pretende-se assim desenvolver esforços para uma maior independência entre a *web* e as tecnologias que a suportam, mantendo o *browser* como interpretador universal, possibilitando e motivando a criação de aplicações mais complexas e de resposta mais fluída recorrendo a estas tecnologias.

Este documento detalha o processo de criação de uma plataforma para a partilha de conhecimento, denominada AnyKB, baseada apenas em tecnologias *web*. Esta plataforma foi construida com o objetivo de provar a possibilidade de separação total da componente visual e lógica inerente, do servidor remoto que mantém todos os dados da aplicação. A solução apresentada inspira-se na forma como aplicações nativas são distribuidas e servidas ao utilizador, e adapta esses conceitos ao *browser*, criando uma aplicação hibrida que compila os melhores aspetos das duas vertentes. A implementação desta solução resultou no protótipo atualmente disponível através do endereço http://anykb.net.

Finalmente, este documento conclui com um conjunto de testes e dados estatísticos que pretendem estudar os ganhos de desempenho e fluidez conseguidos pela implementação da solução apresentada, e avaliar a reação do público-alvo ao comportamento da aplicação.

Palavras-chave: web, Internet, multiplataforma, desempenho, fluidez, partilha de conhecimento

Abstract

The Internet brought forth the most globally accessible platform based on open technologies, called the *world wide web*, or just the web. In this context, the browser is an application with the objective of presenting the web in the same way regradless of the operating system or hardware used. This concept of a cross-plarform language interpreter isn't new, being proved by the existance of programming languages whose main purpose is to be cross-platform. However, the way a web application is served to the user couldn't be more different from the way a native application is served. One example is the complete dependecy a web application has on a remote server, and the constrainst that brings. Even the visual components of an application, that aren't in any way intepreted by the server, are completely dependent on a remote server in order to reach the user's *browser*. This constrainsts comparatively result in less performance and responsiveness for a web application. Given this, it is intended to make efforts towards the detachment of the *web* from the technologies that support it, keeping the *browser* as the universal interpreter, encouraging the creation of more complex and responsive applications based on these technologies.

This document details the creation process of a platform for knowledge sharing, named AnyKB, based solely on web techonlogies. This platform was build to prove the possibility of complete separation between the visual component of the application, incluind it's logic, and the remote server that keeps the application data. The presented solution draws inspiration from the way native application are distributed and served to the user, and adapts those concepts to the *browser*, creating an hybrid application that comprises the best of both types of application. The implementation of the proposed solution resulted in the prototype currently available at http://anykb.net.

Lastly, this document draws conclusions from the execution of several tests that originated sets of statistical data. The performance and responsiveness of the application are studied and analyzed to assess the gains achieved by the implemented solution, as well as the reaction of the target audience to the way the application behaves.

Palavras-chave: web, Internet, cross-platform, performance, responsiveness, knowledge sharing

Índice de Figuras

Índice de Tabelas

Índice de Gráficos

Lista de Siglas

ADB	Android Debug Bridge
API	Application Programming Interface
AJAX	Asynchronous JavaScript and XML
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DB	Database
DDR	Double Data Rate
DOM	Document Object Model
FTP	File Transfer Protocol
GB	Gigabyte
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
KB	Kilobyte
MB	Megabytes
PHP	PHP: Hypertext Preprocessor
PoC	Proof of Concept
RAM	Random Access Memory
REST	Representational State Transfer
SAN	Storage Area Network
SDK	Software Development Kit
SGBD	Sistema de Gestão de Base de Dados
SGML	Standard Generalized Markup Language
SQL	Stuctured Query Language
SSE	Server-sent Events
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
WHATWG	Web Hypertext Application Technology Working Group
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

Índice

1 Introdução

A plataforma descrita neste documento, de nome AnyKB, é uma aplicação *web* de partilha de conhecimento em formato de artigos, criada com o objetivo de implementar e testar um conceito de cache inovador. O sistema de cache tem como objetivo a otimização da aplicação cliente da plataforma (executada no *browser*) por forma a fornecer ganhos significativos de fluidez e tempo de resposta ao utilizador. O conceito a implementar tem inspiração no processo de instalação e distribuição de aplicações nativas (i.e. aplicações "clássicas" geralmente compiladas para um determinado sistema operativo). Ao aceder à página *web* da plataforma AnyKB existirá um processo semelhante ao da instalação de uma aplicação nativa, que tirará partido de tecnologias emergentes recentemente disponíveis nos *browsers* mais populares, tais como *Local Storage* [1], AJAX [2] e *push notifications*. Espera-se com este processo conseguir ganhos de desempenho significativos comparativamente a uma aplicação *web* equivalente que não faça uso destas otimizações.

Em termos de funcionalidade, a plataforma AnyKB aspira a um novo conceito de plataforma *online* híbrida, no sentido que implementa um conjunto de características presentes noutras plataformas com objetivos semelhantes. O conteúdo será idêntico ao de blogues, com o espírito de entreajuda encontrado em fóruns, e proporcionando o sentimento de realização encontrado em plataformas pergunta-resposta como a rede Stack Exchange [3].

A implementação de uma plataforma com utilidade e complexidade reais permite a recolha de resultados estatísticos realistas sobre o desempenho da plataforma em geral. A plataforma AnyKB é composta por conteúdo estático e dinâmico, e o sistema de cache deverá ser capaz de lidar mesmo com grandes quantidades de conteúdo dinâmico criado pela comunidade, com ganhos de desempenho significativos.

1.1 Objetivos

É objetivo principal do projeto AnyKB a criação de uma plataforma *web* com utilidade real que permita a recolha de dados realistas sobre os ganhos de desempenho conseguidos com a implementação de várias técnicas que recorrem a tecnologias *web* emergentes. A aplicação,

na sua totalidade, será composta por duas vertentes: aplicação cliente executada no *browser*, e aplicação servidora executada num servidor *cloud* numa localização remota. Com a implementação desta plataforma espera-se conseguir especificamente:

- Criação de uma plataforma colaborativa para partilha de conhecimento orientada à comunidade;
- Otimização substancial do tempo de carregamento das várias páginas *web* da plataforma;
- Otimização substancial da fluidez da aplicação *web* cliente.

Importa denotar que no âmbito deste documento é empregue o termo fluidez com a conotação da rapidez que é percecionada pelo utilizador quando interage com a aplicação em apreço.

1.2 Organização do relatório

Este documento começa por apresentar no capítulo "Revisão da literatura" o estado atual das tecnologias envolvidas no projeto. Este capítulo descreve o objetivo e utilidade destas tecnologias, juntamente com exemplos de utilização e informação estatística de desempenho, quando aplicável.

O capítulo seguinte, "Implementação", apresenta de forma técnica o problema que incentivou à criação do projeto AnyKB e descreve os detalhes técnicos da implementação das duas componentes principais da plataforma: cliente e servidor. No subcapítulo dedicado à componente servidora encontra-se descrita a arquitetura global da aplicação juntamente com a descrição de como as várias ferramentas utilizadas no servidor foram implementadas e/ou configuradas. No subcapítulo seguinte, dedicado à componente cliente, encontra-se a descrição do interface gráfico da aplicação e dos esforços desenvolvidos por forma a otimizar o desempenho desta componente.

De seguida encontramos o capítulo "Discussão de Resultados" onde são estudados os ganhos de desempenho conseguidos após a implementação do projeto. Os dados estatísticos apresentados compreendem as duas componentes da aplicação (cliente e servidor) ao longo de vários dispositivos móveis (*smartphone*) e não-móveis (computador pessoal). Este capítulo subdivide-se ainda em dois subcapítulos, o primeiro dedicado à apresentação de informação quantitativa, e o segundo de informação qualitativa conseguida através de questionário.

Finalmente, no capítulo "Conclusão" é apresentado um resumo dos resultados conseguidos juntamente com algumas conclusões finais, e feito um levantamento do trabalho futuro no contexto da plataforma AnyKB.

1.3 Contributos

O conceito e solução apresentados pretendem contribuir para uma maior generalização das tecnologias *web* na construção de aplicações informáticas, disponíveis ou não através da Internet. A generalização destas tecnologias contribui para um maior número de aplicações construídas recorrendo a linguagens, ferramentas e infraestruturas abertas (*open-source*) que evoluem graças aos esforços da própria comunidade. Adicionalmente, a solução apresentada contribui para o desenvolvimento de uma *web* mais rápida, fluída e móvel ao incentivar uma maior separação entre *browser* e servidor *web*.

A plataforma AnyKB, que surge como resultado da implementação desse conceito, pretende contribuir em duas vertentes. Em primeiro a vertente de exemplo e base de estudo para a observação e análise do comportamento da solução implementada, permitindo avaliar o grau de sucesso da mesma. Em segundo contribui com uma nova plataforma de partilha de informação, onde a comunidade pode criar artigos tecnológicos por iniciativa própria com o apoio direto da comunidade.

Por fim, enquanto trabalho científico de investigação, o projeto resultou na escrita deste documento e na publicação de um artigo de revista *online* portuguesa de divulgação informática sem revisão por pares, contribuindo para a comunidade científica.

2 Revisão da literatura

A Internet, com pouco menos de 45 anos de existência [4], conta com 55.6% da população portuguesa e 32.8% da população mundial [5], tornando-se assim a plataforma baseada em tecnologias abertas e multiplataforma mais acessível a um nível global. Não tão óbvio é que as mesmas tecnologias que possibilitam a existência da *web*, permitem também a implementação e execução de aplicações tão funcionais como aplicações *desktop* e *mobile*, totalmente baseadas em tecnologias abertas e acessíveis por mais de 14 mil milhões de dispositivos [6] com os mais variados sistemas operativos, **mesmo sem acesso à Internet**, tendo como único requisito a presença de uma aplicação *browser*.

A confirmação da potencialidade destas tecnologias é visível, por exemplo, nas extensões (ou *add-ons*) [7] [8] que os *browsers* mais populares, como o Google Chrome ou Mozilla Firefox, suportam, e que são implementadas recorrendo a tecnologias *web*, como JavaScript (ECMAScript [9]) e HTML [10]. Inicialmente, tais extensões começaram por suportar pequenas aplicações utilitárias, evoluindo recentemente para aplicações mais complexas como, por exemplo, clientes FTP [11], editores de som [12] e jogos [13]. Adicionalmente, provas da polivalência de tecnologias como o HTML não estão apenas presentes na *web*: plataformas de desenvolvimento de *software desktop* e *mobile* - como Android SDK [14] e Microsoft .NET (XAML) [15] – possibilitam a definição de interfaces de utilizador recorrendo a XML [16], uma norma para a definição de linguagens de marcação que surgiu como evolução do HTML [17], que por sua vez é a linguagem de marcação utilizada para definir interfaces de utilizador de aplicações *web*.

Na verdade, se dividirmos um modelo aplicacional em três camadas, à semelhança do que acontece numa arquitetura *3-Tier* [18] – apresentação, lógica de negócio e acesso a dados – existem tecnologias, que surgiram e evoluíram graças à *web*, que satisfazem quaisquer uma das três camadas na implementação de uma aplicação local, sem o requisito de acesso à Internet. Abaixo estão alguns exemplos destas tecnologias, divididos pelas camadas descritas.

Camadas	Tecnologias aplicáveis
Apresentação	HTML, XML, JavaScript, CSS
Lógica de Negócio	JavaScript, CoffeeScript [19], Dart [20]
Acesso a dados	Cookies [21], Local Storage [1], Indexed Database [22]

Tabela 1 – As três camadas do modelo *3-Tier* e tecnologias relacionadas.

Mais recentemente, com as inovações presentes na nova normal HTML5 [23], está a pavimentar-se o caminho para a implementação de aplicações complexas, escritas recorrendo apenas a tecnologias *web*, sem a necessidade de instalar extensões no *browser*. São estas as tecnologias chave para o desenvolvimento do projeto AnyKB, e que iremos analisar em maior detalhe em seguida.

Este capítulo encontra-se organizado da seguinte forma: de seguida são exploradas as tecnologias *web* emergentes mais relevantes para o projeto AnyKB – organizadas pelas três camadas descritas acima - e na secção seguinte são analisadas várias plataformas web empregues por todo o tipo de profissionais para expor as suas dúvidas e solucionar os problemas de colegas da mesma área.

2.1 HTML5 e tecnologias relacionadas

Nesta secção apresentam-se, de forma sucinta, as tecnologias *web* emergentes consideradas interessantes ao desenvolvimento deste projeto. A abordagem é separada em três categorias que correspondem às três camadas de um modelo aplicacional *3-Tier*: camada de apresentação, lógica de negócio e acesso a dados.

2.1.1 Tecnologias para a camada de apresentação

A camada de apresentação é responsável por todos os elementos visuais de uma aplicação, denominada interface de utilizador, bem como de toda a lógica necessária para manter a integridade desse interface. Abaixo estão descritas as tecnologias inseridas nesta camada interessantes para a implementação do projeto AnyKB.

HTML

O HTML é uma norma para linguagem de marcação baseado em SGML [24] [25], que definiu um novo conjunto de símbolos de marcação (*markup*). Criado por Sir Tim Berners-Lee [24] em 1989, o HTML veio possibilitar a implementação de um novo conceito: a interligação de ficheiros de texto, permitindo a navegação de ficheiro em ficheiro através de elementos visuais que estariam em linha com o conteúdo (botões), com os quais se poderia interagir [24]. Desde então, a utilização do HTML como linguagem de marcação evoluiu maioritariamente para a definição de interfaces de utilizador para a *web*, tendo já passado pela definição de cinco versões principais (2.0, 3.2, 4.0, 4.0.1, 5) [26] até à última versão: HTML5. O termo HTML5, para além de estar relacionado com a norma que define a última versão do HTML, é também uma *buzzword* muitas vezes associada a um conjunto de tecnologias emergentes, não necessariamente relacionadas com a norma, mas que são novidade e que introduzem um conjunto de funcionalidades revolucionárias.

Apesar de a W3C definir ainda a norma HTML5 como *"working draft"* (rascunho), a adesão por parte dos *browsers* mais populares tem sido bastante boa. O *website "The HTML5 Test"* [27] implementou uma série de testes onde é verificada a capacidade de um determinado *browser* suportar funcionalidades relacionadas com HTML5, atribuindo uma pontuação no final do teste. O gráfico abaixo demonstra a evolução da maturidade dos *browsers* mais populares na implementação do HTML5 e tecnologias relacionadas, desde janeiro de 2009 até janeiro de 2013.

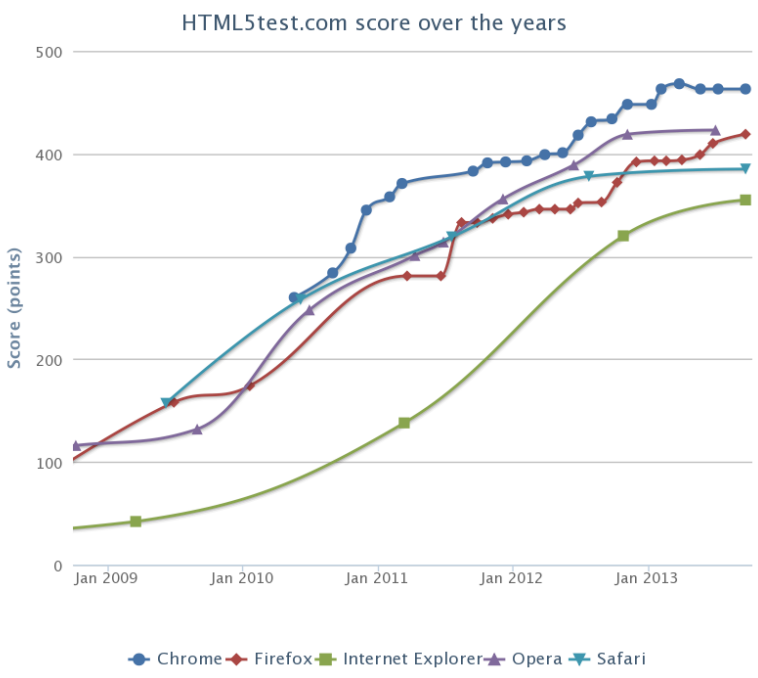


Figura 1 – Evolução da maturidade na implementação de HTML5 e tecnologias relacionadas por parte dos *browsers* mais populares, numa escala de 0 a 500. Fonte: http://html5test.com/results/desktop.html.

A Figura 2 mostra um exemplo de um documento HTML5 simples, onde é possível analisar a definição de um título e parágrafo de texto. Um documento HTML é estruturado utilizando *tags* que são definidas pelos símbolos "<" e ">", sendo que cada *tag* tem uma função e/ou comportamento específico a si associados. Tomando o exemplo em apreço, podemos verificar que o início e fim de um documento HTML é definido pelas *tags* "<html>" e "</html>", respetivamente; e no corpo do documento estão definidos um título através da *tag* "<h1>" e um parágrafo através da *tag* "<p>". Estas *tags* definem apenas a estrutura do documento, indicando que o conteúdo que surge entre duas quaisquer *tags* de início e fim tem um significado específico, e deve ser tratado de acordo com esse mesmo significado.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Título</title>
  </head>
  <body>
    <h1>Título da página</h1>
      <p>Parágrafo de texto</p>
  </body>
</html>
```

Figura 2 – Exemplo de um documento HTML simples (à esquerda), e o aspeto do mesmo quando visualizado com o *browser* Firefox versão 16.0.2 (à direita).

Apesar de não ter sido atribuída qualquer definição de aparência ao documento, verifica-se que o *browser* Firefox predefiniu alguns aspetos visuais, como o título a negrito e o espaçamento que existe entre um título e um parágrafo. Este é um dos resultados do significado atribuído às *tags*: o *browser* entende que um título é algo que deverá estar em evidência, e então coloca-o em destaque utilizando um tipo de letra maior. Estas predefinições visuais estão muitas vezes na origem de inconsistências que se encontram na forma como páginas *web* são apresentadas por *browsers* diferentes, pois nem sempre existem regras que definam como um elemento deverá ser apresentado quando este não tem qualquer informação de aparência associado. Apesar de ser possível definir a aparência de um documento através de *tags* HTML, tal é desencorajado visto esta responsabilidade ter sido transferida para a tecnologia CSS [28], descrita na secção seguinte.

CSS (Cascading Style Sheets)

O CSS [28] (*cascading style sheets*) é um mecanismo para a definição da aparência de páginas *web*, permitindo a separação da definição da estrutura da página (HTML) da aparência da mesma (CSS).

Antes da existência do CSS, com norma em 1996 pela W3C [29], a informação relativa à aparência das páginas *web* era definida recorrendo a HTML e inserida junto com a informação sobre a estrutura da página. À medida que a *web* crescia, e consigo as necessidades dos *web developers*, o HTML foi suportando a definição de cada vez mais elementos de aparência como cores, espaçamento, altura, largura, tipo de letra, etc. Eventualmente, com a existência de um grande número de *tags* para a definição da aparência, e dado estas informações estarem juntas com a informação sobre a estrutura do documento, rapidamente os documentos ficaram complexos demais para serem interpretados de igual forma por diferentes *browsers*. Este é um problema que se prolonga até aos dias de hoje, no entanto de uma forma mais controlada e menos complexa dada a existência do CSS.

```
body
{
  background-color: lightgrey;
}
h1
{
  text-decoration: underline;
}
p
{
  border: solid 2px black;
  padding: 5px;
  width: 220px;
}
```

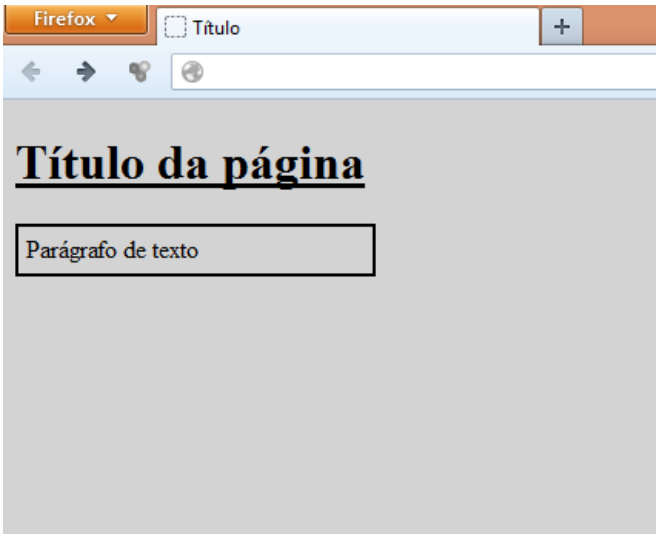


Figura 3 – Exemplo de uma folha de estilos CSS (à esquerda) aplicada ao documento HTML descrito na Figura 2, e o aspeto do documento quando visualizado com o *browser* Firefox versão 16.0.2 (à direita).

Na Figura 3 é possível verificar a definição da aparência de um documento HTML sem, no entanto, ter modificado a sua estrutura. Na prática, é possível redesenhar **completamente** a

aparência de um documento HTML, incluindo a ordem de aparência dos elementos, sem nunca recorrer à alteração da estrutura do mesmo.

O CSS encontra-se na sua terceira versão, denominada "CSS Level 3" (ou apenas CSS3) [28], existindo uma larga adesão por parte dos *browsers* mais populares. À semelhança do teste de funcionalidades HTML5 descrito em [27], existe também uma página *web* que permite testar as capacidades CSS3 de um determinado *browser*, denominada *"The CSS3 Test"* [30], e o gráfico abaixo mostra a maturidade da implementação do CSS3 pelos *browsers* mais populares.

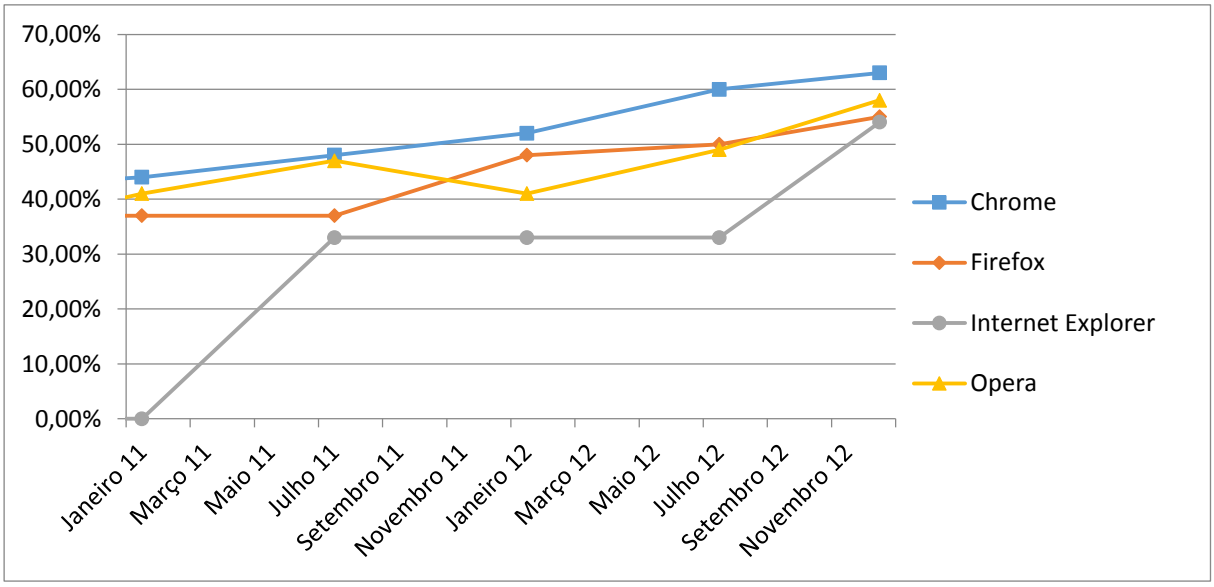


Figura 4 – Evolução da maturidade na implementação de CSS3 e tecnologias relacionadas por parte dos *browsers* mais populares. Fonte: http://css3test.com.

DOM (Document Object Model)

O DOM [31] é uma interface programática que permite aceder e alterar dinamicamente a estrutura de um documento, neste caso de documentos HTML, possibilitando a implementação da lógica de apresentação de uma aplicação *web* recorrendo, por exemplo, ao JavaScript.

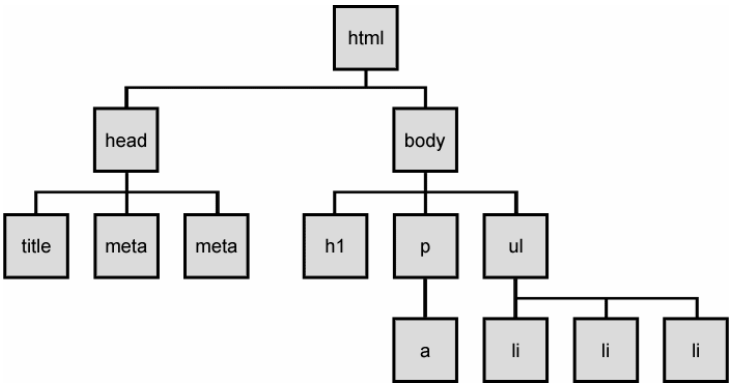


Figura 5 – *DOM Tree*, ou árvore DOM, onde é possível visualizar a estrutura hierárquica dos nós de um documento HTML. Fonte: http://www.webstepbook.com.

O DOM expõe ao programador uma interface em forma de árvore hierárquica, como é possível analisar na Figura 5, que este pode então percorrer, de forma sequencial ou recorrendo a *selectors* [32], por forma a manipular os seus nós, que são constituídos pelas *tags* HTML e todo o conteúdo envolvente. Assim que um elemento é selecionado através do DOM é possível a manipulação das propriedades dos mesmos, ou a execução de funções que permitem manipular, de forma específica, uma propriedade de um determinado nó, sendo tais alterações visíveis pelo utilizador em tempo real. Tal comportamento possibilita não só a manipulação dos nós da árvore mas também a implementação de animações dos elementos visuais do ecrã, por exemplo.

2.1.2 Tecnologias para a camada de lógica de negócio

A camada da lógica de negócio é responsável por conter todo o código que permite o funcionamento das funcionalidades de uma aplicação. Abaixo estão descritas as tecnologias inseridas nesta camada consideradas interessantes para a implementação do projeto AnyKB, nomeadamente da aplicação cliente.

Linguagens de Programação

Nesta secção analisamos as linguagens de programação possíveis de serem executadas no *browser* do utilizador, e que se consideram candidatas para a implementação da aplicação cliente.

JavaScript

JavaScript é a designação mais popular do dialeto ECMAScript [9], uma linguagem de *scripting* atualmente baseada na especificação ECMA-262 [33], suportada pela generalidade dos *browsers* por forma a permitir a manipulação de páginas *web* e a criação de interfaces de

utilizador e experiências *web* mais ricas.

Tratando-se o ECMAScript de uma especificação, as implementações concretas designam-se de dialetos, existindo atualmente três principais designações de dialetos utilizadas pelos *browsers* mais populares. O Mozilla Firefox, o Google Chrome e o Safari utilizam a designação JavaScript para o seu dialeto; o *browser* Opera suporta um dialeto de nome igual à norma que lhe deu origem, ECMAScript; e finalmente o *browser* Internet Explorer designa o seu dialeto de JScript. Apesar das diferentes nomenclaturas todos os dialetos são apenas implementações distintas de uma mesma linguagem, em conformidade com uma determinada versão do ECMAScript. A sintaxe de um determinado *script* é, independentemente do dialeto, interoperável com qualquer *browser*, desde que o dialeto e motor de JavaScript utilizado pelo *browser* tenham sido implementados em conformidade com a mesma norma ECMAScript.

Os *browsers* interpretam JavaScript através de motores JavaScript (JavaScript *engines*), que variam de *browser* para *browser* e apresentam diferentes desempenhos, um fator bastante importante em aplicações que façam uso intensivo de JavaScript, como é o caso do projeto AnyKB. Para medir o desempenho de um determinado motor a Mozilla fornece uma ferramenta *web*, denominada "Kraken JavaScript Benchmark" [34], que compila um conjunto de testes que o *browser* deve desempenhar num determinado intervalo de tempo, sendo esse intervalo empregue como métrica do desempenho do motor. A Figura 6 mostra o desempenho dos *browsers* mais populares, na sua última versão estável, executados num computador com sistema operativo Windows 8 suportado por um processador Intel Core i3 2.13Ghz e 4GB de memória RAM.

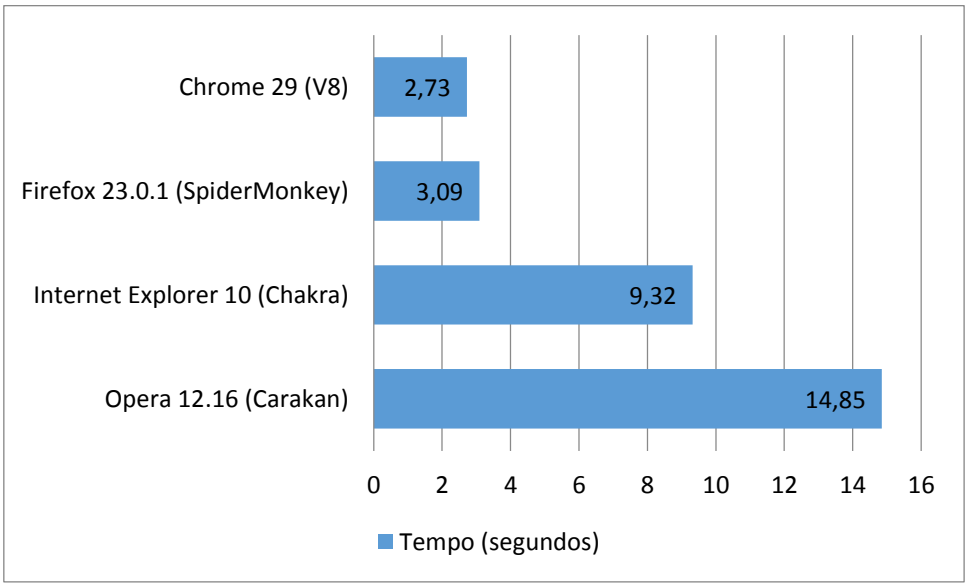


Figura 6 – Desempenho (em segundos) dos motores JavaScript (entre parêntesis) dos *browsers* mais populares. Dados provenientes de http://krakenbenchmark.mozilla.org/.

No projeto AnyKB, o JavaScript é empregue para implementar a lógica de interface, e ainda um sistema de cache de conteúdo, descrito em mais detalhe no decorrer deste documento, que pretende acelerar substancialmente o carregamento do conteúdo da aplicação.

Dart

O Dart é uma linguagem de programação orientada a objetos, desenvolvida pela Google [35], com principal foco em aplicações *web*. O Dart tem como propósito final *"substituir o JavaScript como linguagem franca do desenvolvimento web na plataforma web aberta"* [36].

O Dart diferencia-se principalmente ao ser uma linguagem orientada a objetos, ao contrário do JavaScript que é orientado ao protótipo [37], por opcionalmente suportar *static types*, e por fornecer nativamente um conjunto considerável de bibliotecas (*libraries*) com elevada funcionalidade. No entanto, a principal razão que levou a Google a iniciar este projeto prende-se com a necessidade de evoluir para além do JavaScript, algo que a Google não acredita ser possível de conseguir ao estender o JavaScript dada a *"bagagem histórica [do JavaScript] que não consegue ser resolvida sem um começo limpo"* [38].

O Código escrito em Dart é executado numa máquina virtual, denominada Dart VM [39], sendo também possível a compilação do código para JavaScript através do compilador "dart2js" [40], garantindo assim a interoperabilidade da linguagem com os *browsers* mais populares. No entanto, a utilização da Dart VM fornece um desempenho acrescido de cerca de 21% às aplicações escritas em Dart, comparativamente a código equivalente escrito em JavaScript, como é possível analisar na Figura 7.

Benchmark	v8	dart	dart2js	dart	dart2js
DeltaBlue	284.86	363.14	186.63	127.48%	65.52%
Richards	400.10	565.72	279.22	141.39%	69.79%
NBody	15945.00	17436.50	10936.50	109.35%	68.59%
BinaryTrees	9.02	9.33	8.24	103.49%	91.38%
Mandelbrot	169.08	167.92	138.36	99.31%	81.83%
Fannkuch	3458.50	4202.00	3172.00	121.50%	91.72%
Meteor	6.68	5.96	2.25	89.29%	33.73%
BubbleSort	25248.51	27160.00	15850.50	107.57%	62.78%
Fibonacci	9156.00	13570.50	9405.50	148.21%	102.72%
Loop	34392.03	35560.00	35302.50	103.40%	102.65%
Permute	11084.50	15921.50	7584.00	143.64%	68.42%
Queens	118474.98	184505.01	103320.00	155.73%	87.21%
QuickSort	17138.49	17723.50	9771.00	103.41%	57.01%
Recurse	13990.50	19994.50	14439.50	142.91%	103.21%
Sieve	102273.54	117566.50	106883.50	114.95%	104.51%
Sum	74409.74	60180.50	75387.00	80.88%	101.31%
Tak	3062.50	4731.50	2487.00	154.50%	81.21%
Takl	8917.50	17101.00	8941.00	191.77%	100.26%
Towers	4915.50	5559.00	3232.50	113.09%	65.76%
TreeSort	7043.50	8672.50	5417.00	123.13%	76.91%
Geo. mean	4009.51	4855.15	3136.27	121.09%	78.22%

Figura 7 – Teste comparativo da performance de vários algoritmos quando executados num motor JavaScript (coluna V8), na Dart VM (coluna dart), ou no motor JavaScript utilizando o resultado do compilador dart2js (coluna dart2js). Valores indicam o número de execuções por segundo. Fonte: http://dartlang.org.

Apesar das vantagens indicadas, o único *browser* atualmente a suportar a execução nativa de código Dart é o Dartium, uma versão especial do Chromium, o *browser open-source* desenvolvido pela Google, que inclui a Dart VM. Por enquanto o único suporte para a interoperabilidade com outros *browsers* depende do compilador dart2js. Dados estes fatores, a escolha recai sobre o JavaScript para a implementação do projeto AnyKB. Apesar de existir a possibilidade de conversão de código Dart em JavaScript, o código que resulta deste processo demonstra, na maioria dos casos, perda de desempenho, que não se pretende comprometer.

API's JavaScript

As API's JavaScript são extensões à linguagem que permitem a introdução de novas funcionalidades em aplicações *web*, servindo o JavaScript como ferramenta intermediária para o acesso a tais funcionalidades. Neste capítulo estão descritas as API's mais interessantes ao desenvolvimento do projeto AnyKB, que se inserem na camada de lógica de negócio.

AJAX e Manipulação do Histórico

Na sua forma clássica, o carregamento de páginas *web* através de um *browser* é feita de uma forma estática, isto é, sempre que um utilizador pede uma página a um servidor *web* essa mesma página é descarregada na sua totalidade, não havendo possibilidade de carregar em tempo real dinamicamente apenas parte da página. Para permitir o carregamento parcial de uma página a W3C padronizou uma API JavaScript denominada "XMLHttpRequest" [41] com o objetivo de introduzir na *web* a funcionalidade de comunicação assíncrona entre um cliente e servidor *web*. Esta tecnologia, quando utilizada em conjunto com HTML, CSS e a interface DOM - com o objetivo de alteração dinâmica de partes de uma página *web* - tem a denominação de AJAX [2]. Exemplos populares da utilização da tecnologia AJAX em aplicações *web* vão desde o carregamento de *websites* completos fazendo uso apenas de AJAX, a clientes de *chat* em tempo real [42] [43]. Esta tecnologia introduz claramente uma grande potencialidade para a otimização do desempenho de uma aplicação *web* no que toca ao tamanho do conteúdo a ser transferido do servidor, especialmente quando a diferença entre dois estados distintos de uma aplicação é mínima, sendo apenas necessário carregar do servidor as partes divergentes.

Com a possibilidade de carregamento dinâmico de páginas *web* fazendo uso de AJAX, torna-se necessário manter a integridade do URL, pois idealmente este deverá sempre identificar de forma única um determinado recurso num servidor *web*. No entanto, através de AJAX é possível alterar na totalidade o conteúdo da página que se está a visualizar sem que o URL seja atualizado conforme o novo conteúdo, o que poderá levar a que o URL fique fora de contexto. Outro problema introduzido pelo uso de AJAX prende-se na existência de conteúdo que pode até estar acessível através do carregamento dinâmico, mas sem que tenha um URL associado, tornando impossível o seu acesso direto através do mesmo [44]. Por forma a dar solução a este problema a WHATWG padronizou uma interface denominada "History" [45] que define um conjunto de funções que possibilitam a manutenção do URL por parte do programador através de JavaScript, sendo assim possível manter a integridade do URL em conformidade com o conteúdo carregado de forma dinâmica através de AJAX.

Push Notifications

Um sistema de *push notifications*, no contexto *web*, permite o envio de dados por parte do servidor para a aplicação cliente, servida no *browser*, sem que a ligação seja iniciada pela aplicação cliente.

O paradigma de obtenção de dados em vigor na *web* é maioritariamente *pull*, na medida em que a aplicação cliente tem de iniciar uma ligação com o servidor caso pretenda receber dados provenientes deste. Até recentemente não existia a possibilidade de implementar um sistema

de *push notifications* na *web* pois as tecnologias utilizadas simplesmente não estavam preparadas: o protocolo HTTP não oferece qualquer suporte a este tipo de funcionalidade e o JavaScript não oferecia qualquer modo de manter uma porta de ligação aberta à qual o servidor se poderia ligar de livre vontade a qualquer momento.

Isto não impediu, no entanto, utilizadores de tentarem contornar as limitações das tecnologias existentes, ou mesmo o aparecimento de entidades responsáveis por padronizar tecnologias que o permitam. Exemplo do sucesso das tentativas da criação de um sistema destes é o COMET, um termo utilizado para descrever um conjunto de tecnologias e técnicas que, quando utilizadas em conjunto, permitem implementar um comportamento **semelhante** àquele de um sistema *push* real [46]. A principal técnica utilizada em COMET explora o funcionamento do protocolo HTTP, nomeadamente a tolerância que é dada ao servidor *web* no tempo que este tem para responder, para utilizar o tempo em que a ligação está ativa como se se tratasse de uma ligação persistente para receber informação do servidor em tempo real. Esta técnica tem o nome de *long-polling* e a sua utilização acarreta alguns problemas, como a manutenção da ligação que poderá requerer a renovação da ligação com o servidor em prazos tão curtos como 30 segundos por forma a prevenir a possível expiração do tempo limite de ligação [47].

Mais recentemente, como resposta à falta de uma solução deste tipo na *web*, a W3C padronizou uma API JavaScript denominada Server-Sent Events (SSE) [48], que permite *"abrir uma ligação HTTP com o objetivo de receber push notifications a partir de um servidor na forma de eventos DOM"* [48]. Através da utilização desta API é possível a implementação de suporte a *push notifications* na aplicação *web* cliente de forma bastante mais simplificada comparativamente a COMET.

Em relação à aplicação AnyKB, um sistema de *push notifications* será considerado por forma a atualizar o modelo de dados presente do *browser* quando ocorram alterações a este no servidor, fazendo esta funcionalidade parte de um conjunto de esforços para a otimização do carregamento da informação por parte da aplicação *web* cliente, descritos em maior detalhes no decorrer deste documento.

2.1.3 Tecnologias para a camada de acesso a dados

Descrevem-se de seguida, as API's JavaScript para a manipulação de dados de forma persistente, que sejam consideradas interessantes ao projeto AnyKB.

Indexed database

Indexed Database é uma API JavaScript padronizada pela W3C [49] que fornece uma interface para armazenamento de dados de forma persistente em formato de base de dados, e é

uma alternativa ao WebSQL, um padrão entretanto considerado obsoleto pela W3C [50].

À semelhança do *Local Storage* [1], descrito de seguida, o armazenamento de dados é feito na forma de pares chave/valor, possibilitando também o armazenamento direto de estruturas complexas, como objetos JavaScript, sem necessidade de serialização. Outra vantagem oferecida pelo *Indexed Database* é a possibilidade de aceder síncrona ou assincronamente à interface de persistência, tendo assim o programador a escolha de evitar um potencial bloqueio no interface do utilizador em operações mais pesadas ao utilizar o acesso assíncrono. No entanto, como desvantagens esta API apresenta uma interface programática de maior complexidade comparativamente ao *Local Storage*, e falta de suporte por alguns *browsers* populares como o Safari, iOS Safari, Opera Mini, e Android Browser [51]. Adicionalmente, existe a hipótese se ser necessária a permissão do utilizador para que uma aplicação *web* utilize esta tecnologia em determinados *browsers*. Este fator não parece ter ainda um comportamento consistente ao longo dos *browsers* mais populares [52] [53].

File System API

A *File System* API é uma norma da W3C [54] que define uma tecnologia para o armazenamento de dados, de forma temporária ou persistente, no formato de ficheiros.

A API funciona ao disponibilizar uma interface programática JavaScript que expõe um *filesystem* virtual e *sandboxed* no qual é possível criar pastas e ficheiros, que são escritos para o disco do computador do utilizador [55]. Ou seja, a API não tem como objetivo permitir a manipulação dos dados já existentes no computador do utilizador, mas sim a criação e manutenção de ficheiros criados pela aplicação *web* com o objetivo de persistência de dados relativos a essa mesma aplicação.

Comparativamente aos restantes métodos descritos para a persistência de dados, a *File System* API necessita de permissão do utilizador por forma a ser possível guardar dados de forma persistente [56], e apresenta também a possibilidade de utilizar interfaces síncronas ou assíncronas, podendo o programador escolher qual das duas será a mais indicada consoante a implementação em causa. Esta é, no entanto, a API menos suportada pelos *browsers* atuais, estando apenas disponível no *browser* Google Chrome e Google Chrome para Android [57].

Local Storage

Local Storage é uma interface que integra a API JavaScript de nome "*Storage"*, padronizada pela W3C, que possibilita o armazenamento de dados de forma persistente. Este sistema, especialmente aquando do seu aparecimento, foi recorrentemente anunciado como alternativa viável à utilização de *cookies*, chegando até algumas fontes a citá-lo como o *"cookie killer"* [58] o que contribuiu para a sua popularidade.

Esta API expõe um interface simples que dá acesso a um sistema de armazenamento baseado em pares chave/valor onde o programador pode guardar texto como valor de uma determinada chave, sendo possível o armazenamento de objetos JavaScript através da serialização dos mesmos. Os dados armazenados são visíveis apenas pelas aplicações *web* executadas no mesmo domínio ou subdomínio que lhes deu origem. Algumas fontes citam a tecnologia como tendo um fraco desempenho em leitura/escrita [58], no entanto existem testes que demonstram exatamente o contrário: melhor desempenho com Local Storage do que com IndexedDB ou WebSQL [59].

Apesar de apenas possibilitar acesso síncrono de leitura/escrita, e do suposto fraco desempenho, *Local Storage* é o sistema que expõe a interface mais simples e direta comparativamente com os métodos de armazenamento persistente estudados neste capítulo, e é também o sistema mais suportado, estando disponível em todos os principais *browsers* à exceção do Opera Mini para dispositivos móveis [60]. Por estes motivos esta foi a tecnologia escolhida para a implementação dos requisitos de persistência de dados na plataforma AnyKB.

Síntese

A seguinte tabela agrupa as várias características das API's para a persistência de dados anteriormente descritas, para comparação.

Nome da API	**Permissões especiais**	**Espaço disponível**	***Browsers* Compatíveis**
Indexed Database	Alguns *browsers* podem requerer permissões do utilizador.	50MB	Internet Explorer, Mozilla Firefox (*desktop* e móvel), Google Chrome (*desktop* e móvel) e Opera (*desktop* e móvel).
***File System* API**	Requer permissão do utilizador.	Espaço pedido ao utilizador.	Google Chrome (*desktop* e móvel), Opera (*desktop* e móvel) e Blackberry Browser.
Local Storage	Nenhuma.	5MB	Internet Explorer, Mozilla Firefox (*desktop* e móvel), Google Chrome (*desktop* e móvel), Safari, Opera (*desktop* e móvel) e Blackberry Browser.

Tabela 2 – Tabela síntese das tecnologias para a persistência de dados no *browser* estudadas neste documento. Foram consideradas as seguintes versões: Internet Explorer 10, Mozilla Firefox 23.0.1 (*desktop* e móvel), Google Chrome 29 (*desktop* e móvel) e Opera 16 (*desktop* e móvel).

2.2 Plataformas *online* para a solução de problemas

Nesta secção estão descritas de forma sucinta as principais plataformas atualmente presentes na Internet que visam a resolução de problemas, principalmente tecnológicos, ao facilitar a partilha de informação entre entendidos da mesma área. Estas são as plataformas que serviram como ponto de partida para o desenvolvimento do projeto AnyKB.

2.2.1 Blogues

Um blogue é uma plataforma onde um ou mais autores publicam artigos, geralmente utilizando um registo linguístico mais casual, sobre os mais variados assuntos, que abrangem desde opiniões pessoais, passando por notícias, até conteúdo mais técnico.

O autor de um blogue pode publicar um artigo sobre como solucionar um determinado problema, por exemplo incentivado pela descoberta própria da solução, sendo esse conteúdo visualizado recorrentemente por leitores assíduos do blogue, ou esporadicamente por leitores que encontrem o artigo através de motores de busca.

É comum encontrar um sistema de comentários neste tipo de plataforma, o que incentiva a discussão entre leitores, ou mesmo entre leitor e autor. No entanto este tipo de plataforma não incentiva nem coloca em evidência a entreajuda entre leitor e utilizadores, ou entre

utilizadores, para a solução de problemas. O assunto em discussão encontra-se geralmente limitado ao tema do artigo relacionado.

2.2.2 Fóruns

Um fórum é uma plataforma comunitária dividida por categorias (ou temas) que permite a interação entre vários utilizadores na forma de tópicos (um assunto iniciado por um utilizador). Um determinado tópico de discussão pode ser iniciado por qualquer utilizador, e qualquer utilizador é igualmente livre de responder a um determinado tópico, construindo o que se intitula de *thread*: uma discussão entre vários utilizadores que se vai desenvolvendo como uma conversa de grupo virtual, moderada por um conjunto seleto de utilizadores que mantém a integridade da plataforma.

No que toca à solução de problemas, esta plataforma pode ser utilizada como uma ferramenta que permita utilizadores formularem perguntas e obter soluções de outros utilizadores, com a potencialidade de gerar um tópico de discussão interessante. Deste modo, pode-se entender um fórum como uma evolução das *mailing lists*, que caem cada vez mais em desuso. A motivação dos utilizadores está maioritariamente do lado de quem formula a questão, pois tem o interesse em resolver o problema, partindo a ajuda dos restantes utilizadores e da sua boa vontade.

2.2.3 Pergunta - Resposta: Rede StackExchange

A rede StackExchange [61] criou um novo paradigma ao construir uma plataforma orientada especificamente à pergunta e resposta, colocando estes elementos em evidência. Esta plataforma difere das anteriormente analisadas ao servir apenas um propósito: a solução de problemas, ao contrário das demais plataformas que podem ser utilizadas para outros fins. Para obter a solução a um determinado problema um utilizador pode procurar no *site* se a sua questão já foi respondida e, em caso negativo, formular ele mesmo uma pergunta. Após colocada a pergunta basta esperar pelas respostas de outros utilizadores.

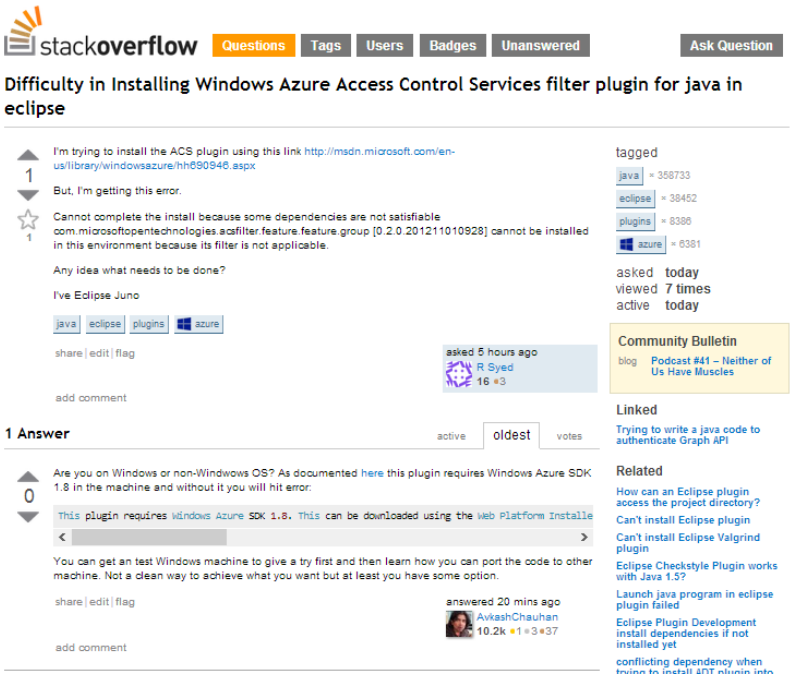


Figura 8 – Exemplo de questão colocada na plataforma Stack Overflow [62], um *site* pertencente à rede StackExchange, para a solução de problemas relacionados com programação.

A rede StackExchange conta neste momento com dezenas de *websites* orientados a diversos temas (não necessariamente tecnológicos), que se tornaram um ponto de encontro essencial para muitos entendidos na matéria [3].

O sistema de recompensa da plataforma está na base do sucesso da rede: os utilizadores ganham pontos de reputação ao fazer boas perguntas e ao dar boas respostas, e podem também ir "colecionando" *badges* (medalhas) à medida que o seu percurso pela rede vai progredindo. Estes elementos incentivam a entreajuda e conferem aos utilizadores uma sensação de recompensa, que é bastante real dada a credibilidade e popularidade que a plataforma tem vindo a ganhar entre profissionais [63]. É ainda na reputação que se baseia o sistema de moderação da rede, pois a integridade da plataforma é mantida pelos próprios utilizadores, que vão ganhando mais poder de moderação quanto maior a sua reputação [64].

2.3 Síntese

A seguinte tabela resume os aspetos mais importantes das plataformas anteriormente descritas, com o objetivo de comparar e colocar em evidência os conceitos que serão utilizados pela plataforma AnyKB.

Plataforma	Motivação do autor da questão/artigo	Motivação do autor da resposta	Conceitos interessantes
Bloque	Descoberta de um novo conceito ou solução	Geralmente não se aplica	Autor da questão ou artigo escreve por vontade própria sem limitações ou exigências externas
Fórum	Obter uma solução ou opinião por parte de outros utilizadores	Vontade de ajudar	Incentivo à colaboração entre utilizadores para a construção de uma resposta mais ideal
Rede StackExchange	Obter uma solução por parte de outros utilizadores	Vontade de ajudar e a possibilidade de se destacar como profissional dentro da sua comunidade	Incentivo baseado em pontos e medalhas

Tabela 3 – Tabela síntese das plataformas para solução de problemas analisadas.

3 Implementação

A plataforma AnyKB, sendo uma aplicação *web*, é composta por dois componentes principais – cliente e servidor - fisicamente separados, interligados através de um *webservice* REST [65].

Numa abordagem clássica, cada pedido ao servidor proveniente da aplicação cliente (*browser*) resulta não só na transferência da informação interessante ao pedido do cliente, mas também da estrutura e apresentação do documento (HTML e CSS), bem como qualquer lógica de apresentação que o documento possa conter (JavaScript). A plataforma AnyKB não utiliza esta abordagem para obter os vários elementos da plataforma a partir do servidor. É neste aspeto que o projeto pretende diferenciar-se pelo desempenho elevado da aplicação, ao implementar um conjunto de otimizações que fazem uso intensivo da tecnologia Local Storage.

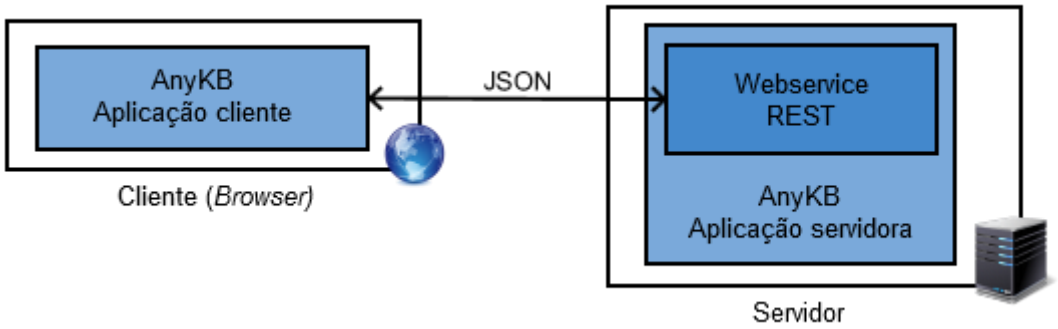


Figura 9 – Diagrama alto nível da plataforma AnyKB.

Nesta secção é detalhado o problema da *web* não persistente, e são apresentados detalhes de implementação das várias componentes da plataforma AnyKB.

3.1 O problema da *web* não persistente

As aplicações web executadas no *browser* têm como característica a completa dependência de um servidor que fornece toda a lógica da aplicação. Isto significa que em cada acesso o *browser* tem de descarregar todos os elementos da aplicação, independentemente de qualquer acesso anterior. Esta redundância de dados que circulam na rede agrava-se ao verificarmos que a componente visual (HTML, CSS & JavaScript) não serve qualquer interesse ao servidor. O ideal seria persistir estes elementos no *browser* no primeiro acesso, eliminando a

necessidade de nova transferência em acessos posteriores.

Os sistemas de cache implementados pelos *browsers* assistem ao guardar localmente alguns elementos, no entanto apenas reduzem a quantidade de informação trocada, não eliminando por completo a transferência da componente visual da aplicação. Adicionalmente, não existe uma interface programática que permita a gestão deste recurso.

A utilização de AJAX é outra técnica que permite a redução da quantidade de dados transferidos ao possibilitar ligações assíncronas ao servidor. É possível utilizar esta técnica para, por exemplo, descarregar a componente visual da aplicação na abertura da página web, e posteriormente utilizar ligações AJAX para obter apenas o resultado do servidor. No entanto introduz-se um novo problema: cada nova abertura da aplicação torna-se mais lenta e pesada. Para além disso, o problema inicial mantém-se pois continuamos a depender do servidor para fornecer a componente visual da aplicação em cada novo acesso.

O Local Storage vem oferecer uma solução mais completa ao permitir que o programador persista no *browser* todos os elementos de apresentação da aplicação, eliminando completamente a transferência desta componente a partir do servidor. Excetuam-se apenas dois casos incontornáveis: quando ainda não foi persistida e quando existem atualizações. Pode ser feita uma analogia a aplicações nativas: o primeiro acesso corresponde à "instalação", existindo "atualizações" posteriores apenas quando necessário. Após "instalada", a aplicação comunica com o servidor utilizando AJAX, consistindo a comunicação apenas da informação relevante ao pedido do utilizador, geralmente estruturada recorrendo a JSON [66] ou XML. Esta é a abordagem utilizada pela plataforma AnyKB para obter os vários elementos da aplicação e toda a informação associada a partir do servidor.

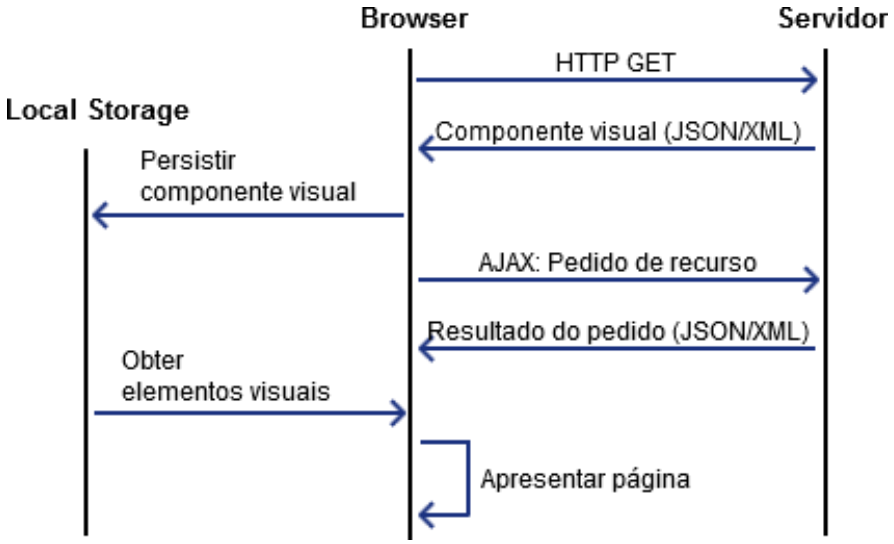


Figura 10 - Diagrama da interação entre *browser* e servidor utilizando Local Storage para persistência da componente visual da aplicação.

3.2 Componentes da aplicação

Neste capítulo apresentamos os detalhes de implementação da plataforma AnyKB, agrupados primariamente pelas componentes cliente e servidor.

3.2.1 Servidor

O servidor utilizado para servir a plataforma AnyKB trata-se de uma máquina virtualizada recorrendo a Xen [67], assente em infraestrutura privada escalável de última geração, recorrentemente referenciada como *Cloud*. O servidor apresenta as seguintes características:

- **CPU:** Quad-core Intel® Xeon® CPU E5649 @ 2.53GHz
- **RAM:** 1.5GB DDR3 @ 1033MHz
- **Disco:** 25GB (10K)

Abaixo está listado o *software* instalado no servidor e utilizado pela plataforma AnyKB:

- CentOS Linux 6.4 (64-bit) [68]
- Apache Web Server 2.2.15 (64-bit) [69]
- PHP 5.3.3 (64-bit) [70]
- MySQL 14.14 (64-bit) [71]
- Apache Solr 4.3.0 (64-bit) [72]

Figura 11 – Hardware de suporte à máquina virtual utilizada: SAN Dell Equalogic e Powervault (cima) e central de processamento composta por servidores Dell Blade (baixo).

REST Webservice

A separação dos elementos e da lógica de apresentação dos restantes componentes da aplicação requer uma abordagem distinta de acesso ao servidor. Deste modo, foi desenvolvido um *webservice* REST capaz de obter, criar, alterar e apagar a informação associada à aplicação. Um *webservice* REST é uma aplicação servidora que expõe um conjunto de métodos remotamente acessíveis através do protocolo e métodos HTTP, identificados pelo URI [65].

No caso específico da plataforma AnyKB, o *webservice* assenta na linguagem de programação PHP para a implementação da lógica da aplicação, e no Apache como servidor *web* capaz de interpretar e redirecionar os pedidos (URI's) para o controlador PHP responsável. Este *webservice* é responsável por fornecer os métodos que possibilitam a aplicação cliente obter, criar, editar as entidades que constituem a aplicação, como artigos, utilizadores, votos, favoritos, etc.

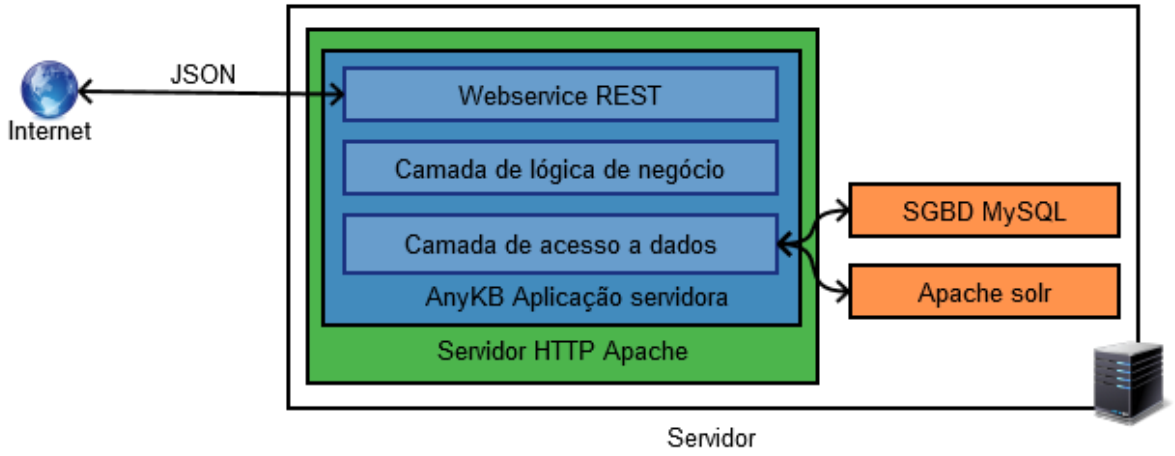


Figura 12 – Diagrama da componente servidora da plataforma AnyKB.

A especificação técnica completa do *webservice* desenvolvido está disponível no anexo "7.1 Documentação do *webservice* REST" deste documento.

Base de Dados

Por forma a persistir toda a informação da aplicação no servidor foi utilizado o SGBD MySQL em conjunto com o mecanismo de armazenamento InnoDB [73] pelo suporte a chaves estrangeiras, o que auxilia a manutenção da integridade da base de dados.

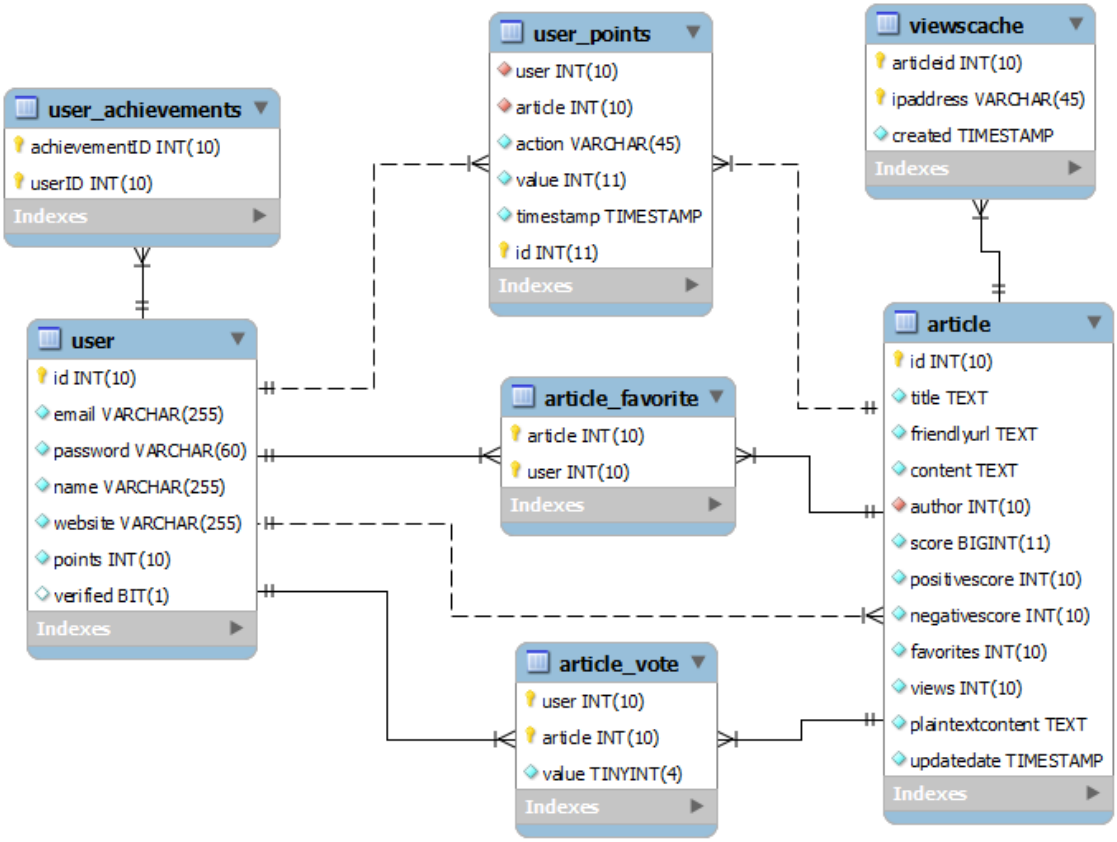


Figura 13 – Diagrama entidade relacionamento da base de dados da plataforma AnyKB.

A base de dados não se encontra normalizada na medida em que foi mantida alguma redundância, como é o caso do campo "*points*" da tabela "*user*" que consiste no valor total dos pontos igualmente presente na tabela "*user_points*". Esta redundância foi criada de forma controlada com o objetivo de obter ganhos de desempenho na obtenção de dados, ao eliminar a necessidade de calcular repetidamente os mesmos campos em cada acesso. A integridade da redundância existente é mantida pela própria base de dados recorrendo a *triggers*.

Motor de Pesquisa e Indexação de Artigos

A plataforma AnyKB possui um sistema de pesquisa que pretende facilitar a descoberta de artigos, utilizando a escrita de termos relacionados com o conteúdo pretendido no processo de pesquisa. É utilizado o motor de pesquisa e indexação Apache Solr neste processo, por forma a otimizar o tempo de resposta desta funcionalidade.

Apache Solr, tal como descrito no site oficial do projeto, é "*(...) a popular, super rápida plataforma open source de pesquisa baseada no projeto Apache Lucene.*" [72] [74]. Esta ferramenta disponibiliza um *webservice* acessível através do protocolo HTTP para a consulta de pesquisas e processos de gestão da plataforma. No caso concreto do projeto AnyKB, esta ferramenta está configurada por forma a indexar periodicamente (num espaço de 5 segundos

após a última indexação, repetidamente) novos artigos introduzidos na base de dados da plataforma e fornecer funcionalidades de pesquisa de artigos com destaque para os termos pesquisados, apresentado os excertos mais relevantes para a pesquisa, bem como de classificação e organização dos artigos por relevância.

A seguinte tabela coloca MySQL e Apache Solr lado a lado numa comparação de desempenho. Os dados estatísticos foram recolhidos ao efetuar várias pesquisas numa base de dados com 50 000 registos (345 MB) devidamente otimizada. O computador utilizado consiste numa máquina virtual com as seguintes características:

- **CPU:** Quad-core Intel® Xeon® CPU E5649 @ 2.53GHz
- **RAM:** 1.5GB DDR3 @ 1033MHz
- **Sistema Operativo:** CentOS Linux 6.4 (64-bit) [68]

N.º de pesquisas	**MySQL**		**Apache Solr**	
	Tempo total	Média por pesquisa	Tempo Total	Média por pesquisa
5	9848 ms	1970 ms	125 ms	25 ms
25	41359 ms	1654 ms	213 ms	9 ms
50	92170 ms	1843 ms	240 ms	5 ms
100	202192 ms	2022 ms	385 ms	4 ms

Tabela 4 – Estatísticas de desempenho MySQL e Apache Solr.

É possível visualizar através da tabela anterior o ganho de desempenho óbvio conseguido pela utilização de Apache Solr, comparativamente a MySQL. Os ganhos em tempo total variam entre um mínimo de 79 vezes a um máximo de 525 vezes mais rápido, numa média de 296 vezes mais rápido. Em média os ganhos são equivalentes, variando entre um mínimo de 79 vezes a um máximo de 505 vezes mais rápido, numa média de 284 vezes mais rápido. Esta ferramenta mostra-se assim fulcral de entre um conjunto de opções para o desempenho da plataforma, tomadas ao longo de todo o projeto.

3.2.2 Cliente

A aplicação cliente é a cara da plataforma AnyKB, e é executada no *browser* do utilizador. É esta aplicação que apresenta o interface gráfico que permite o utilizador visualizar e gerir o conteúdo da plataforma. Como descrito anteriormente, todos os aspetos visuais e de lógica da aplicação cliente são descarregados para o *browser* no primeiro acesso, o que significa que apenas é obtido do servidor a informação relevante ao pedido do utilizador. De seguida estão detalhados o interface gráfico da aplicação e otimizações adicionais implementadas por forma a melhorar o desempenho da aplicação.

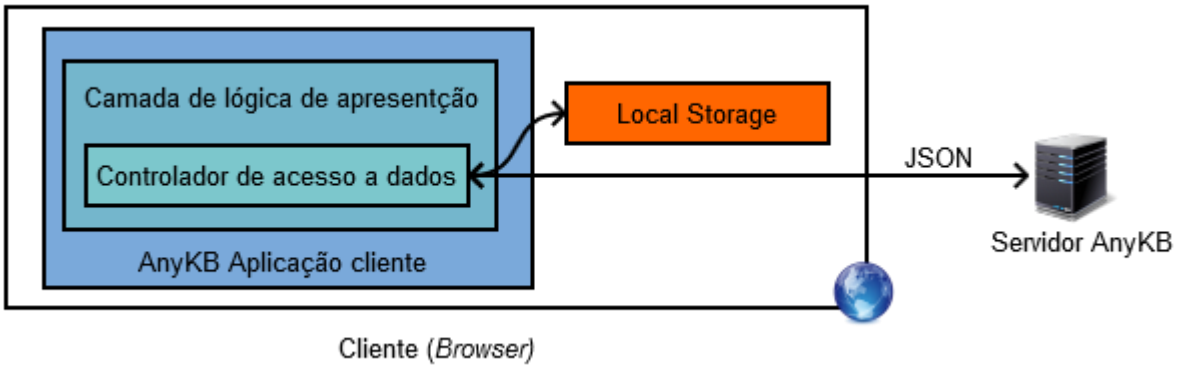


Figura 14 – Diagrama da componente cliente da plataforma AnyKB.

Interface do utilizador (*user interface*)

Neste subcapítulo são apresentados e descritos os interfaces gráficos desenvolvidos no âmbito da aplicação cliente da plataforma AnyKB, orientados para o computador pessoal (interface *desktop*) e para dispositivos móveis (interface *móvel*).

A interface gráfica da aplicação cliente tem como objetivos a simplicidade, consistência e fluidez. O utilizador deve ser capaz de identificar exatamente o que pretende ao longo de todas as páginas da plataforma, sem ser distraído por qualquer elemento visual que não tenha um objetivo funcional. Deste modo, todas as páginas estão primariamente divididas em duas secções: cabeçalho, e conteúdo. A seguinte imagem auxilia na contextualização visual destas duas secções.

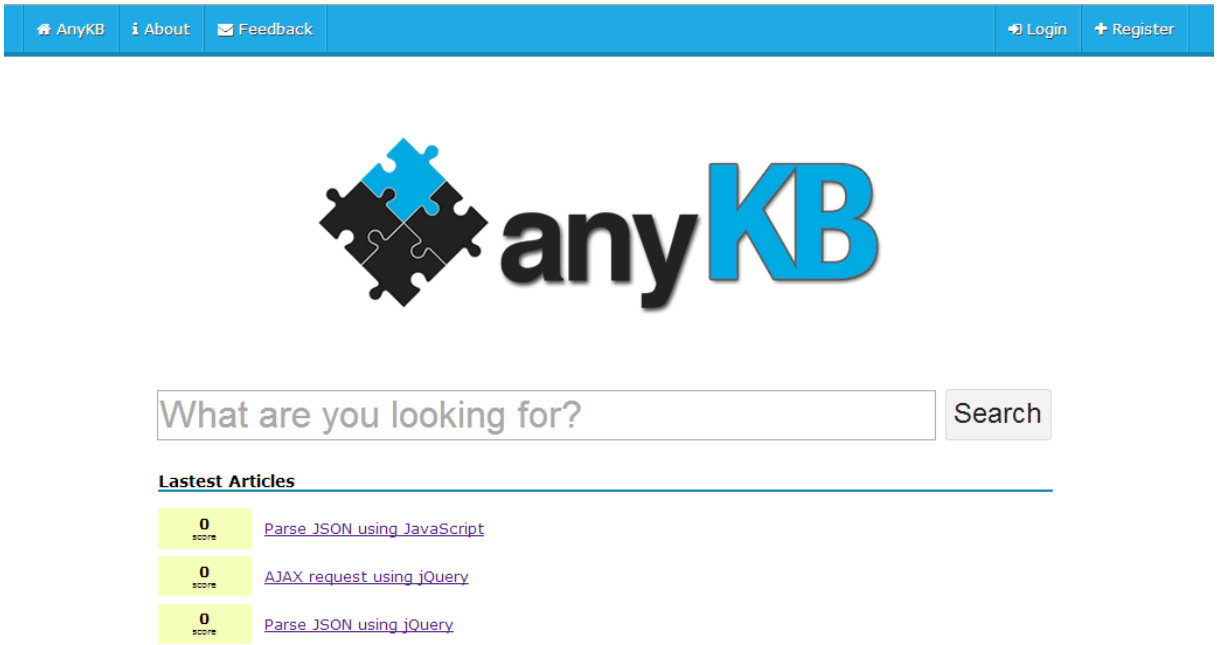


Figura 15 – Página principal da plataforma AnyKB (versão *desktop*).

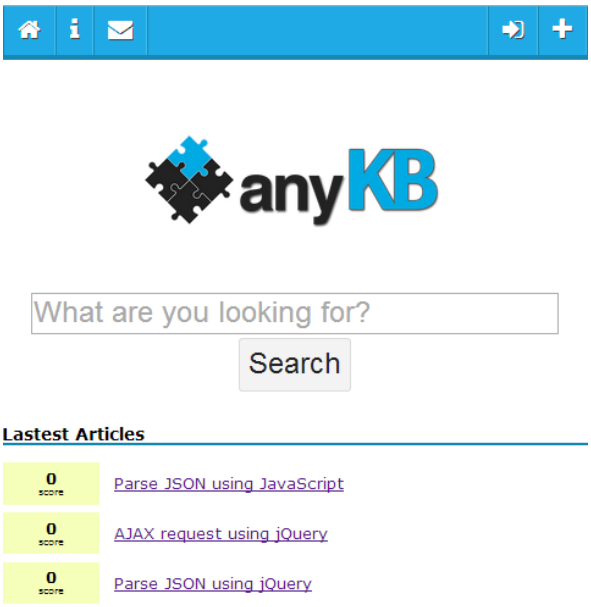


Figura 16 - Página principal da plataforma AnyKB (versão móvel).

As imagens acima correspondem à página principal da plataforma AnyKB, onde é visível a divisão anteriormente descrita. A barra de cor azul, no topo, faz parte do cabeçalho da aplicação e acompanha o utilizador ao longo de todas as páginas. O cabeçalho contém informação que se considera globalmente útil no contexto aplicacional, bem como *links* para as várias áreas do site, constituído assim também o menu principal da aplicação. A área abaixo do cabeçalho corresponde ao conteúdo da aplicação. Esta área poderá ser ainda divida em subáreas e contém todo o conteúdo específico a um determinado pedido do utilizador.

A versão móvel mostra um cabeçalho alterado como forma de adaptação ao tamanho reduzido do dispositivo móvel. É possível verificar que se mantêm apenas os ícones das várias opções do menu, que tentam ser o mais explícitos possível, e foi removida a opção de criação de novo artigo. Esta funcionalidade não mostrou qualquer problema ao nível do funcionamento em dispositivos móveis, no entanto a opção foi removida por se considerar o dispositivo móvel um dispositivo primariamente para o consumo de conteúdo, não estando o tamanho reduzido do ecrã apto à criação de conteúdo devidamente estruturado.

De forma global, a página principal é apenas constituída pelo logótipo da aplicação, um formulário de pesquisa e uma listagem dos últimos artigos criados, colocando em evidência a forma principal de navegação e descoberta de artigos dentro da plataforma.

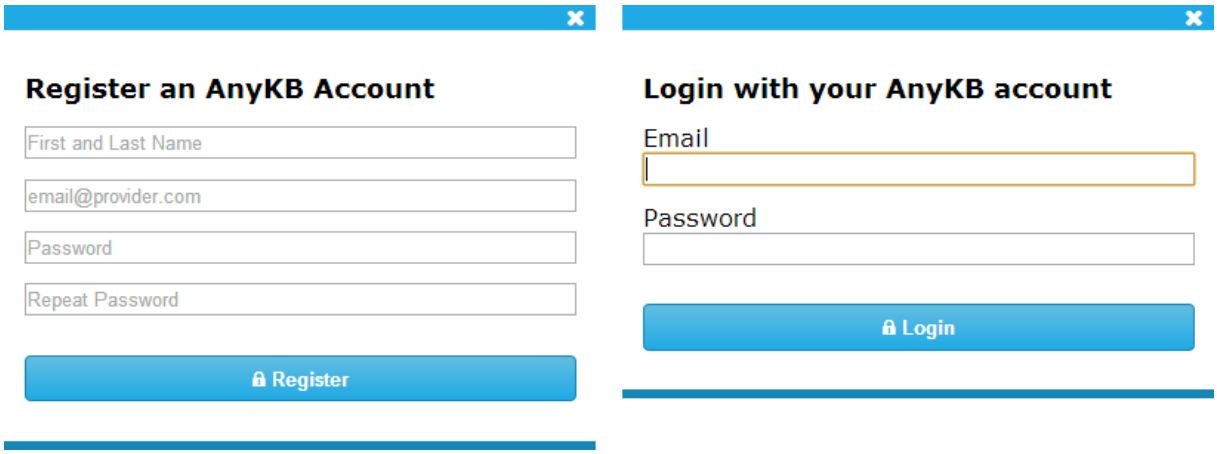


Figura 17 – *Popups* de registo, à esquerda, e autenticação na plataforma, à direita (versão *desktop* e móvel)

A imagem acima apresenta a única forma alternativa de apresentação de conteúdo, utilizada para mostrar ou pedir informação transitória ao utilizador, que não obriga a navegação para uma página completamente nova. Este formato tem a denominação de *popup* e apresenta-se em forma de pequena caixa que sobrepõe o conteúdo da aplicação, desativando qualquer interação com os restantes elementos da aplicação enquanto o *popup* estiver visível no ecrã. Na plataforma AnyKB os *popups* são utilizados para mostrar os formulários de registo e autenticação, não impedindo que o utilizador continue a visualizar qualquer página após efetuar uma destas ações.

As seguintes imagens correspondem à página de resultados de pesquisa, que seguindo o objetivo de simplicidade apresenta apenas os elementos considerados necessários e importantes para que o utilizador possa escolher o artigo que ache de maior interesse. Assim, é possível visualizar o título do artigo, pequenos excertos do texto com ênfase para os termos de pesquisa, e finalmente a pontuação (ou *score*) do artigo.

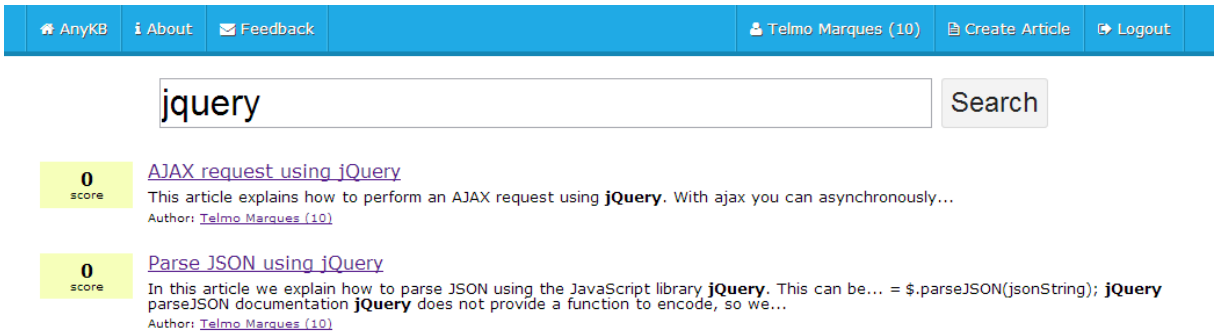


Figura 18 – Página de resultados da pesquisa (versão *desktop*).

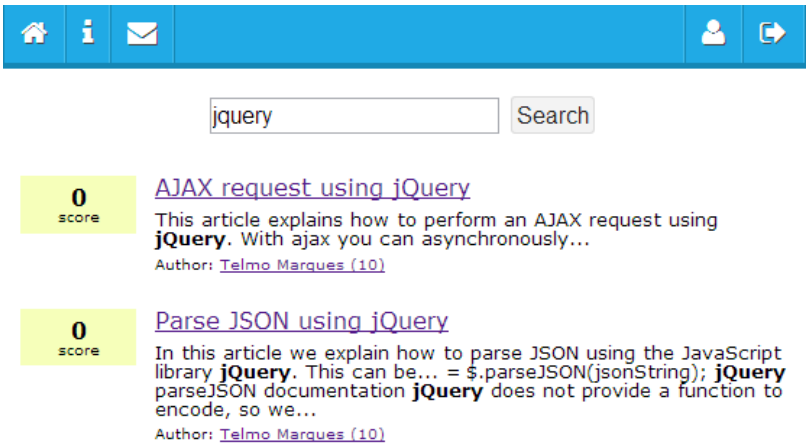


Figura 19 - Página de resultados da pesquisa (versão móvel).

As imagens abaixo representam a página de visualização de artigo, que apresenta todo o conteúdo e informação adjacentes ao artigo selecionado. Para além do título e conteúdo do artigo, que constituem os elementos principais da página, é possível analisar o autor do artigo, os votos dados ao artigo pela comunidade, quantas vezes foi adicionado aos favoritos e quantas vezes foi visualizado. Como ações é possível votar, adicionar aos favoritos e editar o artigo. Estes dados e ações relacionadas estão agregados numa caixa (à direita) que acompanha sempre o utilizador à medida que se faz *scroll* no artigo. Na versão móvel esta caixa está disponível no fundo da página como adaptação às dimensões reduzidas do dispositivo móvel. Adicionalmente, a opção de edição de artigo não se encontra disponível pela mesma razão que levou à remoção da opção de criação de artigo, descrita no início deste subcapítulo.

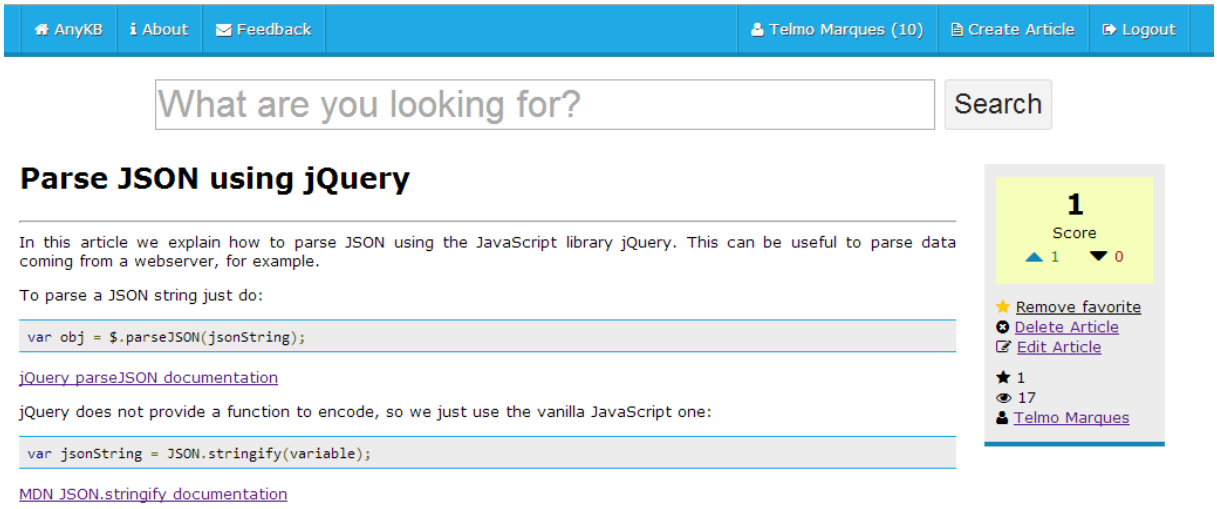


Figura 20 – Página de visualização de artigo (versão *desktop*).

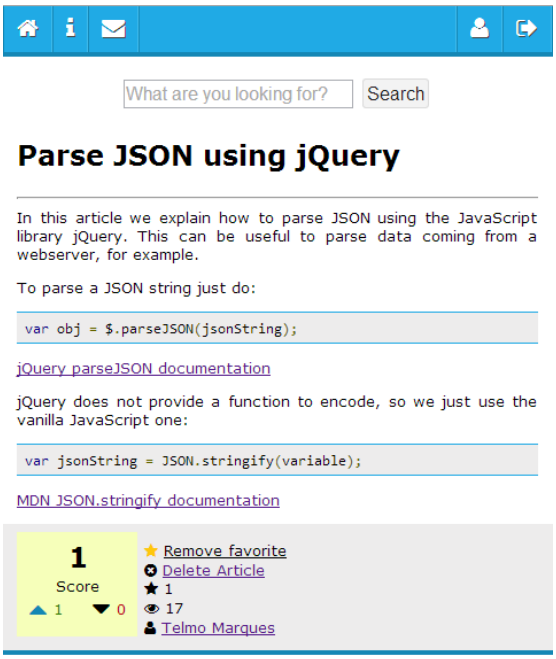


Figura 21 - Página de visualização de artigo (versão móvel).

Ao aceder às opções de criar ou editar um artigo, através da opção presente no menu superior ou página de visualização de artigo, a página visível na seguinte imagem é apresentada. Esta contém apenas dois elementos: à esquerda uma área de texto onde o utilizador poderá redigir o seu artigo, e à direita uma pré-visualização instantânea do resultado da redação.

Figura 22 – Página de criação ou edição de artigo (versão *desktop*).

O conteúdo redigido é formatado recorrendo a *Markdown* [75], uma linguagem de marcação para conteúdo em texto puro (*plain text)* que permite a sua conversão posterior para HTML

equivalente e pré-visualização imediata da estrutura e apresentação do artigo a ser redigido. A caixa de redação (à esquerda) disponibiliza um conjunto de controlos que auxiliam na estruturação em *Markdown* do documento.

Finalmente, a última área funcional da plataforma designa-se de perfil, e permite o utilizador gerir as suas informações pessoais, bem como visualizar informação pública dos restantes utilizadores da plataforma. As seguintes imagens representam esta área da plataforma.

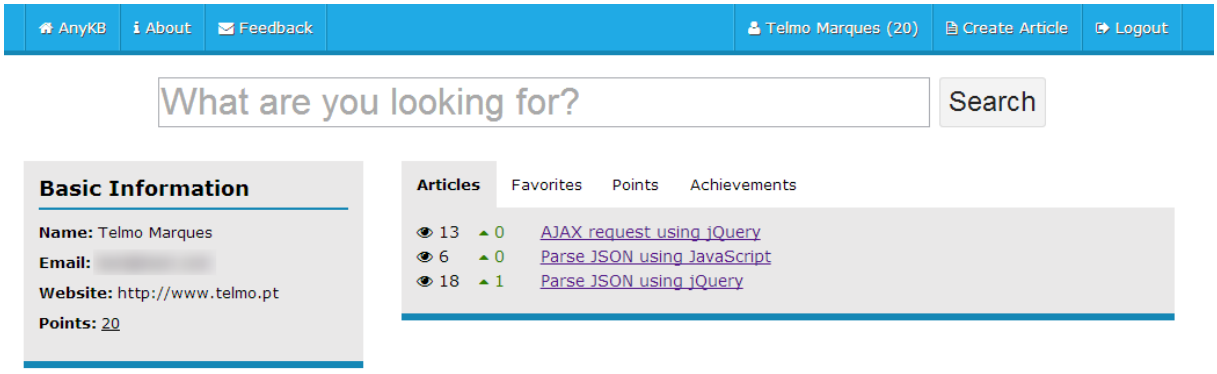


Figura 23 – Página de perfil de utilizador da plataforma (versão *desktop*).

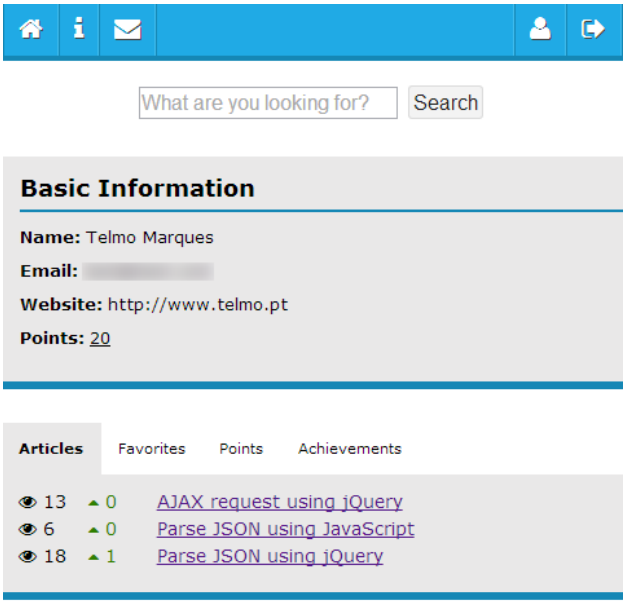


Figura 24 - Página de perfil de utilizador da plataforma (versão móvel).

Para além de informação pessoal, como nome, email e página *web* pessoal, é possível ainda ser verificado o histórico de artigos criados e favoritos, bem como de pontos e *achievements*, que constituem o produto final do sistema de recompensa da plataforma.

Sincronização do *template* e modelo de dados

Como anteriormente indicado, a componente visual da aplicação (denominada *template*) é obtida pelo *browser* no primeiro acesso à aplicação, sendo persistida recorrendo ao Local Storage para utilização em posteriores acessos. Deste modo, o *webservice* anteriormente detalhado implementa um método específico que permite a obtenção desta componente. A lógica de implementação da persistência e apresentação do *template* é semelhante à apresentada na seguinte listagem.

```
//Ponto de entrada na aplicação.
//Quando o documento estiver pronto...
$(document).ready(function()
{
    //Se o template já existe no browser...
    if(templateExists())
    {
        //... mostrar aplicação
        showApplication();
    }
    else
    {
        //Senão, fazer download do template
        getTemplate(function()
        {
            //Quando concluído, mostrar aplicação
            showApplication();
        });
    }
});

/**
Verificar se o template existe no browser
@method templateExists
**/
```

```
function templateExists()
{
    //Neste caso verificamos a existência do parâmetro "version" como teste da presença do template
    return localStorage.getItem("version") !== null;
}

/**
Obter template e guardar em Local Storage
@method getTemplate
@param {function} callback Função chamada após terminado o download do template
**/
function getTemplate(callback)
{
    //Fazer pedido ao servidor pelo template
    $.ajax(
    {
        url: "api.php?getTemplate=1",
        type: "get",
        success: function(data)
        {
            //Resposta JSON
            //Fazer parse do resultado do servidor
            var jsonData = $.parseJSON(data);
            //Iterar os elementos enviados pelo servidor
            $.each(jsonData, function(key, value)
            {
                //Guardar em Local Storage
                localStorage.setItem(key, value);
            });
```

```
            //Chamar função callback
            callback();
        }
    });
}

/**
Carregar pela primeira vez os elementos visuais para o ecrã
@method showApplication
**/
function showApplication()
{
    //Carregar folha de estilos do template
    $("style").append(localStorage.getItem("css_styles"));
    //Mostrar página inicial (index)
    changePage("index", true);
}

/**
Carregar uma determinada página guardada em localStorage
@method changePage
@param {String} pageName Nome da página a carregar
@param {boolean} replaceState Indica se deve ser criado ou reutilizado um estado do histórico
**/
function changePage(pageName, replaceState)
{
    //Carregar para o corpo do documento o conteúdo presente em Local Storage correspondente à página indicada
    $("body").html(localStorage.getItem("html_"+pageName));
    //Construir objecto de estado stateObject, que indica o nome da página destino
    var stateObject = {page: pageName};
```

```
    //Se foi indicado que o estado deve ser substituído...
    if(replaceState)
    {
        //...substituir o estado do histórico utilizando o objecto de estado stateObject
        history.replaceState(stateObject, null, pageName);
    }
    else
    {
        //Em caso contrário, criar um novo estado do histórico utilizando o objecto de estado stateObject
        history.pushState(stateObject, null, pageName);
    }
}
```

Listagem 1 – Lógica de implementação, em JavaScript, da persistência e apresentação do *template* da plataforma AnyKB.

A implementação concreta desta funcionalidade na plataforma AnyKB apresenta alguma complexidade adicional, relativamente à listagem anterior, ao implementar um número de versão que é utilizado para acompanhar as alterações ao *template*: sempre que a aplicação cliente detete, durante acesso à plataforma, que o número de versão do *template* localmente persistido difere da existente no servidor, é feita uma atualização. Desta forma é garantida a sincronização do *template* entre cliente e servidor.

Com a implementação deste conceito pretende-se conferir à plataforma um ganho significativo de desempenho e fluidez, no entanto é possível otimizar adicionalmente estes fatores ao manter uma *cache* local de parte do modelo de dados persistido no servidor. Estas otimizações baseiam-se na recorrência com que determinada informação é apresentada, e na probabilidade do utilizador aceder a essa informação no curto prazo. Caso a recorrência e/ou probabilidade de acesso à informação seja considerada elevada, o recurso torna-se candidato a ser persistido de forma temporária em *cache* local.

Deste modo, encontram-se implementadas as seguintes funcionalidades de sincronização do modelo de dados:

- A informação pessoal do utilizador é carregada para *cache* local aquando do processo de autenticação. Elementos pessoais do utilizador como o nome, pontos, artigos

favoritos, etc., são recorrentemente apresentados ao longo das várias páginas da plataforma. Adicionalmente, a página de perfil do utilizador apresenta todas estas informações. Deste modo é sacrificado parte do desempenho do processo de autenticação, compensado pela diminuição do número de acessos ao servidor dada a recorrência da informação.

- O conteúdo dos artigos do topo do resultado de uma pesquisa são imediatamente colocados em *cache* local. Os artigos são apresentados ao utilizador ordenados por relevância, significando que existe grande probabilidade que o artigo pretendido se encontre entre os primeiros resultados da pesquisa. Assim, enquanto o utilizador analisa os resultados apresentados um processo em *background* coloca um ou mais artigos em *cache* local, permitindo que o utilizador consiga visualizar o artigo completa de forma imediata.
    - A versão móvel da plataforma deixa as imagens, caso existam, de fora deste processo. Pretende-se assim poupar recursos na forma de tráfego de Internet.
- Os artigos visualizados são colocados em *cache* local. Deste modo, um utilizador que esteja a analisar vários artigos por forma a encontrar o que considere de maior relevância consegue voltar a qualquer um dos artigos anteriores de forma imediata.
    - A versão móvel da plataforma deixa as imagens, caso existam, de fora deste processo. Pretende-se assim poupar recursos na forma de tráfego de Internet. No entanto, o sistema de cache nativo do *browser* não é de qualquer modo invalidado.
- As imagens dos artigos são serializadas (através de codificação em Base64 [76]) e guardadas em *cache* local, juntamente com o conteúdo do artigo. Este é um processo necessário dado que em HTML as imagens são referenciadas através da sua localização, e não num formato compatível com a persistência imediata da imagem em si *inline* com o conteúdo do artigo.

Qualquer recurso que seja colocado em *cache* local implementa mecanismos que permitem a deteção de alterações no modelo de dados do servidor. No caso da plataforma AnyKB, sempre que o utilizador solicita um determinado recurso anteriormente *cached* a aplicação questiona o servidor por uma versão mais recente do recurso, atualizando o mesmo se necessário.

4 Discussão de Resultados

Neste capítulo são apresentados os resultados dos testes qualitativos e quantitativos efetuados à plataforma AnyKB, com o objetivo de avaliar o seu desempenho e fluidez geral, com especial enfoque na aplicação cliente (*browser*) onde incidiu o maior esforço de inovação.

4.1 Estudo sobre a escalabilidade da aplicação

Por forma a estudar a escalabilidade da plataforma AnyKB foram comparadas duas implementações equivalentes do mesmo projeto. A primeira, denominada "*Proof of Concept*" (ou PoC) implementa o conceito detalhado neste documento, e a segunda, denominada "clássica" implementa o formato clássico de interação com o servidor utilizando apenas pedidos HTTP GET ou POST. Por forma a serem igualmente equivalentes no que toca à lógica de negócio presente no servidor, ambas as plataformas utilizam exatamente o mesmo *webservice*, alterando apenas a forma como a aplicação cliente comunica com este último.

As características do servidor onde está presente o *webservice* desenvolvido são as mesmas que as descritas no capítulo "3.2.1 Servidor". O computador utilizado para executar a aplicação cliente tem as seguintes características:

- **CPU:** Intel® Core™ i3 CPU M330 @2.13GHz
- **RAM:** 6GB DDR3 @ 1066MHz
- **Disco:** 350GB (5.4K)
- **Sistema Operativo:** Windows 8
- ***Browser*:** Google Chrome 28

O desempenho das duas versões foi comparado ao criar um caso de teste que consiste na navegação pelas várias páginas da aplicação, de forma a tirar partido dos benefícios que ambas as abordagens têm ao seu dispor. A seguinte tabela enumera os vários passos do caso de teste executado e, onde aplicável, detalha a importância que determinada ação tem para a comparação em questão.

Passo N.º	Página	Descrição
1	Página inicial	Ponto de entrada na aplicação.

2	Sobre a aplicação	Página de conteúdo estático. A versão clássica pode beneficiar de sistemas de *cache* do *browser*, e a versão PoC do facto de ter a página completa guardada em Local Storage.
3	Página inicial	Página de conteúdo dinâmico. A versão clássica pode beneficiar de sistemas de cache do *browser*, e a versão PoC do facto de ter a estrutura, apresentação e lógica da página guardados em Local Storage.
4	Resultados de pesquisa	A pesquisa efetuada coloca como primeiro resultado um artigo de teste composto apenas por texto.
5	Artigo	Visualização do primeiro artigo que resultou da pesquisa anterior. A versão clássica pode beneficiar de sistemas de *cache* antecipada do *browser*, e a versão PoC do facto de guardar imediatamente o primeiro resultado em Local Storage.
6	Editar artigo	Permite a versão PoC beneficiar do facto de ter o conteúdo do artigo guardado em Local Storage para pré preencher mais rapidamente o formulário de edição.
7	Artigo	Visualização do artigo anteriormente editado após guardadas as alterações.
8	Resultados de pesquisa	A pesquisa efetuada coloca como primeiro resultado um artigo composto por texto e imagens.
9	Artigo	Visualização do primeiro artigo que resultou da pesquisa anterior. A versão clássica pode beneficiar de sistemas de *cache* antecipada do *browser*, e a versão PoC do facto de guardar imediatamente o primeiro resultado em Local Storage, incluindo imagens.
10	Resultados de pesquisa	A pesquisa efetuada mostra vários resultados de artigos compostos por texto.
11	Artigo	Visualização de um resultado que não o primeiro da

		pesquisa anterior. A versão clássica pode beneficiar de sistemas de *cache* antecipada do *browser*.
12	Envio de *Feedback*	Página de conteúdo estático. A versão clássica pode beneficiar de sistemas de *cache* do *browser*, e a versão PoC do facto de ter a página completa guardada em Local Storage.
13	Perfil do utilizador	Página de conteúdo dinâmico, com várias subpáginas organizadas em abas. Versão PoC beneficia do facto de carregar antecipadamente o conteúdo das várias abas, proporcionando uma transição entre abas imediata.

Tabela 5 – Caso de teste para a comparação quantitativa das versões clássica e PoC da plataforma AnyKB.

Várias estatísticas foram recolhidas através da execução do caso de teste descrito, nomeadamente de tempo de carregamento das páginas pelo *browser*, número de pedidos ao servidor, quantidade de dados trocados e quantidade de memória (RAM e disco) utilizada pela plataforma no *browser*. A métrica temporal utilizada corresponde ao tempo decorrido desde a ação do utilizador (clique num *link*, por exemplo) até ao carregamento completo da página pelo *browser*. Os valores foram recolhidos com o auxílio da ferramenta "Developer Tools" [77] disponível no *browser* Google Chrome. Os resultados médios e medianos apresentados consideram as 5 melhores execuções do caso de teste, para cada uma das versões.

4.1.1 Primeiro acesso

Como descrito anteriormente, do ponto de vista da versão PoC o primeiro acesso consiste na "instalação" da plataforma: *download* de toda a informação do interface do utilizador e persistência do mesmo utilizando a tecnologia Local Storage. Como este processo acontece apenas no primeiro acesso e é exclusivo à versão PoC, analisamos as estatísticas sobre o mesmo de forma independente dos acessos posteriores.

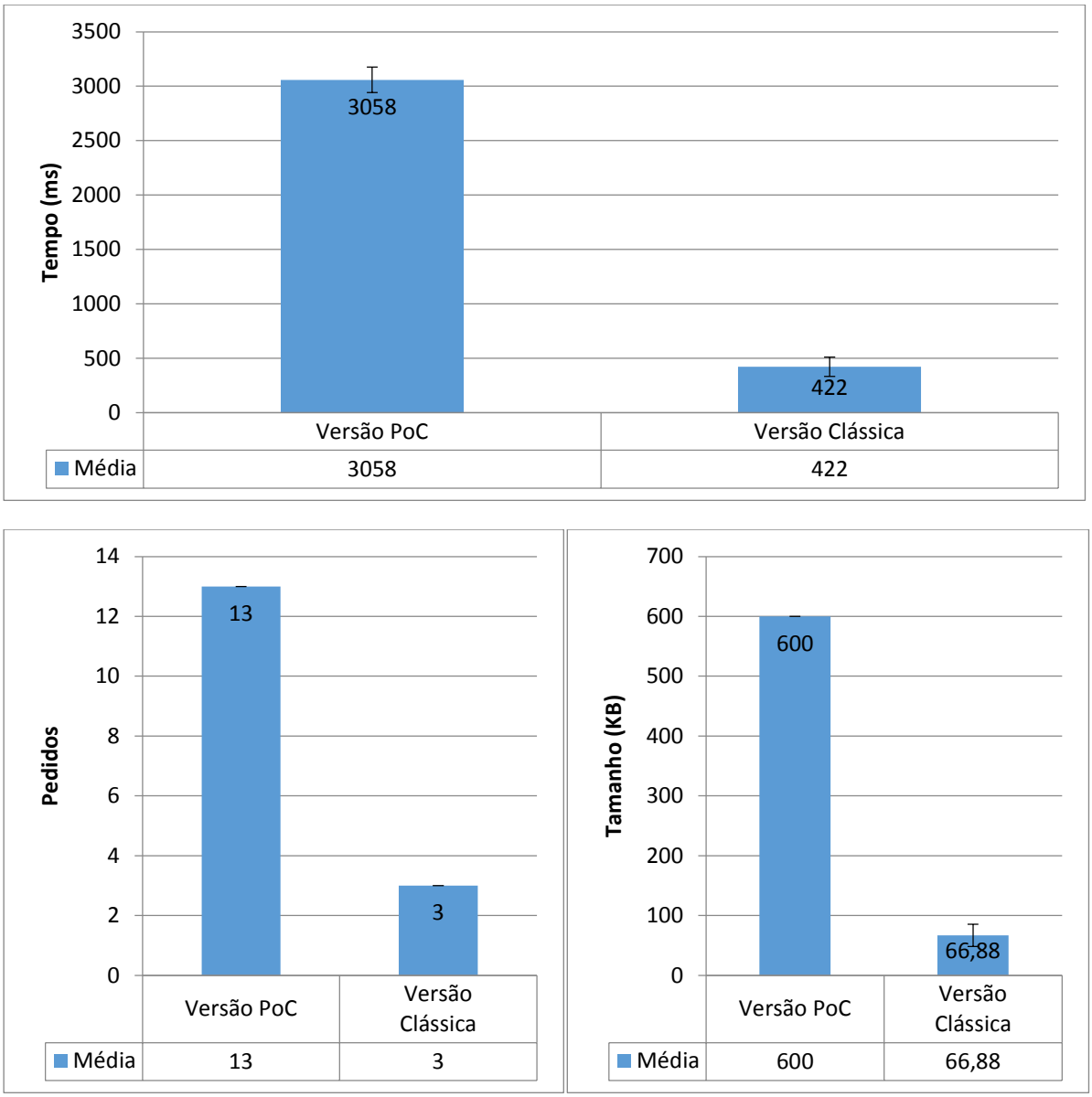


Gráfico 1 – De cima para baixo e da esquerda para a direita: Média do tempo (em milissegundos) de carregamento da página, número de pares pedido/resposta ao servidor e tamanho (em kilobytes) desses pedidos.

Como é possível analisar no gráfico acima, à esquerda, o primeiro acesso é substancialmente mais demorado na versão PoC comparativamente à clássica. Este aumento de tempo em cerca de 7.2 vezes deve-se à quantidade superior de dados que é necessário descarregar do servidor, no entanto o tempo obtido de 3 segundo é ainda assim inferior à média global de 6 segundos no carregamento de uma página *web* [78]. Relativamente ao número de pares pedido/resposta ao servido, estes são cerca de 4 vezes superiores, resultado em cerca de 9 vezes mais dados transferidos do servidor, comparativamente à versão clássica. São valores comparativamente elevados, mas que ainda assim se mantêm inferiores à média global de 42.63 [79] pedidos por página, e superior em apenas 2 vezes à média global de 312.04KB de dados trocados por página [79]. Estes fatores refletem-se principalmente no tempo de carregamento da página,

sendo possível concluir que apesar das estatísticas serem comparativamente superiores, o tempo de espera está dentro do considerado normal na *web* hoje em dia.

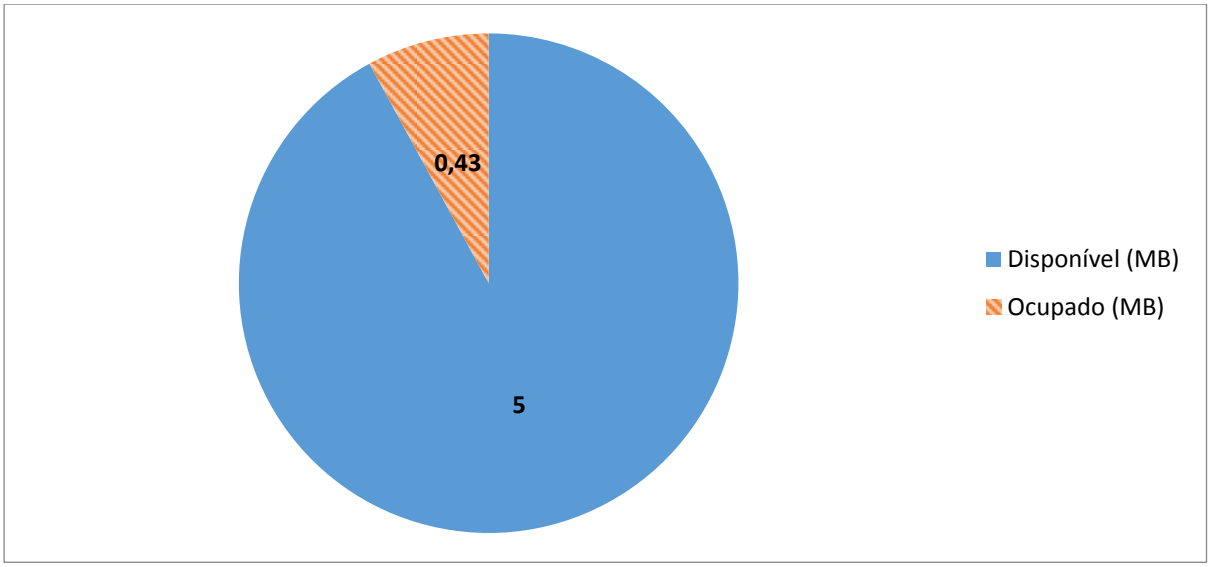
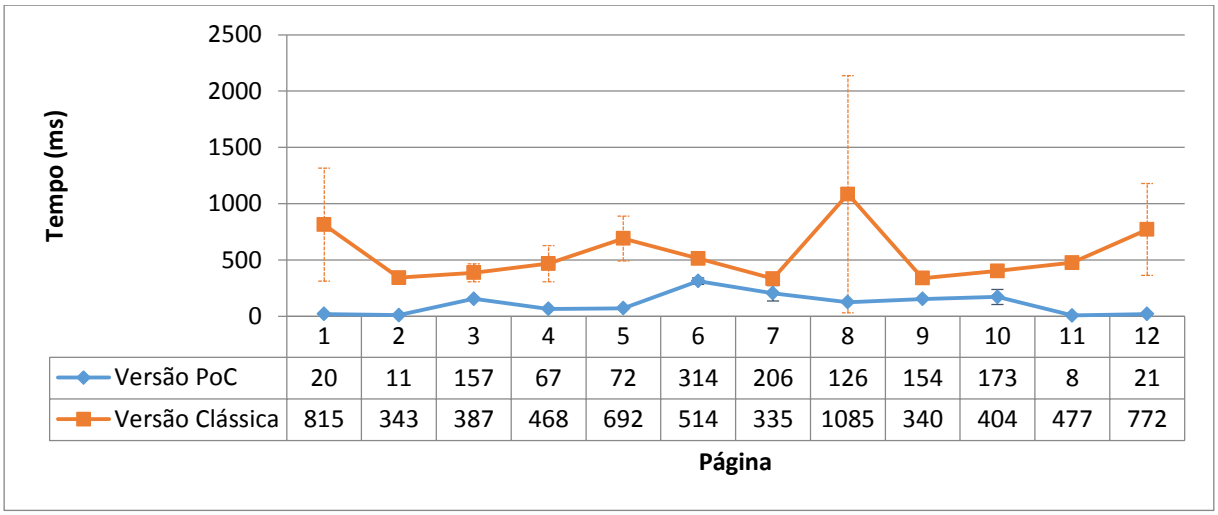


Gráfico 2 – Espaço total disponível e utilizado recorrendo à tecnologia Local Storage.

Foi também analisada a percentagem de espaço utilizado dos 5MB disponibilizados pelo Local Storage, visível no gráfico acima, onde se pode verificar uma utilização de apenas 8.6% (0.43MB) do espaço disponível. Este valor ficou aquém do previsto para a ocupação de toda a informação do interface de utilizador, que se esperaria bastante superior. Conclui-se assim que o espaço disponibilizado pelo Local Storage é suficiente para a persistência de interfaces gráficas de complexidade média-elevada.

4.1.2 Acessos subsequentes

Após concluída a fase de "instalação" da versão PoC é possível comparar diretamente ambas as plataformas por se considerarem em pé de igualdade. À semelhança do subcapítulo anterior, começamos por analisar o tempo de carregamento da plataforma. No gráfico abaixo está descriminado o tempo médio de carregamento de cada uma das páginas do caso de teste para ambas as versões.



	1	2	3	4	5	6	7	8	9	10	11	12
Versão PoC	20	11	157	67	72	314	206	126	154	173	8	21
Versão Clássica	815	343	387	468	692	514	335	1085	340	404	477	772

Gráfico 3 – Tempo médio (em milissegundos) e desvio padrão do carregamento de várias páginas de ambas as versões.

É possível verificar que o tempo de carregamento de cada página da versão PoC é sempre inferior comparativamente à versão clássica, atingindo mínimos de 1.63 vezes e máximos de 56.06 vezes mais rápido. As transições n.º 1, 2, 11 e 12 são transições para páginas de conteúdo estático (i.e. todo o conteúdo está persistido no *browser*), por esse motivo encontramos as maiores distâncias entre os valores de ambas as plataformas exatamente nestes pontos. É possível ainda verificar um desvio padrão bastante elevado para a versão clássica na transição n.º 8, que compreende a transição para uma página constituída por texto e imagens. Este valor elevado deve-se à diferença verificada no tempo no carregamento antes e depois do *browser* colocar a página em cache. Tratando-se de uma página relativamente pesada o primeiro acesso é consideravelmente mais demorado que os posteriores, onde o *browser* poderá então recorrer à cache.

Para avaliar como o tempo de carregamento das páginas é influenciado pela carga do servidor, que por sua vez representa uma utilização mais ou menos pesada da plataforma, foi simulada carga artificial, composta por operações de leitura e escrita para o disco e ocupação da CPU. Posteriormente, o caso de teste acima descrito foi executado em vários destes cenários, tendo sido o servidor levado ao limiar da capacidade de resposta. Os valores de carga apresentados correspondem ao formato conhecido como "Unix CPU Load" [80].

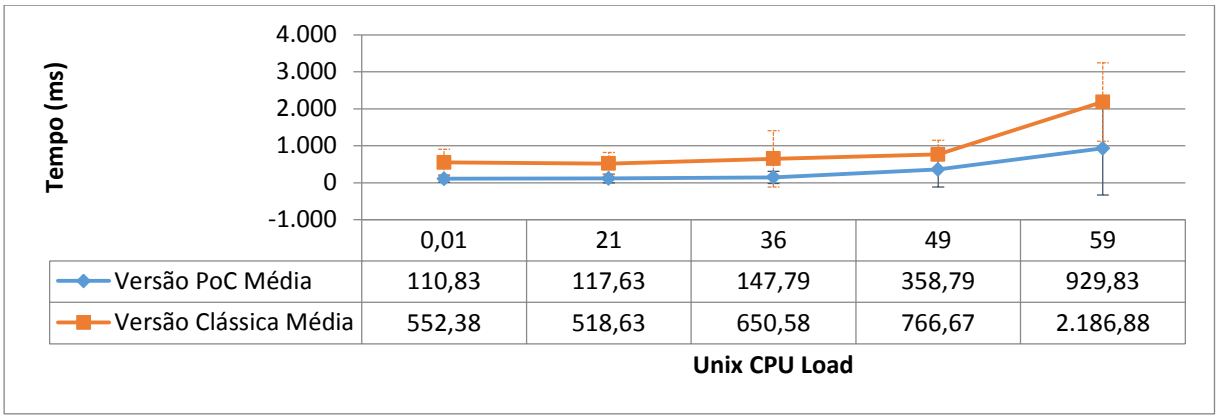


Gráfico 4 – Tempo médio (em milissegundos) e desvio padrão do carregamento de várias páginas de ambas as versões, ao longo de vários cenários de carga do servidor.

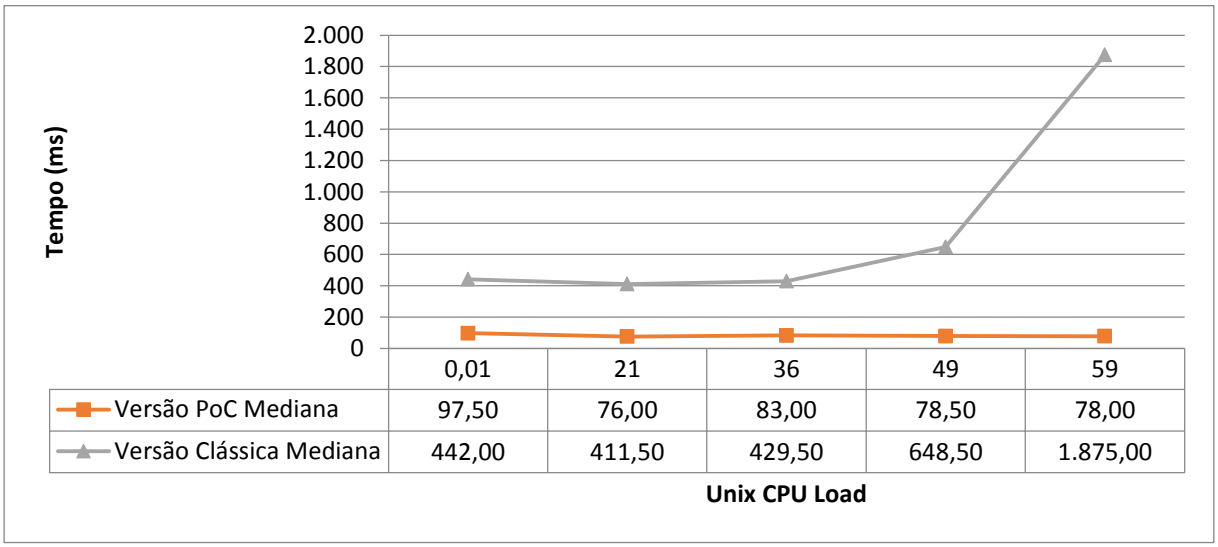


Gráfico 5 - Tempo mediano (em milissegundos) e desvio padrão do carregamento de várias páginas de ambas as versões, ao longo de vários cenários de carga do servidor.

Os gráficos acima mostram a evolução do tempo médio e mediano do carregamento das páginas da aplicação, ao longo dos vários cenários de carga. Como esperado, é possível verificar que o valor médio e mediano da versão clássica cresce à medida que a carga do servidor aumenta. O valor médio da versão PoC cresce igualmente com a carga do servidor, mas mostra uma curva mais suave, enquanto que a mediana da versão PoC se mantém estável em cerca de 78ms. Isto acontece porque existe uma compensação do tempo que os dados demoram a chegar do servidor pela instantaneidade de reposta do interface, e pela redução substancial de dados trocados com o servidor. Em termos mais genéricos, o gráfico abaixo representa a evolução do *speedup* médio e mediano conseguido pela versão PoC relativamente à clássica, ao longo dos vários cenários de carga.

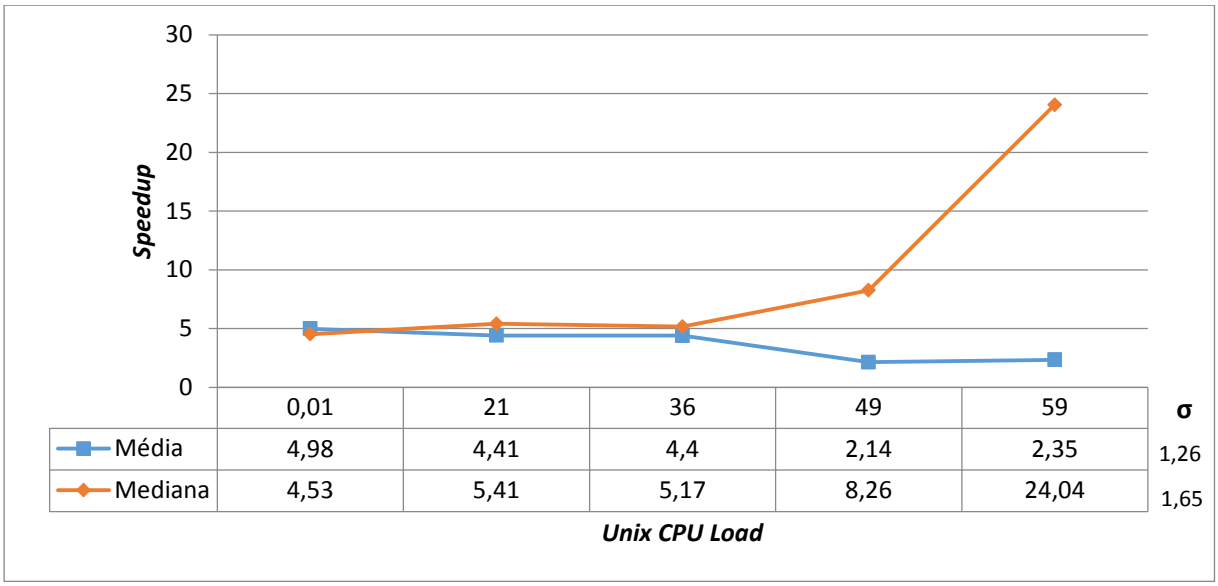


Gráfico 6 - *Speedup* baseado nos valores médios e medianos do tempo de carregamento global das páginas de ambas as versões. Resultados mostram o ganho da versão PoC relativamente à versão clássica, ao longo de vários cenários de carga do servidor.

É possível verificar que em condições de carga normais a versão PoC consegue um desempenho cerca de 4 a 5 vezes superior. O desempenho baixa para cerca de 2 a 3 vezes superior à medida que a carga do servidor aumenta, no entanto a mediana mostra-nos a tendência que a versão PoC tem para um desempenho mais elevado, conseguindo resultados até cerca de 24 vezes superiores.

Utilizando ainda o caso de teste acima descrito, foram também obtidas estatísticas sobre o número de pares pedido/resposta feitos pela aplicação cliente ao servidor, e do tamanho total do carregamento de cada página. Os seguintes gráficos mostram os dados recolhidos sobre ambas as versões da plataforma.

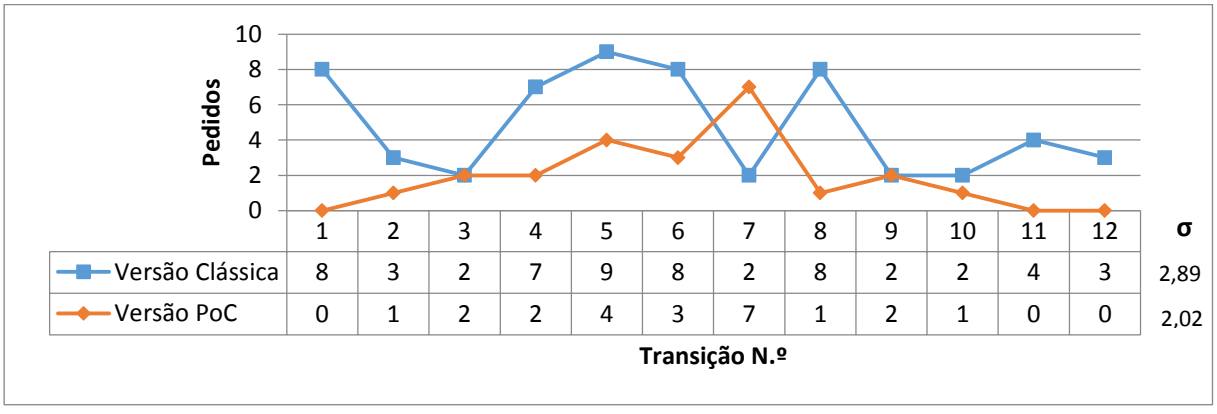
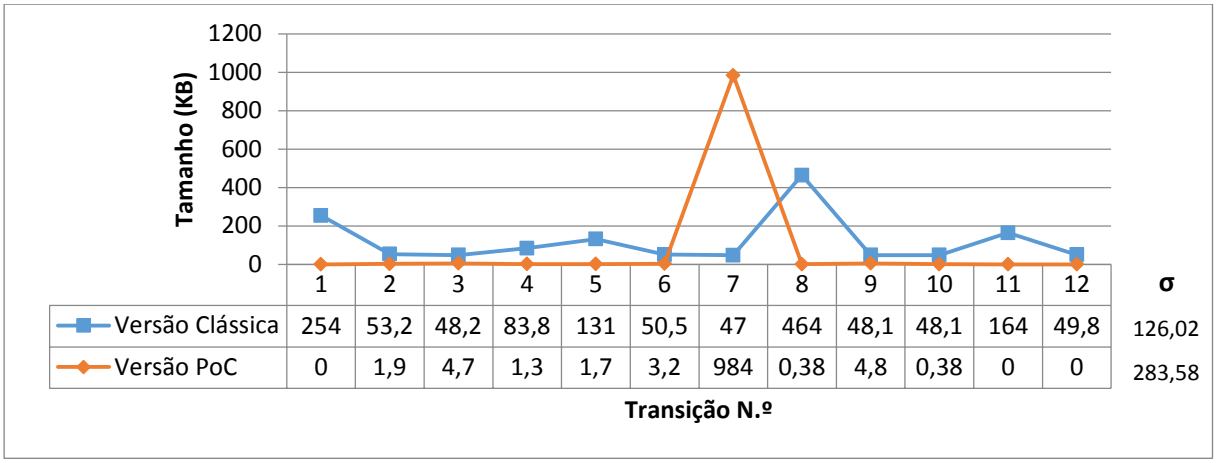


Gráfico 7 – Número de pares pedido/resposta ao servidor de ambas as versões.

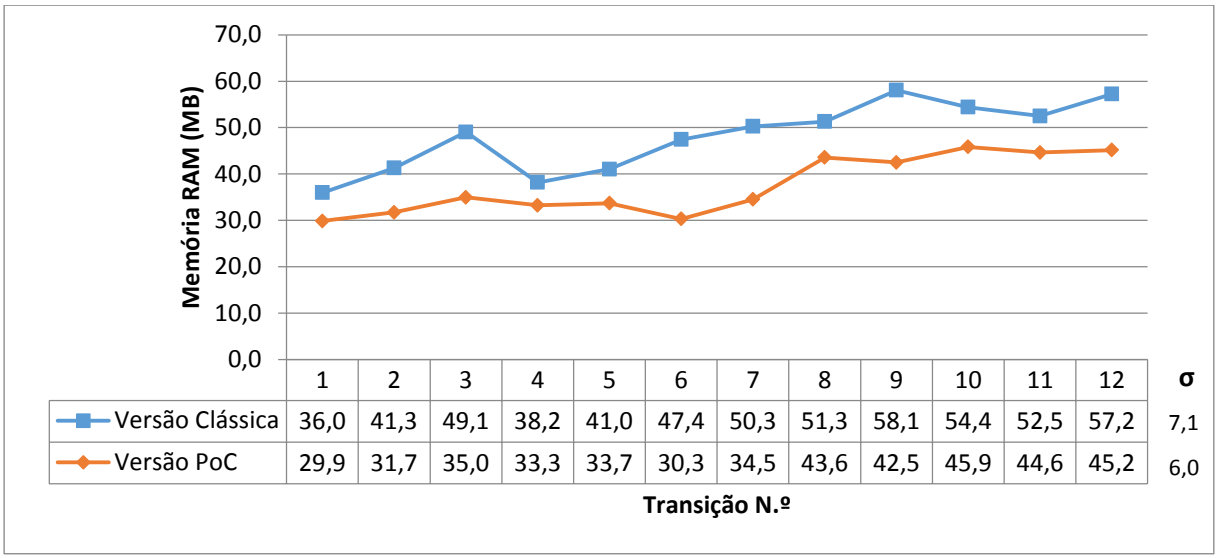


	1	2	3	4	5	6	7	8	9	10	11	12	σ
Versão Clássica	254	53,2	48,2	83,8	131	50,5	47	464	48,1	48,1	164	49,8	126,02
Versão PoC	0	1,9	4,7	1,3	1,7	3,2	984	0,38	4,8	0,38	0	0	283,58

Gráfico 8 – Tamanho dos dados trocados em cada par pedido/resposta com o servidor, de ambas as versões.

É possível verificar que os valores para ambas as estatísticas mantêm-se iguais ou inferiores na versão PoC, comparativamente à versão clássica. Excetua-se apenas um caso interessante referente à transição n.º 7, que corresponde à execução de uma pesquisa que coloca um artigo constituído por texto e imagens na primeira posição dos resultados. Nesta transição é possível verificar um aumento considerável do número de pedidos e dados trocados pela versão PoC, numa tentativa de obter o(s) artigo(s) mais provável(eis) de serem acedidos. No entanto, este esforço está a ser conseguido em *background*, não influenciando o tempo de carregamento dos resultados da pesquisa pois acontece numa fase posterior. Na transição seguinte vê-se a inversão dos papéis: a versão PoC mostra agora valores inferiores. Fazendo a referência cruzada com os valores disponíveis no Gráfico 3 é possível verificar que a versão PoC consegue valores de carregamento de página inferiores em ambas as transições, mesmo com o aumento de pedidos e tamanho dos dados obtidos do servidor.

Finalmente, foi analisado o *footprint* de memória RAM utilizada pelo *browser*, após a execução do mesmo caso de teste, recorrendo à ferramenta "*Task Manager*" disponível no *browser* Google Chrome [81]. O seguinte gráfico coloca lado-a-lado a média da utilização deste recurso por ambas as versões da plataforma.



	1	2	3	4	5	6	7	8	9	10	11	12	σ
Versão Clássica	36,0	41,3	49,1	38,2	41,0	47,4	50,3	51,3	58,1	54,4	52,5	57,2	7,1
Versão PoC	29,9	31,7	35,0	33,3	33,7	30,3	34,5	43,6	42,5	45,9	44,6	45,2	6,0

Gráfico 9 – Consumo de memória RAM por ambas as versões.

É possível analisar que o consumo de memória RAM por ambas as versões cresce globalmente com a utilização da plataforma, no entanto os valores da versão PoC são consistentemente mais baixos. Supõe-se que apesar de ambas as plataformas serem equivalentes, tanto em conteúdo como em funcionalidade, a versão PoC é mais económica ao reduzir o número de pedidos e tamanho da informação trocada com o servidor.

4.1.3 Dispositivo móvel

Neste subcapítulo avaliamos o desempenho de ambas as versões da aplicação, PoC e clássica, num dispositivo de recursos comparativamente reduzidos, executando o caso de teste acima descrito num dispositivo móvel (*smartphone*) de gama média-baixa. As especificações do *hardware* e *software* utilizado são as mesmas, com a substituição do computador pessoal por um *smartphone* com as seguintes características:

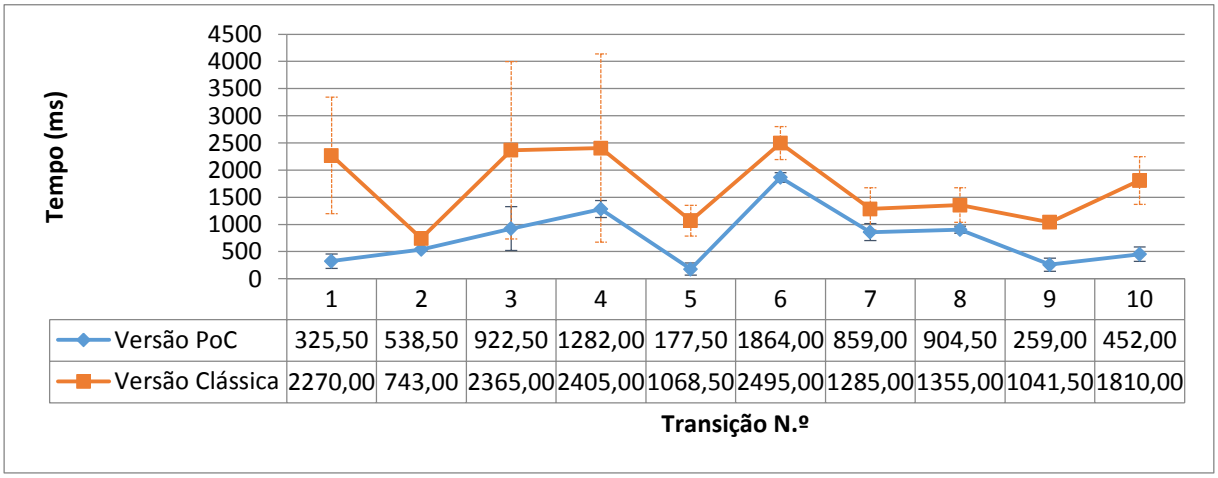
- **Marca / Modelo:** LG P970
- **CPU:** Cortex-A8 @1.00GHz
- **RAM:** 512MB
- **Espaço interno:** 2GB
- **Sistema Operativo:** Android 4.0.4
- ***Browser*:** Google Chrome 27

O caso de teste executado é ligeiramente diferente, na medida em que foram excluídas as páginas relacionadas com a criação de conteúdo, pelas razões explicadas no subcapítulo "3.2.2 Cliente".

Por forma a obter os resultados no mesmo formato que os testes anteriores, foi utilizada a

extensão para o *browser* Google Chrome denominada "ADB Plugin" [82], que permite utilizar a ferramenta "Developer Tools" da mesma forma que num computador.

Abaixo estão apresentados os valores para o carregamento médio e mediano das várias páginas de ambas as versões, do *speedup* médio e mediano conseguido no dispositivo móvel, e finalmente do número de pedidos ao servidor e tamanho dos mesmos. Foram estudados estes indicadores por se considerarem dependentes dos recursos disponíveis no dispositivo em que a aplicação cliente é executada. Parte-se do princípio que quaisquer indicadores que variem consoante a disponibilidade do servidor (como é o caso do CPU *Load*, por exemplo) irão variar proporcionalmente aos resultados acima apresentados, razão pela qual não foram estudados neste dispositivo. Adicionalmente, os indicadores de memória RAM consumida e espaço em disco ocupado não foram novamente analisados por se considerarem independentes do dispositivo utilizado.



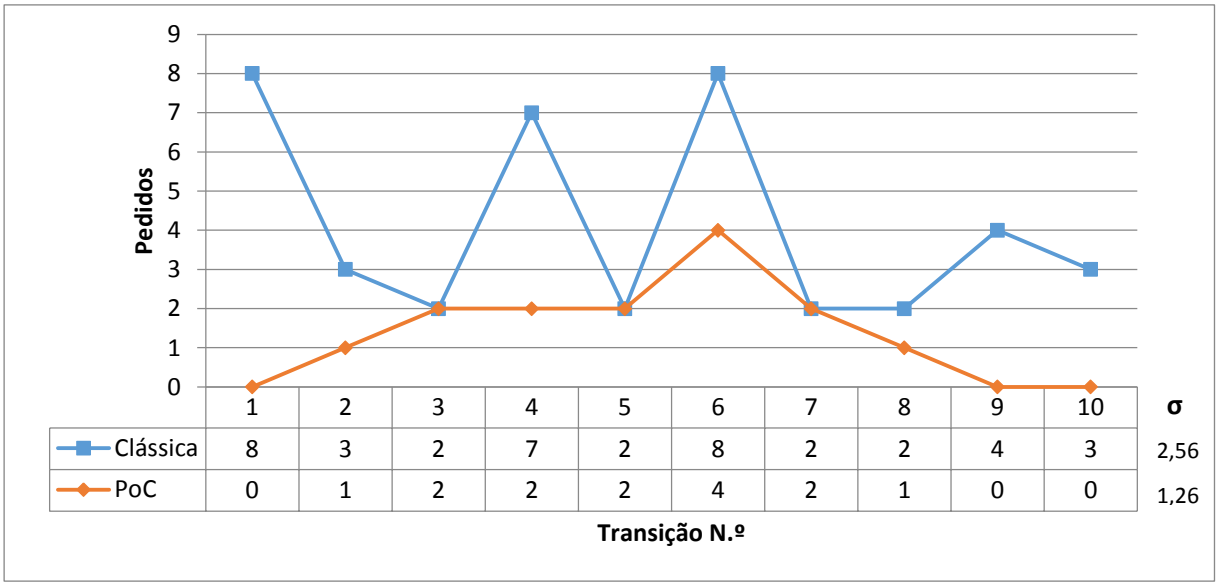
	1	2	3	4	5	6	7	8	9	10
Versão PoC	325,50	538,50	922,50	1282,00	177,50	1864,00	859,00	904,50	259,00	452,00
Versão Clássica	2270,00	743,00	2365,00	2405,00	1068,50	2495,00	1285,00	1355,00	1041,50	1810,00

Gráfico 10 - Tempo médio de carregamento (em milissegundos) de várias páginas de ambas as versões.

Através do gráfico acima é possível verificar que a versão PoC continua em vantagem no que toca ao tempo de abertura, médio e mediano, atingindo um aumento de desempenho mínimo de 1.34 vezes e máximo de 6.97 vezes. Comparativamente aos valores obtidos com a execução no computador, o valor mínimo do aumento de desempenho é bastante idêntico, no entanto o valor máximo está bastante abaixo, limitado pelas capacidades inferiores do dispositivo móvel utilizado. A média de abertura global das páginas da aplicação subiu também em comparação, em cerca de 684% para a versão PoC e 305% para a versão clássica.

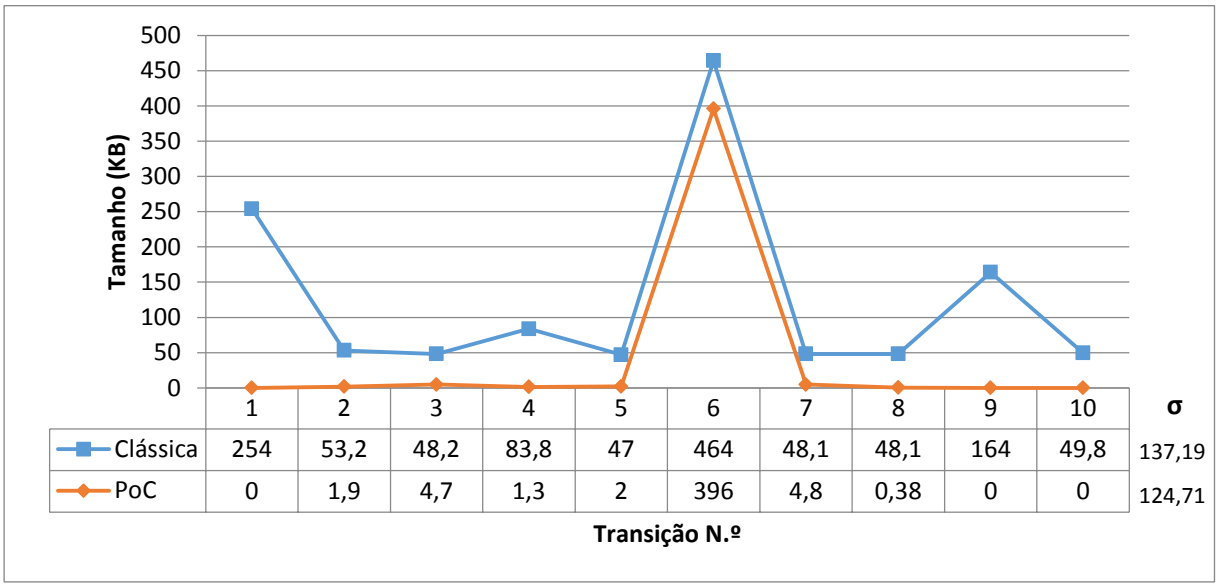
Os dois gráficos abaixo apresentam o número de pares pedido/resposta feitos ao servidor pela aplicação móvel da plataforma, e tamanho dos mesmos. Voltamos a analisar estes fatores à luz de um dispositivo móvel pelas diferenças existentes entre as aplicações *desktop* e móvel anteriormente detalhadas no subcapítulo "3.2.2 Cliente". As diferenças existentes têm como principal objetivo a poupança de recursos do dispositivo móvel dada a geral escassez de

recursos como tráfego de Internet, bateria, etc. É assim esperada uma redução de desempenho da versão PoC móvel face à versão PoC *desktop*, especialmente nas transições de n.º 5 e 6, onde é feito uso de otimizações que deixam um *footprint* pesado no tráfego de Internet.



	1	2	3	4	5	6	7	8	9	10	σ
Clássica	8	3	2	7	2	8	2	2	4	3	2,56
PoC	0	1	2	2	2	4	2	1	0	0	1,26

Gráfico 11 - Número de pares pedido/resposta ao servidor de ambas as versões.



	1	2	3	4	5	6	7	8	9	10	σ
Clássica	254	53,2	48,2	83,8	47	464	48,1	48,1	164	49,8	137,19
PoC	0	1,9	4,7	1,3	2	396	4,8	0,38	0	0	124,71

Gráfico 12 - Tamanho dos dados trocados em cada par pedido/resposta com o servidor, de ambas as versões.

À semelhança dos resultados obtidos da aplicação *desktop*, a versão PoC mantém a sua utilização de tráfego de Internet e pedidos ao servidor geralmente abaixo da versão clássica, no entanto agora esta utilização mantém-se reduzida em todos os pontos do gráfico. Esta diferença deve-se à inexistência de cache de imagens na aplicação móvel, que entraria em

funcionamento na transição n.º 5. Assim a obtenção de imagens, e provável cache das mesmas pelo *browser* ocorre apenas na transição seguinte (n.º 6), mas ainda assim com um consumo inferior à versão clássica. Estes resultados são especialmente benéficos para dispositivos móveis pois é reduzido o número de pedidos ao servidor, poupando a bateria do dispositivo, e também o tráfego de Internet utilizado, recurso que é geralmente limitado e cujo uso resulta em custos acrescidos para o utilizador.

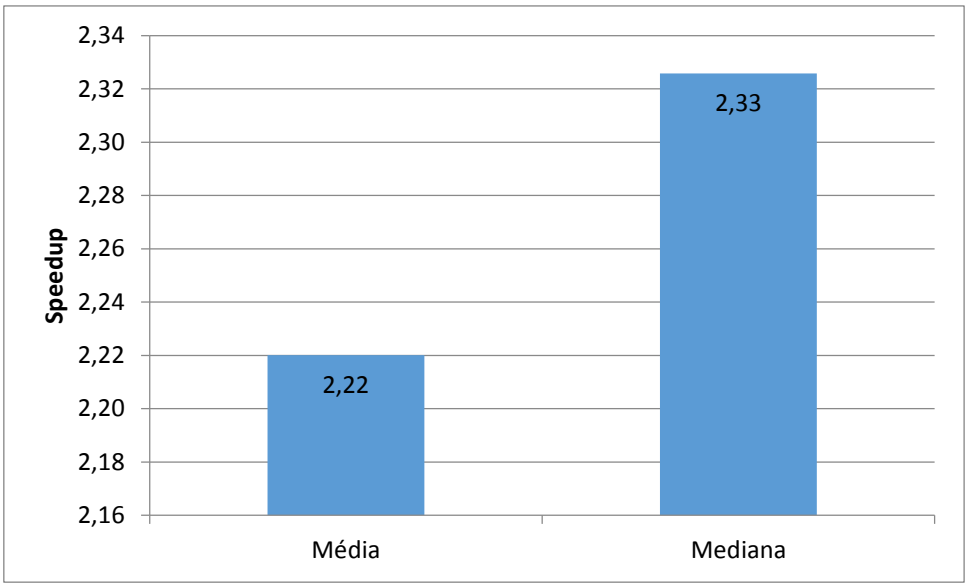


Gráfico 13 - *Speedup* baseado nos valores médios e medianos do tempo de carregamento global das páginas de ambas as versões. Resultados mostram o ganho da versão PoC relativamente à versão clássica

Finalmente, o gráfico acima mostra o *speedup* médio e mediano conseguido no dispositivo móvel para a versão PoC, comparativamente à versão clássica. Considera-se significativo o aumento de desempenho em 2.27 vezes (média) e 2.33 vezes (mediana), apesar de se encontrarem algo abaixo dos valores conseguidos com a execução no computador de 4.98 vezes (média) e 4.53 vezes (mediana).

É possível concluir assim que enquanto as características do dispositivo cliente utilizado afetam significativamente o desempenho geral da plataforma, em nenhuma instância se verifica uma perda de desempenho da versão PoC comparativamente à versão clássica, que poderia eventualmente acontecer devido à complexidade acrescida do interface do utilizador existente na versão PoC. A versão PoC mostra-se assim a mais otimizada para as categorias dos dispositivos testados, tanto pelo desempenho superior como na utilização mais poupada de recursos escassos.

4.2 Resultados qualitativos

Por forma a obter a reação do público-alvo da plataforma, que nesta fase é constituído principalmente por profissionais e entusiastas da área tecnológica, foi desenvolvido um questionário com o objetivo de recolher informação qualitativa sobre o ganho de fluidez conseguido comparativamente a uma aplicação *web* "clássica".

Deste modo, foram questionadas 10 pessoas com idades entre os 22 e 28 anos, na sua maioria estudantes de vários níveis na área da informática, que tiveram a oportunidade de utilizar ambas as versões da plataforma AnyKB (clássica e PoC) e dar a sua opinião respondendo ao questionário disponível no Anexo "Questionário qualitativo" deste documento. Alguns dos questionados possuem conhecimentos da *web* na perspetiva do programador, o que resultou na elaboração de um questionário mais técnico. O questionário começa por fazer uma breve descrição do projeto e do seu objetivo, e define um conjunto de passos a executar em ambas as plataformas. No final, o entrevistado é convidado a responder a uma série de questões com o objetivo de obter a sua opinião sobre um conjunto de comportamentos de ambas as plataformas, e do grau de importância dado à existência, ou não existência, de determinado comportamento. Abaixo estão detalhadas as perguntas presentes no questionário, a informação estatística das respostas obtidas e conclusões retiradas destas.

Perguntas 1 & 2

- *Na primeira plataforma, notou se na transição entre páginas o browser mostra uma página em branco antes de carregar totalmente o conteúdo?*

- *Em caso positivo, descreve este comportamento como "normal", ou já está habituado ao mesmo?*

É comportamento recorrente em virtualmente todos os *browsers* o aparecimento de uma página em branco enquanto o *rendering* da nova página está a ser executado. É objetivo destas questões entender se o entrevistado reconhece este comportamento, e se notou o mesmo durante o teste à primeira plataforma (clássica).

Enquanto apenas 30% dos entrevistados disseram notar o comportamento durante o teste, 100% disseram reconhecer o mesmo. Quando questionados se consideram este comportamento normal ou habitual, 100% dos entrevistados dizem estar habituados ao mesmo, admitindo não ter notado este comportamento no decorrer do teste por possível hábito ao mesmo. Conclui-se assim que o comportamento de interfaces *web* executados num *browser* é menos fluido do que, por exemplo, um interface de uma aplicação nativa; no entanto esta falta de fluidez é de algum modo compensada, tornando-se presentemente um fator normal da navegação na *web* através de um *browser*.

Perguntas 3 & 4

- *Diria que este comportamento é indesejado?*

- *Notou o mesmo comportamento na segunda plataforma?*

Pretendia-se com estas questões entender se o comportamento anteriormente descrito seria totalmente indesejado, ou se existiria alguma utilidade em mantê-lo.

Nesta questão 100% dos entrevistados consideram que se trata de um comportamento indesejado, e que a versão PoC não mostrou qualquer indício deste. No entanto, quando confrontados com uma transição imediata da versão PoC (transição para página estática, por exemplo), que não oferece *feedback* para além da alteração do conteúdo, 10% dos entrevistados (1 utilizador) indicaram que a existência da página branca transitória serve como confirmação que o seu pedido está a ser executado pelo *browser*, e que uma transição imediata sem qualquer tipo de *feedback* pode ser, devido a habituação, confusa. Nos pedidos de maior duração (ao guardar um novo documento, por exemplo), a versão PoC exibe *feedback* que o pedido está a ser executado, o que foi bem recebido por esta percentagem dos entrevistados.

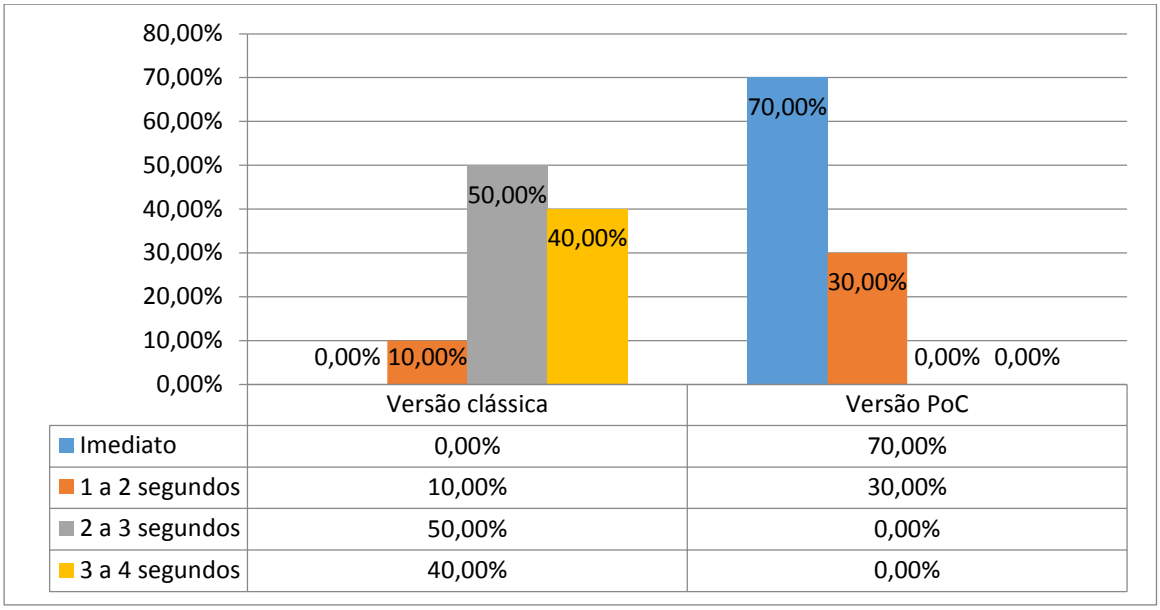
Tratando-se de uma questão de hábito não se considera a falta de *feedback* entre transições rápidas razão para reduzir intencionalmente o desempenho da plataforma, que seria necessário para a inclusão de *feedback* intermédio. Esta conclusão é reforçada pela recorrente não existência de um elemento extra de confirmação de transição noutros tipos de interfaces que respondem de forma imediata (por exemplo em aplicações nativas).

Perguntas 5 & 6

- *Qual das duas plataformas ofereceu, na generalidade, uma experiência mais rápida e fluída?*

- *Qual a sua perceção do valor do tempo de abertura, em média, de uma página da primeira plataforma? E da segunda plataforma? Pode indicar valores temporais quantitativos (segundos, milissegundos, etc.) ou qualitativos (ex. "imediato").*

Pretendia-se com estas duas últimas questões expor por meios quantitativos a opinião geral dos entrevistados sobre ambas as plataformas, na forma de tempo médio de carregamento das várias páginas de ambas as plataformas. Aqui 100% dos entrevistados indicaram a versão PoC como a mais fluída na generalidade, e quando solicitado um valor quantitativo como média de carregamento foi obtida a distribuição apresentada pelo seguinte gráfico.



	Versão clássica	Versão PoC
Imediato	0,00%	70,00%
1 a 2 segundos	10,00%	30,00%
2 a 3 segundos	50,00%	0,00%
3 a 4 segundos	40,00%	0,00%

Gráfico 14 - Distribuição do tempo de carregamento médio percecionado pelos entrevistados para ambas as versões da plataforma.

Através do gráfico acima é possível verificar a tendência da plataforma PoC para a rapidez e fluidez, em deterioração da versão clássica, confirmando qualitativamente a tendência quantitativa obtida no subcapítulo anterior. Relativamente à versão clássica, os valores médios obtidos encontram-se acima dos dados quantitativamente recolhidos. Neste caso a maioria dos entrevistados (50%) responderam com um tempo médio de carregamento entre 2 a 3 segundos, contra o resultado médio de 552ms relatado no subcapítulo anterior. Supõe-se que a menor fluidez deste interface e maior distância entre valores máximos e mínimos para o carregamento das várias páginas contribua negativamente com a perceção de um valor superior ao real. É possível encontrar uma tendência idêntica nos valores médios obtidos para a versão PoC com 30% dos entrevistados e indicar um tempo médio entre 1 a 2 segundos, contra os 110ms apresentados no subcapítulo anterior. No entanto, quanto à versão PoC, a maioria dos entrevistados (70%) indicou como "imediato" o carregamento das páginas por não conseguir quantificar um tempo médio, o que auxilia a confirmar a tendência igualmente apresentada no subcapítulo anterior.

5 Conclusão

O *browser* é atualmente visto como uma ferramenta dispensável sem a presença de uma ligação á Internet. É assim óbvia a dependência que as tecnologias *web* têm a servidores remotos por forma a fornecer serviços aos utilizadores. Esta dependência pode ser vista como exagerada, ou mesmo até desnecessária, em aplicações hoje em dia servidas através da *web*. Isto gera redundância na informação trocada, perda de desempenho e fluidez, e uma menor disponibilidade das aplicações.

Este projeto focou-se no desenvolvimento de uma plataforma *web* complexa com utilidade real, constituindo um desafio na implementação de uma nova solução para o aumento do desempenho e fluidez da aplicação. As atuais abordagens mais populares, como os sistemas de cache do *browser* ou a utilização de AJAX, apesar de apresentarem resultados bastante positivos, estão aquém do potencial máximo que o *browser* consegue atingir.

A solução proposta neste documento não vai contra as atuais abordagens, nem necessariamente compete com as mesmas. A solução apresentada pode ser vista como uma evolução das abordagens existentes, mais completa pela utilização de tecnologias *web* emergentes ainda não popularizadas. Esta solução permite a redução da informação trocada com o servidor, aumento do desempenho e fluidez, e impulsiona as tecnologias *web* para um caminho de maior disponibilidade e de menor dependência com a Internet. Neste projeto o *browser* é visto como um interpretador universal de tecnologias abertas (*open source*) ao invés de uma ferramenta que apenas serve para navegar a *web*.

A plataforma desenvolvida como prova de conceito foi testada e comparada a uma aplicação "clássica" equivalente, mostrando resultados bastante positivos em todas as vertentes analisadas. A plataforma desenvolvida mostra consistentemente menor utilização de recursos como a quantidade de dados transferidos, quantidade de pedidos feitos ao servidor e menor utilização de memória RAM. Esta redução resulta num aumento de desempenho da aplicação, justificada quantitativamente e também pelas reações positivas do público-alvo submetido a um questionário. Os resultados positivos verificam-se tanto em computadores pessoais como em dispositivos móveis (*smartphone*s), sendo os resultados particularmente importantes para estes últimos, pois são geralmente dispositivos de características inferiores.

5.1 Trabalho Futuro

A plataforma desenvolvida encontra-se publicamente disponível e serviu completamente o seu propósito no contexto de investigação científica, no entanto encontra-se ainda na sua primeira versão *alpha*. Existe um conjunto de *bugs* conhecidos, e outros tantos por conhecer, que necessitam de ser abordados com um esforço continuado de desenvolvimento.

Algumas funcionalidades inicialmente propostas, no entanto não criticas ao desenvolvimento do estudo científico, encontram-se atualmente num estado mais simples ou não existiu a oportunidade de serem implementadas. Destaca-se a inexistência de um mecanismo de colaboração em tempo real na redação dos artigos, a quantidade reduzida de *achievements* disponíveis. Algumas outras funcionalidades que não estavam inicialmente previstas mas que seriam interessantes de implementar, ou se encontram num estado inicial de implementação, seriam:

- Possibilitar o registo de utilizadores utilizando sistemas disponibilizados pela Google, Facebook e OpenID;
- Possibilitar a associação de uma imagem ao perfil de utilizador;
- Criação de uma tabela classificativa (*leaderboard*) dos utilizadores da plataforma;
- Criação de um sistema que facilite a descoberta de artigos interessantes ao utilizador;
- Manutenção de versões dos artigos, por forma a possibilitar a recuperação de versões anteriores;
- Implementação de um sistema que coloque em evidência os artigos que necessitam de revisão (disponível apenas a utilizadores com reputação suficiente).

6 Bibliografia

[1] W3C, “Web Storage,” 8 Agosto 2011. [Online]. Available: http://www.w3.org/TR/webstorage/#the-localstorage-attribute. [Acedido em 29 Novembro 2012].

[2] Mozilla Developer Network, “AJAX | MDN,” [Online]. Available: https://developer.mozilla.org/en/docs/AJAX. [Acedido em 30 Agosto 2013].

[3] Stack Exchange, inc., “All Sites - Stack Exchange,” [Online]. Available: http://stackexchange.com/sites. [Acedido em 17 Julho 2013].

[4] National Geographic, “Internet's 40th "Birdthday" Marked,” 31 Agosto 2009. [Online]. Available: http://news.nationalgeographic.com/news/2009/08/090831-internet-40th-video-ap.html. [Acedido em 26 Novembro 2012].

[5] The World Bank, “Internet users (per 100 people),” [Online]. Available: http://data.worldbank.org/indicator/IT.NET.USER.P2/. [Acedido em 26 Novembro 2012].

[6] M. Warman, “Telegraph,” 31 Janeiro 2012. [Online]. Available: http://www.telegraph.co.uk/technology/internet/9051590/50-billion-devices-online-by-2020.html. [Acedido em 26 Novembro 2012].

[7] Google Inc., “Chrome Web Store,” [Online]. Available: https://chrome.google.com/webstore/. [Acedido em 28 Novembro 2012].

[8] Mozilla, “Add-ons for Firefox,” [Online]. Available: https://addons.mozilla.org/. [Acedido em 28 Novembro 2012].

[9] ECMA International, “TC-39 - ECMAScript,” [Online]. Available: http://www.ecma-international.org/memento/TC39.htm. [Acedido em 26 Dezembro 2012].

[10] W3C, “HTML 4.01 Specification,” 24 Dezembro 1999. [Online]. Available: http://www.w3.org/TR/REC-html40/. [Acedido em 5 Dezembro 2012].

[11] Nightlight, “FireFTP - The Free FTP Client for Mozilla Firefox,” [Online]. Available: http://fireftp.mozdev.org/. [Acedido em 28 Novembro 2012].

[12] TwistedWave, “TwistedWave, an Audio Editor,” [Online]. Available: http://twistedwave.com/. [Acedido em 28 Novembro 2012].

[13] Google Inc., “Chrome Web Store - Games,” [Online]. Available: https://chrome.google.com/webstore/category/app/3-games. [Acedido em 28 Novembro 2012].

[14] Google Inc. e Open Handset Alliance, “Layouts | Android Developers,” [Online]. Available: http://developer.android.com/guide/topics/ui/declaring-layout.html. [Acedido em 29 Novembro 2012].

[15] Microsoft, “XAML Overview (WPF),” [Online]. Available: http://msdn.microsoft.com/en-us/library/ms752059.aspx. [Acedido em 29 Novembro 2012].

[16] W3C, “Extensible Markup Language (XML) 1.0 (Fifth Edition)),” 26 Novembro 2008. [Online]. Available: http://www.w3.org/TR/xml/. [Acedido em 17 Julho 2013].

[17] T. Anderson, “Introducing XML, its history, what it it, its significance,” Janeiro 2004. [Online]. Available: http://www.itwriting.com/xmlintro.php. [Acedido em 29 Novembro 2012].

[18] Microsoft, “Using a Three-Tier Architecture Model (COM+),” 16 Outubro 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/ms685068(v=vs.85).aspx. [Acedido em 29 Novembro 2012].

[19] “CoffeeScript,” [Online]. Available: http://coffeescript.org/. [Acedido em 29 Novembro 2012].

[20] “Dart: Structured web apps,” [Online]. Available: http://www.dartlang.org/. [Acedido em 29 Novembro 2012].

[21] Internet Engineering Task Force, “RFC 6265 - HTTP State Management Mechanism,” Abril 2001. [Online]. Available: http://tools.ietf.org/html/rfc6265. [Acedido em 17 Julho 2013].

[22] W3C, “Indexed Database API,” 24 Maio 2012. [Online]. Available: http://www.w3.org/TR/IndexedDB/. [Acedido em 29 Novembro 2012].

[23] W3C, “HTML5,” 25 Maio 2011. [Online]. Available: http://www.w3.org/TR/2011/WD-html5-20110525/. [Acedido em 1 Dezembro 2012].

[24] A. W. Longman, “A history of HTML,” 1998. [Online]. Available: http://www.w3.org/People/Raggett/book4/ch02.html. [Acedido em 1 Dezembro 2012].

[25] W3C, “On SGML and HTML,” 24 Dezembro 1999. [Online]. Available: http://www.w3.org/TR/html4/intro/sgmltut.html. [Acedido em 1 Dezembro 2012].

[26] W3Schools, “W3C HTML Activities,” [Online]. Available: http://www.w3schools.com/w3c/w3c_html.asp. [Acedido em 1 Dezembro 2012].

[27] The HTML5 test, “The HTML5 test - How well does your browser support HTML5?,” [Online]. Available: http://html5test.com/. [Acedido em 1 Dezembro 2012].

[28] W3C, “Cascading Style Sheets (CSS) Snapshot 2010,” 12 Maio 2011. [Online]. Available: http://www.w3.org/TR/CSS/. [Acedido em 5 Dezembro 2012].

[29] W3C, “Cascading Style Sheets, level 1,” 17 Dezembro 1996. [Online]. Available: http://www.w3.org/TR/2008/REC-CSS1-20080411/. [Acedido em 5 Dezembro 2012].

[30] “The CSS3 Test,” [Online]. Available: http://css3test.com/. [Acedido em 5 Dezembro 2012].

[31] W3C, “W3C Document Object Model,” 19 Janeiro 2005. [Online]. Available: http://www.w3.org/DOM/. [Acedido em 18 Dezembro 2012].

[32] Mozilla, “Locating DOM elements using selectors - Document Object Model (DOM) | MDN,” 19 Agosto 2012. [Online]. Available: https://developer.mozilla.org/en-

US/docs/DOM/Locating_DOM_elements_using_selectors. [Acedido em 18 Dezembro 2012].

[33] ECMA International, “Standard ECMA-262,” Junho 20122. [Online]. Available: http://www.ecma-international.org/publications/standards/Ecma-262.htm. [Acedido em 26 Dezembro 2012].

[34] Mozilla, “Kraken JavaScript Benchmark (version 1.1),” [Online]. Available: http://krakenbenchmark.mozilla.org/. [Acedido em 28 Dezembro 2012].

[35] Google, “Dart: Structured web apps,” [Online]. Available: http://www.dartlang.org/. [Acedido em 30 Dezembro 2012].

[36] T. Claburn, “Google Launches Dart Programming Language - Development - Web,” InformationWeek, 10 Outubro 2011. [Online]. Available: http://www.informationweek.com/development/web/google-launches-dart-programming-languag/231900449. [Acedido em 30 Dezembro 2012].

[37] Mozilla Developer Network, “Details of the object model - JavaScript | MDN,” 2012 Novembro 2012. [Online]. Available: https://developer.mozilla.org/en-US/docs/JavaScript/Guide/Details_of_the_Object_Model. [Acedido em 30 Dezembro 2012].

[38] Desconhecido, “Leaked internal google dart email,” [Online]. Available: https://gist.github.com/1208618. [Acedido em 31 Dezembro 2012].

[39] Google, “dart: The Standalone Dart VM | Dart: Structured web apps,” [Online]. Available: http://www.dartlang.org/docs/standalone-dart-vm/. [Acedido em 30 Dezembro 2012].

[40] Google, “dart2js: The Dart-to-JavaScript Compiler | Dart: Structured web apps,” [Online]. Available: http://www.dartlang.org/docs/dart2js/. [Acedido em 30 Dezembro 2012].

[41] W3C, “XMLHttpRequest,” 6 Dezembro 2012. [Online]. Available: http://www.w3.org/TR/XMLHttpRequest/. [Acedido em 31 Dezembro 2012].

[42] Facebook, “Chat Sidebar | Facebook Help Center,” [Online]. Available: https://www.facebook.com/help/260938680677469/. [Acedido em 30 Agosto 2013].

[43] Omegle, “Omegle: Talk to strangers!,” [Online]. Available: http://www.omegle.com/. [Acedido em 30 Agosto 2013].

[44] M. Pilgrim, “History API - Dive Into HTML5,” [Online]. Available: http://diveintohtml5.info/history.html. [Acedido em 30 Dezembro 2012].

[45] WHATWG, “6.5 Session and history navigation - HTML Standard,” 30 Dezembro 2012. [Online]. Available: http://www.whatwg.org/specs/web-apps/current-work/multipage/history.html#the-history-interface. [Acedido em 30 Dezembro 2012].

[46] Rob Gravelle, “Comet Programming: Using Ajax to Simulate Server Push | WebReference,” WebReference, [Online]. Available: http://www.webreference.com/programming/javascript/rg28/index.html. [Acedido em 3 Janeiro 2013].

[47] e. a. Loreto, “RFC6202 - Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP,” Internet Engineering Task Force (IETF), [Online]. Available: http://tools.ietf.org/html/rfc6202. [Acedido em 3 Janeiro 2013].

[48] W3C, “Server-Sent Events,” 11 Dezembro 2012. [Online]. Available: http://www.w3.org/TR/eventsource/. [Acedido em 3 Janeiro 2013].

[49] W3C, “Indexed Database API,” 24 Maio 2012. [Online]. Available: http://www.w3.org/TR/IndexedDB/. [Acedido em 4 Janeiro 2013].

[50] W3C, “Web SQL Database,” 18 Novembro 2010. [Online]. Available: http://www.w3.org/TR/webdatabase/. [Acedido em 4 Janeiro 2013].

[51] A. Deveria, “Can I use IndexedDB,” [Online]. Available: http://caniuse.com/indexeddb. [Acedido em 4 Janeiro 2013].

[52] Mozilla Developer Network, “IndexedDB | MDN,” 9 Junho 2013. [Online]. Available: https://developer.mozilla.org/en/docs/IndexedDB. [Acedido em 17 Julho 2013].

[53] Google, “Managing HTML5 Offline Storage - Google Chrome - Google Developers,” 7 Junho 2012. [Online]. Available: https://developers.google.com/chrome/whitepapers/storage#unlimited. [Acedido em 17 Julho 2013].

[54] W3C, “File API: Directories and System,” 17 Abril 2012. [Online]. Available: http://www.w3.org/TR/file-system-api/. [Acedido em 4 Janeiro 2013].

[55] Mozilla Developer Network, 14 Março 2012. [Online]. Available: https://developer.mozilla.org/en-US/docs/DOM/File_API/File_System_API. [Acedido em 4 Janeiro 2013].

[56] Mozilla Developer Network, “LocalFileSystem - Web API Reference | MDN,” 20 Maio 2013. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/LocalFileSystem. [Acedido em 17 Julho 2013].

[57] A. Deveria, “Can I use Filesystem & FileWriter API,” [Online]. Available: http://caniuse.com/filesystem. [Acedido em 4 Janeiro 2013].

[58] C. Heilmann, “There is no simple solution for local storage - Mozilla Hacks - the Web developer blog,” Mozilla, 5 Março 2012. [Online]. Available: https://hacks.mozilla.org/2012/03/there-is-no-simple-solution-for-local-storage/. [Acedido em 4 Janeiro 2013].

[59] e. a. Mathias Bynens, “Write Test: IndexedDB vs localStorage vs webSQL | jsPerf,” Dezembro 2012. [Online]. Available: http://jsperf.com/indexeddb-vs-localstorage/15. [Acedido em 4 Janeiro 2013].

[60] A. Deveria, “Can I use,” [Online]. Available: http://caniuse.com/namevalue-storage. [Acedido em 4 Janeiro 2013].

[61] Stack Exchange, inc., “Stack Exchange - Free, Community-Powered Q&A,” [Online]. Available: http://stackexchange.com/. [Acedido em 17 Julho 2013].

[62] Stack Exchange, inc., “Stack Overflow,” [Online]. Available: http://stackoverflow.com/. [Acedido em 17 Julho 2013].

[63] Stack Exchange, inc., “How does “Reputation” work? - Meta Stack Overflow,” 25 Junho 2013. [Online]. Available: http://meta.stackoverflow.com/questions/7237/how-does-reputation-work. [Acedido em 17 Julho 2013].

[64] J. Atwood, “A Theory of Moderation « Blog - Stack Exchange,” 5 Maio 2009. [Online]. Available: http://blog.stackoverflow.com/2009/05/a-theory-of-moderation/. [Acedido em

17 Julho 2013].

[65] A. Rodriguez, “RESTful Web services: The basics,” 6 Novembro 2008. [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-restful/. [Acedido em 17 Julho 2013].

[66] Internet Engineering Task Force, “RFC 4627 - The application/json Media Type for JavaScript Object Notation (JSON),” Julho 2006. [Online]. Available: http://tools.ietf.org/html/rfc4627. [Acedido em 17 Julho 2013].

[67] Xen Project, “The Xen Project, the powerful open source industry standard for virtualization,” [Online]. Available: http://www.xenproject.org/. [Acedido em 01 07 2013].

[68] CentOS Project, “www.centos.org - The Community ENTerprise Operating System,” [Online]. Available: http://www.centos.org/. [Acedido em 17 Julho 2013].

[69] The Apache Software Foundation, “Welcome! - The Apache HTTP Server Project,” [Online]. Available: http://httpd.apache.org/. [Acedido em 17 Julho 2013].

[70] The PHP Group, “PHP: Hypertext Preprocessor,” [Online]. Available: http://php.net/. [Acedido em 17 Julho 2013].

[71] Oracle Corporation, “MySQL :: The world's most popular open source database,” [Online]. Available: http://www.mysql.com/. [Acedido em 17 Julho 2013].

[72] The Apache Software Foundation, “Apache Lucene - Apache Solr,” [Online]. Available: http://lucene.apache.org/solr/. [Acedido em 17 Julho 2013].

[73] Oracle Corporation, “MySQL :: MySQL 5.5 Reference Manual :: 14.3 The InnoDB Storage Engine,” [Online]. Available: http://dev.mysql.com/doc/refman/5.5/en/innodb-storage-engine.html. [Acedido em 17 Julho 2013].

[74] Apache Software Foundation, “Apache Lucene - Welcome to Apache Lucene,” [Online]. Available: http://lucene.apache.org/. [Acedido em 21 Agosto 2013].

[75] The Daring Fireball Company LLC, “Daring Fireball: Markdown,” [Online]. Available: http://daringfireball.net/projects/markdown/. [Acedido em 13 Julho 2013].

[76] Internet Engineering Task Force, “RFC 4648 - The Base16, Base32, and Base64 Data Encodings,” Outubro 2006. [Online]. Available: http://tools.ietf.org/html/rfc4648. [Acedido em 13 Julho 2013].

[77] Google Developers, “Chrome DevTools -- Google Developers,” [Online]. Available: https://developers.google.com/chrome-developer-tools/. [Acedido em 14 Agosto 2013].

[78] Google Analytics, “Global Site Speed Overview: How Fast Are Websites Around The World? - Analytics Blog,” 19 Abril 2012. [Online]. Available: http://analytics.blogspot.pt/2012/04/global-site-speed-overview-how-fast-are.html. [Acedido em 5 Agosto 2013].

[79] Google Developers, “Web metrics: Size and number of resources - Make the Web Faster - Google Developers,” 26 Maio 2010. [Online]. Available: https://developers.google.com/speed/articles/web-metrics. [Acedido em 5 Agosto 2013].

[80] Linux Kernel Organization, Inc., “The Linux Kernel Archives,” [Online]. Available: https://www.kernel.org/doc/Documentation/cpu-load.txt. [Acedido em 2 Agosto 2013].

[81] Google, “Force Browser processes to close,” [Online]. Available: https://support.google.com/chrome/answer/95672?hl=en. [Acedido em 21 Setembro 2013].

[82] Google, “Chrome Web Store - ADB,” [Online]. Available: https://chrome.google.com/webstore/detail/adb/dpngiggdglpdnjdoaefidgiigpemgage. [Acedido em 17 Agosto 2013].

[83] Linux Kernel Organization, Inc., “The Linux Kernel Archives,” [Online]. Available: https://www.kernel.org/doc/Documentation/cpu-load.txt. [Acedido em Junho 2013].

[84] A. Deveria, “Can I use... Support tables for HTML5, CSS3, etc,” [Online]. Available: http://caniuse.com/#feat=namevalue-storage. [Acedido em Junho 2013].

[85] World Wide Web Consortium (W3C), “W3C Proposed Recommendation 9 April 2013,” [Online]. Available: http://www.w3.org/TR/webstorage/. [Acedido em Julho 2013].

[86] J. Kumar, “Benchmarking results of mysql, lucene and sphinx,” 6 Fevereiro 2006. [Online]. Available: http://jayant7k.blogspot.pt/2006/06/benchmarking-results-of-mysql-

lucene.html. [Acedido em 21 Agosto 2013].

[87] A. Ternovskiy, “Chatroulette,” [Online]. Available: http://chatroulette.com/. [Acedido em 30 Agosto 2013].

7 Anexos

7.1 Documentação do *webservice* REST

Este anexo fornece uma descrição técnica sobre o *webservice* REST constituinte da plataforma AnyKB. Estão presentes neste documento detalhes sobre acesso, formato de dados, entidades e recursos fornecidos.

7.1.1 Acerca do *webservice*

O *webservice* REST descrito neste anexo tem como objetivo a integração das aplicações cliente e servidor da plataforma AnyKB. É um *webservice* de uso privado pela própria plataforma, razão pela qual não se encontra neste anexo detalhes de como obter uma chave de autenticação no *webservice*.

7.1.2 URL base, recursos e parâmetros

O *webservice* está acessível pela própria plataforma através do seguinte URL:

```
https://anykb.net/api
```

Através deste URL é possível obter **entidades** ao indicar um determinado **recurso** disponibilizado, adicionando o caminho do recurso no final do URL. Como exemplo, o recurso de nome "exemplo/um" é acedido através do URL:

```
https://anykb.net/api/exemplo/um
```

Os constituintes do recurso, que no caso do exemplo anterior são `exemplo` e `um` têm a designação de **elementos**.

Parâmetros são passados através da *querystring* do URL para pedidos `HTTP GET`, ou seja, adicionando uma cadeia de pares chave/valor no formato `?chave=valor&chave2=valor2` ao final do URL. Para pedidos utilizando qualquer outro método HTTP, os parâmetros devem ser passados no corpo do pedido HTTP.

7.1.3 Formato dos dados

O *webservice* responde em JSON e utiliza o *header* HTTP *status* para indicar o estado do pedido (sucesso/erro). Informações detalhadas sobre os valores possíveis para o *status* e códigos de erro encontram-se no subcapítulo "7.1.6 Erros".

7.1.4 Entidades

Neste capítulo estão descritas as entidades devolvidas pelo *webservice* e a sua estrutura.

User			
Descrição	Representa um utilizador registado na plataforma.		
Atributos	**Nome**	**Tipo**	**Descrição**
	id	Integer	Identificador único do utilizador dentro da plataforma
	name	String	Nome do utilizador
	email	String	Email do utilizador
	website	String	*Website* do utilizador
	points	Integer	Pontos do utilizador dentro da plataforma

Article			
Descrição	Representa um artigo criado por um utilizador.		
Atributos	**Nome**	**Tipo**	**Descrição**
	id	Integer	Identificador único do artigo dentro da plataforma
	title	String	Título do artigo
	friendlyURL	String	URL de acesso ao artigo
	content	String	Corpo do artigo, formatado em MarkDown
	authorID	Integer	Identificador do utilizador autor do artigo
	authorName	String	Nome do utilizador autor do artigo
	score	Integer	Pontuação do artigo, atribuída pelos utilizadores através da função de voto
	positiveScore	Integer	Número de votos positivos recebidos pelo artigo
	negativeScore	Integer	Número de votos negativos recebidos pelo artigo
	favorites	Integer	Número de vezes que o artigo foi marcado como favorito pelos utilizadores
	views	Integer	Número de visualizações do artigo pelos utilizadores

SearchResult			
Descrição	Representa um resultado proveniente da pesquisa na plataforma.		
Atributos	**Nome**	**Tipo**	**Descrição**
	id	Integer	Identificador único do artigo relacionado com o resultado da pesquisa
	title	String	Título do artigo
	friendlyURL	String	URL de acesso ao artigo
	articleScore	Integer	Pontuação do artigo
	author	Integer	Identificador do utilizador autor do artigo
	authorName	String	Nome do utilizador autor do artigo
	authorPoints	Integer	Pontos do utilizador autor do artigo
	snippets	Array<String>	Conjunto de trechos do corpo do artigo relevantes para os termos de pesquisa utilizados

UserArticleExtras			
Descrição	Detalhes sobre um determinado artigo, específicos a um utilizador.		
Atributos	**Nome**	**Tipo**	**Descrição**
	isFavorite	Boolean	Indica se o artigo foi marcado como favorito pelo utilizador
	voteDirection	Integer	Indica a direção (positiva = 1, negativa = -1) do voto do utilizador ao artigo, caso exista. Um valor de 0 indica que não existe voto.

UserArticleResume			
Descrição	Representa um resumo de um artigo criado por um utilizador.		
Atributos	**Nome**	**Tipo**	**Descrição**
	id	Integer	Identificador único do artigo relacionado
	title	String	Título do artigo
	friendlyURL	String	URL de acesso ao artigo
	score	String	Pontuação do artigo, atribuída pelos utilizadores através da função de voto
	positiveScore	Integer	Números de votos positivos recebidos do artigo
	negativeScore	Integer	Número de votos negativos recebidos do artigo
	favorites	Integer	Número de vezes que o artigo foi marcado como favorito pelos utilizadores
	views	Integer	Número de visualizações do artigo pelos utilizadores

UserPoint			
Descrição	Representa um ponto ganho pelo utilizador pela sua participação na plataforma.		
Atributos	**Nome**	**Tipo**	**Descrição**
	article	Integer	Identificador único do artigo relacionado
	action	String	Descrição da ação que levou à atribuição do ponto
	value	Integer	Valor número do ponto. Atualmente qualquer ponto tem o único valor de "1".
	timestamp	Integer	Data e hora da atribuição do ponto, formatado como o número de segundos desde 1 de janeiro de 1970.

UserAchievement			
Descrição	Representa um *achievement* (conquista) ganho pelo utilizador pela sua participação na plataforma.		
Atributos	**Nome**	**Tipo**	**Descrição**
	userID	Integer	Identificador único do utilizador relacionado
	achivementID	String	Identificador único do *achivement* (conquista) relacionado

7.1.5 Recursos

Neste capítulo estão descritos os recursos disponibilizados pelo *webservice*. O título de cada recurso representa o método HTTP e entidade a indicar no URL de acesso ao *webservice*. Elementos que deverão ser substituídos por um determinado valor estão representados entre "[" e "]".

GET /user	
Descrição	Obtém o utilizador com sessão atualmente iniciada na plataforma.
Devolve	Entidade User

GET /user/article/resume	
Descrição	Obtém um resumo de todos os artigos criados pelo utilizador com sessão atualmente iniciada na plataforma.
Devolve	Array de entidades UserArticleResume

GET /user/article/favorite	
Descrição	Obtém um resumo de todos os artigos marcados como favoritos pelo utilizador com sessão atualmente iniciada na plataforma.
Devolve	Array de entidades UserArticleResume

GET /user/points	
Descrição	Obtém os detalhes dos pontos atribuídos ao utilizador com sessão atualmente iniciada na plataforma.
Devolve	Array de entidades UserPoint

GET /user/achievements	
Descrição	Obtém os detalhes dos *achievements* (conquistas) atribuídos ao utilizador com sessão atualmente iniciada na plataforma.
Devolve	Array de entidades UserAchievement

GET /article/[ID]	
Descrição	Obtém os detalhes dos pontos atribuídos ao utilizador com sessão atualmente iniciada na plataforma.
Elementos	**ID** Identificador único ao artigo dentro da plataforma
Devolve	Entidade Article Entidade UserArticleExtras, caso o utilizador esteja autenticado

GET /article/latest/[num]	
Descrição	Obtém os últimos artigos criados na plataforma.
Elementos	**num** Número de artigos a obter
Devolve	Array de entidades Article

GET /search/[termos]	
Descrição	Executar uma pesquisa de artigos.
Elementos	**termos** Termos de pesquisa
Devolve	Array de entidades SearchResult

POST /user/login	
Descrição	Cria uma nova sessão de utilizador na plataforma.
Parâmetros	**username** Email do utilizador **password** Password do utilizador
Devolve	Entidade User

POST /user/register	
Descrição	Cria um novo utilizador na plataforma.
Parâmetros	**name** Nome do utilizador **email** Email do utilizador **password** Password do utilizador **repeatPassword** Password do utilizador, repetida por questões de confirmação

POST /user/favorite	
Descrição	Cria um novo favorito para o utilizador atualmente autenticado na plataforma.
Parâmetros	**articleid** Identificador do artigo em questão

POST /article	
Descrição	Cria um novo artigo na plataforma.
Parâmetros	**title** Título do artigo **content** Corpo do artigo, formatado em MarkDown **plainTextContent** Corpo do artigo em texto puro

PUT /user	
Descrição	Altera detalhes pessoais do utilizador atualmente autenticado na plataforma.
Parâmetros	**field** Nome do detalhe pessoal a alterar. Os valores válidos para este campo são "name" e "website", para alterar o nome e *website* do utilizador. **value** Novo valor para a o "field" indicado

PUT /user/recoverpassword	
Descrição	Altera a password do utilizador e notifica o utilizador por email.
Parâmetros	**email** Email do utilizador em questão

PUT /article	
Descrição	Alterar detalhes de artigo.
Parâmetros	**articleID** Identificado único do artigo a alterar **title** Título do artigo **content** Corpo do artigo, formatado em MarkDown **plainTextContent** Corpo do artigo em texto puro

PUT /article/vote	
Descrição	Fazer voto em artigo, associado ao utilizador atualmente autenticado na plataforma.
Parâmetros	**articleID** Identificador único do artigo em questão **voteDirection** Direção (positiva, negativa) do voto. Um valor de "1" indica um voto positivo e um valor de "-1"

DELETE /user	
Descrição	Fecha a sessão do utilizador atualmente autenticado na plataforma.

DELETE /user/favorite	
Descrição	Elimina um artigo favorito do utilizador
Parâmetros	**articleID** Identificador único do artigo em questão

DELETE /article	
Descrição	Elimina um artigo. Apenas é possível eliminar artigos pertencentes ao utilizador atualmente autenticado na plataforma.
Parâmetros	**articleID** Identificador único do artigo em questão

DELETE /article/vote	
Descrição	Elimina o voto do utilizador atualmente autenticado na plataforma para um determinado artigo
Parâmetros	**articleID** Identificador único do artigo em questão

7.1.6 Erros

É possível determinar se um pedido ao *webservice* terminou corretamente, ou não, avaliando o código HTTP recebido na resposta do servidor. A seguinte tabela descrimina os valores possíveis devolvidos pelo *webservice*.

Código	Descrição	Significado
200	OK	O pedido terminou com sucesso
400	Bad Request	Pedido por um recurso inválido/inexistente, ou pedido indevidamente especificado. O pedido não deve ser repetido sem alterações.
401	Unauthorized	Este pedido está apenas disponível para utilizadores autenticados.
501	Not Implemented	Método HTTP utilizado não está implementado. O *webservice* aqui especificado faz uso apenas dos métodos GET, POST, PUT e DELETE.

7.2 Questionário qualitativo

Questionário

Recolha de estatísticas qualitativas no âmbito do projeto AnyKB (MEI-CM, ESTG, IPleiria)

Este questionário tem como objetivo a recolha de dados qualitativos sobre a aplicação *web* AnyKB, desenvolvida no âmbito do projeto de final de curso em Mestrado em Computação Móvel. A aplicação foca-se no desempenho e fluidez da aplicação, sendo estes os aspetos primários a serem avaliados por este questionário.

Breve descrição da aplicação

A aplicação *web* AnyKB é uma plataforma para a partilha de artigos na área da tecnologia, desde tutoriais, resoluções de problemas, artigos de investigação, etc. Os artigos são criados pela comunidade, sendo os autores recompensados através da aprovação dos seus artigos pela comunidade, na forma de votos. Esta aprovação permite o autor destacar-se de entre os restantes membros.

A plataforma foi criada como base útil para testar um novo conceito que permite o aumento do desempenho e fluidez da aplicação cliente (executada no *browser*), recorrendo a várias tecnologias *web* emergentes.

Teste qualitativo

O seguinte teste compara duas versões da mesma plataforma: a primeira recorre a tecnologias *web* clássicas, e a segunda implementa os mecanismos de aumento de desempenho indicados anteriormente. A sua opinião será baseada na comparação destas duas versões.

Os passos descritos abaixo deverão primeiramente ser executados na plataforma disponível em http://html.anykb.telmo.pt, e de seguida em http:// anykb.telmo.pt.

1. Efetue o registo na plataforma através da opção "*Register*" do menu de topo.
2. Encontra-se agora registado e autenticado na plataforma.
3. Aceda à opção "*About*" do menu de topo. Analise o conteúdo da página que se abre.
4. Volte à página inicial, utilizando o botão retroceder do *browser*.
5. Localize a caixa de pesquisa no centro da página. Pesquise pelos termos "jquery ajax".
6. Analise os resultados da pesquisa. Aceda ao artigo de título "Ajax request using jQuery".
7. Analise o conteúdo da página que se abre. Adicione o artigo aos favoritos, utilizando a opção correspondente do menu lateral.
8. De seguida vamos editar o artigo, para tal aceda à opção "*Edit article*" do menu lateral direito.
9. Efetue algumas alterações ao conteúdo do artigo (não importa o conteúdo destas alterações), e de seguida grave o artigo utilizando a opção "Save Article" no fundo da página.
10. Volte à página inicial, utilizando o botão "AnyKB" do menu de topo.
11. Pesquise por "parse json javascript".
12. Analise os resultados da pesquisa. Aceda ao artigo de título "Parse JSON using jQuery".

13. Analise o conteúdo do artigo. Faça um voto utilizando a opção correspondente do menu lateral.
14. Aceda ao seu perfil, através da opção com o seu nome disponível no menu de topo. Navegue pelos vários separadores disponíveis nesta página – "*Articles*", "*Favorites*", "*Points*" e "*Achievements*".

Recolha dos resultados

Após conclusão do teste anterior em ambas as plataformas, responda às seguintes questões.

1. Na primeira plataforma, notou se na transição entre páginas o *browser* mostra uma página em branco antes de carregar totalmente o conteúdo?

2. Em caso positivo, descreve este comportamento como "normal", ou já está habituado ao mesmo?

3. Diria que este comportamento é indesejado?

4. Notou o mesmo comportamento na segunda plataforma?

5. Qual das duas plataformas ofereceu, na generalidade, uma experiência mais rápida e fluída?

6. Qual a sua perceção do valor do tempo de abertura, em média, de uma página da primeira plataforma? E da segunda plataforma?
   Pode indicar valores temporais quantitativos (segundos, milissegundos, etc.) ou qualitativos (ex. "imediato").

7.3 Artigo publicado em revista *online*

PROGRAMAR

REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.PT

EDIÇÃO #42 - SETEMBRO 2013

ISSN 1647-0710

A PROGRAMAR

DEBUG DE APLICAÇÕES EM C

ARDUINO ACCIONAMENTO E CONTROLO DE SERVOS VIA TECLADO

INTRODUÇÃO AO WEB2PY

LISTAS SIMPLESMENTE LIGADAS E EXEMPLO DE IMPLEMENTAÇÃO EM C

PASCAL TIPO DE DADOS VARIANT E TIPOS GENÉRICOS

COLUNAS

EVENTOS & HANDLERS **VISUAL(NOT)BASIC**

COMUNIDADES

IMPLEMENTANDO PUBLICIDADE USANDO NOKIA NAX EM APLICAÇÕES WINDOWS PHONE **NETPONTO**

NO CODE

SER OU SER RECONHECIDO PELA GOOGLE **DISPOSITIVO ANDROID PREMIUM 11**

NAVICAT

ANÁLISES

WINDOWS SERVER 2012 CURSO COMPLETO

REDES DE COMPUTADORES CURSO COMPLETO

TEMA DE CAPA

Web Persistente: Local Storage

Web Persistente: Local Storage

Introdução

Neste artigo vamos estudar uma solução para o desenvolvimento de aplicações web mais fluídas recorrendo ao Local Storage. Esta tecnologia possibilita a persistência de dados no browser, o que permite a implementação de aplicações web que respondem visualmente de forma imediata – uma caraterística que sempre foi possível em aplicações nativas, mas que a web apenas recentemente começou a suportar.

No decorrer deste artigo introduzimos a utilização do Local Storage e analisamos em maior detalhe a técnica utilizada para tirar partido da potencialidade da tecnologia. Finalmente, é feita a apresentação dos resultados de desempenho conseguidos, capturados recorrendo à plataforma de nome AnyKB desenvolvida como prova de conceito para esse objectivo.

Buzzword: HTML5

O HTML5, para além de ser uma especificação, é uma buzzword que engloba todas as tecnologias emergentes que estão a revolucionar a Web. Apesar do nome, algumas destas tecnologias não estão relacionadas com a linguagem de marcação, mas tratam-se de APIs (Application Programming Interface) JavaScript que permitem a implementação de funcionalidades inovadoras: este é o caso do Local Storage.

A especificação W3C “Web Storage” [1] define esta API do seguinte modo:

“This specification defines an API for persistent data storage of key-value pair data in Web clients.

Na prática, Local Storage é uma API JavaScript que permite guardar de forma persistente dados no formato chave-valor. Isto é, os dados são mantidos no browser mesmo após o utilizador fechar a página web.

O principio básico do Local Storage é idêntico ao dos cookies, mas de utilização simplificada e com limites de espaço mais generosos. De momento é possível guardar cerca de 5MB por domínio ou subdomínio, estando o acesso aos dados limitado igualmente ao domínio ou subdomínio originador dos mesmos.

Apesar de recente, o suporte a esta tecnologia pelos browsers mais populares é bastante boa. O Local Storage é suportado em todas as actuais versões dos browsers Chrome, Firefox, Internet Explorer, Safari e Opera, e ainda pelas versões móveis iOS Safari, Android Browser e Blackberry Browser [1].

Introdução ao Local Storage

Aceder ao Local Storage através de JavaScript é bastante simples e directo. A especificação define a interface localStorage que contém os métodos setItem e getItem para guardar e obter dados, respectivamente. Vejamos o exemplo abaixo.

```
//Guardar
localStorage.setItem("chave", "valor");
//Obter
var valor = localStorage.getItem("chave");
```

Listagem 1: Exemplo de acesso ao interface localStorage.

Infelizmente, a persistência de objectos não é nativamente suportada, para tal é necessário serialização e desserialização dos objectos. Uma solução muito popular passa pela tradução dos objectos para JSON (JavaScript Object Notation) - uma notação para intercâmbio de dados derivada do JavaScript. Este processo é suportado pelo objecto JSON utilizando as funções stringify e parse para serializar e desserializar, respectivamente. Vejamos o exemplo abaixo, onde aproveitámos para estender os métodos setItem e getItem para suportar a persistência de objectos.

```
//Guardar referência às funções originais
Storage.prototype._setItem =
Storage.prototype.setItem;
Storage.prototype._getItem =
Storage.prototype.getItem;

//Override da função setItem
Storage.prototype.setItem = function (key,
value) {
    //Se for um objecto, serializar
    if (value instanceof Object) {
        value = JSON.stringify(value);
    }
    //Guardar
    this._setItem(key, value);
}

/* Override da função getItem
* O parâmetro isObject é utilizado para
* indicar se o valor a devolver é um
* objecto
*/
Storage.prototype.getItem = function (key,
isObject) {
    //Obter
    var item = this._getItem(key);
    //Se for um objecto, desserializar
    if (isObject) {
        item = JSON.parse(item);
    }
    return item;
}

//Guardar um objecto
var obj = { foo: "bar" };
localStorage.setItem("chave", obj);
```

WEB PERSISTENTE: LOCAL STORAGE

```
//Obter um objecto
var obj =
localStorage.getItem("chave", true);
```

Listagem 2: Persistência de objectos serializados em Local Storage.

Conseguimos assim uma interface de armazenamento persistente capaz de suportar não só tipos de dados primitivos mas também objectos.

De seguida analisamos o contexto em que pretendemos aplicar esta tecnologia.

O problema da Web não-persistente

As aplicações web executadas no browser têm como característica a completa dependência de um servidor que fornece toda a lógica da aplicação. Isto significa que em cada acesso o browser tem de descarregar todos os elementos da aplicação, independentemente de qualquer acesso anterior. Esta redundância de dados que circulam na rede agrava-se ao verificarmos que a componente visual (HTML, CSS & JavaScript) não serve qualquer interesse ao servidor. O ideal seria persistir estes elementos no browser no primeiro acesso, eliminando a necessidade de nova transferência em acessos posteriores.

Os sistemas de cache implementados pelos browsers assistem ao guardar localmente alguns elementos, no entanto apenas reduzem a quantidade de informação trocada, não eliminando por completo a transferência da componente visual da aplicação. Adicionalmente, não existe uma interface programática que permita a gestão deste recurso.

A utilização de AJAX (Asynchronous Javascript and XML) é outra técnica que permite a redução da quantidade de dados transferidos ao possibilitar ligações assíncronas ao servidor. É possível utilizar esta técnica para, por exemplo, descarregar a componente visual da aplicação na abertura da página web, e posteriormente utilizar ligações AJAX para obter apenas o resultado do servidor. No entanto introduz-se um novo problema: cada nova abertura da aplicação torna-se mais lenta e pesada. Para além disso, o problema inicial mantém-se pois continuamos a depender do servidor para fornecer a componente visual da aplicação em cada novo acesso.

O Local Storage vem oferecer uma solução mais completa ao permitir que o programador persista no browser todos os elementos de apresentação da aplicação, eliminando completamente a transferência desta componente a partir do servidor. Exceptuam-se apenas dois casos incontornáveis: quando ainda não foi persistida e quando existem actualizações. Pode ser feita uma analogia a aplicações nativas: o primeiro acesso corresponde à "instalação", existindo "actualizações" posteriores apenas quando necessário. Após "instalada", a aplicação comunica com o servidor utilizando AJAX, consistindo a comunicação apenas da informação relevante ao pedido do utilizador, geralmente estruturada recorrendo a JSON ou XML.

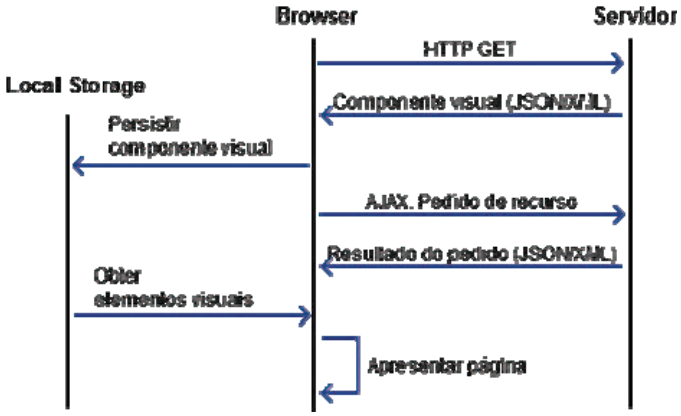


Figura 1: Diagrama da interacção entre browser e servidor utilizando Local Storage para persistência da componente visual da aplicação.

Exemplo de Implementação

Como demonstração concreta deste conceito vamos implementar uma aplicação simples, recorrendo à linguagem PHP para a implementação de um webservice.

O seguinte diagrama auxilia na contextualização dos ficheiros que compõem este exemplo, indicando a posição final dos mesmos após a primeira abertura da plataforma pelo browser.

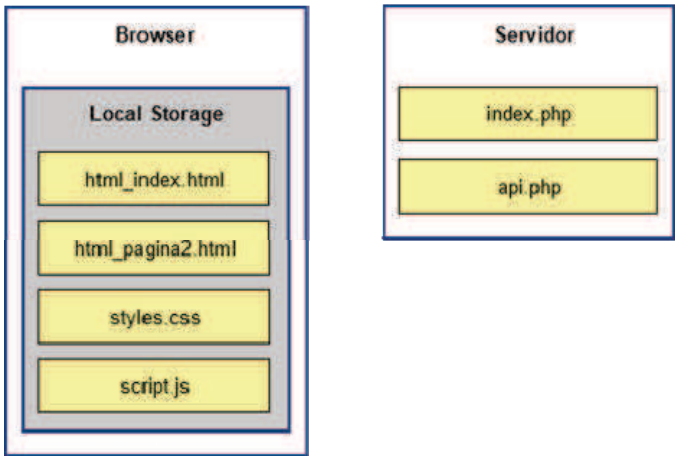


Figura 2: Contextualização dos ficheiros que compõem o exemplo de implementação.

Todo o código de seguida apresentado foi cuidadosamente documentado para melhor compreensão.

Ficheiro index.html

```
<!DOCTYPE html>
<html>
 <head>
  <title>Aplicação Exemplo</title>
<meta http-equiv="Content-Type"
                content="text/html; charset=UTF-8">
  <!-- Carregar jQuery -->
<script src="http://code.jquery.com/
                              jquery-2.0.0.min.js">
```

```
</script>
<!-- Carregar script com a lógica de
apresentação da aplicação -->
<script src="script.js"></script>
  <style type="text/css"></style>
 </head>
 <body>
 </body>
</html>
```

Listagem 3: Ficheiro index.html. Ponto de entrada para a aplicação.

Ficheiro html_index.html

```
<div>
 <p>
Página de inicio.
<a href="pagina2">Ir para a Página 2.</a>
 </p>
</div>
```

Listagem 4: Ficheiro html_index.html. Conteúdo da página principal da aplicação.

Ficheiro html_pagina2.html

```
<div>
 <p>
Página 2.
<a href="index">Voltar ao início.</a>
 </p>
 <p id="serverContent"></p>
</div>

<script type="text/javascript">
/**
Exemplo de pedido de recursos ao servidor através
de AJAX.
O resultado é carregado para o parágrafo de ID
"serverContent".
**/
$("#serverContent").load("api.php?getExampl
eText=1");
</script>
```

Listagem 5: Ficheiro html_pagina2.html. Conteúdo da segunda página da aplicação.

Ficheiro styles.css

```
body
{
    background-color: #262626;
}

div
{
    padding: 10pt;
    margin: 0 auto;
    width: 800pt;
    text-align: center;
    background-color: white;
}


p
{
    border: solid 2px orange;
    padding: 5pt;
}
```

Listagem 6: Ficheiro styles.css. Folha de estilos da aplicação

Ficheiro api.php

```
<?php
/**
Pedido de template (componente visual da aplicação)
**/
if($_GET["getTemplate"] == "1")
{
    //Construir um array com os elementos
visuais
    $result = array
    (
        "version" => "0.1",
        //Páginas HTML
        "html_index" =>
              file_get_contents("html_index.html"),
        "html_pagina2" =>
            file_get_contents("html_pagina2.html"),
        //Folhas de estilo
        "css_styles" =>
                    file_get_contents("styles.css")
    );

    //Traduzir o array para JSON e enviar
   //como resposta
    echo json_encode($result);
    //Parar
    die();
}

/**
Exemplo de pedido de um determinado recurso
**/
if($_GET["getExampleText"] == "1")
{
    //Responder com texto demonstrativo
    echo "Este texto é o resultado de um
pedido AJAX ao servidor.";
    //Parar
    die();
}
?>
```

Listagem 7: Ficheiro api.php. Webservice fornecedor do resultado da lógica da aplicação.

Ficheiro script.js

```
//Quando o documento estiver pronto...
$(document).ready(function()
{
    captureLinkClick();
    captureOnPopState();

    //Se o template já existe no browser...
    if(templateExists())
    {
        //... mostrar aplicação
        showApplication();
    }
    else
    {
        //Senão, fazer download do template
        getTemplate(function()
        {
            //Quando concluído, mostrar
aplicação
            showApplication();
        });
    }
});

/**
```

```
Capturar click em links
Esta acção irá ter um comportamento diferente do
normal @method captureLinkClick
**/
function captureLinkClick()
{
    //Ao clicar num link...
    $(document).on("click", "a",
    function(e)
    {
        //...impedir o comportamento por defeito
        e.preventDefault();
        //Mudar para a página definida no attributo
                                        "href" do link
        changePage($(e.currentTarget).attr("href"),
                                                false);
    });
}

/**
Capturar eventos retroceder e avançar do histórico
do browser@method captureOnPopState
**/
function captureOnPopState()
{
    //Ao retroceder/avançar...
    window.onpopstate = function(e)
    {
        var pageName = "";

        //Obter objecto state que contém o nome da
                                        página destino
        var state = e.state;
        //Caso state seja null...
        if(state == null)
        {
            //...é o primeiro acesso à página.
            //Mostrar página principal (index)
            pageName = "index";
        }
        else
        {
            //Senão, mostrar página indicada no
                                        objecto state
            pageName = state.page;
        }

        //Mudar para a página destino
        changePage(pageName, true);
    }
}

/**
Verificar se o template existe no browser
@method templateExists
**/
function templateExists()
{
    //Neste caso verificamos a existência do
      parâmetro "version" como teste da presença do
                                                template
    return
localStorage.getItem("version") !== null;
    //É possível aproveitar este parâmetro para
//verificar se existe uma nova versão do template
//no servidor
}

/**
Obter template e guardar em Local Storage
@method getTemplate
@param {function} callback Função chamada após ter-
minado o download do template
**/
function getTemplate(callback)
{
    //Fazer pedido ao servidor pelo
template
    $.ajax(
    {
        url: "api.php?getTemplate=1",
        type: "get",
        success: function(data)
        {
            //Resposta JSON
            //Fazer parse do resultado do servidor
            var jsonData = $.parseJSON(data);
            //Iterar os elementos enviados pelo
                                            //servidor
            $.each(jsonData, function(key, value)
            {
                //Guardar em Local Storage
                localStorage.setItem(key, value);
            });

            //Chamar função callback
            callback();
        }
    });
}

/**
Carregar pela primeira vez os elementos visuais
para o ecrã
@method showApplication
**/
function showApplication()
{
    //Carregar folha de estilos do template
    $("style").append(localStorage.getItem
("css_styles"));
    //Mostrar página inicial (index)
    changePage("index", true);
}

/**
Carregar uma determinada página guardada em localS-
torage
@method changePage
@param {String} pageName Nome da página a carregar
@param {boolean} replaceState Indica se deve ser
criado ou reutilizado um estado do histórico
**/
function changePage(pageName, replaceState)
{
    //Carregar para o corpo do documento o conteúdo
presente em Local Storage correspondente à pagina
indicada
    $("body").html(localStorage.getItem
                                    ("html_+pageName));
    //Construir objecto de estado stateObject, que
indica o nome da página destino
    var stateObject = {page: pageName};

    //Se foi indicado que o estado deve ser
                                        //substituido...
    if(replaceState)
    {
        //...substituir o estado do histórico
       //utilizando o objecto de estado stateObject
        history.replaceState(stateObject, null,
pageName);
    }
    else
    {
        //Em caso contário, criar um novo estado do
histórico utilizando o objecto de estado
stateObject
```

```
        history.pushState(stateObject, null, page-
Name);
    }
}
```

Listagem 8: Ficheiro script.js. Lógica de apresentação da aplicação.

Caso prático: Plataforma AnyKB

AnyKB é uma plataforma web colaborativa para a partilha de conhecimento no formato de artigos criados pela comunidade, vocacionada para a área tecnológica. A plataforma surge da necessidade de implementar o conceito descrito neste artigo numa aplicação de utilidade real com o objectivo de gerar resultados num ambiente de utilização o mais intensivo possível.

Figura 3: Página inicial da plataforma AnyKB.

Na plataforma AnyKB, qualquer utilizador pode partilhar com a comunidade as suas descobertas, experiências e soluções, ou ainda melhorar o conteúdo já existente. A participação é recompensada na forma de pontos que reflectem a reputação do utilizador dentro da plataforma e a qualidade dos conteúdos que cria. O utilizador enriquece a comunidade com o seu conhecimento e é recompensado com a aprovação desta, num formato quantitativo que destaca o utilizador entre os profissionais da sua área.

Deixo assim o convite para visitar e experimentar o desempenho da plataforma em http://www.anykb.net e contribuir para a comunidade com os seus artigos!

Resultados Estatísticos

Por forma a estudar a escalabilidade da aplicação foram comparadas duas implementações equivalentes do mesmo projecto. A versão "*Proof of Concept*" (PoC) implementa o conceito detalhado neste documento, e a versão "Clássica" funciona como a maioria dos *websites* de hoje: cada página corresponde a um pedido novo (GET) ao servidor.

O desempenho das duas versões foi comparado ao criar um caso de teste que consiste na navegação pelas várias páginas da aplicação, de forma a tirar partido dos benefícios que ambas as abordagens têm ao seu dispor. A métrica utilizada corresponde ao tempo decorrido desde a acção do utilizador (clique num *link*, por exemplo) até ao carregamento completo da página pelo *browser*. Adicionalmente, o caso de teste foi executado em vários cenários de carga do servidor, por forma a analisar como este factor afecta o desempenho de ambas as versões. Os valores de CPU *Load* apresentados correspondem ao formato conhecido como Unix CPU Load [2], tendo sido o servidor levado ao limiar da capacidade de resposta.

O servidor utilizado é uma máquina virtualizada recorrendo ao software para virtualização Xen [4], assente em infra-estrutura escalável de última geração, recorrentemente referenciada como *Cloud*. O servidor apresenta as seguintes características:

CPU: Quad-core Intel® Xeon® CPU E5649 @ 2.53GHz ou equivalente

RAM: 1.5GB DDR3 (1033Mhz)

Disco: 25GB (10K)

Software utilizado:

CentOS Linux 2.6.32 (64-bit)

Apache 2.2.15 (64-bit)

PHP 5.3.3 (64-bit)

MySQL 14.14 (64-bit)

Google Chrome 28 (64-bit)

Para simular, de forma controlada, os vários cenários de carga no servidor foi utilizado um script que executa, sem objectivo, operações pesadas de I/O e CPU.

O gráfico abaixo mostra, em média e mediana, o *speedup* conseguido pela versão PoC relativamente à versão Clássica, ao longo dos vários cenário de carga. É possível verificar que, em condições normais, a versão PoC consegue um desempenho cerca de 5 vezes superior. O desempenho baixa para cerca de 2 a 3 vezes superior à medida que a carga do servidor aumenta. No entanto, a mediana mostra-nos a tendência que a versão PoC tem para um desempenho mais elevado, conseguindo resultados até cerca de 24 vezes superiores.

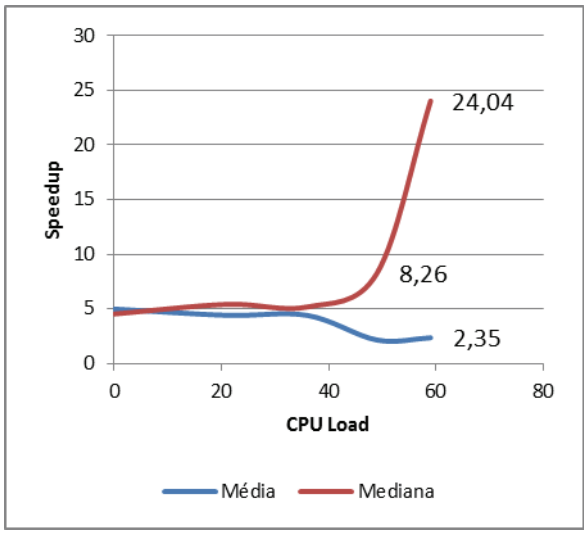


Gráfico 1: *Speedup* médio e mediano da versão PoC, comparativamente à versão Clássica.

WEB PERSISTENTE: LOCAL STORAGE

O seguinte gráfico mostra a evolução do tempo médio e mediano do carregamento das páginas da aplicação, ao longo dos vários cenários de carga. Como esperado, é possível verificar que o valor médio e mediano da versão clássica cresce à medida que a carga do servidor aumenta. O valor médio da versão PoC cresce igualmente com a carga do servidor mas mostra uma curva mais suave, enquanto que a mediana da versão PoC se mantém estável em cerca de 78ms. Isto acontece porque existe uma compensação do tempo acrescido que a informação demora a chegar do servidor pela instantaneidade de reposta do interface, e pela redução substancial de dados enviados pelo servidor.

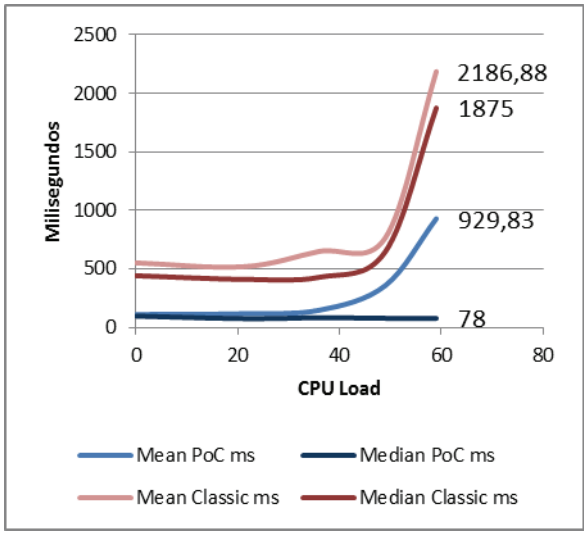


Gráfico 2: Tempo médio e mediano (em milissegundos) do carregamento de várias páginas de ambas as versões.

Conclusão

Concluímos assim que a separação da lógica da apresentação da restante lógica da aplicação, fazendo recurso ao Local Storage, possibilita ganhos de desempenho cerca de 5 vezes superior, comparativamente à mesma aplicação servida de forma clássica. Em casos extremos de carga o valor médio do desempenho baixa para 2 a 3 vezes superior, enquanto que o valor mediano mostra uma capacidade 24 vezes superior de resposta, colocando em evidencia a tendência para a fluidez, e não o contrário.

A implementação deste conceito possibilita assim a criação de aplicações mais rápidas e fluidas, fornecendo uma experiencia mais agradável ao utilizador.

> ***"O Local Storage vem oferecer uma solução mais completa ao permitir que o programador persista no browser todos os elementos de apresentação da aplicação, eliminando completamente a transferência desta componente a partir do servidor."***

Referências

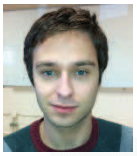
[1] World Wide Web Consortium (W3C), "W3C Proposed Recommendation 9 April 2013," [Online]. Available: http://www.w3.org/TR/webstorage/. [Acedido em Julho 2013].

[2] A. Deveria, "Can I use... Support tables for HTML5, CSS3, etc," [Online]. Available: http://caniuse.com/#feat=namevalue-storage. [Acedido em Junho 2013].

[3] Linux Kernel Organization, Inc., "The Linux Kernel Archives," [Online]. Available: https://www.kernel.org/doc/Documentation/cpu-load.txt. [Acedido em Junho 2013].

[4] Xen Project, "The Xen Project, the powerful open source industry standard for virtualization," [Online]. Available: http://www.xenproject.org/. [Acedido em Julho 2013].

AUTOR



Escrito por Telmo Marques

Estudante licenciado em eng.ª informática com um afecto especial por tecnologias *web*, encontra-se atualmente a concluir o Mestrado em Computação Móvel (MEI-CM) do Instituto Politécnico de Leiria (ESTG-IPLeiria), sob orientação do professor Patrício Domingues, no âmbito do qual se encontra inserido este artigo. Profissionalmente, após uma curta passagem pelo Instituto de Telecomunicações como investigador, é atualmente o responsável pelo desenvolvimento de *software* numa empresa portuguesa de serviços *web.*